

## **SESSION**

# **NUMERICAL METHODS + APPROXIMATION AND ESTIMATION TECHNIQUES + OPTIMIZATION METHODS**

**Chair(s)**

**TBA**





# Numerical Solutions of Heat and Mass Transfer with the Third Kind Boundary and Initial Conditions in Capillary Porous Media Using Programmable Graphics Hardware

Hira Narang, Fan Wu and Aswad Abdul Shakur

Computer Science Department

Tuskegee University

Tuskegee, AL 36088

E-mail: narang@mytu.tuskegee.edu, wuf@mytu.tuskegee.edu and aswadat@gmail.com

**Abstract**—Nowadays, a heat and mass transfer simulation plays an important role in various engineering and industrial fields. To analyze physical behaviors of a thermal environment, we have to simulate heat and mass transfer phenomena. However to obtain numerical solutions to heat and mass transfer equations is much time-consuming. In this paper, therefore, one of acceleration techniques developed in the graphics community that exploits a graphics processing unit (GPU) is applied to the numerical solutions of heat and mass transfer equations. Implementation of the simulation on GPU makes GPU computing power available for the most time-consuming part of the simulation and calculation. The nVidia CUDA programming model provides a straightforward means of describing inherently parallel computations. This paper improves the computational performance of solving heat and mass transfer equations with the third kind boundary and initial conditions numerically running on GPU. We implemented simulation of heat transfer using the novel CUDA platform on nVidia Quadro FX 4800 and compared its performance with an optimized CPU implementation on a high-end Intel Xeon CPU. The experimental results of heat transfer clearly show that GPU can perform heat transfer simulation accurately and significantly accelerate the numerical calculation with the maximum observed speedups 20 times. Therefore, the GPU implementation is a promising approach to acceleration of the heat transfer simulation.

**Keywords-Genereal:** Numerical Solution; Heat and Mass Transfer; High Performance Computation; General Purpose Graphics Processing Unit; CUDA.

## I. INTRODUCTION

During the last 4-5 decades, many scientists and engineers working in Heat and Mass Transfer processes have focused their attention to finding solutions both analytically/numerically, and experimentally. To precisely analyze physical behaviors of thermal environments, we need to simulate several heat and mass transfer phenomena such as heat conduction, convection, and radiation. A heat transfer simulation is accomplished by combining multiple computer simulations of such heat and mass transfer phenomena. With the advent of computer, initially the sequential solutions were found, and later when super-computers became available, fast solutions were obtained to above mentioned problems. However, the simulation of heat and mass transfer requires much longer execution time than the other simulations. Therefore, acceleration of the heat and

mass transfer simulation is essential to realize a practical large-scale heat and mass transfer simulation.

This paper exploits the computing power of graphics processing units (GPUs) to accelerate the heat and mass transfer simulation. GPUs are cost-effective in terms of theoretical peak floating-point operation rates [1]. Therefore, comparing with expensive cluster, GPUs is a powerful co-processor on a common desktop PC that is ready to achieve a large-scale heat and mass transfer simulation at a low cost. The GPU has several key advantages over CPU architectures for highly parallel, compute intensive workloads, including higher memory bandwidth, significantly higher floating-point throughput. The GPU can be an attractive alternative to CPU clusters in high performance computing environments.

Recent announcement like CUDA [2] by nVidia proved their effort to extend both programming and memory models. CUDA (Compute Unified Device Architecture) is a new data-parallel, C-language programming API that bypasses the rendering interface and avoids the difficulties of classic GPGPU. Parallel computations are instead expressed as general-purpose, C-language kernels operating in parallel over all the points in a domain.

This paper investigates the numerical solutions to Two-point Initial-Boundary Value Problems (TIBVP) of Heat transfer with the third boundary and initial conditions arising in capillary porous media. These problems find applications in drying processes, under-ground contaminants transport, absorption of nutrients in human bodies, transpiration cooling of space vehicles at re-entry into atmosphere, and many other science and engineering problems. Although traditional approaches of parallel-distributed processing have been applied with advantage to the solutions of some of these problems, no more seem to have explored the high performance solutions to these problems with compact multi-processing capabilities of GPU, which is multi-processors technology on a chip. With the power of this compact technology and develop relevant algorithms to find the solution of TIBVP with the third boundary and initial conditions and compare with some of the existing solutions to simple known problems. All of our experimental results show satisfactory speedups. The maximum observed speedups are about 10 times.

The rest of the paper is organized as follow: Section II introduces some previous related work; Section III describes the background on GPU and CUDA briefly; Section IV presents the mathematical model of heat and mass transfer and numerical solutions to heat and mass transfer equations;

Our experimental results of heat transfer are presented in Section V; Finally Section VI concludes this paper with our future direction.

## II. RELATED WORK

The simulation of heat and mass transfer has received much attention for years. And there is much work related to this field, such as modeling and dynamic simulation. Here we just refer to some recent work closely related.

Soviet Union was in the fore-front for exploring the coupled Heat and Mass Transfer in Porous media was researched as a part of chemical engineering discipline, and major advances were made at Heat and Mass Transfer Institute at Minsk, BSSR. Later England and India took the lead and made further advances in terms of analytical and numerical solutions to certain problems. Later Narang and Rajiv [4] explored the wavelet solutions and Ambethkar [5] explored the numerical solutions to some of these problems.

With the programmability of fragments on GPU, Krüger et al. [6] computed the basic linear algebra problems, and further computed the 2D wave equations and NSEs on GPU. Bolz et al. [7] rearranged the sparse matrix into textures, and utilized them multigrid method to solve the fluid problem. Similarly, Goodnight et al. [8] used the multigrid method to solve the boundary value problems on GPU. Harris [9, 10] solved the PDEs of fluid motion to get cloud animation.

GPU is also used to solve other kinds of PDEs. For example, Kim et al. [11] solved the crystal formation equations on GPU. Lefohn et al. [12] packed the level-set isosurface data into a dynamic sparse texture format, which was used to solve the PDEs. Another creative usage was to pack the information of the next active tiles into a vector message, which was used to control the vertices and texture coordinates needed to send from CPU to GPU. To learn more applications about GPU for general-purpose computations, readers can refer to [13].

## III. AN OVERVIEW OF CUDA ARCHITECTURE

The GPU that we have used in our implementations is nVidia's Quadro FX 4800, which is DirectX 10 compliant. It is one of nVidia's fastest processors that support the CUDA API and as such all implementations using this API are forward compatible with newer CUDA compliant devices. All CUDA compatible devices support 32-bit integer processing. An important consideration for GPU performance is its level of occupancy. Occupancy refers to the number of threads available for execution at any one time. It is normally desirable to have a high level of occupancy as it facilitates the hiding of memory latency.

The GPU memory architecture is shown in figure 1.

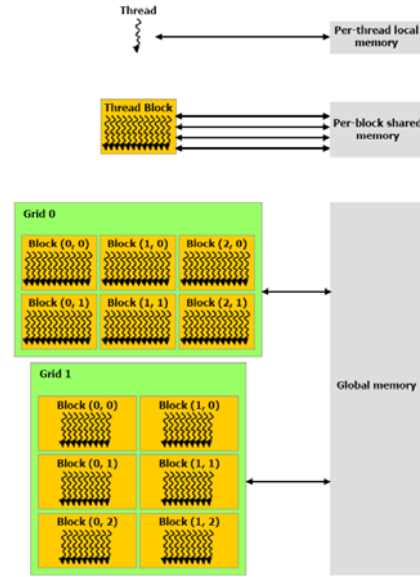


Figure 1: GPU Memory Architecture [2]

## IV. MATHEMATICAL MODEL AND NUMERICAL SOLUTIONS OF HEAT AND MASS TRANSFER

### A. Mathematical Model

Consider the Heat and Mass Transfer through a porous slab with boundary conditions of the third kind. The third kind of boundary condition which is also referred to as convective boundary condition, is a more common practical situation, where a heat transfer occurs at the boundary surface to or from a fluid flowing on the surface of a slab at a known temperature and a known heat transfer coefficient, eg. in heat exchangers, condensers, reboilers etc.

Let the x-axis be directed upward along the slab and the y-axis normal to the slab. Let  $u$  and  $v$  be the velocity components along the x- and y- axes respectively. Let us assume that the slab is accelerating with a velocity  $u = Ut$  in its own plane at time  $t \geq 0$ . Then the heat and mass transfer equations in the Boussinesq's approximation, are:

$$\frac{\partial v_1}{\partial x_1} = 0 \quad (1)$$

$$\frac{\partial u_1}{\partial t_1} + v_1 \frac{\partial u_1}{\partial x_1} = g\beta(T_1 - T_\infty) + \frac{v\partial^2 u_1}{\partial x_1^2} - \frac{\sigma B_0^2 u_1}{\rho} \quad (2)$$

$$\frac{\partial T_1}{\partial t_1} + v_1 \frac{\partial C_1}{\partial x_1} = \frac{k\partial^2 T_1}{\partial x_1^2} \quad (3)$$

$$\frac{\partial C_1}{\partial t_1} + v_1 \frac{\partial C_1}{\partial x_1} = \frac{D'\partial^2 C_1}{\partial x_1^2} \quad (4)$$

Equating the conduction heat flux to convection heat flux at the left surface of the slab and taking into consideration

the direction of heat flow (i.e. whether it is in the positive X-direction or negative X-direction), we can represent the initial and boundary conditions mathematically as follows,

$$t_1 \leq 0, u_1(x_1, t_1) = 0 \quad (5)$$

$$T_1(x_1, t_1) = T_\infty, C_1(x_1, t_1) = C_\infty$$

$$t_1 > 0, u_1(0, t_1) = V_0 \quad (6)$$

$$h(T_1 - T)|_{x_1=0} = -k \frac{\partial T}{\partial x_1}|_{x_1=0}$$

$$\frac{\partial C(0, t_1)}{\partial x_1} + k \frac{\partial T(0, t_1)}{\partial x_1} = 0$$

$$t_1 > 0 \quad (7)$$

$$u_1(\infty, t_1) \rightarrow 0, T_1(\infty, t_1) \rightarrow T_\infty$$

$$C_1(\infty, t_1) \rightarrow C_\infty \text{ as } x_1 \rightarrow \infty$$

Since the slab is assumed to be porous, Equation (1) integrates to  $v_1 = -v_0$  is the constant velocity. Here,  $\mu_1$  is the velocity of the fluid,  $T_p$  the temperature of the fluid near the slab,  $T_\infty$  the temperature of the fluid far away from the slab,  $C_p$  the concentration near the slab,  $C_\infty$  the concentration far away from the slab,  $g$  the acceleration due to gravity,  $\beta$  the coefficient of volume expansion for heat transfer,  $\beta'$  the coefficient of volume expansion for concentration,  $\nu$  the kinematic viscosity,  $\sigma$  the scalar electrical conductivity,  $\omega$  the frequency of oscillation,  $k$  the thermal conductivity,  $h$  is heat transfer coefficient and  $t_1$  is the time.

From Equation (1) we observe that  $v_1$  is independent of space co-ordinates and may be taken as constant. We define the following non-dimensional variables and parameters.

$$t = \frac{t_1 V_0^2}{4\nu}, x = \frac{V_0 x_1}{4\nu} \quad (8)$$

$$u = \frac{u_1}{V_0}, T = \frac{T_1 - T_\infty}{T_p - T_\infty}, C = \frac{C_1 - C_\infty}{C_p - C_\infty}, P_r = \frac{\nu}{k}, S_c = \frac{\nu}{D}$$

$$M = \frac{\sigma B_0^2 \nu}{\rho V_0^2}, G_r = \frac{\nu g \beta (T_p - T_\infty)}{V_0^3}$$

$$G_m = \frac{\nu g \beta' (C_p - C_\infty)}{V_0^3}, \omega = \frac{4\nu \omega_i}{V_0^2}$$

Now taking into account Equations (5), (6), (7), and (8), equations (2), (3) and (4) reduce to the following form:

$$\frac{\partial u}{\partial t} - 4 \frac{\partial u}{\partial x} = 4 \frac{\partial^2 u}{\partial x^2} + 4G_r T - 4Mu + 4G_m C \quad (9)$$

$$\frac{\partial T}{\partial t} - 4 \frac{\partial T}{\partial x} = \frac{4}{P_r} \frac{\partial^2 T}{\partial x^2} \quad (10)$$

$$\frac{\partial C}{\partial t} - 4 \frac{\partial C}{\partial x} = \frac{4}{S_c} \frac{\partial^2 C}{\partial x^2} \quad (11)$$

with

$$t \leq 0 \quad (12)$$

$$u(x, t) = 0, T(x, t) = 0$$

$$t > 0 \quad (13)$$

$$u(0, t) = 0,$$

$$h(T_1 - T)|_{x_1=0} = -k \frac{\partial T}{\partial x_1}|_{x_1=0}$$

$$\frac{\partial C(0, t_1)}{\partial x_1} + k \frac{\partial T(0, t_1)}{\partial x_1} = 0$$

$$t > 0 \quad (14)$$

$$u(\infty, t) = 0, T(\infty, t) = 0$$

$$C(\infty, t) = 0, \text{ as } x_1 \rightarrow \infty$$

## B. Numerical Solutions

Here we sought a solution by finite difference technique of implicit type namely Crank- Nicolson implicit finite difference method which is always convergent and stable. This method has been used to solve Equations (9), (10), and (11) subject to the conditions given by (12), (13) and (14). To obtain the difference equations, the region of the heat is divided into a grid or mesh of lines parallel to  $x$  and  $t$  axes. Solutions of difference equations are obtained at the intersection of these mesh lines called nodes. The values of the dependent variables  $T$ ,  $u$  and  $C$  at the nodal points along the plane  $x = 0$  are given by  $T(0, t)$ ,  $u(0, t)$  and  $C(0, t)$  hence are known from the boundary conditions.

In the figure 2,  $\Delta x$ ,  $\Delta t$  are constant mesh sizes along  $x$  and  $t$  directions respectively. We need an algorithm to find single values at next time level in terms of known values at an earlier time level. A forward difference approximation for the first order partial derivatives of  $u$ ,  $T$  and  $C$ . And a central difference approximation for the second order partial derivative of  $u$ ,  $T$  and  $C$  are used. On introducing finite difference approximations for:

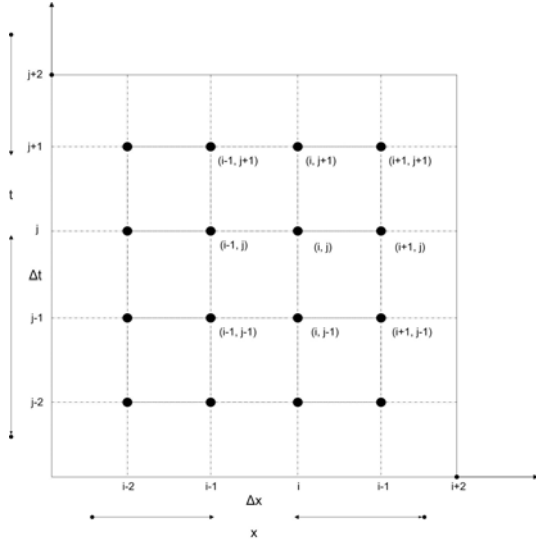


Figure 2: Finite Difference Grid

$$\left(\frac{\partial T}{\partial x}\right)_{i,j} = \frac{T_{i+1,j} - T_{i-1,j} + T_{i+1,j+1} - T_{i-1,j+1}}{4(\Delta x)} \quad (15)$$

$$\left(\frac{\partial C}{\partial x}\right)_{i,j} = \frac{C_{i+1,j} - C_{i-1,j} + C_{i+1,j+1} - C_{i-1,j+1}}{4(\Delta x)}$$

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{u_{i+1,j} - u_{i-1,j} + u_{i+1,j+1} - u_{i-1,j+1}}{4(\Delta x)}$$

$$\left(\frac{\partial T}{\partial t}\right)_{i,j} = \frac{T_{i,j+1} - T_{i,j}}{\Delta t}, \left(\frac{\partial C}{\partial t}\right)_{i,j} = \frac{C_{i,j+1} - C_{i,j}}{\Delta t}, \left(\frac{\partial u}{\partial t}\right)_{i,j} = \frac{u_{i,j+1} - u_{i,j}}{\Delta t}$$

$$\left(\frac{\partial^2 T}{\partial x^2}\right)_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j} + T_{i+1,j+1} + T_{i-1,j+1} - 2T_{i,j+1}}{2(\Delta x)^2}$$

$$\left(\frac{\partial^2 C}{\partial x^2}\right)_{i,j} = \frac{C_{i+1,j} + C_{i-1,j} - 2C_{i,j} + C_{i+1,j+1} + C_{i-1,j+1} - 2C_{i,j+1}}{2(\Delta x)^2}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} = \frac{u_{i+1,j} + u_{i-1,j} - 2u_{i,j} + u_{i+1,j+1} + u_{i-1,j+1} - 2u_{i,j+1}}{2(\Delta x)^2}$$

The finite difference approximation of Equations (9), (10) and (11) are obtained with substituting Equation (15) into Equations (9), (10) and (11) and multiplying both sides by

$\Delta t$  and after simplifying, we let  $\frac{\Delta t}{(\Delta x)^2} = r' = 1$  (method is always stable and convergent), under this condition the above equations can be written as:

$$2u_{i,j+1} \left(\frac{1}{2} + \frac{\Delta t}{\Delta x}\right) u_{i+1,j+1} + \left(\frac{\Delta t}{\Delta x} - \frac{1}{2}\right) u_{i-1,j+1} \\ = \left(\frac{1}{2} + \frac{\Delta t}{\Delta x}\right) u_{i+1,j} + \left(\frac{1}{2} + \frac{\Delta t}{\Delta x}\right) u_{i-1,j+1} + \\ + 4G_r \Delta t T_{i,j} + 4G_m \Delta t C_{i,j} - 4M \Delta t u_{i,j} \quad (16)$$

$$\left(1 + \frac{4}{P_r}\right) T_{i,j+1} + \left(\frac{\Delta t}{\Delta x} - \frac{2}{P_r}\right) T_{i-1,j+1} - \left(\frac{\Delta t}{\Delta x} - \frac{2}{P_r}\right) T_{i+1,j+1} \\ = \left(\frac{\Delta t}{\Delta x} - \frac{2}{P_r}\right) T_{i+1,j} + \left(\frac{2}{P_r} - \frac{\Delta t}{\Delta x}\right) T_{i-1,j} + \left(1 - \frac{4}{P_r}\right) T_{i,j} \quad (17)$$

$$\left(1 + \frac{4}{S_c}\right) C_{i,j+1} + \left(\frac{\Delta t}{\Delta x} - \frac{2}{S_c}\right) C_{i-1,j+1} - \left(\frac{\Delta t}{\Delta x} - \frac{2}{S_c}\right) C_{i+1,j+1} \\ = \left(\frac{2}{S_c} + \frac{\Delta t}{\Delta x}\right) C_{i+1,j} + \left(\frac{2}{S_c} - \frac{\Delta t}{\Delta x}\right) C_{i-1,j} + \left(1 - \frac{4}{S_c}\right) C_{i,j} \quad (18)$$

## V. EXPERIMENTAL RESULTS AND DISCUSSION

### A. Setup and Device Configuration

The experiment was executed using the CUDA Runtime Library, Quadro FX 4800 graphics card, Intel Core 2 Duo. The programming interface used was Visual Studio.

The experiments were performed using a 64-bit Lenovo ThinkStation D20 with an Intel Xeon CPU E5520 with processor speed of 2.27 GHZ and physical RAM of 4.00GB. The Graphics Processing Unit (GPU) used was an NVIDIA Quadro FX 4800 with the following specifications:

CUDA Driver Version:	3.0
Total amount of global memory:	1.59 Gbytes
Number of multiprocessors:	24
Number of cores:	92
Total amount of constant memory:	65536 bytes
Total amount of shared memory per block:	16384 bytes
Total number of registers available per block:	16384
Maximum number of threads per block:	512
Banwidth:	

Host to Device Bandwith: 3412.1 (MB/s)

Device to Host Bandwith: 3189.4 (MB/s)

Device to Device Bandwith: 57509.6 (MB/s)

In this experiment, we considered a slab of thickness  $L$  and thermal conductivity  $K$ . At the left surface ( $x = 0$ ), a hot fluid of temperature  $T_1$  is flowing with a heat transfer coefficient  $h$ , supplying heat into the slab. Assuming the initial temperature of the slab is also the surrounding temperature denoted as  $T_\infty$ .

In the implementation, we assumed a fixed length of the slab and the region of the heat was divided into sample nodal points  $N$ . We wrote a C function and a kernel function to be executed on the CPU and GPU respectively for the purposes of obtaining the temperature distribution across the length of the slab at each nodal point using the Forward Euler Method (FEM). The logic and behavior of the two functions were similar in calculating the desired results. Furthermore, to obtain accurate results with minimal error, we performed the calculation using several iterations.

For the first iteration in each computation, we initialized the temperature  $T_0$  at the surface of the slab, where the hot fluid temperature  $T_1$  was set to some value. Then using  $T_0$  as the initial surface temperature, the heat transfer differential equation was used to propagate the heat across the length of the slab and the temperature at each node was held in a temporarily array to be overwritten in the next and subsequent iterations.

In the next and subsequent iterations,  $T_0$  was updated each time and then the heat was propagated across the length of the slab, at the same time the temperature distribution at each node was determined and temporarily stored. The computation was performed for several iterations until the temperature at each node stabilized. The final values of temperature at each nodal point were then recorded.

Finally, for each value of  $N$ , the corresponding GPU and CPU processing times were determined. To compare the performance of the GPU and CPU, we varied the number of sample points  $N$  to obtain different processing times for the GPU and CPU.

### B. Experimental Results

In this section, we show our results in solving heat transfer with the third initial and boundary conditions with CPU and GPU. For the purpose of implementation, the following constant values were used:

Thermal conductivity of slab (K):	0.55 W/mk
Heat transfer coefficient (h):	5 W/m <sup>2</sup> K
Specific heat capacity (c):	1300 J/kg K
Density of material ( $\rho$ ):	900 kg/m <sup>3</sup>
Temperature $T_1$ :	45 K
Temperature $T_\infty$ :	0 K

For the first part of our results, we used a slab of length 220 and the number of sample nodes  $N$  was set to 64. We performed the computation for 100 iterations and the results obtained is shown and discussed below.

The surface temperature at  $x = 0$  plotted against time is depicted in Figure 3. We can immediately see that the temperature gradually decreases from a maximum value at time equal zero and converges to the surrounding temperature  $T_\infty$  (0 K) as the time elapsed.

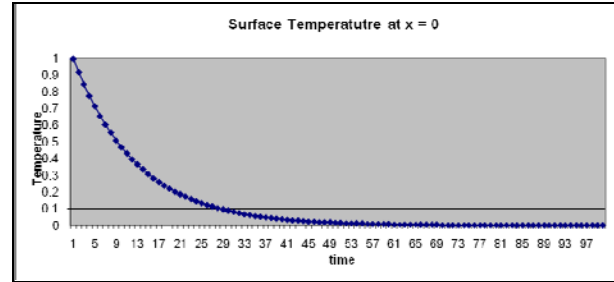


Figure 3: Surface temperature at  $x = 0$ ,  $T_\infty = 0$  K.

In addition, to obtain the temperature distribution across the length of the slab, we plotted the temperature against the number of nodes  $N$ . Figures 4 and 5 below show the temperature distribution for  $N = 64$  and  $N = 96$  respectively. From the graphs, we observed that the temperature is maximum at the surface of the slab (node 0) where the hot fluid is constantly applied and gradually decreases as we move away from the surface. The temperature approaches zero ( $T_\infty$ ) as we get closer to the far end of the slab. To add to the above, we also noticed that as we increased the length of the slab, the temperature distribution got infinitesimally closer to zero.

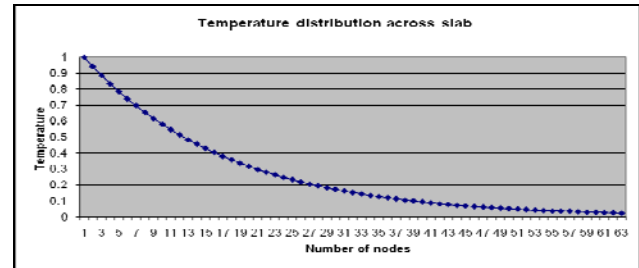


Figure 4: Temperature distribution across slab, length = 220,  $N = 64$ .

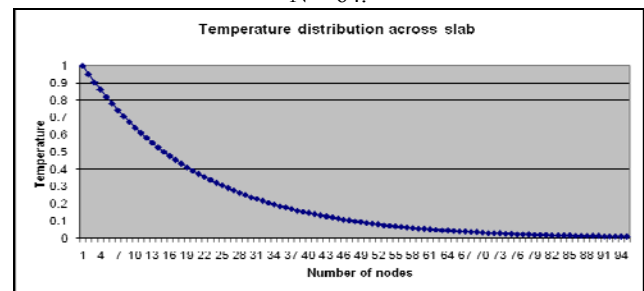


Figure 5: Temperature distribution across slab, length = 220,  $N = 96$ .

In our test, we also compared that both the results obtained from the GPU and CPU were the same. To minimize the error between the GPU and CPU results, we implemented similar functions in the GPU and CPU and the parameters passed to both functions were the same. We also implemented error checking procedures to keep track of errors that resulted from the computations. Furthermore, we observed that, large number of iterations resulted in accurate results. Hence, after running several tests using different values of  $N$ , we observed that the difference between the

GPU and CPU calculated values was very negligible. Table 1 shows the normalized numerical results obtained from the GPU and CPU.

TABLE I. COMPARISON OF GPU AND CPU RESULTS (TEMPRETURE)

GPU Results	CPU Results
1.00000	1.00000
0.92960	0.92955
0.86416	0.86345
0.74677	0.74265
0.69419	0.69389
0.51840	0.52876
0.44798	0.4555
0.35987	0.35987
0.26874	0.26889
0.18656	0.18908
0.13931	0.13123
0.10404	0.10409
0.08990	0.08680
0.07769	0.07809
0.06714	0.06435
0.05393	0.05710

For the second part of our results, we benchmarked the GPU (NVIDIA Quadro FX 4800) against the CPU (Intel Xeon E5520) in terms of processing or execution time.

For a fixed length of the slab, we varied N between 512 (29) and 65536 (216) and obtained the corresponding GPU and CPU processing times as follows:

- For values of N between 512 and 2048, the GPU was slower than the CPU. This is due to the fact that the GPU executes parallel instructions more efficiently whereas the CPU executes sequential instructions more efficiently. Therefore, for small values of N, the GPU has fewer blocks to execute and does not employ concurrent execution hence it is slow.
- For values of N greater than 2048, the GPU was considerably faster than the CPU.

The maximum speed up observed in this test was around 20 times. However, the speed up increased for increasing value of N.

Figure 6 depicts the graph of the GPU speed up.

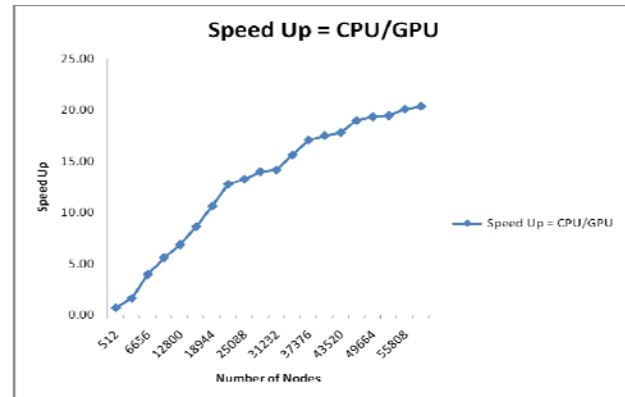


Figure 6: Performance of GPU and CPU Implementations

Finally, the accuracy of our numerical solution was dependent on the number of iterations we performed in calculating each nodal point, where more iteration mean more accurate results. In our experiment, we observed that after 14 or 15 iterations, the solution to the heat and mass equation at a given point became stable. For optimal performance, and to keep the number of iterations the same for both CPU and GPU, we used 15 iterations.

## VI. CONCLUSION AND FUTURE WORK

In this research, we found numerical solutions for heat transfer differential equations with convective boundary conditions (boundary conditions of the third kind) using the Finite Difference Method (FDM) on GPGPUs. We also implemented the Forward Euler Method (FEM) for iterative computations of the temperature distribution at various points in a slab. Furthermore, we benchmarked the performance of the GPGPU against the CPU in terms of execution or processing time.

In conclusion, our results show that FDM is well appropriate for parallel computation on GPUs. In addition, we have demonstrated that GPU- based implementations can give considerable performance improvement over CPU- based implementations. This is evident in the test case presented in the results section, where the maximum speed up of the GPU recorded was about 20 times over the CPU.

There are several avenues for future work. We would like to extend out results to mass transfer with the third kind initial and boundary conditions. We also would like to test our algorithm on different GPUs and explore the new performance opportunities offered by newer generations of GPUs. It would also be interesting to explore more tests with large scale data set. Finally, further attempts will be made to explore more complicated problems both in terms of boundary conditions as well as geometry.

## VII. ACKNOWLEDGMENT

This work has been supported in part by US. NSF grant HBCU-UP.

**REFERENCES**

- [1] J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A.E. Lefohn, T.J. Purcell.: A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum* 26(1) (2007) 80-113.
- [2] NVIDIA Corporation. NVIDIA Programming Guide 2.3. Retrieved July, 2009. [www.nvidia.com](http://www.nvidia.com).
- [3] A. V. Luikov. *Heat and Mass Transfer in Capillary Porous Bodies*, Pergamon Press, 1966.
- [4] Hira Narang and Rajiv Nekkanti. Wavelet-based Solution to Time-dependent Two-point Initial Boundary Value Problems with Non-Periodic Boundary Conditions, Proceedings of the IATED International Conference Signal Processing, Pattern Recognition & Applications July 3-6 2001, Rhodes, Greece.
- [5] Hira Narang and Rajiv Nekkanti. Wavelet-based Solution of Boundary Value Problems involving Hyperbolic Equations, Proceedings from the IATED International Conference Signal Processing, Pattern Recognition & Applications June 25-26, 2002
- [6] Hira Narang and Rajiv Nekkanti. Wavelet-based solutions to problems involving Parabolic Equations, Proceedings of the IATED International Conference Signal Processing, Pattern Recognition & Applications 2001, Greece.
- [7] Hira Narang and Rajiv Nekkanti. Wavelet-Based Solution to Elliptic Two-Point Boundary Value Problems with Non-Periodic Boundary Conditions, Proceedings from the WSEAS international conference in Signal, Speech, and Image processing Sept 25-28, 2002
- [8] Hira Narang and Rajiv Nekkanti. Wavelet-Based Solution to Some Time-Dependent Two-Point Initial Boundary Value Problems with Non-Linear Non-Periodic Boundary Conditions, International Conference on Scientific computation and differential equations, SCICADE 2003, Trondheim, Norway, June 30. July 4, 2003
- [9] Hira Narang and Rajiv Nekkanti. Wavelet based Solution to Time-Dependent Two Point Initial Boundary Value Problems with Non-Periodic Boundary Conditions involving High Intensity Heat and Mass Transfer in Capillary Porous Bodies, IATED International Conference proceedings, Gainesville, FL 2004.
- [10] Vishwavidyalaya Ambethkar. Numerical Solutions of Heat and Mass Transfer Effects of an Unsteady MHD Free Convective Flow Past an Infinite Vertical Plate With Constant Suction. *Journal of Naval Architecture and Marine Engineering*, pages 28-36, June, 2008.
- [11] Jens Krüger and Rüdiger Westermann. Linear Algebra Operators for GPU Implementation of Numerical Algorithms. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 908-916, July 2003.
- [12] Jeff Bolz, Ian Farmer, Eitan Grinspun and Peter Schröder. Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 917-924, July 2003.
- [13] Nolan Goodnight, Cliff Woolley, David Luebke and Greg Humphreys. A Multigrid Solver for Boundary Value Problems Using Programmable Graphics Hardware. In *Proceeding of Graphics Hardware*, pages 102-111, July 2003.
- [14] Mark Harris, William Baxter, Thorsten Scheuermann and Anselmo Lastra. Simulation of Cloud Dynamics on Graphics Hardware. In *Proceedings of Graphics Hardware*, pages 92-101, July 2003.
- [15] Mark Harris. *Real-Time Cloud Simulation and Rendering*. PhD thesis, 2003.
- [16] Theodore Kim and Ming Lin. Visual Simulation of Ice Crystal Growth. In *Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 86-97, July 2003.
- [17] Aaron Lefohn, Joe Kniss, Charles Hansen and Ross Whitaker. Interactive Deformation and Visualization of Level Set Surfaces Using Graphics Hardware. In *IEEE Visualization*, pages 75-82, 2003.
- [18] GPGPU website. <http://www.gpgpu.org>.

# Padé Approximants as Numerical Models for Mesoscopic Phenomena

M. George

Department of Physics, Southwestern College, Chula Vista, CA, USA

**Abstract** - *Padé approximants are useful in numerical analysis and computational science. In this paper they provide the basis for numerical modeling for mesoscopic systems. We place this in the context of an application in time series analysis. We discuss a novel point of view for the approximants that focuses on the boundary conditions for the time series. We regard the approximants as supplying a mean-field theory approach to mesoscopic systems. Conceptually, this centers around using the polynomial parts of the approximants as indicators of randomness. An approximant with a polynomial part of just low degree is highly influenced by the presence of the polynomial. This means that the resulting mean field theory differs significantly from approximant to approximant.*

**Keywords:** Padé approximants, time series, mesoscopic systems, mean field theory

## 1 Introduction

Padé approximants have proven themselves to be useful in computational science [6]. In this paper, we are concerned with the application of Padé approximants to time series related to mesoscopic phenomena. Mesoscopic systems include biosystems, fusion reactors, economic systems, and a great variety of other systems of current interest. This is a novel application of the approximants.

Padé approximants are ideally suited for use in modeling of mesoscopic systems. They provide analytic continuation and can supply a model suitable for describing changes over many different scales. This provided the basis for their first major application in the theory of critical phenomena [6]. In addition, the determination of Padé approximants is computationally efficient.

Both the theory and applications of Padé approximants are discussed in the book by Baker and Graves-Morris [6]. Our intention is to discuss some of the theory associated with the approximants in the context of a mean field theory. An expanded version of this paper is in preparation [11].

It is frequently emphasized that Padé approximants, as rational approximations, have a demonstrated usefulness in applications such as in statistical mechanics [4] that require analytic continuation of functions, a certain number of terms of the Maclaurin expansion of which are known. This focus on analyticity is primary in the monograph of Baker and

Graves-Morris, and they discuss much in detail that relates to Padé approximants.

On the other hand, the article by Kumar [3] points in a direction of broader concerns. This article is groundbreaking in applying Padé approximants to time series. Kumar utilizes random variables.

We emphasize the fact that each Padé approximant can be regarded as a numerical model of a zero-dimensional field. As a rational function, it can be interpreted as a mean-field theory. We suggest that this can be applied to mesoscopic systems. We have previously presented [9] a brief theoretical treatment of computation, itself, as a mesoscopic phenomenon. Typical mesoscopic systems are, at their base level, quantum-mechanical systems. However, at other scales of space or time than the microscopic, mesoscopic systems must be modeled in different ways, and the interfaces between models may be difficult to define and address.

In our treatment of computation, we used quantum Ising games [10]. The motivation for this arose in that, as computation evolves, the realm of quantum computing will not be the first to be encountered. Rather, with developments such as miniaturization, a mesoscopic regime will first be encountered. Furthermore, one must decide, in far-from-equilibrium systems, how to approach the dynamics of such systems. Games (and even single-person games) can provide a way of modeling in this non-equilibrium context. The question of what it means to “win” can be difficult to define, but there is no question that games can succeed in driving systems far from equilibrium.

The Padé approximants can be regarded as providing mean field theories throughout a certain scale addressed by the series data. As one shifts within this scale from the fine-grained to the coarse-grained, i.e. within the range of variation at this scale, one must address boundary concerns with adjacent scales. This is very similar to what is considered in the Hilbert-Huang transform and empirical mode decomposition [7]. The poles in Padé approximants (which represent mean-field approximations) yield information about the boundary at the next larger scale, while the polynomial part of the rational approximation addresses the fine-scale variation at the boundary at the next lower scale. Therefore, Padé approximants represent an interpolation throughout a given scale using boundary behavior.

The paper is arranged as follows. The next section addresses the aspect of analyticity. In the following section,



we discuss time series. The heart of the theory we discuss here is to applications in mesoscopic systems, which is taken up in Sec. 4. The last section is the conclusion where, among other things, we discuss possible improvements on the approach of Padé approximants for developing models at a particular scale of a mesoscopic system using series data.

## 2 Analyticity

The basic definition and properties of Padé approximants can be found in Baker and Graves-Morris [6]. We discuss briefly, in this section, some important aspects of Padé approximants relating to the approach that is traditional with series analysis, i.e. a focus on analyticity.

We assume that we are supplied with a data sequence:

$$a_0, a_1, \dots, a_N \quad (1)$$

where these quantities are assumed to be certain real numbers, and we have exact or extremely accurate estimates of these  $N + 1$  quantities. In traditional series analysis, these numbers are assumed to give the first  $N + 1$  coefficients of a Maclaurin series in some variable  $x$ . In general, for example, when considering time series, we cannot make any such assumption. However, we can always introduce some variable  $x$ , called a conjugate variable, just as if we were dealing with a Maclaurin series. We also assume that the data, at least in principle, can be extended indefinitely, and we wish to develop a model to predict additional quantities in the sequence. A Padé approximant is one such model.

It is reasonable, based on the assumptions we are making, to consider a formal power series, in the conjugate variable,

$$y = \sum_{n=0}^{\infty} a_n x^n \quad (2)$$

(whether or not this series converges). A Padé approximant  $\hat{y}_{[K/M]}(x)$  for  $y$ , is a rational function with numerator polynomial  $p_K(x)$  of degree  $K$  and a denominator polynomial  $q_M(x)$ , of degree  $M$ , such that

$$p_K(x) = \left(\sum_{n=0}^{\infty} a_n x^n\right)q_M(x) + O(x^{N+1}) \quad (3)$$

We allow the usual Padé approximant to be determined by this condition and a normalization condition. Note that (3) can determine the approximant only to within an overall factor. We can fix this factor by requiring the leading coefficient of  $q_M(x)$  to be equal to one.

The Padé approximant is an appealing mathematical object for a number of reasons. First, as a rational approximation it can potentially contain more sophisticated information than a mere polynomial approximation. Second, because it can have singularities that are just ordinary poles (and finite in number) it supplies an analytic approximation that in principle could be used to provide an analytic continuation of (2) if this series has a nonzero radius of convergence. Third, (3) resolves into just a sequence of simple linear equations to determine the polynomial

coefficients of the rational function. This makes its determination numerically straightforward. Lastly, this type of rational approximation is the “best possible” as it uses all available information about the series, and contains no extraneous information.

On the other hand, Padé approximants can be difficult to interpret. The diagonal approximants, with  $K = M$ , are often stressed. The famous Padé conjecture, which has been shown to have counterexamples (see Baker and Graves-Morris [6]), states that given diagonal approximants for an analytic function, some subsequence of the sequence of such approximants will converge to the function. In this paper, diagonal approximants have no special significance (other than the fact that they are “maximally complicated” approximants).

## 3 Time series

Kumar [3], in ground-breaking work, discussed the Padé approximant in economics for time series analysis, with respect to ARMA models (see Refs. [1], [2] and [5]). Analyticity is not a central issue with respect to time series, as one expects a certain level of “noisy” data. Kumar uses, instead, random variables and white noise in which to couch his theoretical discussion. Our approach is very different, as it follows a viewpoint of zero-dimensional field theory and mean field theory [12]. This more physical point of view has the advantage that one can develop a mean field theory for mesoscopic systems, something that is entirely outside the reach of Kumar’s approach.

For time series, we do not want to interpret the Padé approximants as analytic continuations for Maclaurin series. Instead, we identify a polynomial part of the Padé approximant as containing information about randomness, and we regard the associated rational expression as containing information about processes occurring at the scale of the mesoscopic system being considered.

With respect to time series, we must confront a new concept in Padé approximants: The idea of the boundary condition (in time). We regard the Padé approximant itself as a numerical mathematical model, rather than something representing possible analytic continuation. Thus, this concept becomes suddenly meaningful, and is actually important in considering time series, as opposed to analytic continuation.

With respect to boundary conditions, we are concerned with the overlap between scales of a mesoscopic system. This entails consideration of physical processes: Specifically, energy transfers. The modes of interactions leading to energy transfers are formulated in particular ways at each scale considered of relevance to the production of the time series data. Thus, even if we consider mathematical modeling in biology, or economics, or (even) history, we must take up issues that, at base, involve non-equilibrium thermodynamics.

There are six types of boundary conditions we will discuss in the next section. These may be denoted: analytic, periodic, inductive, inter-model, shock and physical (related

to a physical theory). All of these modes of discussing boundary conditions demonstrate the complexity of time series analysis in the context of mesoscopic systems.

The six different types of boundary conditions relate to boundary overlaps, and are not meant to be an exhaustive enumeration of possibilities. For example, it is perfectly possible to conceive of combinations of types: Say combining the inductive with the shock modes. From the point of view of dimensionality, one must ultimately formulate these overlaps geometrically. For example, a time series consists of a field in a zero-dimensional space (as a point embedded in three dimensions) that is constantly being updated, through instrumentation, in time.

We are monitoring the activity at just a single point, with respect to a certain measurement system, and the measured level of the activity. This is a field that, in the mathematical model of the system, is representative of some manifold, which we can think of as akin to a domain in a magnet. The time boundary of the domain is just a point (or a system of points), or a time interval (or a system of time intervals) or a combination of points and intervals. If we assume that this boundary cannot interact with things in the past (causality), it can interact with future domains, and its boundary can be shared with the boundary of other domains. The boundary, if it is extended, can even interact with itself.

Lastly, we point out that the effects of measurement devices in determining the level of activity at each time in a time series cannot be neglected. It is as if an intermediate, activation state is formed involving the device prior to a final record of the time series data is made. One should not assume that because this activation threshold may not be a quantum threshold, that significant uncertainty is not being introduced into time series data merely by the act of measurement.

We cannot adequately address some of the topics we have introduced in this section in our discussion in the next section. In a short paper, one can treat only a few concepts adequately. Here, we need to discuss the physics involved in time series analysis using Padé approximants, with respect to mesoscopic systems. It is this we take up in the following section.

## 4 Mesoscopic systems

Each Padé approximant supplies us with a numerical model. In general, if we use all available quantities in the time series, or at least some fairly large number, there will result numerous Padé approximants that yield a diversity of models. As we discussed above, the use of the approximants as models for mesoscopic systems comes down to a discussion of boundary conditions, of which we mentioned six in the preceding section.

The principal goal of using Padé approximants on time series data is to make predictions. From the previous section, noting that uncertainties can arise merely from measurement (and, one must add, from any “games” that the data collection and analysis group are “playing”), and that each Padé approximant supplies a very distinct and different mathematical model (point of view) from others, one must

always bear in mind that however one proceeds, there simply may be no meaningful pattern of prediction to extract from a particular approximant, or, indeed, from any of the approximants. Having said this, on the other hand, by considering boundary conditions, one can construct procedures for making predictions and for assessing predictions.

We assume the Padé approximant is being used to model time series from a mesoscopic system. The simplest type of boundary condition is analytic. Here, the analytic form of the rational function is used to predict future values. This was the original use that Baker put the approximants to in his application to critical phenomena (see Ref. 6).

The Ising model in one- and two-dimensions admits exact solutions in zero external field for thermodynamic functions that display analyticity [13]. Using series data, and assuming analyticity, Baker was able to estimate critical exponents for the three-dimensional Ising model.

To show how analyticity can be used to predict series data, suppose that  $R(x)$  is any rational approximation that has a Maclaurin series whose coefficients match the  $N + 1$  known coefficients listed in (1). Then, we can write this rational function in a partial fraction decomposition with a polynomial added to a series of terms of the general form,

$$\frac{a}{(1-\alpha x)^m} \quad (4)$$

Here,  $a$  and  $\alpha$  are constants, and  $m$  is some positive integer. (Note that  $a$  and  $\alpha$  can be complex.) By expanding all such terms in power series and combining the results with the background polynomial, we can predict the value of any coefficient of the formal series (2).

Thus, the analytic boundary condition simply utilizes the rational function to make series-coefficient predictions. As we point out with the use Baker put the approximants to, this is a reasonable approach in investigating models where we suspect analytic behavior.

The next type of boundary condition, the periodic boundary condition, is also very simple. Once again, as above, we assume that we have a partial fraction decomposition, and a background polynomial for the rational approximation. We ignore the coefficients of the background polynomial in our predictions. For the other terms, of the form (4), if  $\alpha$  is complex, signifying that there is a periodicity (whether or not some exponential decay or growth is also involved), we simply generate coefficients by way of a power series expansion, as in the analytic case, and we combine these to yield a prediction, ignoring cases where  $\alpha$  is real, i.e. no periodicity is involved. This type of prediction is plausible if, on the whole, “noise” is small, and the real  $\alpha$  are less than one in magnitude (i.e. yield exponential decay).

In general, we cannot expect this type of prediction to be very useful. One possible improvement, which we will not discuss in this short paper, is to replace the polynomial by white noise, as Kumar [3] does and use random variables. In any case, a suitable metric can be based on using this type of prediction for the tail of the coefficients in (1), where we take the last few coefficients and compare with predictions.

The inductive boundary conditions are similar to the periodic boundary conditions, but approach the issue of prediction in a slightly more sophisticated way. This approach is similar to the empirical mode decomposition associated with the Hilbert-Huang transform [7]. The polynomial background is used to define the mean and standard deviation of white noise, and white noise of these characteristics replaces the polynomial. The other terms in the partial fraction decomposition (4) are studied, considering  $a$ , the amplitude, and  $n$ , for each term, in addition to  $\alpha$ . These factors combine in a combinatorial term in prediction of a specific exponent, and if  $\alpha$  is complex, this entails an oscillation. Thus, these quantities define “empirical” modes. We can form groupings of these modes, and extract an “empirical mode decomposition”. Since this requires a detailed discussion of the Hilbert-Huang transform, we will not pursue this further in this short paper.

We will very briefly describe characteristics of other boundary conditions. The reader is referred to the extended paper [11], under preparation, for more details. We are thinking of time series as zero-dimensional fields. The terms of the form (4) provide a lowest-order perturbative theory, and so we refer to the theory, obtained by this type of modeling, using Padé approximants as numerical models, as a mean field theory. This is consistent with the idea implicit in the Hilbert-Huang transform, which the inductive boundary conditions utilize in developing a decomposition.

Inter-model boundary conditions combine several or all of the Padé approximants, for a given  $N$  in (1), together. This is a pseudo-quantum model. The amplitude associated with each approximant is determined by ignoring the polynomial background and using the remainder to predict the last several coefficients of the known coefficients in (1). Just as in quantum mechanics, we regard the model as supplying a superposition of states, the wavefunction for each state being the approximant (minus its polynomial background). The polynomial background is replaced by white noise.

In considering quantum mechanics, we can ask ourselves if a quantum game [11] can be used for boundary conditions. This requires a quantum model for the system, such as the quantum Ising model.

The shock boundary condition results from combining all Padé approximants as a multifunction. This will merely reproduce the original time series up to  $t = N$ . However, beyond this, since each approximant has a different “noise” background, as given by the polynomial part of the approximant, when we replace the polynomial contribution by white noise beyond this point, we are going to obtain a multifunction that displays a spread of values.

Finally, the physical boundary condition utilizes a physical theory to set the boundary conditions. In such a

situation, we would be able to relate the expected behavior of the time series to physical processes.

## 5 Conclusions

Padé approximants have proven themselves to be useful in computational science [6]. We have discussed the application of Padé approximants to time series related to mesoscopic phenomena. Mesoscopic systems include biosystems, fusion reactors, economic systems, and a great variety of other systems of current interest.

Our approach of mean field analysis of time series from mesoscopic systems by Padé approximants is completely new. This approach requires a focus on boundary conditions, and we have discussed, in our paper, six different types of boundary conditions. The analytic structure of Padé approximants and their ease of computation make them good candidates for numerical modeling of mesoscopic systems.

A main point here is that the Padé approximant splits into a background polynomial that is used to model noise, and a rational expression, characterizing the evolving pattern as a mean field approach to modeling mesoscopic systems. The analysis reduces to a consideration of boundary conditions, of which there are several of interest. This requires a much fuller treatment than we are able to present in this short paper.

Padé approximants have an analytic structure of poles. There are more general approaches [8] to approximation of a similar nature that allow more complex analytic structures, such as branch cuts. These more general approaches are also worth consideration, and will be the subject of future work.

Modern theories of mesoscopic systems (see Ref. 9), as a topic of non-equilibrium thermodynamics, need adequate discussions of both games and quantum mechanics. In this short paper, we have been unable to extend the discussion to either of these important topics. This must await future work.

## 6 References

- [1] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel, “Time Series Analysis : Forecasting and Control”, fourth ed., Wiley, 2008.
- [2] Wikipedia, “Autoregressive-moving-average model”, [http://en.wikipedia.org/wiki/Autoregressive\\_moving\\_average\\_model](http://en.wikipedia.org/wiki/Autoregressive_moving_average_model).
- [3] K. Kumar, “Padé Approximation and its Application in Time Series Analysis”, Applied Math. and Computation, 48 :139-151, 1992.
- [4] M. George, “Series Analysis in Statistical Mechanics”, Ph.D. Thesis, Univ. of Washington, 1985.
- [5] M.B. Priestley, “Spectral Analysis and Time Series”, Academic, 1981.

- [6] G.A. Baker, Jr. And P. Graves-Morris, "Padé Approximants", second ed., Cambridge, 1996.
- [7] N.E. Huang and S.S.P. Shen, eds., "Hilbert-Huang Transform and Its Applications", World Scientific, 2005.
- [8] B.G. Nickel and J.J. Rehr, "High Temperature Series for Scalar Field Lattice Models", J. Stat. Phys., **61**, 1-50, 1990.
- [9] M. George, "Computation as a Mesoscopic Phenomenon", Int'l Conf. Scientific Computing, 43-47, 2011.
- [10] M. George, "Classical and Quantum Ising Games", Int. J. of Pure and Applied Math., **42**, 529-534, 2008.
- [11] M. George, "A Mean Field Theory for Mesoscopic Phenomena", in preparation.
- [12] D.J. Amit, "Field Theory, Renormalization Group and Critical Phenomena", McGraw-Hill, 1978.
- [13] K. Huang, "Statistical Mechanics", second ed., Wiley, 1987.

# A New Discrete Collocation Method For Nonlinear Fredholm Integral Equations

K. Maleknejad<sup>1</sup>, K. Nedaiasl<sup>1</sup>, and L. Torzkadeh<sup>1</sup>

<sup>1</sup>Department of Mathematics, Iran University of Science & Technology, Tehran, Iran

**Abstract**—In this paper, the numerical solution of nonlinear Fredholm integral equations of second kind is considered by Sinc method. This numerical method combines a discrete Sinc collocation method with the Newton iterative process that involves solving a nonlinear system of equations. We provide an error analysis for the method. So far approximate solutions with polynomial convergence have been reported for this equation. This method improve conventional results and achieve exponential convergence. Some numerical examples are given to confirm the accuracy and the ease of implementation of the method.

**Keywords:** Nonlinear Fredholm Integral Equation; Sinc Approximation; Collocation Method.

## 1. Introduction

In this paper, high order numerical method has been developed to approximate the solution of the nonlinear Fredholm integral equations of the form

$$u(t) = g(t) + \int_a^b k(t, s, u(s))ds, \quad a \leq s \leq b \quad (1)$$

where  $k(t, s, u)$ ,  $g(t)$  are known functions and  $u(t)$  is an unknown function. Eq.(1) was introduced for the first time by Pavel Urysohn in [1]. Equations of this type appear in many applications. For example, they arise as a reformulation of two-point boundary value problems with a certain nonlinear boundary condition [5], [4]. Several authors have considered the numerical solving of this equation with different methods [4-11].

Atkinson has investigated the use of piecewise polynomials of order  $n$  as an approximate subspace and obtained the convergence of polynomial order [6]. The aim of this work is to present a numerical scheme by discrete collocation method based on Sinc functions. The method is given by extending Stenger's idea to nonlinear Fredholm integral equation. It is shown that this method confirms the convergence rate  $O(\exp(-C\sqrt{N}))$ . For a comprehensive study of Sinc methods, we refer to [12], [13], and [14],

Eq.(1) can be rewritten in the operator form

$$u = \mathcal{K}u + g, \quad (2)$$

where  $(\mathcal{K}u)(t) = \int_a^b k(t, s, u(s))ds$ . The operator is defined on the Banach space  $X = \mathbf{Hol}(D) \cap C(\overline{D})$ . In this notation,  $D$  is a simply connected domain which satisfies

$(a, b) \subset D$  and  $\mathbf{Hol}(D)$  denotes the family of all functions  $f$  that are analytic in domain  $D$ . Furthermore, Eq.(1) has at least one solution, if the right hand side of Eq.(1) be completely continuous operator [16]. So it is assumed that the kernel  $k(t, s, u)$  and the forcing function  $g(t)$  are sufficiently smooth [11] such that the right hand side of Eq.(2) be completely continuous. Additionally, suppose that the solution  $u^*(t)$  to be determined is geometrically isolated [3], in the other words, there is some ball

$$\mathbf{B}(u^*, r) = \{u \in X : \|u - u^*\| \leq r\},$$

with  $r > 0$ , that contains no solution of Eq.(1) other than  $u^*$ . It is assumed that the linear operator  $\mathcal{K}'(u^*)$  does not have 1 as an eigenvalue, then there is a geometrically isolated solution for Eq.(1) [4]. Let  $\|u\| = \sup\{|u(t)| : t \in [0, 1]\}$  and  $\|\mathcal{K}\| = \sup\{\|\mathcal{K}u\| : u \in \mathbf{B}\}$  where  $\mathbf{B} = \{u \in X : \|u\| \leq 1\}$ .

The layout of this paper is as follows. In section 2, the basic definitions, assumptions and preliminaries of the Sinc method are stated. The smoothness properties of the solution are discussed in section 3. The discrete Sinc collocation scheme is considered in section 4. In section 5, the order of convergence of the schemes using the new approaches is described. Finally, section 6 contains two numerical experiments.

## 2. Basic Definition

The Sinc function is defined on the whole real line by

$$\text{Sinc}(t) = \begin{cases} \frac{\sin(\pi t)}{\pi t}, & t \neq 0, \\ 1, & t = 0. \end{cases}$$

Originally, Sinc approximation for a function  $f$  is expressed as

$$f(t) \approx \sum_{j=-N}^N f(jh)S_j(t), \quad t \in \mathbb{R}, \quad (3)$$

where the basis function  $S_j(t)$  is defined by

$$S_j(t) = \text{Sinc}\left(\frac{t}{h} - j\right), \quad (4)$$

and  $h$  is a step size appropriately chosen depending on a given positive integer  $N$ , and  $j$  is an integer and (4) is called  $j$ th Sinc function. The approximation (3) is valid on  $\mathbb{R}$ , whereas the Eq.(1) is defined on finite interval  $[a, b]$ . The Eq.(3) can be adapted to approximate on general intervals

with the aid of appropriate variable transformations  $t = \varphi(x)$ . This transformation and its inverse can be introduced respectively as below

$$\varphi(x) = \frac{b-a}{2} \tanh\left(\frac{x}{2}\right) + \frac{b+a}{2},$$

$$\phi(t) = \log\left(\frac{t-a}{b-t}\right).$$

In order to define a convenient function space, the strip domain  $D_d = \{z \in \mathcal{C} : |\text{Im}z| < d\}$  for some  $d > 0$  is introduced. When incorporated with the transformation, the conditions should be considered on the translated domain

$$\varphi(D_d) = \{z \in \mathcal{C} : |\arg\left(\frac{z-a}{b-z}\right)| < d\}.$$

The following definitions and theorems are considered for further details of the procedure. Let  $D$  be a simply connected domain which satisfies  $(a, b) \subset D$  and  $\alpha$  and  $C$  be positive constant. Then,  $\mathcal{L}_\alpha(D)$  denotes the family of all functions  $f \in \mathbf{Hol}(D)$  which satisfy

$$|f(z)| \leq C|Q(z)|^\alpha, \quad (5)$$

for all  $z$  in  $D$  where  $Q(z) = (z-a)(b-z)$ . The next theorem shows the exponential convergence of the Sinc approximation. ([13]) Let  $f \in \mathcal{L}_\alpha(\varphi(D))$  for  $d$  with  $0 < d < \pi$ . Suppose that  $N$  be a positive integer, and  $h$  be given by the formula  $h = \sqrt{\frac{\pi d}{\alpha N}}$ . Then there exists a constant  $C$  independent of  $N$ , such that

$$\|f(t) - \sum_{j=-N}^N f(\varphi(jh))S_j(\phi(t))\| \leq C\sqrt{N} \exp(-\sqrt{\pi d \alpha N}).$$

Sinc approximation can be applied to definite integration based on the function approximation described above. ([13]) Let  $(fQ) \in \mathcal{L}_\alpha(\varphi(D_d))$  for  $d$  with  $0 < d < \pi$ . Suppose that  $N$  be a positive integer and  $h$  is selected by the formula

$$h = \sqrt{\frac{\pi d}{\alpha N}}.$$

Then there exists a constant  $C$  which is independent of  $N$ , such that

$$\left| \int_a^b f(s) ds - h \sum_{j=-N}^N f(\varphi(jh))\varphi'(jh) \right| \leq C \exp(-\sqrt{\pi d \alpha N}). \quad (6)$$

### 3. Properties of the Solution

In this part, the analytical solution of Eq.(1) is briefly discussed. In the case of complex Banach spaces, the operator  $\mathcal{K}$  is analytic in  $\Omega$ , if it is Frechét differentiable at each point of  $\Omega$ . Having analytic integral operator gives us analytical solution to Eq.(1) [14]. Reference [16] includes conditions in which the nonlinear operators are Frechét differentiable. But in the case of real Banach space, determination of analytical

solution to Eq.(1) is generally difficult. Atkinson [6] has introduced a special class of nonlinear integral equation. This class has been denoted by  $g_1(\eta, \mu)$ . In this notation,  $\eta$  and  $\mu$  are related to the continuity order of partial derivatives of the kernel of integral equation with respect to the third variable.

([6]) Let  $k$  the kernel of integral equation be of class  $g_1(\eta, \mu)$  and consider the nonlinear integral equation (1). If  $u^*(t)$  is a solution of Eq.(1), then  $u^*(t) \in C^\eta[a, b]$ .

It is not difficult to see that

$$u^{(n)}(t) = \int_a^b \frac{\partial^n K_1(t, s, u(s))}{\partial t^n} ds,$$

where  $K_1(t, s, u) = g(t) + k(t, s, u)$ . So a sequence of functions  $\{u^{(n)}(t)\}$  is obtained. If the sequence  $\left\{\frac{\partial^n K_1(t, s, u)}{\partial t^n}\right\}$  be uniformly bounded and  $\eta$  can be taken infinity, then we have an analytic solution for Eq.(1).

### 4. Sinc-collocation method

A Sinc approximation  $u_N$  to the solution  $u \in \mathcal{M}_\alpha(\varphi(D_d))$  of Eq.(1) is described in this part. Let us define the operator  $\mathcal{P}_N : \mathcal{M}_\alpha \rightarrow X$  as follows

$$\mathcal{P}_N[u](t) = \mathfrak{L}u(t) + \sum_{j=-N}^N [u(t_j) - (\mathfrak{L}u)(t_j)]S_j(\phi(t)),$$

where

$$\mathfrak{L}[u](t) = \frac{b-t}{b-a}u(a) + \frac{t-a}{b-a}u(b),$$

and the points  $t_j$  are defined by the formula

$$t_j = \begin{cases} a, & j = -N - 1, \\ \varphi(jh), & j = -N, \dots, N, \\ b, & j = N + 1. \end{cases}$$

It should be noticed that  $\mathcal{P}_N u$  is an interpolation of  $u$  by Sinc functions with the above points and  $\mathcal{P}_N$  is called collocation operator. The approximate solution  $u_N$  is considered that has the form

$$u_N(t) = c_{-N-1} \frac{b-t}{b-a} + \sum_{j=-N}^N c_j S_j(\phi(t)) + c_{N+1} \frac{t-a}{b-a}. \quad (7)$$

Applying the operator  $\mathcal{P}_N$  to both sides of Eq.(1) gives us the following approximate equation in operator form

$$z_N = \mathcal{P}_N \mathcal{K} z_N + \mathcal{P}_N g, \quad (8)$$

so collocation method for solving Eq.(1) amounts to solve (8) for  $N$  sufficiently large. We are interested in approximating the integral operator in (8) by the quadrature formula presented in (6). So the following discrete operator can be defined

$$\mathcal{K}_N(u)(t) = h \sum_{j=-N}^N k(t, t_j, u(t_j))\varphi'(jh). \quad (9)$$

This numerical procedure leads us to replace (8) with

$$u_N = \mathcal{P}_N \mathcal{K}_N u_N + \mathcal{P}_N g. \quad (10)$$

By substituting  $u_N$  into Eq.(1) and approximating the integral by means of Sinc quadrature formula and considering its collocation on  $2N + 3$  sampling points at  $t = t_i$ , for  $i = -N - 1, -N, \dots, N, N + 1$ , the following nonlinear system of equations

$$u_N(t_i) = h \sum_{j=-N}^N k(t_i, t_j, u_N(t_j)) \varphi'(jh) + g(t_i), \quad (11)$$

is obtained. This nonlinear system of equations is equivalent to (10). By solving this system, the unknown coefficients in  $u_N$  are determined.

The nonlinear system of equations in (10) can be rewritten as

$$\mathcal{F}_N(u_N) = 0, \quad (12)$$

where  $\mathcal{F}_N : \mathbb{R}^{2N+3} \rightarrow \mathbb{R}^{2N+3}$  with  $\mathcal{F}_N(u_N) = u_N - \mathcal{P}_N \mathcal{K}_N u_N - \mathcal{P}_N g$ . All practical approaches to solve such a nonlinear system are iterative, there are much interests for finding a more efficient method for solving such nonlinear systems, e.g. see [17], [18], and references in [23]. In this paper Newton's iterative procedure is applied. The Newton's method reads as follows: Choose an initial guess  $u_{N,(0)}$ ; for  $m = 0, 1, \dots$ , compute

$$u_{N,(m+1)} = u_{N,(m)} - [(\mathcal{F}_N)'(u_{N,(m)})]^{-1} \mathcal{F}_N(u_{N,(m)}). \quad (13)$$

## 5. Convergence Analysis

Sinc-collocation method is discussed in the present section. It is assumed that  $u_N$  is the exact solution of Eq.(10) and  $u_{N,(m)}$  is an approximation of  $u_N$  obtained from Newton's iterative process.

Firstly, we state the following lemma which is used subsequently. ([13]) Let  $h > 0$ . Then it holds that

$$\sup_{x \in \mathbb{R}} \sum_{j=-N-1}^{N+1} |S_j(x)| \leq \frac{2}{\pi} (3 + \log(N+1)). \quad (14)$$

Based on this lemma, it has been concluded  $\|\mathcal{P}_N\| \leq C \log(N+1)$  where  $C$  is constant independent of  $N$ .

Assume that there exists a constant  $d$  with  $0 < d < \pi$  such that  $k(t, \cdot, \cdot) \in \mathbf{Hol}(\varphi(D_d))$  for all  $t$  in  $[a, b]$ . Furthermore, suppose that there exists a constant  $C_1$  for all  $t$  in  $[a, b]$  such that  $\|k(t, \cdot, \cdot)\| \leq C_1$ . Then there exists a constant  $C$  which is independent of  $a, b$  and  $N$  such that

$$\|\mathcal{K}u - \mathcal{K}_N u\| \leq C \exp(-\sqrt{\pi d \alpha N}).$$

In the following theorem, we will find an upper bound for the error.

Suppose that  $u^*(t)$  is an exact solution of Eq.(1) and  $[I - \mathcal{K}'(u^*)]$  is nonsingular and  $\frac{\partial^2 k}{\partial u^2}(t, s, u)$  exists and continuous

on its domain. Furthermore, let the assumptions of Lemma 2 be fulfilled and  $g \in \mathcal{M}_\alpha(\varphi(D_d))$  and  $\mathcal{K}u \in \mathcal{M}_\alpha(\varphi(D_d))$  for all  $u \in \mathbf{B}(u^*, r)$  with  $r > 0$ . Then there exists a constant  $C$  independent of  $N$  such that

$$\|u^* - u_N\| \leq C \lambda_N \sqrt{N} \log(N+1) \exp(-\sqrt{\pi d \alpha N}), \quad (15)$$

where  $\lambda_N = \|(I - \mathcal{P}_N(\mathcal{K}_N)'(u^*))^{-1}\|$ . *Proof:* The estimation (15) is obtained as follows:

$$u^* - u_N = g - \mathcal{P}_N g + \mathcal{K}u^* - \mathcal{P}_N \mathcal{K}_N u_N.$$

We call the right side of the above term  $RS$  which can be rewritten as

$$\begin{aligned} RS &= (g - \mathcal{P}_N g) + (\mathcal{K}u^* - \mathcal{P}_N \mathcal{K}u^*) \\ &\quad + \mathcal{P}_N(\mathcal{K}u^* - \mathcal{K}_N u^*) + \mathcal{P}_N(\mathcal{K}_N u^* - \mathcal{K}_N u_N) \end{aligned}$$

So the following relation is achieved:

$$\begin{aligned} u^* - u_N &= (I - \mathcal{P}_N(\mathcal{K}_N)'(u^*))^{-1} \{ (g - \mathcal{P}_N g) \\ &\quad + (\mathcal{K}u^* - \mathcal{P}_N \mathcal{K}u^*) + \mathcal{P}_N(\mathcal{K}u^* - \mathcal{K}_N u^*) \\ &\quad + \mathcal{P}_N(\mathcal{K}_N u^* - \mathcal{K}_N u_N - (\mathcal{K}_N)'(u^*)(u^* - u_N)) \}. \end{aligned} \quad (16)$$

By applying  $\|\cdot\|$  on both side of (22), we obtain the relation

$$\begin{aligned} \|u^* - u_N\| &\leq \|(I - \mathcal{P}_N(\mathcal{K}_N)'(u^*))^{-1}\| \{ \|g - \mathcal{P}_N g\| \\ &\quad + \|\mathcal{K}u^* - \mathcal{P}_N \mathcal{K}u^*\| + \|\mathcal{P}_N\| \|\mathcal{K}u^* - \mathcal{K}_N u^*\| \\ &\quad + \|\mathcal{P}_N\| O(\|u^* - u_N\|^2) \}. \end{aligned} \quad (17)$$

Because of  $g, \mathcal{K}u^* \in \mathcal{M}_\alpha(\varphi(D_d))$ , we can apply Theorem 1 and get

$$\begin{aligned} \|g - \mathcal{P}_N g\| &\leq C_1 \sqrt{N} \exp(-\sqrt{\pi d \alpha N}), \\ \|\mathcal{K}u^* - \mathcal{P}_N \mathcal{K}u^*\| &\leq C_2 \sqrt{N} \exp(-\sqrt{\pi d \alpha N}). \end{aligned}$$

By using Lemma 2, the following result is concluded

$$\|\mathcal{K}u^* - \mathcal{K}_N u^*\| \leq C_3 \exp(-\sqrt{\pi d \alpha N}),$$

and finally  $\|\mathcal{P}_N\|$  is estimated by conclusion of Lemma 1. So

$$\|u^* - u_N\| \leq C \lambda_N \log(N+1) \sqrt{N} \exp(-\sqrt{\pi d \alpha N}).$$

In the following we are trying to discuss the conditions in which the Newton's method is convergence. For this reason we will state and prove Lemma 3 and Theorem 5. It is well known that

$$F'(u_N)(v)(t) = v(t) - \mathcal{P}_N(\mathcal{K}_N)'(u_N)v(t),$$

where

$$(\mathcal{K}_N)'(u)(v)(t) = h \sum_{j=-N}^N \frac{\partial k}{\partial u}(t, t_j, u(t_j)) \varphi'(jh) v(t_j).$$

So,  $[F'(u_N)]^{-1}$  exists if and only if

$$[I - \mathcal{P}_N(\mathcal{K}_N)'(u_N)],$$

is invertible where  $I$  is the identity operator. In the following lemma, we are trying to find conditions in which  $[I - \mathcal{P}_N(\mathcal{K}_N)'(u_N)]$  is invertible. Suppose that  $k_u(t, s, u^*)$  satisfies

$$|k_u(t, s, u^*) - k_u(\tau, s, u^*)| \leq C_4|t - \tau|^\beta, \quad (18)$$

where  $k_u(t, s, u) \equiv \frac{\partial k(t, s, u)}{\partial u}$  and  $C$  is a constant. Furthermore, assume that there exists a  $r > 0$  such that

$$|k_u(t, s, u) - k_u(t, s, v)| \leq C_5|u - v|, \quad (19)$$

for all  $u, v \in \mathbf{B}(u^*, r)$  and  $k_u(t, s, u^*)$  satisfies the hypotheses of Lemma 2. Then

$$\|\mathcal{P}_N(\mathcal{K}_N)'(u_N) - \mathcal{K}'(u^*)\| \rightarrow 0,$$

as  $N \rightarrow \infty$ . *Proof:* By triangular inequality the following relation is obtained

$$\begin{aligned} \|\mathcal{P}_N(\mathcal{K}_N)'(u_N) - \mathcal{K}'(u^*)\| &\leq \|\mathcal{P}_N \mathcal{K}'(u^*) - \mathcal{K}'(u^*)\| \\ &\quad + \|\mathcal{P}_N(\mathcal{K}_N)'(u_N) - \mathcal{P}_N(\mathcal{K}_N)'(u^*)\| \\ &\quad + \|\mathcal{P}_N(\mathcal{K}_N)'(u^*) - \mathcal{P}_N \mathcal{K}'(u^*)\|. \end{aligned} \quad (20)$$

According to [19], under the assumption (18),  $\|\mathcal{P}_N \mathcal{K}'(u^*) - \mathcal{K}'(u^*)\|$  approaches to zero. The second term in right side of (20) is easily evaluated by condition (19) as follows

$$\|\mathcal{P}_N(\mathcal{K}_N)'(u_N) - \mathcal{P}_N(\mathcal{K}_N)'(u^*)\| \leq \|\mathcal{P}_N\| \|u_N - u^*\|.$$

So, for sufficiently large  $N$ , it is concluded that

$$\|\mathcal{P}_N(\mathcal{K}_N)'(u_N) - \mathcal{P}_N(\mathcal{K}_N)'(u^*)\| \rightarrow 0.$$

The third term approaches to zero by applying Lemma 2 to  $k_u(t, s, u^*)$  and Lemma 1. ■

Now, suppose that 1 is not an eigenvalue of  $\mathcal{K}'(u^*)$  then under the assumptions of Lemma 3, we can conclude from Lemma 2.2 in [20] that for sufficiently large  $N$ ,  $[I - \mathcal{P}_N(\mathcal{K}_N)'(u_N)]$  is invertible. Next theorem deals with the local convergence of Newton's iterative method applied to Eq.(12). Assume  $u_N$  is the exact solution of the Eq.(12) and  $[I - \mathcal{K}'(u^*)]$  is invertible. Furthermore, let the assumptions of Lemma 3 be fulfilled. Then there exists a  $\varepsilon > 0$  such that if  $\|u_{N,(0)} - u_N\| \leq \varepsilon$ , the Newton's sequence  $\{u_{N,(m)}\}$  is well-defined and convergence to  $u_N$ . Furthermore, for some constant  $l$  with  $l\varepsilon < 1$ , we have the error bounds

$$\|u_{N,(m)} - u_N\| \leq \frac{(l\varepsilon)^{2^m}}{l} = C(m). \quad (21)$$

*Proof:* The conclusion is straightforwardly achievable by applying Theorem 5.4.1 in [22] and above discussion. ■ In the following final theorem, we summarize the conclusions of theorems and lemmas proved in this section. Assume that  $u^*$  is an isolated solution of Eq.(1), Furthermore,  $u_N$  and  $u_{N,(m)}$  are the solution of Eq.(10) and Eq.(13),

respectively. Suppose that hypotheses of Theorem 4 and Theorem 5 are satisfied. Then there exists a positive constant  $C(m)$  independent of  $N$  and depend on  $m$  such that

$$\|u^* - u_{N,(m)}\| \leq C(m)\lambda_N\sqrt{N}\log(N+1)\exp(-\sqrt{\pi d\alpha N}). \quad (22)$$

*Proof:* The conclusion is obtained by using triangular inequality and conclusions of Theorem 4 and Theorem 5. ■

## 6. Numerical Experiments

In this section, the theoretical results of the previous sections are used for some numerical examples. In order to analyze the error of the method the following notations are introduced:

$$e_{max} = \max\{|u(t_i) - u_N(t_i)| : t_i = \frac{i}{1000}, i = 1(1)1000\},$$

and

$$\rho_N = \log_2\left(\frac{e_{max}^i}{e_{max}^{(i+1)}}\right),$$

which  $e_{max}$  approximate  $\|u - u_N\|$  and  $\rho_N$  estimate the convergence rate. In second formula  $e_{max}^i$  denotes the  $e_{max}$  in  $(i+1)$ th column of tables. In these examples, the Newton's method is iterated until the accuracy  $10^{-6}$  is obtained. In tables,  $m$  denotes the number of iterations. Initial point for Newton's iteration is selected by steepest descend method [21].

It is assumed that  $\alpha = 1$ . The  $d$  values is  $\frac{\pi}{2}$  for the Sinc method. The absolute value of the errors of the two methods for  $N = 5, 10, 15, 20$  and  $25$  is reported. These tables show that increasing  $N$  the error significantly is reduced.

*Example 1.* we interest in approximating the solution of the following nonlinear Fredholm integral equation

$$u(t) = \frac{1}{5} \int_0^1 \cos(\pi t) \sin(\pi s) [u(s)]^3 ds + \sin(\pi t), \quad (23)$$

with the exact solution

$$u(t) = \sin(\pi t) + \frac{1}{3}(20 - \sqrt{391}) \cos(\pi t),$$

where  $0 \leq t \leq 1$ . The numerical solution of this equation is considered in [26] via Newton - Kantorovich - quadrature method. Table 1 shows the absolute error of SE-Sinc method for  $N = 25$  and the results of Newton - Kantorovich - quadrature method. As this table illustrates the Sinc method is more efficient than their method.

Table 1: Comparison of the results in [26] to Sinc Method.



t	Results in [26]	Sinc method
0	4.98E - 2	4.15E - 7
0.1	4.73E - 2	1.33E - 5
0.2	4.03E - 2	6.85E - 6
0.3	2.92E - 2	6.22E - 6
0.4	1.54E - 2	2.19E - 6
0.5	0.00E - 0	2.15E - 7
0.6	1.54E - 2	5.85E - 6
0.7	2.92E - 2	1.02E - 5
0.8	4.03E - 2	8.25E - 6
0.9	4.73E - 2	1.66E - 5
1	4.98E - 2	4.14E - 7

Table 2: Absolute errors of the Sinc method for Example 2.

t	N = 5	N = 15	N = 25
0.1	2.92E - 3	2.99E - 6	6.24E - 7
0.3	7.27E - 4	1.43E - 5	1.49E - 7
0.5	1.38E - 3	1.25E - 5	9.58E - 7
0.7	4.55E - 4	1.22E - 5	1.31E - 6
0.9	2.38E - 3	1.17E - 6	1.06E - 6
m	3	3	3
$e_{max}$	2.94E - 3	2.74E - 5	2.01E - 6
$\rho_N$	-	2.99	2.15
Results in [24]	3.3E - 3	1.1E - 3	1.1E - 4

Example 2. We consider the nonlinear integral equation (see [24], Example 3)

$$u(t) = t \int_0^1 s \sqrt{u(s)} ds + 2 - \frac{1}{3}(2\sqrt{2} - 1)t - t^2,$$

with the exact solution  $u(t) = 2 - t^2$ . The Table 2 illustrates the numerical results obtained here and the numerical results of [24] for this example.

## 7. Conclusions

Finding exact solutions for nonlinear Fredholm integral equations are often not available. So, approximating these solutions are very important. Many authors have proposed different methods. In this research, a numerical method based on Sinc function has been suggested. It has been shown theoretically and numerically that the scheme is extremely accurate and achieve exponential convergence with respect to  $N$ .

## References

- [1] P. S. Urysohn, "On a type of nonlinear integral equations," *Mat. Sb.*, vol. 31, pp. 236–255, 1924.
- [2] M. A. Krasnoselskii, P. P. Zabreiko, E. I. Pustyl'nik and P. E. Sbolevskii, *Integral operators in spaces of summable functions*, Noordhoff international publishing, Leyden, 1976.
- [3] H. B. Keller, "Geometrically isolated nonisolated solutions and their approximation," *SIAM J. Numer. Ana.*, vol. 18, pp. 822–838, 1981.
- [4] S. Kumar, I. H. Sloan, "A New Collocation-Type Method for Hammerstein Integral Equations," *J. Mathematics of Computation*, vol. 48, pp. 585–593, 1987.
- [5] K. E. Atkinson, "A survey of numerical methods for solving nonlinear integral equation," *J. Int. Equ. Appl.*, vol. 4 (1992) pp. 15–47.
- [6] K. E. Atkinson, F. A. Potra, "Projection and iterated projection methods for nonlinear integral equation," *SIAM J. Numer. Ana.*, vol. 24, pp. 1352–1373, 1989.
- [7] K. E. Atkinson, "Numerical evaluation of fixed points for completely continuous operator," *SIAM J. Numer. Ana.*, vol. 10, pp. 799–807, 1973.
- [8] K. E. Atkinson, J. Flores, "Discrete collocation methods for nonlinear integral equations," *IMA J. Numer. Ana.*, vol. 13, pp. 195–213, 1993.
- [9] K. E. Atkinson, F. Potra, "Discrete Galerkin methods for nonlinear integral equations," *J. Int. Equ.*, vol. 13, pp. 17–54, 1998.
- [10] H. Kaneko, Y. Xu, "Super convergence of the iterated Galerkin methods for Hammerstein integral equations," *SIAM, J. Numer. Ana.*, vol. 33, pp. 1048–1064, 1996.
- [11] Y. C. Song, "On the discrete Galerkin methods for nonlinear integral equation," *J. Korean Math. Soc.*, vol. 29, pp. 297–315, 1992.
- [12] J. Lund, K. Bowers, *Sinc methods for quadrature and differential equations*, SIAM, Philadelphia, 1992.
- [13] F. Stenger, *Numerical methods based on Sinc and analytic functions*, Springer, New York, 1993.
- [14] F. Stenger, "Summary of Sinc numerical methods," *J. Comp. Appl. Math.*, vol. 121, pp. 379–420, 2000.
- [15] F. Stenger, "Matrices of Sinc methods," *J. Comp. Appl. Math.*, vol. 86, pp. 297–310, 1997.
- [16] M. A. Krasnoselskii, P. P. Zabreiko, *Geometrical methods of nonlinear analysis*, Springer-Verlag, 1984.
- [17] J. J. Moré, M. Y. Cosnard, "Numerical solution of nonlinear equations," *ACM Trans. Math. Software* vol. 5, pp. 64–85, 2009.
- [18] S. Schlenkrich, A. Walthner, "Global convergence of quasi-Newton methods based on adjoint Broyden updates," *Appl. Numer. Math.*, vol. 59, pp. 1120–1136, 2009.
- [19] K. E. Atkinson, *The numerical solution of integral equations of second kind*, Cambridge University of Press, New York, 1997, pp. 62–64.
- [20] P. M. Anselone, *Collectively compact operator approximation theory*, Printice-Hall, Inc., Englewood Cliffs, N. J., 1971.
- [21] R. L. Burden, J. Douglas Faires, *Numerical analysis*, Brooks/Cole Publishing Company, 1997.
- [22] K. E. Atkinson, W. Han, *Theoretical numerical analysis: a functional analysis framework*, Springer, New York, 2001, p. 249.
- [23] G. A. Zakeri, M. Navab, "Sinc collocation approximation of non-smooth solution of a nonlinear weakly singular Volterra integral equation," *J. Comput. Phys.*, vol. 229 pp. 6548–6557, 2010.
- [24] Ü. Lepik, E. Tamme, "Solution of nonlinear Fredholm integral equations via the Haar wavelet method," *Proc. Estonian Acad. Sci. Phys. Math.* vol. 56, pp. 11–27, 2007.
- [25] A. H. Borzabadi, O. S. Fard, "A numerical scheme for a class of nonlinear Fredholm integral equations of the second kind," *J. Comp. Appl. Math.* vol. 232, pp. 449–454, 2009.
- [26] J. Saberi-Nadjafi, M. Heidari, "Solving nonlinear integral equations in the Urysohn form by Newton-Kantorovich-quadrature method," *Computers and Mathematics with Applications*, vol. 60, pp. 2058–2065, 2010.

# Multiobjective Optimization of Nonlinear Circuits

J. Dobeš<sup>1</sup>, J. Míchal<sup>2</sup>, and V. Paňko<sup>1,3</sup>

<sup>1</sup>Department of Radio Engineering, Czech Technical University in Prague, Praha, Czech Republic

<sup>2</sup>Division of IC Design, CertiCon Corporation, Praha, Czech Republic

<sup>3</sup>SCG Czech Design Center, ON Semiconductor, Rožnov pod Radhoštěm, Czech Republic

**Abstract**—This paper suggests an enhancement of an existing method for the multiobjective optimization known as GAM (goal attainment method). In our proposal, the GAM algorithm is combined with a mechanism that automatically provides a set of parameters (weights, coordinates of the reference point) for which the method generates noninferior solutions uniformly spread over a suitably selected part of the Pareto front. The resulting set of solutions is then presented in a suitable graphic form so that the solution representing the most satisfactory tradeoff can be easily chosen. The whole algorithm was implemented as a program and tested on various design examples. The most difficult one was a multiobjective optimization of a C-class power amplifier in the time domain. Therefore, this task is thoroughly described in the paper as a demonstration of the program capability.

**Keywords:** Multiobjective optimization, Pareto front, Pareto optimal set, noninferior solutions, goal attainment method, LDMOS.

## 1. Introduction

The process of electronic circuit design usually strongly relies on the use of computers. One class of methods for circuit design not only uses them as a circuit simulation tool, but also uses numerical optimization algorithms as a means of determining parameter values in order to bring the designed circuit as close as possible to some prescribed behavior or a set of characteristics. Multiobjective optimization solves the situations in circuit design where there are two or more possibly contradictory requirements on a circuit and thus a suitable tradeoff needs to be found. Such a tradeoff solution should best belong to a set of noninferior solutions, also called Pareto optimal set or Pareto front. Noninferior solutions are characterized by the property that any further improvement in one objective value can only be achieved at the expense of disimprovement in at least one other objective value.

### 1.1 Multiobjective Optimization Problem

In practical designs, there are often multiple mutually contradicting requirements on the designed circuit. In such cases, our aim is to solve the corresponding *multiobjective optimization problem* (MOP). This can be formally written as

$$\underset{\mathbf{x} \in S}{\text{minimize}} \quad \{ f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}) \}, \quad (1)$$

where we have  $k$  objective functions  $f_i: \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $k \geq 2$ . As in the case of SOP, the decision vectors  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$  belong to the (nonempty) feasible region  $S$ ,  $S \subseteq \mathbb{R}^n$ , which can also be defined by a number of equality constraints, inequality constraints, and/or bounds on the decision variables  $x_i$ . The vector of objective functions is denoted by  $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})]^T$  and the image of the feasible region, also called the *feasible objective region*, is denoted by  $Z = \mathbf{f}(S)$ ,  $Z \subseteq \mathbb{R}^k$ . The elements of  $Z$  are called *objective vectors* and are denoted by  $\mathbf{f}(\mathbf{x})$  or  $\mathbf{z} = [z_1, z_2, \dots, z_k]^T$ , where  $z_i = f_i(\mathbf{x})$  for all  $i = 1, 2, \dots, k$  are *objective values*. The geometrical representation of both sets  $S$  and  $Z$  and of the mapping  $\mathbf{f}(\mathbf{x})$  between them can easily be illustrated on a two-dimensional case, as shown in Fig. 1 for  $n = 2$  and  $k = 2$ .

### 1.2 Pareto Optimality

The word “minimize” in (1) means that we want to minimize all the objective functions simultaneously. However, because of the contradiction between the objective functions, it is not possible to find a single solution that would be optimal for all the objectives simultaneously. The concept of *noninferiority* also called *Pareto optimality* must be used to characterize the objective vectors. A noninferior solution is the one in which an improvement in one objective requires a deterioration of another. The set of all noninferior solutions is also called the *Pareto front*. In Fig. 1 it is marked by the thick curve segment between points  $z_A$  and  $z_B$ .

By solving the problem (1) we understand obtaining a sufficient number of noninferior solutions covering parts of the Pareto front that are of interest to the designer. This will allow him or her to fully understand the available trade-offs and to take a qualified decision based on this knowledge.

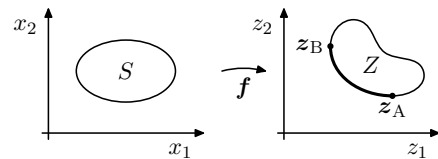


Fig. 1: Feasible region ( $S$ ), feasible objective region ( $Z$ ), and Pareto front.

### 1.3 Ranges of the Pareto Front

For normalizing purposes we may need to know the minimum and maximum values of the individual objectives achieved over the Pareto Front. We will assume that individual objective functions  $z_i = f_i(\mathbf{x})$  are bounded over the feasible region  $S$ .

*Ideal objective vector*  $\mathbf{z}^* = [z_1^*, z_2^*, \dots, z_k^*]$  is the objective vector independently minimizing each objective function:

$$\mathbf{z}^* = \left[ \min_{\mathbf{x} \in S} f_1(\mathbf{x}), \min_{\mathbf{x} \in S} f_2(\mathbf{x}), \dots, \min_{\mathbf{x} \in S} f_k(\mathbf{x}) \right]. \quad (2)$$

It can easily be seen that if the ideal objective vector is feasible ( $\mathbf{z}^* \in Z$ ), it is a solution of the multiobjective optimization problem and the Pareto front is reduced to it. But even in the usual cases when ideal objective vector is not feasible, it can still be considered a useful reference point.

Maximum objective function values achieved over the Pareto front are represented by a *nadir objective vector*. Because such maxima are difficult to find, an approximate nadir vector  $\mathbf{z}^{\text{nad}}$  is instead defined as

$$\mathbf{z}^{\text{nad}} = \left[ \max_i (z_i^*)_1, \dots, \max_i (z_i^*)_k \right], \quad (3)$$

i.e., as the vector of the largest respective components  $(z_i^*)_j$  found in all  $k$  ideal objective vectors. Nadir vector may and may not be feasible.

### 1.4 Goal Attainment Method

Typically used approaches to the multiobjective optimization are either a method based on a weighted sum or optimization of a single objective function while the others serve as constraints (also known as  $\varepsilon$ -constraint method) [1], [2]. The goal attainment method [3] provides a better control over obtained solutions. It is defined as a scalar constrained optimization problem of the form

$$\begin{aligned} & \underset{\gamma \in \mathbb{R}, \mathbf{x} \in S}{\text{minimize}} && \gamma \\ & \text{subject to} && f_i(\mathbf{x}) - w_i \gamma \leq \bar{z}_i, \\ & && i = 1, \dots, k, \end{aligned} \quad (4)$$

where  $f_i$  are the  $k$  objective functions to be minimized (design goals),  $S$  is the set of acceptable solutions (the feasible region),  $\bar{z}_i$  are predefined reference goal values associated with the objective functions  $f_i$ ,  $w_i \in \mathbb{R}$  are predefined weighting coefficients, and  $\gamma$  is an auxiliary variable making the new single objective function. The method requires  $2k$  input parameters, but only uses  $2k-1$  degrees of freedom as shown in Fig. 2. Any solution of this optimization problem is noninferior. Its location on the Pareto front can be controlled by the weighting vector  $\mathbf{w}$  and/or by the reference vector  $\bar{\mathbf{z}}$ .

Note that as all more complicated multiobjective problems are time consuming, it is suitable to run them in batch mode.

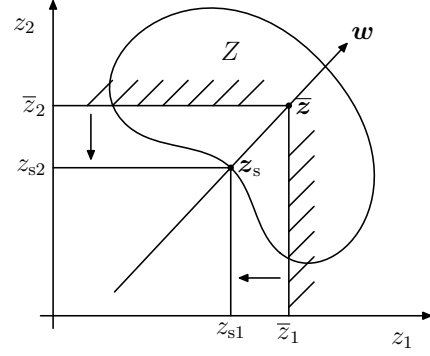


Fig. 2: Geometrical representation of GAM.

## 2. A Semiautomatic A Posteriori Method

The goal is to arrange a reliable general-purpose multiobjective optimization tool that could be used in circuit design.

Requirements on the multiobjective optimization method will include:

- arbitrary number of objectives
- arbitrary number of inequality constraints (but no need for equality constraints)
- provisions for maximizing some functions while minimizing others
- possibility of non-differentiable objectives
- automatic generation of Pareto optimal solutions
- automatic determining of what the covered part of the whole Pareto front will be
- even density of coverage of selected part of the Pareto front
- little or no a priori knowledge of the DM's preferences required
- possibility of monitoring and user interference during computation run
- support for graphical presentation of the solutions

These requirements clearly suggest the use of an a posteriori method or a method of another class converted into an a posteriori method to be able to automatically generate Pareto optimal solutions.

The authors' choice is the Goal Achievement Method, which is also one of the Achievement Scalarizing Function approaches and a variety of Goal Programming [1].

Its advantages as opposed to Weighted Method or Method of Weighted Metrics [1] are that all Pareto optimal solutions are accessible even for non-convex problems and that it works for both feasible as well as infeasible reference points. As a disadvantage could be seen the need for a subroutine for constrained optimization, but we want to be able to work with constraints, anyhow.

There are two possible ways of controlling the location of the Pareto optimal solutions found by GAM: either (a) by the choice of the reference point or (b) by the choice of the weighting vector (or (c) a combination of both). Our

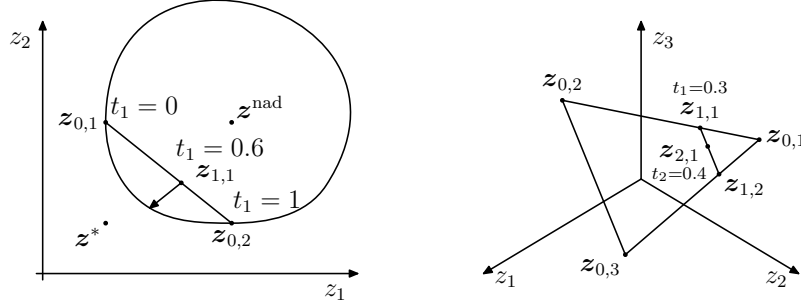


Fig. 3: Reference set  $A$  in the 2D and 3D objective spaces.

approach uses (a) because this seems to have a better chance of even coverage of the Pareto front.

Here is the implemented equivalent form of the GAM, also with the used normalization:

$$\underset{\mathbf{x} \in S}{\text{minimize}} \quad \max_{i=1, \dots, k} \frac{f_i(\mathbf{x}) - \bar{z}_i}{z_i^{\text{nad}} - z_i^*}. \quad (5)$$

Note that this very formula works for both the minimized as well maximized objective functions: for the maximized ones we have  $z_i^{\text{nad}} - z_i^* < 0$  and the denominator thus automatically provides the correct sign. Also, as a result of the choice (a) mentioned above, there are no explicit weighting coefficients  $w_i$  used in (4)<sup>1</sup>.

Now let us consider the choice of a set  $A$  in the  $k$  dimensional objective space from which the reference points are taken. We will call it the *reference set*.

It should not be too far away from the Pareto front (in Euclidian sense) so that it is not too difficult for the user to predict where the corresponding Pareto optimal solution will be from the knowledge of the reference point.

If the feasible objective set is bounded, the Pareto front  $P$  will usually be a subset of the  $k$ -dimensional interval  $B = \prod_{i=1}^k [z_i^*, z_i^{\text{nad}}]$ , where the product operator represents the Cartesian product.

We could put  $A = B$  and simply randomly generate the coordinates from the intervals  $[z_i^*, z_i^{\text{nad}}]$ . However, this approach would lead to many reference points that have no projection on the Pareto front. Such points could still provide Pareto optimal solutions but those would be concentrated along the border of Pareto front and not evenly spread over the interior. Also many points would be quite far from the Pareto front.

Therefore, we try to limit the size of the set  $A$  and select it such that it is likely to be not very far from Pareto front.

One such a choice of the reference set  $A$ , that has actually been implemented in the proposed method, is the

<sup>1</sup>The missing weighting coefficients in (5) that are a result of the choice (a) above also exclude the possibility to introduce hard constraints simply by setting the particular weight to zero, but this really poses no practical limitation in our implementation as any goal can easily be switched to directly play the role of an objective, a constraint, or even both of them simultaneously.

$k$ -dimensional convex body with  $k$  vertices (segment of straight line, triangle, tetrahedron, etc.) whose vertices are composed of one component of the ideal vector  $z_i^*$  and the rest are corresponding components of the nadir vector  $z^{\text{nad}}$ :

$$z_l^{\text{vert}} = [z_1^{\text{nad}}, \dots, z_{l-1}^{\text{nad}}, z_l^*, z_{l+1}^{\text{nad}}, \dots, z_k^{\text{nad}}]^T \quad \forall l = 1, \dots, k. \quad (6)$$

This set  $A$  is randomly sampled with the uniform distribution all over its  $k-1$ -dimensional volume. This is done with the intent to uniformly cover the corresponding part of the Pareto front. Fig. 3 illustrates the location of this reference set in the two- and three-dimensional objective space.

The random generation of reference points belonging to  $A$  can be performed in this way: starting with  $k$  vertices  $z_{0,1}, z_{0,2}, \dots, z_{0,k}$  and a  $(k-1)$ -tuple of uniformly distributed and mutually independent random numbers  $r_i \in [0, 1)$  for  $i = 1, \dots, k-1$ , we perform the following sequence of assignments to calculate a point  $z_{k-1,1} \in A$ :

$$\begin{aligned} t_1 &= \sqrt[k-1]{r_1} \\ z_{1,1} &= (1-t_1)z_{0,1} + t_1z_{0,2} \\ z_{1,2} &= (1-t_1)z_{0,1} + t_1z_{0,3} \\ &\vdots \\ z_{1,k-1} &= (1-t_1)z_{0,1} + t_1z_{0,k} \\ \hline t_2 &= \sqrt[k-2]{r_2} \\ z_{2,1} &= (1-t_2)z_{1,1} + t_2z_{1,2} \\ z_{2,2} &= (1-t_2)z_{1,1} + t_2z_{1,3} \\ &\vdots \\ z_{2,k-2} &= (1-t_2)z_{1,1} + t_2z_{1,k-1} \\ \hline &\vdots \\ \hline t_{k-1} &= \sqrt[r_{k-1}]{} \\ z_{k-1,1} &= (1-t_{k-1})z_{k-2,1} + t_{k-1}z_{k-2,2}. \end{aligned} \quad (7)$$

This procedure therefore calculates a total of  $N = \sum_{i=1}^{k-1} i = k(k-1)/2$  points  $z_i^j$  in the  $k$ -dimensional objective space (i.e.,  $N = 1, 3$  and  $6$  points for  $k = 2, 3$  and  $4$ ) from  $k$  initially known vertices of the reference set. An example of generation of a point ( $z_{3,1}$ ) from a reference set in the shape of a tetrahedron (i.e., when  $k = 4$ ) is shown in Fig. 4.

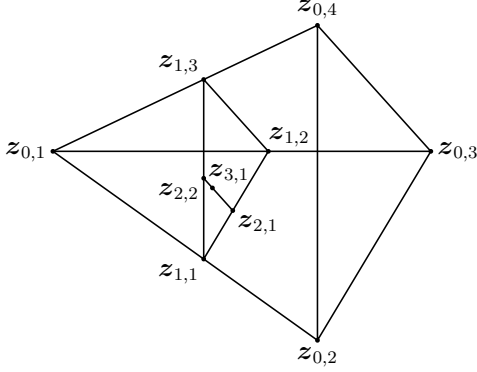
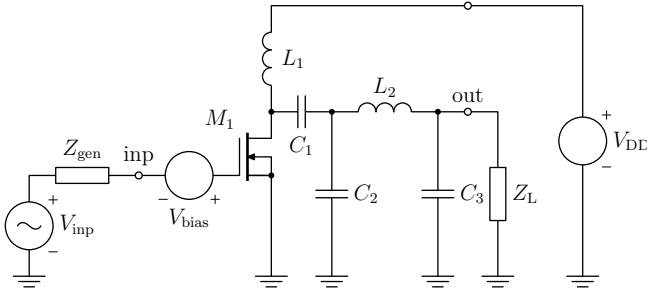
Fig. 4: Obtaining a point from the reference set for  $k = 4$ .

Fig. 5: RF C-class power amplifier schematic.

### 3. RF C-Class Power Amplifier Design

As a sophisticated example, let us try to design the last stage of an RF power amplifier for a narrow-band signal with an analog modulation at the frequency  $f_1 = 300$  MHz. The source and load impedances should be both  $50 \Omega$  and supply voltage  $V_{DD} = 12$  V. Our goal will be to explore the trade-offs between achievable output power, power efficiency and total harmonic distortion.

#### 3.1 Schematic

We use an RF N-channel LDMOS as an active component and a topology that is typical for C-class mode of operation, see Fig. 5. The transistor is followed by an LC filter to suppress harmonic distortion and provide good impedance matching. (Even though impedance matching at the output is not directly required, it is enforced indirectly by maximizing output power.) The combination of elements  $L_1$ ,  $C_1$  and  $C_2$  can also be seen as a tapped resonant circuit. As for the transistor, our choice will be LP821 (Polyfet RF Devices), a silicon LDMOS device for frequencies of up to 500 MHz, with a maximal total dissipated power of 50 W.

As our goal is exploration of the output trade-offs rather than obtaining a complete design, no input impedance matching circuit is considered, and no stability-ensuring measures are taken (other than rather small reactance of the capacitance between gate and source of the transistor itself).

Table 1: Design variables for the power amplifier.

No.	Symbol	Bound		Unit	Coverage Type
		Lower	Upper		
1	$V_{gs \max}$	2	20	V	lin.
2	$V_{gsACm}$	0.4	12	V	lin.
3	$L_1$	3 n	30 n	H	log.
4	$C_1$	10 p	300 p	F	log.
5	$C_2$	3 p	300 p	F	log.
6	$L_2$	3 n	100 n	H	log.
7	$C_3$	3 p	100 p	F	log.

#### 3.2 Design Variables

As design variables we have two kinds of parameters: (A) parameters of the gate voltage that directly determine the operating mode of the transistor, and (B) all LC-component values of the filter.

A simple way to define the design variables of the former group would be the combination of the input AC voltage  $V_{inp}$  and its DC offset  $V_{bias}$ . That, however, would not provide direct control over the voltage between gate and source, which must not exceed 20 V (as given by maximum ratings of the device). This requirement would have to be enforced by means of a special constraint, which would increase simulation time. In order to avoid this need, an estimated peak  $V_{gs}$  voltage, denoted by  $V_{gs \max}$ , was chosen as one design variable and the amplitude of its AC component as another one. The gate voltage estimate is defined using the equation of the voltage divider formed by the driver output resistance  $R_d$  and input capacitive reactance of the gate  $X_i \approx 10 \Omega$

$$V_{gsACm} = V_{inpACm} \frac{X_i}{\sqrt{X_i^2 + R_d^2}}, \quad (8)$$

where  $V_{inpACm}$  is the amplitude of the input AC voltage component from the preceding driver stage (open-circuited). From given values of design variables  $V_{gs \max}$  and  $V_{gsACm}$  we then obtain

$$V_{bias} = V_{gs \max} - V_{gsACm}, \quad V_{inpACm} = V_{gsACm} \frac{\sqrt{X_i^2 + R_d^2}}{X_i}. \quad (9)$$

Table 1 gives a summary of all design variables including their ranges and types of coverage.

#### 3.3 Design Goals

There is a total of five design goals, three of which are the three objective functions to be optimized and two constraints representing maximum ratings of the LDMOS. All goals are defined in terms of waveforms of voltages and currents in the periodic steady state, which was obtained by the steady-state analysis of the simulator CIA [4] (necessary time-domain sensitivity analysis is also described in [4]). Table 2 gives a complete summary of all design goals. The individual design goal definitions are as follows:

Table 2: Design goals for the power amplifier.

No.	Symbol	Type	Direction	Optimum/ Bound	Unit
1	$P_{\text{out1}}$	obj.	max.	31.1	W
2	$\eta$	obj.	max.	83.0	%
3	$THD$	obj.	min.	0.0783	%
4	$I_{\text{d avg}}$	constr.	$\leq$	5	A
5	$P_{\text{diss}}$	constr.	$\leq$	50	W

a) Average output power at the first harmonic frequency  $P_{\text{out1}}$ :

$$P_{\text{out1}} = \frac{|\hat{v}_{\text{out1}}|^2}{2R_L} = \frac{a_1^2 + b_1^2}{2R_L}, \quad (10)$$

where  $\hat{v}_{\text{out1}}$  is the phasor of the output voltage  $v_{\text{out}}(t)$ ,  $a_k$  and  $b_k$  are generally the coefficients of the  $k$ -th cosine and sine harmonic (Fourier) components of the periodic steady-state output voltage  $v_{\text{out}}(t)$  of the period  $T$ , respectively:

$$a_k = \frac{2}{T} \int_T v_{\text{out}}(t) \cos \frac{2\pi k}{T} t dt, \quad b_k = \frac{2}{T} \int_T v_{\text{out}}(t) \sin \frac{2\pi k}{T} t dt.$$

Here the integrals over period  $T$  were computed using the trapezoidal method of numeric integration.

b) Power efficiency  $\eta$ : it is defined as the ratio of output power at the first harmonic frequency and the total average power from power supply and from the input driver. Such a form of definition encourages not only lower power dissipation on the transistor, but also lower input power and thus higher power gain

$$\eta = \frac{P_{\text{out1}}}{\frac{V_{\text{DD}}}{T} \int i_{\text{DD}} dt + \frac{1}{T} \int v_{\text{inp}}(t) i_{\text{inp}}(t) dt} \times 100\%. \quad (11)$$

c) Total harmonic distortion  $THD$ :

$$THD = \sqrt{\frac{P_{\text{out higher}}}{P_{\text{out}}}} \times 100\%, \quad (12)$$

where  $P_{\text{out higher}}$  is the output power at higher harmonics up to  $n_h$

$$P_{\text{out higher}} = \frac{1}{2R_L} \sum_{k=2}^{n_h} a_k^2 + b_k^2, \quad (13)$$

with  $n_h = 10$ , and where  $P_{\text{out}}$  is the total output power computed with the formula

$$P_{\text{out}} = \frac{1}{R_L} \left[ \frac{1}{T} \int_T v_{\text{out}}^2(t) dt - \left( \frac{1}{T} \int_T v_{\text{out}}(t) dt \right)^2 \right]. \quad (14)$$

The second term here cancels the contribution by a possible false DC component that could emerge as a result of the failure to fully achieve the periodic steady state within the chosen maximum number of iterations. (We know that in reality the DC component of the output voltage must be zero due to the capacitive coupling by  $C_1$  and the load being linear.)

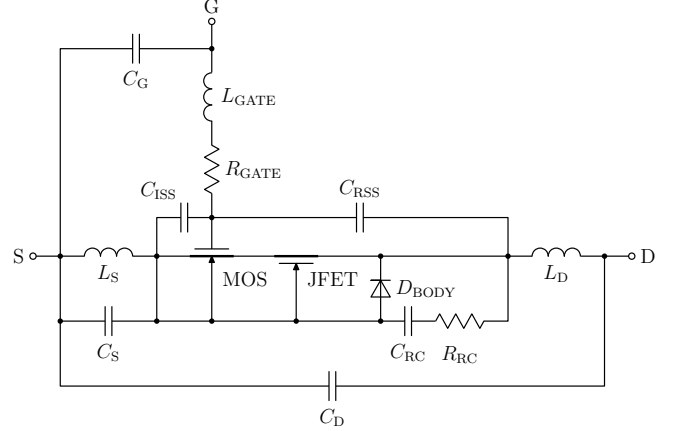


Fig. 6: Model configuration of LDMOS LP821.

d) The maximum ratings of the transistor are applied as constraints: maximum average drain current  $I_{\text{d avg}}$  and the maximum average dissipated power by the transistor  $P_{\text{diss}}$

$$I_{\text{d avg}} = \frac{1}{T} \int_T i_{\text{d}}(t) dt \quad (15)$$

and

$$P_{\text{diss}} = \frac{1}{T} \int_T [v_{\text{gs}}(t) i_{\text{g}}(t) + v_{\text{ds}}(t) i_{\text{d}}(t)] dt, \quad (16)$$

respectively, where  $i_{\text{g}}(t)$ ,  $i_{\text{d}}(t)$  are the instantaneous gate and drain currents, and  $v_{\text{gs}}(t)$ ,  $v_{\text{ds}}(t)$  are the instantaneous voltages between gate and source and drain and source.

### 3.4 Transistor Model

The model used for simulations is based on a SPICE-like one structured as shown in Fig. 6.

The original manufacturer's model is actually only the simple Level 1 SPICE MOSFET one (i.e., Shichman and Hodges). Therefore, its parameters were recalculated for using the semiempirical Level 3 model. They became  $V_{\text{TO}} = 2.4$  V,  $\phi_S = 0.6$  V,  $\phi_O = 0.8$  V,  $W = 0.04$  m,  $L = 1$   $\mu\text{m}$ ,  $X_J = 1$   $\mu\text{m}$ ,  $X_{JL} = 0$   $\mu\text{m}$ ,  $t_{\text{ox}} = 100$  nm,  $N_{\text{FS}} = 0$   $\text{m}^{-2}$ ,  $N_A = 10^{21}$   $\text{m}^{-3}$ ,  $v_{\text{max}} = 5 \times 10^4$  m/s,  $\mu_O = 0.06$   $\text{m}^2/(\text{Vs})$ ,  $\varkappa = 0.22$ ,  $E_P = 5 \times 10^5$  V/m,  $K_P = 1.8 \times 10^{-5}$  A/V<sup>2</sup>,  $\gamma = 0$   $\sqrt{\text{V}}$ ,  $\delta = \eta = \iota = 0$ ,  $\theta = 0$  V<sup>-1</sup>,  $r_D = 0.16$   $\Omega$ , and  $r_S = 0.16$   $\Omega$  – the manufacturer's value of  $W$  (for the composed devices like LDMOS, this number represents the element as a whole) clearly indicates that LP821 is really a power transistor. The manufacturer's values of the JFET and PN diode parameters were  $\lambda = 0.8$  V<sup>-1</sup>,  $\beta = 6$  AV<sup>-2</sup>,  $V_{\text{TO}} = -5.25$  V,  $C_{\text{JOD}} = C_{\text{JOS}} = 0$  F,  $I_S = 10^{-14}$  A,  $n = 1$ ,  $V_B = 45$  V,  $I_B = 10^{-7}$  A,  $C_{\text{JO}} = 60$  pF,  $\phi_O = 0.6$  V, and  $m = 0.25$ .

The RLC component values are the following:  $L_{\text{GATE}} = 0.867$  nH,  $R_{\text{GATE}} = 0.01$   $\Omega$ ,  $C_G = 3.5$  pF,  $C_{\text{RSS}} = 4.5$  pF,  $C_{\text{ISS}} = 22.1$  pF,  $L_S = 0.108$  nH,  $C_S = 0.43$  pF,  $L_D = 0.51$  nH,  $C_D = 0.01$  pF,  $R_{\text{RC}} = 1989$   $\Omega$ , and  $C_{\text{RC}} = 0.381$  nF.

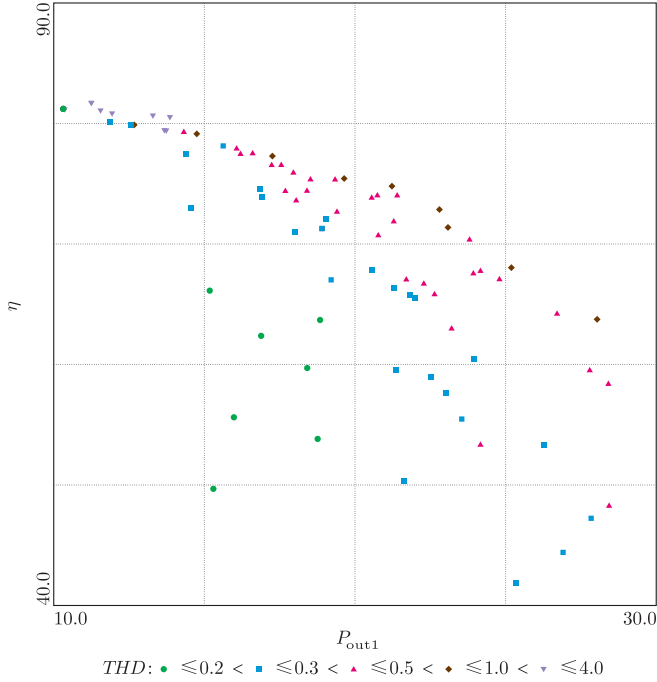


Fig. 7: Obtained Pareto front in the space  $P_{out1}$ ,  $\eta$ ,  $THD$ .

### 3.5 Models of LC Components

Unlike idealized circuit elements used in network theory, real-life components have *parasitics* attached to them that may (and often do) substantially modify the behavior of resulting circuits at higher frequencies. Therefore, their presence should be somehow considered within used design procedures. One approach would be to first assume ideal elements and after determining their values (e.g., by using optimization) to apply a correction to each component value so that its impedance at a chosen frequency (e.g., in the middle of the full frequency band) is equal, or at least close, to that of the ideal element in the design.

A more thorough and correct approach, however, is to introduce the parasitics already before the optimization by using parametrized RF models instead of simple ideal elements. Such a procedure is called *parasitic-aware optimization* [5]. It was chosen to be applied to all LC components in the present example. Fig. 9 shows used model structures for inductors and capacitors.

Only rough estimates of parasitics and their dependences on the main component values have been introduced, as real parameters and functions strongly depend on types and spatial configuration of the real components, their lead lengths, etc. Each inductor  $L_i$ ,  $i = 1$  and  $2$  has a series resistance  $R_{Li}$  representing all kinds of power losses (due to skin effect, eddy currents and/or coil core hysteresis, etc.) and a parallel capacitance  $C_{Li}$  modeling the collective stray capacitance (between the coil's winding turns, leads, etc.),

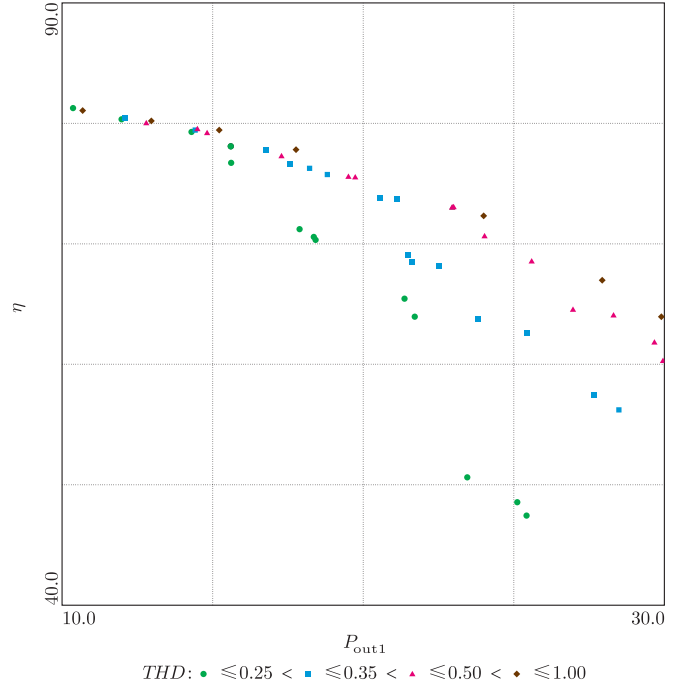


Fig. 8: The 3D Pareto front obtained in the form of contours.

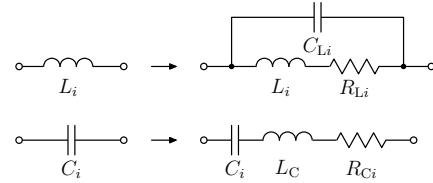


Fig. 9: Modeled parasitics of passive components.

whose values are obtained using formulas

$$R_{Li} = \frac{2\pi f_1 L_i}{Q_{Lmax}} + R_{L0} \quad \text{and} \quad C_{Li} = L_i p_{CL} + C_{L0}. \quad (17)$$

Here the frequency  $f_1 = 300$  MHz; the maximum quality factor  $Q_{Lmax} = 100$ , achievable only when the constant term  $R_{L0} = 10$  m $\Omega$  is negligible; the stray capacitance coefficient  $p_{CL} = 1$  pF/ $\mu$ H and the constant term  $C_{L0} = 100$  fF.

Similarly, each capacitor  $C_i$ ,  $i = 1, 2$ , and  $3$  has a series resistance  $R_{Ci}$  (also known as ESR)

$$R_{Ci} = \frac{1}{2\pi f_1 C_i Q_{Cmax}} + R_{C0}, \quad (18)$$

where  $Q_{Cmax} = 1000$  and  $R_{C0} = 10$  m $\Omega$ ; and a stray series inductance  $L_C$  (ESL) estimated by a constant value of  $3$  nH.

Even though those formulas and parameter values are very approximate, they still represent a significant improvement to the whole method, at least, by helping to keep the component values in the design after optimization in realistic proportions. For example, inductances will not tend to be too large, as their own resonant frequencies need to stay above the basic signal frequency  $f_1$  (and probably also above some higher harmonic frequencies).

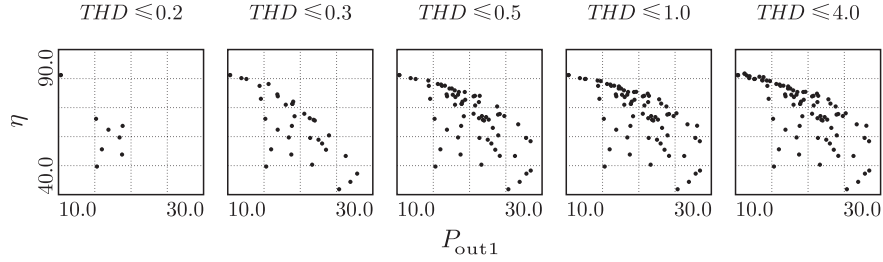


Fig. 10: An alternative way of displaying the 3D Pareto front.

Table 3: Selected solutions from the Pareto front.

No.	Symbol	Solution Number					Unit
		1	2	3	4	5	
1	$V_{gsmax}$	9.97	15.9	20.0	19.2	18.9	V
2	$V_{gsACm}$	4.03	8.05	10.7	9.24	12.0	V
3	$L_1$	7.86 n	11.3 n	4.23 n	3.97 n	5.03 n	H
4	$C_1$	294 p	133 p	299 p	51.6 p	166 p	H
5	$C_2$	22.6 p	5.09 p	27.0 p	300 p	3.41 p	H
6	$L_2$	6.84 n	7.00 n	7.97 n	7.32 n	9.89 n	H
7	$C_3$	20.1 p	2.35 p	18.4 p	22.6 p	17.0 p	H
1	$P_{out1}$	15.3	18.8	22.8	28.4	11.2	W
2	$\eta$	49.7	63.7	72.9	58.4	81.7	%
3	$THD$	0.163	0.239	0.512	0.394	3.03	%
4	$I_{d\ avg}$	2.56	2.44	2.59	4.01	1.15	A
5	$P_{diss}$	13.6	8.42	6.65	16.0	1.84	W

### 3.6 Results

Fig. 7 shows a total of 84 obtained solutions covering the three-dimensional Pareto front. There is obviously a large trade-off between harmonic distortion and power efficiency at the highest output power levels, but it diminishes with decreasing the output power, and for  $P_{out1} \leq 14$  W the requirement of low distortion can be met with almost no penalty on power efficiency.

Instead of by type of points, the different  $THD$  bounds can be distinguished by separating the particular cases into an array of graphs. Such an alternative format is presented in Fig. 10.

A selection of five distinctly different solutions is given in Table 3: number 1 has the lowest distortion  $THD$ , 4 has one of the highest values of  $P_{out1}$ , and 5 the highest efficiency  $\eta$ ; solutions 2 and 3 are located in the middle area at different levels of  $THD$ .

Instead of trying to uniformly cover the three-dimensional Pareto front, it may be preferable to cover only a set of its two-dimensional contours. This can be done by having  $THD$  as a constraint (instead of as objective function) and by repeating the optimizations for different  $THD$  bound values. Alternately, if we already have a set of solutions such as of Fig. 7, we can obtain the solution covering the contours by reoptimizing them to the new set of constraints that includes  $THD$ , i.e., by running a new optimization with each

already available solution as a starting point. A result of such a procedure is shown in Fig. 8.

### 4. Conclusions

An improved semiautomatic multiobjective method based on an asymptotically uniform coverage of the reference set in the combination with goal attainment method was proposed and implemented. The method was successfully tested on more examples, and one – a sophisticated optimization of the highly nonlinear circuit in the time domain – was presented.

### Acknowledgments

This paper has been supported by the Grant Agency of the Czech Republic, grant N° P102/10/1665, and by the Czech Technical University project N° SGS 12/151/OHK3/2T/13.

### References

- [1] K. K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston (MA): Kluwer Academic Publishers, 1998.
- [2] P. Bruschi, D. Navarrini, G. Tarroboiro, and G. Raffa, "A computationally efficient technique for the optimization of two stage CMOS operational amplifiers," in *Proc. European Conference on Circuit Theory and Design*, Cracow, Poland, Sep. 2003, pp. 305–308.
- [3] T. F. Coleman and Y. Zhang, *Optimization Toolbox 4 – User's Guide*. Natick (MA): The MathWorks, Inc., Mar. 2008.
- [4] J. Dobeš, "Advanced types of the sensitivity analysis in frequency and time domains," *AEÜ—International Journal of Electronics and Communications*, vol. 63, no. 1, pp. 52–64, Jan. 2009.
- [5] D. Allstot, K. Choi, and J. Park, *Parasitic-Aware Optimization of CMOS RF Circuits*. New York: Kluwer, 2003.



# A Vertical Splitting Scheme for Nonhydrostatic Atmospheric Model

A. Bourchtein, and L. Bourchtein

Institute of Physics and Mathematics, Pelotas State University, Pelotas, Brazil

**Abstract** - In this study, a numerical solution of nonhydrostatic atmospheric equations is considered. A robust semi-implicit approach with additional time splitting is applied in order to construct computationally efficient and accurate numerical scheme for modeling of large-scale atmospheric dynamics. Description of the designed numerical algorithm is provided and its accuracy and stability are discussed. The main properties of the scheme are compared to the respective properties of more traditional algorithms. Performed numerical experiments with the actual atmospheric data of pressure, temperature and wind show that the developed scheme supplies accurate forecast fields for the increased time steps chosen in accordance with the physical requirements.

**Keywords:** atmospheric modeling, numerical solution, semi-implicit scheme, time splitting

## 1 Introduction

The Earth atmosphere is a complex dynamical system which supports the processes of different time and space scales, including the most important synoptic (weather) processes. There are different approximated physical models of atmosphere dynamics constructed to filter the secondary waves, however all these approximations introduce certain distortions to the principal synoptic modes. For this reason and also due to considerable increasing of computer power and advances in numerical methods, in the last years a great attention in atmospheric modeling is given to non-filtered models called also nonhydrostatic equations. These models include the Euler momentum equations, mass conservation equation and energy conservation equation for compressible inviscid ideal gas considered usually in the rotating frame related to the Earth surface. Analysis of the corresponding linearized nonhydrostatic equations reveals three kinds of the waves - acoustic, gravity and inertial waves - from which only the

last waves are related to the synoptic processes. These waves differ not only in the source of their origin (compressibility, gravity force and Coriolis force together with advection), but also in such important characteristics as propagation velocity and energy contribution. In fact, the characteristic propagation speed of the acoustic waves in the Earth's atmosphere is about 330-340 m/s, while the speed of the gravity waves is always below 330 m/s (with majority of the gravity waves propagating at 100 m/s and below) and the propagation speed of the inertial processes (advection and Rossby waves) is usually about 10 m/s with the (very rare) maximum values below 80 m/s [6,9,11]. At the same time, it is well-known from evaluations of the spectrum of oscillations of the Earth atmosphere that the energy of the acoustic waves is negligible, the gravity modes contain a small part of the evaluable energy for the majority of the large- and meso-scale processes, and the inertial modes contain the main part of the atmospheric energy [9,11]. In this way, the problem of numerical weather forecasting and atmospheric modeling is a stiff problem.

Since the wave filtration can not be performed in the differential form without distortion of the principal inertial modes, the problem stiffness should be addressed in the design of numerical algorithm in order to achieve sufficient efficiency and accuracy of numerical solution. Indeed, the time step of a numerical scheme is generally determined by the Courant-Friedrichs-Lewy (CFL) condition [7]:

$$\tau \leq \frac{h}{c_{\max \exp}},$$

where  $\tau$  is the time step,  $h$  is the mesh size of a spatial grid and  $c_{\max \exp}$  is the maximum propagation speed of the processes treated explicitly in the chosen form of the time differencing. For many totally implicit schemes there is no restriction on the size of the time step due to

numerical approximation. In such cases the time step can be chosen solely on the ground of the physical limitations related to the time scale of the considered processes and the required accuracy of numerical solution. However, implicit schemes require solution of high-order nonlinear systems at each time step, that turns these schemes excessively computationally demanding and inefficient. For this reason, totally implicit time differencing is not practically used in atmospheric modeling.

On the other pole, fully explicit schemes assure simple and fast calculations at each time step of numerical integration, but the size of the time step for non-filtered atmospheric models is too small. For such schemes it is defined by the CFL condition with the maximum speed equal to the propagation speed  $c_{\max acoust}$  of the acoustic waves [7,15]. Therefore, for the typical vertical resolution  $h_v \approx 400$  m used in large- and meso-scale models, the CFL condition

$$\tau \leq \frac{h_v}{c_{\max acoust}}$$

requires the time steps less than 1 sec. Taking in account that the characteristic time scale of the synoptic (large- and meso-scale) models is about a couple of hours, such severe restriction on  $\tau$  makes the fully explicit time differencing computationally inefficient.

A more reasonable approach is a semi-implicit time differencing, in which the linear terms of the governing equations, which are responsible for the fastest waves, are approximated implicitly, while the remaining terms - explicitly. For example, applying an implicit approximation to the main linear terms in the vertical, it is possible to increase the time step almost two orders, because the CFL condition will include the horizontal propagation speed of the gravity waves  $c_{\max grav}$  [13, 15]. For the typical horizontal resolution  $h_h \approx 20$  km, the time restriction

$$\tau \leq \frac{h_h}{c_{\max grav}}$$

will allow the time steps up to 1 min. The price of this improvement is a necessity to solve systems of linear equations with narrow-band matrices at each time step, which can be accomplished efficiently employing the Gelfand-Thomas type algorithm [7]. However, the above time step is still too small as compared to the accuracy requirements.

Implicit differencing of all the main linear terms in the governing equations allows further increasing of the time step up to 5 min (under the same horizontal resolution  $h_h \approx 20$  km) [7,14]:

$$\tau \leq \frac{h_h}{c_{\max adv}},$$

because only the maximum advection speed  $c_{\max adv}$  enters in the CFL condition in this case. This is a reasonable choice for the time step, comparable with the physical requirements of accuracy. However, such time approximation leads (at each time step) to solution of high-order linear systems with wide-band matrices, that pull down the efficiency of this approach.

It is worth to note that the semi-implicit approach assures usually the same level of accuracy of numerical solutions as the explicit schemes with the same approximation order. This is the reason why the semi-implicit schemes are so popular in atmospheric modeling. Nevertheless, when these schemes arrive to the limit of their efficiency, some additional techniques should be applied to speed-up computations. One of such techniques is a splitting by physical processes. Applied in its extreme form, it usually causes substantial additional errors, which grows significantly when the time step approaches the limit allowable by the CFL condition. In such cases a numerical splitting scheme can be stable for rather large time steps and keeps formally the required approximation order, but the practical accuracy of numerical solutions can fall down [7, 13, 15]. Therefore, the splitting techniques should be applied cautiously, frequently using a partial splitting, and checking the accuracy of the obtained results against available atmospheric data and simulation output of other verified models.

In this study, a partial physical splitting is introduced in the semi-implicit scheme in such a way that all the vertical modes are separated according their propagation velocity. Such vertical splitting corresponds to the analytical separation of the spectrum of the atmospheric waves supported by the governing equations. The gravity waves of the fast vertical modes are approximated implicitly, while those of the slow vertical modes are treated explicitly. It allows us to substitute the problem of solution of three-dimensional elliptic equations by solution of a set of the two-dimensional elliptic problems. The solution of the last set of problems can be obtained much more efficiently than for their three-dimensional counterpart. Additionally, a simpler approximation can be applied to slower vertical modes, because they contain a small part of the total energy [9, 16]. The performed numerical experiments show that applied vertical splitting maintain the desired level of the accuracy of the forecasting fields and speeds-up computations required for each forecast.

## 2 Governing nonhydrostatic equations

The momentum equations of inviscid atmosphere in the coordinate system related to the rotating Earth can be written as follows:

$$\begin{aligned} u_t &= \bar{f}v - R\bar{T}P_x + N_u, \\ v_t &= -\bar{f}u - R\bar{T}P_y + N_v, \end{aligned} \quad (1)$$

$$w_t = \frac{g}{\bar{T}}T - R\bar{T}P_z + N_w. \quad (2)$$

The continuity equations for compressible ideal gas is

$$P_t = -\frac{c_p}{c_v}(u_x + v_y + w_z) + \frac{g}{R\bar{T}}w + N_P. \quad (3)$$

The last equation is the thermodynamic one, including the equations of the state for ideal gas,

$$T_t = \frac{R\bar{T}}{c_p} \cdot P_t - \frac{g}{c_p}w + N_T. \quad (4)$$

The following notations are used above:  
the independent variables:  $t$  - the time coordinate,  $x, y, z$  - the spatial Cartesian coordinates;  
the unknown functions:  $u, v, w$  - the velocity components,  $P = \ln p$  - the pressure logarithm,  $T$  - the temperature;  
the given parameters:  $f$  - the Coriolis parameter with the mean value  $\bar{f} = const$ ,  $g$  - the gravitational acceleration,  $\bar{T} = const$  - the mean value of the temperature,  $R$  - the gas constant,  $c_p$  and  $c_v$  - the specific heat at constant pressure and volume, respectively;

The nonlinear terms  $N_u, N_v, N_w, N_T$  represent mainly advective terms in each equation, whose specific form will not be used in the subsequent formulas.

This is the standard form of the governing equations of the nonhydrostatic atmosphere, which can be found in different sources (e.g. [7,11]).

## 3 Time splitting semi-implicit scheme

In this section the time splitting semi-implicit scheme is presented and its analytical properties of accuracy and stability are discussed and compared to the respective properties of the standard semi-implicit scheme.

### 3.1 Standard semi-implicit scheme

The standard three-time-level semi-implicit time differencing of the second order of accuracy for equations (1)-(4) has the following form [8, 14, 15]:

$$\begin{aligned} \frac{u^\tau - u^{-\tau}}{2\tau} &= \bar{f} \frac{v^\tau + v^{-\tau}}{2} - R\bar{T} \frac{P_x^\tau + P_x^{-\tau}}{2} + N_u, \\ \frac{v^\tau - v^{-\tau}}{2\tau} &= -\bar{f} \frac{u^\tau + u^{-\tau}}{2} - R\bar{T} \frac{P_y^\tau + P_y^{-\tau}}{2} + N_v, \end{aligned} \quad (5)$$

$$\frac{w^\tau - w^{-\tau}}{2\tau} = \frac{g}{\bar{T}} \frac{T^\tau + T^{-\tau}}{2} - R\bar{T} \frac{P_z^\tau + P_z^{-\tau}}{2} + N_w, \quad (6)$$

$$\frac{P^\tau - P^{-\tau}}{2\tau} = -\frac{c_p}{c_v} \frac{D^\tau + D^{-\tau}}{2} - \frac{c_p}{c_v} \frac{w_z^\tau + w_z^{-\tau}}{2} + \frac{g}{R\bar{T}} \frac{w^\tau + w^{-\tau}}{2} + N_P, \quad (7)$$

$$\frac{T^\tau - T^{-\tau}}{2\tau} = \frac{R\bar{T}}{c_p} \frac{P^\tau - P^{-\tau}}{2\tau} - \frac{g}{c_p} \frac{w^\tau + w^{-\tau}}{2} + N_T. \quad (8)$$

Here  $D = u_x + v_y$  - the horizontal divergence,  $\tau$  - the time step, the superscript “ $\tau$ ” denotes the quantities on the next time level  $t_{n+1} = (n+1)\tau$  (so-called the prognostic values), the superscript “ $-\tau$ ” denotes the values on the past time level  $t_{n-1} = (n-1)\tau$ , and functions without superscripts are considered on the current time level  $t_n = n\tau$ :

$$\begin{aligned} \varphi^\tau &= \varphi(t_{n+1}, x, y, z), \varphi = \varphi(t_n, x, y, z), \varphi^{-\tau} = \varphi(t_{n-1}, x, y, z), \\ \varphi &= u, v, w, P, T. \end{aligned}$$

It is easy to realize that the scheme (5)-(8) combines the implicit Crank-Nicolson approximation with the double time step for the linear terms, and the explicit leap-frog approximation for the nonlinear and variable coefficient terms.

The analysis of linear stability of the scheme (5)-(8) shows that the time step is restricted by the condition

$$\tau \leq \frac{h_h}{c_{\max adv}}.$$

For the horizontal mesh size  $h_h \approx 20$  km and the maximum advection velocity  $c_{\max adv} \approx 60$  m/s, the maximum allowable time step is about 5 min.

### 3.2 Splitting scheme: semi-implicit step

Although the scheme (5)-(8) is rather efficient as compared to more explicit schemes, it has some drawbacks, the main of which is a necessity to perform inefficient computations at each time step related to solution of three-dimensional elliptic problems arising due to the implicit approximation of the linear terms. Since a part of these terms is responsible for slowly propagating internal gravity waves, the amount of computations can be decreased by applying vertical splitting. Such splitting allows us to separate and, consequently, approximate differently the principal (fast) and secondary (slow)

vertical modes. A similar approach has been applied successfully in the hydrostatic atmospheric models [3, 4, 5, 10, 12]. To perform the vertical splitting, it is convenient to divide each time step in two stages. On the first stage, the explicit leap-frog scheme is applied to all the terms, except for those responsible for vertical propagation of the acoustic waves, which are approximated implicitly by the Crank-Nicolson scheme:

$$\begin{aligned}\frac{\hat{u}^\tau - u^{-\tau}}{2\tau} &= \bar{f}v - R\bar{T}P_x + N_u, \\ \frac{\hat{v}^\tau - v^{-\tau}}{2\tau} &= -\bar{f}u - R\bar{T}P_y + N_v,\end{aligned}\quad (9)$$

$$\frac{\hat{w}^\tau - w^{-\tau}}{2\tau} = \frac{g}{\bar{T}} \frac{\hat{T}^\tau + T^{-\tau}}{2} - R\bar{T} \frac{\hat{P}_z^\tau + P_z^{-\tau}}{2} + N_w, \quad (10)$$

$$\frac{\hat{P}^\tau - P^{-\tau}}{2\tau} = -\frac{c_p}{c_v} D - \frac{c_p}{c_v} \frac{\hat{w}_z^\tau + w_z^{-\tau}}{2} + \frac{g}{R\bar{T}} \frac{\hat{w}^\tau + w^{-\tau}}{2} + N_p, \quad (11)$$

$$\frac{\hat{T}^\tau - T^{-\tau}}{2\tau} = \frac{R\bar{T}}{c_p} \frac{\hat{P}^\tau - P^{-\tau}}{2\tau} - \frac{g}{c_p} \frac{\hat{w}^\tau + w^{-\tau}}{2} + N_T. \quad (12)$$

The computations on this stage are much faster than for the standard semi-implicit scheme (5)-(8). Indeed, the last three equations can be decoupled from the system and solved separately. Unknown functions in these equations are coupled only in the vertical variable, that reduces (10)-(12) to a set of decoupled one-dimensional boundary value problems, each of which can be solved efficiently by the Gelfand-Thomas algorithm [7]. After solution of (10)-(12) is found, the remaining equations (9) have the explicit form with respect to the horizontal velocity components. The deficiency of the scheme (9)-(12) is its weak stability [8, 13]:

$$\tau \leq \frac{h_h}{c_{\max grav}},$$

that means that for the horizontal mesh size  $h_h \approx 20$  km and the maximum propagation speed of the gravity waves  $c_{\max grav} \approx 300$  m/s, the maximum allowable time step is about 1 min.

### 3.3 Splitting scheme: correction equations

To improve the stability of (9)-(12), a more implicit time differencing should be applied, for example, such as in (5)-(8). Equations for the differences (corrections) between solutions (prognostic quantities) of (5)-(8) and (9)-(12), can be written as follows:

$$\begin{aligned}u^* - \bar{q}v^* + \bar{\alpha}R\bar{T}P_x^* &= L_u, \\ v^* + \bar{q}u^* + \bar{\alpha}R\bar{T}P_y^* &= L_v,\end{aligned}\quad (13)$$

$$w^* - \tau \frac{g}{\bar{T}} T^* + \bar{\alpha}R\bar{T}P_z^* = L_w, \quad (14)$$

$$P^* + \tau \frac{c_p}{c_v} D^* + \tau \frac{c_p}{c_v} w_z^* - \tau \frac{g}{R\bar{T}} w^* = L_p, \quad (15)$$

$$T^* - \frac{R}{c_p} \bar{T}P^* + \tau \frac{g}{c_p} w^* = L_T, \quad (16)$$

where  $\bar{\phi}^* = \bar{\phi}^\tau - \hat{\phi}^\tau$ ,  $\phi = u, v, w, T, P$  are the corrections to be found and the linear terms  $L_\phi$  do not include the prognostic values of the standard semi-implicit scheme:

$$L_u = \bar{q}\bar{v} - \bar{\alpha}R\bar{T}\bar{P}_x, \quad L_v = -\bar{q}\bar{u} - \bar{\alpha}R\bar{T}\bar{P}_y,$$

$$L_w = 0, \quad L_p = -\tau \frac{c_p}{c_v} \bar{D}, \quad L_T = 0;$$

$$\bar{\phi} = \hat{\phi}^\tau - 2\phi + \phi^{-\tau}, \quad \phi = u, v, w, T, P.$$

Considering, that the prognostic values of the first stage  $\hat{\phi}^\tau$  are already calculated, the equations (13)-(16) represent the linear system for corrections  $\phi^*$ . Evidently, a solution of this system is equivalent to finding the prognostic values of the standard semi-implicit scheme (5)-(8), but, as it was noted above, such procedure requires solution of the three-dimensional elliptic problems. However, application of the vertical splitting can significantly reduce the amount of the computations without loss of accuracy and stability of numerical solution.

The first step to perform the vertical splitting is elimination of unknown functions  $w^*$  and  $T^*$  from equations (14)-(16), that leads to the following equation for  $P^*$ :

$$-P_{zz}^* + \frac{g}{R\bar{T}} P_z^* + \frac{c_v}{c_p} \frac{1}{\tau^2 R\bar{T}} P^* + \frac{1}{\bar{\alpha}R\bar{T}} \left( 1 + \tau^2 \frac{g^2}{c_p \bar{T}} \right) D^* = L_1. \quad (17)$$

Considering the first two terms of the last equation together with the kinematic boundary conditions  $w = 0$  on the upper ( $z = 0$ ) and lower ( $z = z_{up}$ ) boundaries of the atmosphere rewritten for the function  $P^*$ , one can arrive to the corresponding Sturm-Liouville problem:

$$-P_{zz}^* + \frac{g}{R\bar{T}} P_z^* = \lambda P^*, \quad z \in (0, z_{up}); \quad (18)$$

$$P_z^* - \frac{g}{c_p \bar{T}} P^* = 0, \quad z = 0, \quad z = z_{up}. \quad (19)$$

It can be shown that the spectrum of the last problem is simple and positive, and the eigenvalues  $c_k$  approach fast the only limit point 0 of the spectrum set. This property is important for a selective correction of the vertical modes.

### 3.4 Splitting scheme: vertical decoupling

On the second stage of vertical splitting scheme, the eigenfunctions  $F_k(z)$  of the problem (18)-(19) are used to expand some unknown and known quantities:

$$\varphi^* = \sum_k \varphi_k^+(x, y) F_k(z), \quad \varphi = u, v, P, L.$$

Applying this expansion, the correction equations (13), (17) can be rewritten in the form

$$\begin{aligned} u^+ - \bar{q}v^+ + \bar{\tau}R\bar{T}P^+_x &= L^+_u, \\ v^+ + \bar{q}u^+ + \bar{\tau}R\bar{T}P^+_y &= L^+_v, \\ R\bar{T}P^+ + \tau c_k^2(u^+_x + v^+_y) &= L^+_P. \end{aligned} \quad (20)$$

For a simplicity of notations the subscript "k" is omitted in all quantities, except for  $c_k$ .

Each of the systems (20) can be solved separately from others and represents the time discretization of the linearized shallow water equations with the corresponding gravity wave speed  $c_k$ . Internal vertical modes, with large values of  $k$ , correspond to the slow gravity waves and contain a small part of the atmospheric energy. Therefore, they have no influence on the scheme stability and their contribution to the solution accuracy is very small. Hence, it is sufficient to solve only a few of the principal vertical systems (20) in order to improve considerably the stability of the semi-implicit stage (9)-(12). Analysis of the linear stability of the vertically splitting scheme shows that the restriction on the time step can be expressed as

$$\tau \leq \frac{h_h}{c_{I+1}},$$

where  $c_{I+1}$  is the maximum propagation velocity of the vertical modes that remain uncorrected. For example, applying corrections to the first four vertical modes in the model with 30 vertical levels, it is possible to increase the time step from 1 min to 5 min, because  $c_5 \approx 50$  m/s. The achieved time step is practically equivalent to the maximum allowable time step of the standard semi-implicit scheme (5)-(8). In this way, in order to recover the scheme stability after the semi-implicit stage (9)-(12), it is sufficient to solve only a small fraction of the systems (20). After this, the inverse vertical transformation returns the physical values of the corrections for the pressure and horizontal components of velocity, and, finally, the corrections to the temperature and vertical velocity are found by explicit formulas (14) and (16).

## 4 Numerical experiments

The described numerical scheme was applied to forecasting of atmospheric fields on the horizontal area of 3000x3000 km<sup>2</sup>, covering the South part of Brazil and adjacent territories (the center point was chosen at 30° South and 55° West) and within the vertical layer of the atmosphere extending from the Earth surface up to 15 km. The horizontal grid was chosen to be uniform with the mesh size of 20 km and the vertical layer was divided in 30 sub-layers of different thickness - the finest vertical resolution was used in the planetary boundary layer and near tropopause, where the vertical variations are the highest.

The initial conditions were derived from the objective analysis fields of the National Centers for Environmental Predictions (NCEP), and the boundary conditions were defined based on the global forecast fields of the same center. The 12, 24 and 36-hour forecasts were calculated on the defined above territory, but the evaluation of the forecast skills was made on the inner territory of the size 1000x1000 km<sup>2</sup> in order to minimize the influence of the boundary conditions. The forecasts beyond 36 hours were not calculated, because it is well-known that they are highly dependent on the provided boundary conditions [1,2].

To evaluate the accuracy and efficiency of the constructed scheme, the forecasting results were compared to the respective forecasts obtained with the use of the standard semi-implicit scheme (5)-(8) and the simpler semi-implicit scheme (9)-(12). The standard measures of evaluation of the forecast skills used in the short-range numerical weather prediction were employed: the root-mean-square error (the root-mean-square difference between forecast and analysis fields for the chosen meteorological element and vertical surface) and correlation coefficient between the prognostic and actual tendencies (again for the chosen meteorological element and vertical surface) [1,2].

The root-mean-square errors for the geopotential height of 500 hPa are shown in Fig.1 for the indicated three schemes. This surface is quite characteristic for evaluation of the synoptic processes in mid-troposphere. The root-mean-square errors for the temperature at the pressure surface of 850 hPa are shown in Fig.2. The forecasts on this surface are important for evaluation of convective activity in lower troposphere, which affects the formation of clouds and precipitation in more complete non-adiabatic atmospheric models. One can note a practical identity of the forecast accuracy of the standard and time splitting semi-implicit schemes for the both assessed fields. The evaluations of the correlation coefficients of the forecasts calculated with the indicated

three schemes show similar results (not presented here): the coefficients for the vertically splitting and standard semi-implicit schemes are practically identical, while the coefficient for the scheme (9)-(12) is a bit smaller, showing slightly decreasing quality.

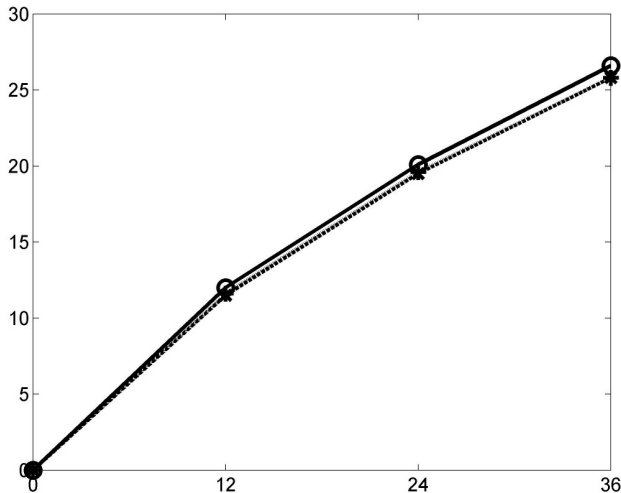


Fig 1. Root-mean-square errors of the forecasts of the geopotential height (in meters) at the pressure surface of 500 hPa presented as a function of the forecast time. Solid line - the semi-implicit scheme (9)-(12), dashed line - the standard semi-implicit scheme (5)-(8), dotted line - the vertical splitting semi-implicit scheme.

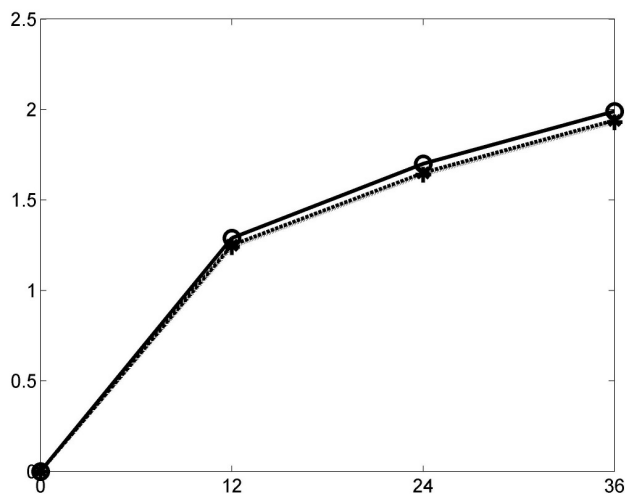


Fig 2. Root-mean-square errors of the forecasts of the temperature (in degrees) at the pressure surface of 850 hPa presented as a function of the forecast time. Solid line - the semi-implicit scheme (9)-(12), dashed line - the standard semi-implicit scheme (5)-(8), dotted line - the vertical splitting semi-implicit scheme.

As it can be seen from the provided results, the vertical splitting does not introduce any visible additional errors and provides numerical solutions of the same level of accuracy as the standard semi-implicit scheme: the differences between evaluations of two forecasts are practically negligible for both root-mean-square error and correlation coefficient, and for both chosen meteorological elements. At the same time, the computational time required for the vertical splitting scheme is almost a half of the forecast time for the standard semi-implicit scheme. It is worth to note that according to the properties of the spectrum of the eigenvalue problem (18)-(19), the number of the fast vertical modes remains almost the same when the number of the vertical levels increases. Therefore, under current tendency of enhancing vertical resolution to 40 or 50 vertical levels, or even higher, the computational speed-up obtained in the vertical splitting scheme can be even stronger.

## 5 Conclusions

In this report we have presented a finite-difference semi-implicit time splitting scheme designed to overcome the problems related to numerical solution of the stiff atmospheric equations based on the non-filtered Euler equations of the ideal compressible gas. Each time step of the constructed scheme consists of two stages: a simple semi-implicit integration and subsequent solution of the corrections equations for the fastest vertical modes. The time step of stable integration is defined by the maximum velocity of processes treated explicitly that allows the use of the time steps comparable with those required by physical accuracy. The provided results of numerical experiments have shown the accuracy of the obtained forecasts and the computational efficiency of the developed numerical scheme.

## 6 Acknowledgements

This research was supported by Brazilian science foundations CNPq and FAPERGS.

## 7 References

- [1] R.A. Anthes. "Regional models of the atmosphere in middle latitudes"; *Mon. Wea. Rev.*, Vol. 111, 1306-1335, Jun 1983.
- [2] R.A. Anthes, Y.H. Kuo, E.Y. Hsie, S. Low-Nam, T.W. Bettge. "Estimation of skill and uncertainty in regional numerical models"; *Q. J. R. Meteorol. Soc.*, Vol. 115, 763-806, Jul 1989.

- [3] A. Bourchtein. "Semi-Lagrangian semi-implicit space splitting regional baroclinic atmospheric model"; *Appl. Numer. Math.*, Vol. 41, 307-326, Jan 2002.
- [4] A. Bourchtein, L. Bourchtein. "Semi-Lagrangian semi-implicit time-splitting scheme for a regional model of the atmosphere"; *J. Comput. Appl. Math.*, Vol. 227, 115-125, May 2009.
- [5] D.M. Burridge. "A split semi-implicit reformulation of the Bushby-Timpson 10 level model"; *Q. J. R. Meteorol. Soc.*, Vol. 101, 777-792, Oct 1975.
- [6] M. Cullen. "Modelling atmospheric flows"; *Acta Numerica*, Vol. 16, 67-154, May 2007.
- [7] D. Durran. "Numerical Methods for Wave Equations in Geophysical Fluid Dynamics". Springer-Verlag, 1999.
- [8] A. Gassmann, H.-J. Herzog. "A consistent time-split numerical scheme applied to the nonhydrostatic compressible equations"; *Mon. Wea. Rev.*, Vol. 135, 20-36, Jan 2007.
- [9] J.R. Holton. "An Introduction to Dynamic Meteorology". Academic Press, 2004.
- [10] V. Kadychnikov, V. Losev. "Application of the alternating direction implicit method to the numerical regional weather forecast"; *Meteorology and Hydrology*, Vol. 9, 26-33, Sep 1991.
- [11] E. Kalnay. "Atmospheric Modeling, Data Assimilation and Predictability". Cambridge University Press, 2002.
- [12] L.M. Leslie, R.J. Purser. "Three-dimensional mass-conserving semi-Lagrangian scheme employing forward trajectories"; *Mon. Wea. Rev.*, Vol. 123, 2551-2566, Aug 1995.
- [13] W.C. Skamarock, J.B. Klemp. "A time-split nonhydrostatic atmospheric model for weather research and forecasting applications"; *J. Comp. Phys.*, Vol. 227, 3465-3485, Mar 2008.
- [14] A. Staniforth, N. Wood. "Aspects of the dynamical core of a nonhydrostatic, deep-atmosphere, unified weather and climate-prediction model"; *J. Comp. Phys.*, Vol. 227, 3445-3464, Mar 2008.
- [15] J. Steppeler, R. Hess, U. Schattler, L. Bonaventura. "Review of numerical methods for nonhydrostatic weather prediction models"; *Meteorol. Atmos. Phys.*, Vol. 82, 287-301, Jan 2003.
- [16] D.L. Williamson, C. Temperton. "Normal mode initialization for a multilevel grid-point model. Part II: nonlinear aspects"; *Mon. Wea. Rev.*, Vol. 109, 744-757, Apr 1981.

# Computational Algorithm for Estimating the Gradient Isocenter of an MRI Scanner

R. Cai<sup>1</sup>, K.E. Schubert<sup>1</sup>, R. Schulte<sup>2</sup>

ritchicai@r2labs.org, keith@r2labs.org, rschulte@dominion.llumc.edu

<sup>1</sup>School of Computer Science and Engineering, California State University, San Bernardino,  
5500 University Parkway, San Bernardino, CA 92407 USA

<sup>2</sup>Department of Radiation Medicine, Loma Linda University Medical Center,  
Loma Linda, CA, 92354, USA

**Abstract**—*The isocenter of the magnetic gradient fields inside a magnetic resonance imaging scanner is an important parameter in gradient non-linearity distortion correction methods for MR images. Currently there is no established method for estimating the gradient isocenter. All existing correction methods assume that the gradient isocenter coincides with the DICOM coordinate center, which is close but not exactly the same as the gradient isocenter. A difference between gradient isocenter and DICOM coordinate center could compromise the accuracy of the correction method. The goal of this research was to develop a reasonably accurate estimation method of the 3D location of the gradient isocenter that was based on the geometry of a custom-designed phantom. We present a two-step algorithm for estimating the gradient isocenter and examine some of the possible numeric methods to be used on each step.*

**Keywords:** Isocenter, MR Imaging, least squares, total least squares

## 1. Introduction

Magnetic resonance imaging (MRI) provides an excellent modality for distinguishing different tissues in the human body, which makes this modality essential for medical applications. MRI can also be used for treatment planning of medical procedures that require a great degree of geometrical accuracy such as functional radiosurgery [2]. Unfortunately, the accuracy of MRI for medical targeting applications is compromised by the presence of geometric distortions such as non-linearities of the magnetic gradient fields or inhomogeneities of the scanner main field [1], [3], [4], [5], [6], [7], [8], [9]. Gradient nonlinearities can cause over 2mm of distortion to the location of features in the image [6], [7].

Modern MRI scanners have built-in distortion correction algorithms, that rely on knowledge of the magnetic field configuration and its distortion. Built-in algorithms, which are usually proprietary, are designed based on assumed knowledge of the geometry and location of the gradient coils. A more practical approach is to use a phantom of known geometry and to derive the distortion by analyzing the MRI images generated with such a phantom [1], [3], [4], [5], [6],

[7], [8], [9]. This method has the advantage that a scanner-specific correction can be applied, which can also take into account potential changes in the gradient fields over time.

Accurate knowledge of the gradient isocenter is essential to very accurate distortion correction methods. To our knowledge, existing distortion correction algorithms, both built-in and phantom based, make the assumption that the gradient isocenter coincides with the origin of the DICOM coordinates. This assumption may not be accurate and should not be used if a high degree of accuracy is necessary.

The goal of this work was to develop and implement a numerical software-based method to estimate the gradient isocenter of the magnetic field inside an MRI scanner using the MRI scan of a custom-built phantom. In our previous work [3], [4], [5], [6], we used an oil filled plexiglas cube with 159.50mm × 159.70mm × 158.11mm dimension that was designed to fit MR scanners that were available at that time. Current MRI scanners have a significantly wider aperture, which necessitates a larger phantom to characterize the field distortion. Furthermore, since the lower part of the scanner aperture is occupied by the scanner table, the scanning area occupied by the phantom or the patient is not centered on the gradient isocenter, making correction methods that assume that the phantom is centered with respect to the gradient isocenter [3], [4], [5], [6], [7] obsolete and potentially inaccurate.

## 2. Distortion Phantom

A new phantom design was developed with the intention to probe the distortion in the 50 cm field-of-view (FOV) of a modern 3T MRI scanner (Magnetom Trio, Siemens). Due to the geometry of the gradient field nonlinearity, the distortion is largest in the outer fringes of the FOV [3], [4], [5], [6]. Capturing distortion data in the periphery of the FOV is important for better accuracy of the distortion correction. Building a very large cubical oil-filled phantom is a challenge due to weight limitations and limited capability to accurately machine a very large flat surface. Thus it was felt to be impractical to simply scale up the original phantom. Another limitation is that the phantom needs to



fit inside the scanner head coil which needs to be used to achieve a sufficiently high signal-to-noise ratio. To meet all requirements, we designed a phantom that is comprised of 64 NMR glass tubes of 3 mm inner diameter, filled with copper sulfate and arranged in 8 surfaces forming an octagon. A water-filled cylinder in the middle of the octagon ensures a good signal generated by the tubes. The distance between tubes on the opposite side of the surface is 205mm, which is about 28% wider than the original phantom. In addition we also added four more surfaces to provide data and better fit the headcoil. There are 8 removable tubes on each surface with 10 mm gap between adjacent tubes. Each tube is about 207 mm long, which is also an increase of about 28% in length from the original phantom. Tubes are placed along the main axis of the magnetic  $B_0$  field to reduce the effects of chemical shift and magnetic susceptibility and resulting in high-quality axial images.

### 3. Computational Algorithm

The algorithm we described here is using simulated data set that is based the geometric property of the phantom and the properties of the MR images. The MR images we are using have 1mm per pixel resolution for every image and 1mm thickness between two adjacent planes. In the simulation, we will be using millimeters as unit for each data point. Since the images we are using to collect data points are all axial scans, we will only add noise to x and y coordinate of each data point.

#### 3.1 Tube Modeling

The tubes is 203 mm long. So our simulation data is ranging from -100 to 100 on z axis for each tube. The arrangement of the tubes could be seen in Figure 1. Our algorithm is based on two standard assumptions [3], [4], [5], [6], [7].: (1) the distortion in MR images are only caused by the magnetic field inside MRI scanner and (2) the magnetic field can be perfectly described using the sum of spherical harmonic.

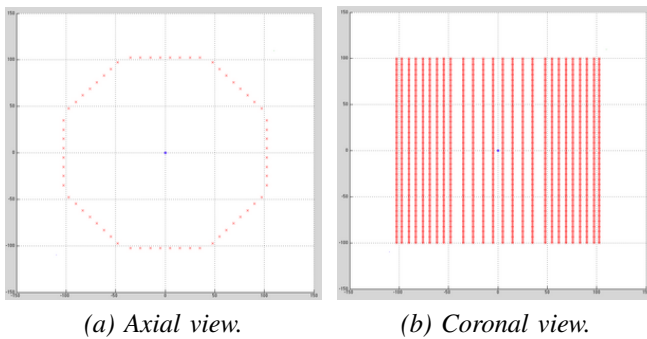


Fig. 1: Phantom modeling without distortion

The spacial coordinate of the phantom must be transformed into a coordinate relative to the gradient isocenter

of the magnetic field, see Equation 1, then using the first 5 terms of the spherical harmonic we can estimate a offset in  $x, y, z$  direction, see Equation 4. Finally this offset is added the original coordinate to create a distorted model, see Equation 5. Also a very key requirement for this algorithm is that the z-axis the phantom which is parallel to the tubes must be aligned with the z-axis of the magnetic field.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{iso} \\ y_{iso} \\ z_{iso} \end{bmatrix} \quad (1)$$

$$K = \begin{bmatrix} K_{x_0} & K_{x_1} & K_{x_2} & K_{x_3} & K_{x_4} \\ K_{y_0} & K_{y_1} & K_{y_2} & K_{y_3} & K_{y_4} \\ K_{z_0} & K_{z_1} & K_{z_2} & K_{z_3} & K_{z_4} \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} \alpha \\ \beta \\ \zeta \end{bmatrix} = K \begin{bmatrix} x'^2 + y'^2 \\ z'^2 \\ z'^2(x'^2 + y'^2)^2 \\ (x'^2 + y'^2)^2 \\ z'^4 \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} x' \alpha \\ y' \beta \\ z' \zeta \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} = \begin{bmatrix} x + \Delta x \\ y + \Delta y \\ z + \Delta z \end{bmatrix} \quad (5)$$

#### 3.2 ISO-Center Coordinate Estimation

Using a exaggerated distortion parameter we can visualize the shape of the distortion model. Without distortion, the phantom appears as in Figure 1. With large distortions, the phantom appears as in Figure 2, which is useful to understand the form of the distortions. A more realistic distortion can be seen in Figure 3, which uses distortion parameters from [6].

Distortion in the sum of spherical harmonics is coupled in the x and y directions (orthogonal to axis), making the z axis independent. Noise and distortion are thus very different in the z direction as opposed to the x-y plane. We break down the gradient isocenter coordinate estimation into two big steps: estimation of z coordinate and estimation of x,y coordinate.

##### 3.2.1 Z-Coordinate Estimation

Since the distortion model is based on sum of spherical harmonics, the shape of the distorted data for each tube is an even polynomial function. Figure 3 shows that the realistic distortion is quite small, being about a maximum of 2 pixels, and the smaller the distortion the more sensitive the problem becomes. With the first four terms of sum of spherical harmonics, the shape of the each tube can be seen as part of a polynomial function of 4th degree. Since the

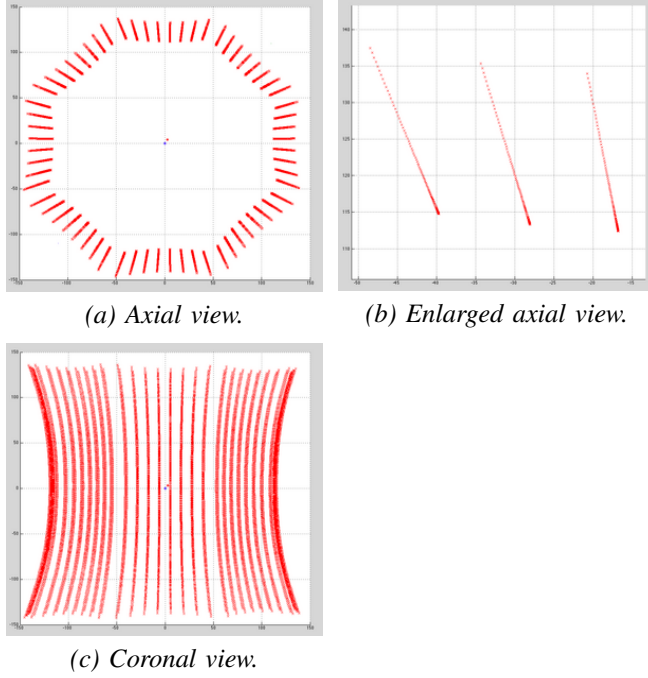


Fig. 2: Phantom modeling with large distortions

data points of tubes only represent the middle section of the 4th degree polynomial function, it is also very similar to a shape of quadratic function. The offset between the largest distortion and the smallest distortion can be as small as 1mm. With such a small margin, it is more practical to fit the data points to simpler model than a 4th degree polynomial. Only the first spherical harmonic is needed in order to estimate the point where gradient is zero to measure the isocenter. Furthermore, the first term of the sum of spherical harmonics has the largest signal to noise ratio, so we use a quadratic model to fit the data points.

We can see the distorted tube models' middle part is bending toward the center with both end bending outward. If we were to fit each tubes to a parabola and locate the point where its gradient is zero, that point's z-coordinate should be the same as the z-coordinate of the gradient isocenter of the magnetic field. The z-coordinate is measured for all 64 tubes and the result averaged. The resulting estimation is within 0.1 mm of the actual z-coordinate.

### 3.2.2 X,Y Coordinate Estimation

The estimation of the x and y coordinates of the isocenter is the more difficult problem. We are assuming that the difference between the distortion on x and y direction are so small that we can treat them as if they are the same. When this is not true, the errors on the isocenter location will be asymmetric, and it will be even more important to maintain a good numerical method to estimate the isocenter. With that in mind, the distorted data of a tube should all stay on a

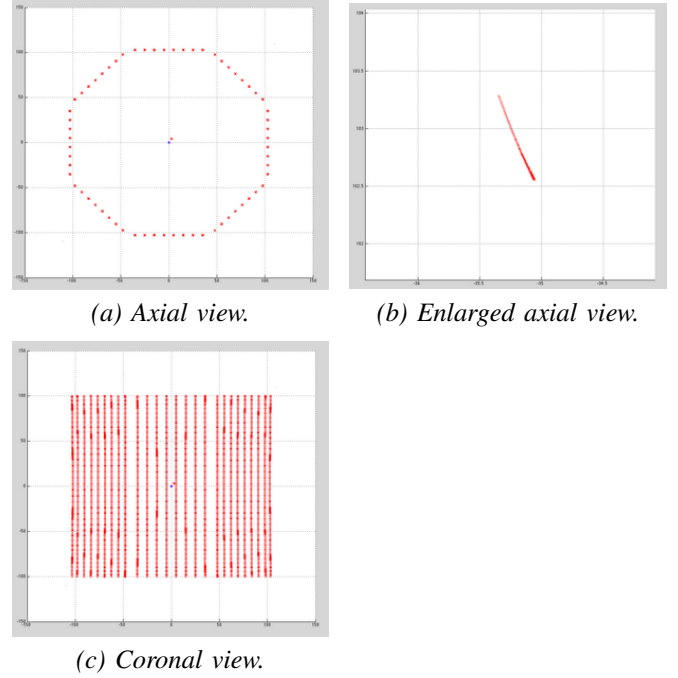


Fig. 3: Phantom modeling with small distortions

plane which isocenter is also in. The intersection of such planes from each tube should be a line that goes through isocenter. Using the isocenter estimated from previous step we should obtain an estimation of x and y coordinate.

Since the tubes were aligned to the z-axis to reduce magnetic field distortion by the tubes, the range of data in the z-direction is of necessity larger. In the x-y plane the range of points is controlled by the distortion, and is thus only a few pixels. The resulting equations are highly sensitive, requiring careful handling in our numerical algorithm.

The equation of a plane in three dimensions is as follows.

$$ax + by + cz + d = 0 \quad (6)$$

Rewriting eq 6 with the measured data, we can solve for the plane each tube lies in. This in turn can be used to find the intersection of the planes, which is the isocenter.

$$-\frac{b}{a}y - \frac{c}{a}z - \frac{d}{a} = x \quad (7)$$

$$\begin{bmatrix} y_0 & z_0 & 1 \\ \vdots & \vdots & \vdots \\ y_n & z_n & 1 \end{bmatrix} \begin{bmatrix} -b/a \\ -c/a \\ -d/a \end{bmatrix} = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}$$

$$-m_x = \frac{b}{a}$$

$$k_x = -\frac{c}{a}z_{iso} - \frac{d}{a}$$

$$x = -m_x y + k_x$$

$$\begin{bmatrix} 1 & m_x \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = k_x \quad (8)$$

Note that eq 6 can be rewritten so either x or y is independent, which affects the error in standard least squares. This becomes particularly important when the non-linearity is not the same in the x and y directions, as scaling is also well known to cause problems for least squares.

$$-\frac{a}{b}y - \frac{c}{b}z - \frac{d}{b} = y \quad (9)$$

$$\begin{bmatrix} x_0 & z_0 & 1 \\ \vdots & \vdots & \vdots \\ x_n & z_n & 1 \end{bmatrix} \begin{bmatrix} -a/b \\ -c/b \\ -d/b \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix}$$

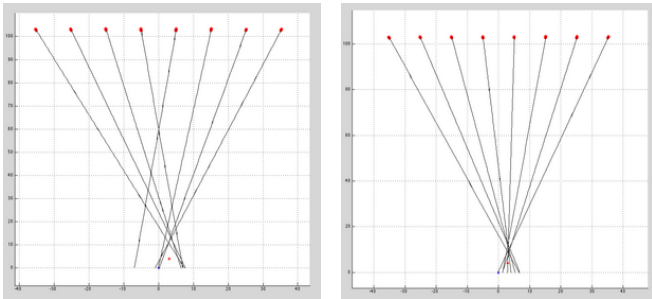
$$-m_y = \frac{a}{b}$$

$$k_y = -\frac{c}{b}z_{iso} - \frac{d}{b}$$

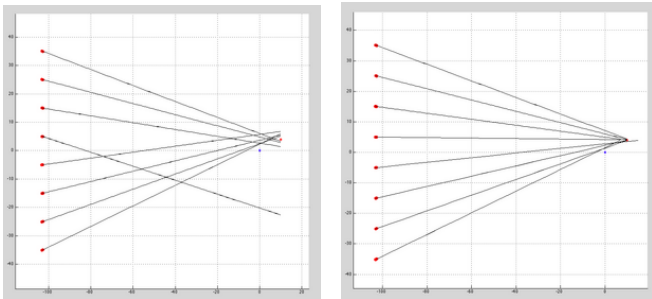
$$y = -m_y x + k_y \quad (10)$$

$$\begin{bmatrix} 1 & m_y \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = k_y \quad (11)$$

As we can see from figure 4, by using equation 7 and equation 9 to estimate planes in figure 4(b) and 4(d) respectively we get quite accurate x and y coordinate estimate. However, when you swap the choice of equations, even though they are mathematically identical, they make a huge numeric difference as shown in 4(a) and 4(b).



(a) LS fit with y orientation. (b) LS fit with x orientation.

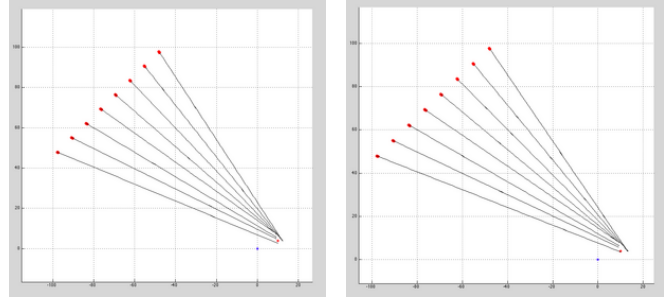


(c) LS fit with x orientation. (d) LS fit with y orientation.

Fig. 4: Phantom modeling with distortion, showing differences in isocenter estimates due to how the LS equation is oriented.

When estimating the x-y coordinate using least square we should keep in mind that least square assumes there is no

observation error, it will only try to correct one side of the equation depending on how it is setup. With this in mind, we use equation 8 for x coordinate estimation and 11 for y estimation. For the tubes on the diagonal planes, as we can see in fig5, they offer neither a good data for x nor y coordinate estimation as compared to fig 4 (b) and (d). So in order to utilize the diagonal tubes we have to rotate these tubes to either x or y plane, and get an estimation of a rotated x-y coordinate, then rotate the rotated x-y coordinate back and average it with original x-y coordinate estimation.



(a) LS using x orientation. (b) LS using y orientation.

Fig. 5: Phantom modeling with distortion, showing how diagonal tube planes without rotation do not improve the estimates produced.

In order to obtain a good estimate, we thus must separate the estimation of x and y, as well as rotate the diagonal oriented tube planes and then separate the estimation of x and y and rotate back. We refer to this algorithm as Rotated Separable Least Squares (RSLs). We now present the RSLs algorithm to estimate x-y coordinate of gradient isocenter as follows:

- 1) Use equation 7 to estimate tube planes for tubes at upper and lower surfaces.
- 2) Solve equation 8 for x and y, but only use x for x coordinate.
- 3) Use equation 9 to estimate tube planes for tubes at left and right surfaces.
- 4) Solve equation 11 for y and x, but only use y for y coordinate.
- 5) Rotate diagonal tubes  $\pi/4$ , and repeat steps 1-4.
- 6) Rotate x-y coordinate obtained from previous step by  $-\pi/4$ .
- 7) Average the x-y coordinate from previous step with x-y coordinate calculated from step 2 and step 4 for final x-y estimation.

Alternatively, after obtaining the plane equation parameters we can put everything into one matrix and do a one time estimation using either least square or total least square. In table 1, we can see the comparison of accuracy of estimating an isocenter at [4 4 3] using different methods. Least square tends to lean toward one coordinate more depends on setup,

	$\bar{x}$	$\sigma_x$	err	$\bar{y}$	$\sigma_y$	err
RLS	3.9709	0.1068	0.0291	3.9677	0.1008	0.0323
LS	3.9950	0.1118	0.0050	3.9414	0.1011	0.0586
TLS	4.0800	0.1035	0.0800	4.0800	0.0964	0.0800

Table 1: Average of 200 runs using different methods for estimating the isocenter at [4 4 3] in the presence of symmetrical distortion

	$\bar{x}$	$\sigma_x$	err	$\bar{y}$	$\sigma_y$	err
RLS	3.8073	0.1117	0.1927	3.9838	0.1059	0.0162
LS	4.2191	0.1117	0.2191	3.8765	0.1003	0.1235
TLS	9.5744	0.2616	5.5744	8.6905	0.2393	4.6905

Table 2: Average of 200 runs using different methods for estimating the isocenter at [4 4 3] in the presence of non-symmetrical distortion

while total least square has an accurate and very balanced result due to the property that it will try to correct errors on both side of the equation. In the contrast, our method has best properties of both methods:

- It is accurate. For both x and y coordinate it does a better job than total least square, much better than least square's worst case and very close to least square's best case.
- It has very balanced result. Both x and y coordinates are very close the correct result equally just like total least square.
- It has very tight error boundaries. After 200 runs, it's error is tighter than standard least square.

Therefore, our estimation method is a good alternative to traditional least square or total least square methods. Although it might require more computation, it could be easily dealt by modern GPU computing. And due to the fact that each least square estimation has relative small matrix, and each estimation is independent of each other, it is very close to "embarrassingly parallel" type of problems and makes it easy to solve.

In table 1, the test is run using a symmetric x-y axis distortion. We can see that all three methods performed very well. RLS method's result is slightly better than Total Least Squares (TLS), and one on y axis it is better than standard Least Squares method.

When the distortion is not symmetrical the issue of proper estimation becomes crucial. In table 2, we show the result of using non-symmetrical distortion in which the y direction was set to be twice the distortion in the x. In this test, TLS method's result is the worst, off by a few millimeters. Both LS and RLS show comparable degradation of performance in x. LS shows degradation in the estimation of y as well, but RLS, since it is separable, does not experience any degradation of it. Only the RLS's result remains quite accurate. This shows the advantage of RLS over the other two traditional methods.

## 4. Conclusion

In this work, we developed a new numerical algorithm to accurately determine the gradient isocenter of MRI scanners based on a new distortion correction phantom. Knowledge of the gradient isocenter is an essential part of gradient-nonlinearity correction methods. We have shown that the method used in this work to estimate the gradient isocenter is a good alternative to more traditional estimation methods such as least squares and total least squares. The new algorithm is particularly suited when the image data are extremely sensitive to the presence of noise and asymmetric distortion. Using simulated (but realistic) distortion data, it was shown, that the resulting estimated isocenter was within 0.2 mm of the actual gradient isocenter, leading to a better estimate than the currently used DICOM coordinate center.

## References

- [1] Doran, S., Charles-Edwards, L., Reinsberg, S., Leach, M.: A complete distortion correction for mr images: I. gradient warp correction. *Phys. Med. Biol.* **50**, 1343–1361 (2005)
- [2] Kondziolka, D.: Functional radiosurgery. *Neurosurgery* **44**, 12–20 (1999)
- [3] Lee, T., Schubert, K., Schulte, R.: Gradient Non-Linearity Correction of MR Images for Functional Radiosurgery. In: ICIS-COMSAR '06: Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse, pp. 338 – 344. IEEE Computer Society, Honolulu, Hawaii, USA (2006). DOI <http://dx.doi.org/10.1109/ICIS-COMSAR.2006.43>
- [4] Lee, T., Schubert, K., Schulte, R.: Software Development For Correction Of Gradient-Nonlinearity Distortions In MR Images. In: P. Dey, M. Amin, T. Gattton (eds.) Proceedings of the International Conference on Computer Science and its Applications, pp. 338 – 344. US Education Service, San Diego (2006)
- [5] Lee, T., Schubert, K., Schulte, R.: Computational Algorithm for Modeling and Correction of Gradient Nonlinearity Distortions in Magnetic Resonance Imaging. In: Proceeding of the World Conference of Engineering and Computer Science, pp. 210–215 (2008)
- [6] Lee, T., Schubert, K., Schulte, R.: Software-Based Algorithm for Modeling and Correction of Gradient Nonlinearity Distortions in Magnetic Resonance Imaging. *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, WCECS '08. I*, 52–61 (2008)
- [7] S. Langlois M. Desvignes, J.C., Revenu, M.: Mri geometric distortion: A simple approach to correcting the effects of non-linear gradient fields. *J. Magn. Reson. Imaging* **9**, 821–831 (1999)
- [8] Wang, D., Strugnell, W., Cowin, G., Doddrell, D., Slaughter, R.: Geometric distortion in clinical mri systems part i: Evaluation using a 3d phantom. *J. Magn. Reson. Imaging* **22**, 1211–1221 (2004)
- [9] Wang, D., Strugnell, W., Cowin, G., Doddrell, D., Slaughter, R.: Geometric distortion in clinical mri systems part ii: Correction using a 3d phantom. *J. Magn. Reson. Imaging* **22**, 1223–1232 (2004)

# Angular power spectrum of scattered radiation in ionospheric plasma with both electron density and magnetic field fluctuations

G. Jandieri<sup>1</sup>, N. Zhukova<sup>2</sup>, Zh. Diasamidze<sup>1</sup>, and M. Diasamidze<sup>1</sup>

<sup>1</sup>Department of Physics, Georgian Technical University, Tbilisi, Georgia

<sup>2</sup>Dynamics of Geophysical fields and Computing Geophysics, TSU M. Nodia Institute of Geophysics, Tbilisi, Georgia

**Abstract** - *The influence of anisotropy of both electron density and external magnetic field fluctuations on the spatial power spectrum (SPS) of scattered electromagnetic waves is considered in this paper. Stochastic differential equation is obtained for the phase fluctuations using smooth perturbation method taking into account diffraction effects. Second order statistical moments are calculated for arbitrary correlation functions of electron density and external magnetic field fluctuations. Numerical calculations were carried out for anisotropic Gaussian correlation function containing nondimensional anisotropic parameter and the angle of inclination of prolate irregularities with respect to the external magnetic field. SPS of scattered radiation has a pronounced gap caused by electron density fluctuations. The influence of an external magnetic field on a double-peaked shape has been analytically and numerically.*

**Keywords:** Ionospheric plasma, Anisotropy, Phase fluctuations, Angular power spectrum

## 1 Introduction

Peculiarities of the electromagnetic waves propagation in randomly inhomogeneous media have been intensively studied [1,2]. However, the large-scale irregularities were considered to be statistically isotropic. In many cases irregularities are anisotropic. Particularly, they are observed in lyotropic crystals with a hexagonal structure [3], in the Earth's ionosphere random plasma inhomogeneities are aligned with the geomagnetic fields [4]. The evolution of the angular distribution of the intensity at light propagation in a randomly unhomogeneous medium with strongly prolated anisotropic irregularities of dielectric permittivity has been investigated in [5,6]. Using the smooth perturbation method it has been shown that the spatial power spectrum (SPS) of multiply scattered waves at oblique illumination of a boundary of a randomly inhomogeneous medium with prolate irregularities by mono-directed incident radiation has a double-peaked shape. Numerical simulation has been carried out by Monte-Carlo method. Second order statistical moments of the SPS in magnetized anisotropic plasma have

been investigated in the complex geometrical optics approximation and perturbation method [7-10].

The features of the SPS of multiply scattered radiation in a randomly inhomogeneous anisotropic ionospheric plasma are investigated analytically and numerically taking into account diffraction effects caused by both electron density and external magnetic field fluctuations. The expressions for phase fluctuations of scattered electromagnetic waves in the principle (wave vector of mono-directed incident radiation and external magnetic field are located in this plane) and perpendicular planes are derived using the smooth perturbation method. Correlation functions of the phase fluctuations are calculated for arbitrary correlation functions of fluctuating magnetized plasma parameters. The influence of an external magnetic field on a gap caused due to electron density fluctuations in the ionospheric plasma is considered for the first time in this paper. Numerical calculations are carried out using satellite and remote sensing data.

## 2 Formulation of the problem

Let us consider the features of the SPS of scattered electromagnetic waves in the anisotropic ionospheric magnetized plasma with both electron density and external magnetic field fluctuations. Initial is the following wave equation:

$$\left( \frac{\partial^2}{\partial x_i \partial x_j} - \Delta \delta_{ij} - k_0^2 \varepsilon_{ij}(\mathbf{r}) \right) \mathbf{E}_j(\mathbf{r}) = 0. \quad (1)$$

Wave field we introduce as  $E_j(\mathbf{r}) = E_{0j} \exp(\varphi_1 + \varphi_2 + i k_{\perp} y + i k_0 z)$  ( $k_{\perp} \ll k_0$ ). If electromagnetic wave propagates along  $z$  axis and the vector of an external magnetic field lies in the coordinate plane ( $\mathbf{k} \parallel z$ ,  $\langle \mathbf{H}_0 \rangle \in yz$ ), components of the second-rank tensor of collisionless magnetized plasma have the following form [11]:

$$\begin{aligned}\varepsilon_{xx} &= 1 - \frac{v}{1-u}, & \varepsilon_{yy} &= 1 - \frac{v(1-u \sin^2 \alpha)}{1-u}, \\ \varepsilon_{zz} &= 1 - \frac{v(1-u \cos^2 \alpha)}{1-u}, & \varepsilon_{xy} &= -\varepsilon_{yx} = i \frac{v \sqrt{u} \cos \alpha}{1-u}, \\ \varepsilon_{yz} &= \varepsilon_{zy} = \frac{u v \sin \alpha \cos \alpha}{1-u}, & \varepsilon_{xz} &= -\varepsilon_{zx} = -i \frac{v \sqrt{u} \sin \alpha}{1-u}\end{aligned}\quad (2)$$

where  $\alpha$  is the angle between the vectors  $\mathbf{k}$  and  $\mathbf{H}_0$ ;  $\varepsilon_{xy} = i \tilde{\varepsilon}_{xy}$ ,  $\varepsilon_{xz} = -i \tilde{\varepsilon}_{xz}$ ,  $u = (e H_0 / m c \omega)^2$ ,  $v = \omega_p^2 / \omega^2$  are the magneto-ionic parameters,  $\omega_p = (4 \pi N e^2 / m)^{1/2}$  is the plasma frequency,  $\Omega_H = e H_0 / m c$  is the electron gyrofrequency. Dielectric permittivity of a turbulent magnetized plasma is a second rank tensor, which is random function of a spatial coordinates  $\varepsilon_{ij}(\mathbf{r}) = \varepsilon_{ij}^{(0)} + \varepsilon_{ij}^{(1)}(\mathbf{r})$ ,  $|\varepsilon_{ij}^{(1)}(\mathbf{r})| \ll 1$ . First component represents zero-order approximation, second one contains fluctuations of both electron density and external magnetic field fluctuations of the ionospheric plasma which are random functions of the spatial coordinates:  $\mathbf{v}(\mathbf{r}) = v_0 [1 + n_1(\mathbf{r})]$ ,  $\mathbf{u}(\mathbf{r}) = u_0 [1 + 2h_1(\mathbf{r})]$ .

In a zero-order approximation we have the following wave equation

$$\left[ \frac{\partial^2 \varphi_0}{\partial x_i \partial x_j} + \frac{\partial \varphi_0}{\partial x_i} \frac{\partial \varphi_0}{\partial x_j} + (k_{\perp}^2 + k_0^2) \delta_{ij} - k_0^2 \varepsilon_{ij}^{(0)} \right] E_{0j} = 0, \quad (3)$$

containing the set of three algebraic equations for the  $E_{0j}$  regular field components:

$$\begin{aligned}\left( \varepsilon_{xx}^{(0)} - 1 - \frac{k_{\perp}^2}{k_0^2} \right) E_{0x} + i \tilde{\varepsilon}_{xy}^{(0)} E_{0y} - i \tilde{\varepsilon}_{xz}^{(0)} E_{0z} &= 0, \\ i \tilde{\varepsilon}_{xy}^{(0)} E_{0x} + (1 - \varepsilon_{yy}^{(0)}) E_{0y} - \left( \frac{k_{\perp}}{k_0} + \varepsilon_{yz}^{(0)} \right) E_{0z} &= 0, \\ i \tilde{\varepsilon}_{xz}^{(0)} E_{0x} + \left( \varepsilon_{yz}^{(0)} + \frac{k_{\perp}}{k_0} \right) E_{0y} - \left( \frac{k_{\perp}^2}{k_0^2} - \varepsilon_{zz}^{(0)} \right) E_{0z} &= 0.\end{aligned}\quad (4)$$

Solution of determinant imposes the restriction on the parameter  $\mu = k_{\perp} / k_0$ :

$$\begin{aligned}(2 - \varepsilon_{yy}) \mu^4 + 2 \varepsilon_{yz} \mu^3 + (2 - 2 \varepsilon_{xx} - \varepsilon_{yy} - \varepsilon_{zz} + \varepsilon_{xx} \varepsilon_{yy} + \\ + \varepsilon_{yy} \varepsilon_{zz} - \varepsilon_{yz}^2 - \tilde{\varepsilon}_{xy}^2) \mu^2 + 2 \left[ \tilde{\varepsilon}_{xy} \tilde{\varepsilon}_{xz} + \varepsilon_{yz} (1 - \varepsilon_{xx}) \right] \mu + \\ + \left[ \varepsilon_{zz} (\varepsilon_{xx} - \varepsilon_{xx} \varepsilon_{yy} - 1 + \varepsilon_{yy}) + 2 \tilde{\varepsilon}_{xy} \tilde{\varepsilon}_{xz} \varepsilon_{yz} + \tilde{\varepsilon}_{xz}^2 (\varepsilon_{yy} - 1) + \right.\end{aligned}$$

$$\left. + \varepsilon_{yz}^2 (\varepsilon_{xx} - 1) + \tilde{\varepsilon}_{xy}^2 \varepsilon_{zz} \right] = 0. \quad (5)$$

Taking into account that fluctuations of the complex phase are of the order  $\varphi_1 \sim \varepsilon_{ij}^{(1)}$ ,  $\varphi_2 \sim \varepsilon_{ij}^{(1)2}$  and the well known conditions characterizing the smooth perturbation method:

$$\begin{aligned}\left| \frac{\partial \varphi_1}{\partial z} \right| \ll k_0 |\varphi_1|, & \quad \left| \frac{\partial^2 \varphi_1}{\partial z^2} \right| \ll k_0 \left| \frac{\partial \varphi_1}{\partial z} \right|, \\ \left| \frac{\partial \varphi_2}{\partial z} \right| \ll k_0 |\varphi_2|, & \quad \left| \frac{\partial^2 \varphi_2}{\partial z^2} \right| \ll k_0 \left| \frac{\partial \varphi_2}{\partial z} \right|,\end{aligned}$$

in the first approximation we obtain:

$$\begin{aligned}\left[ \frac{\partial^2 \varphi_1}{\partial x_i \partial x_j} + \frac{\partial \varphi_0}{\partial x_i} \frac{\partial \varphi_1}{\partial x_j} + \frac{\partial \varphi_1}{\partial x_i} \frac{\partial \varphi_0}{\partial x_j} - \delta_{ij} \left( \Delta_{\perp} + 2 i k_{\perp} \frac{\partial \varphi_1}{\partial y} + \right. \right. \\ \left. \left. + 2 i k_0 \frac{\partial \varphi_1}{\partial z} \right) - k_0^2 \varepsilon_{ij}^{(0)} \right] E_{0j} = 0.\end{aligned}\quad (6)$$

where  $\Delta_{\perp} = (\partial^2 \varphi_1 / \partial x^2) + (\partial^2 \varphi_1 / \partial y^2)$  is the transversal Laplacian.

Two-dimensional Fourier transformation for the phase fluctuations is

$$\varphi_1(x, y, z) = \int_{-\infty}^{\infty} d k_x \int_{-\infty}^{\infty} d k_y \psi(k_x, k_y, z) \exp(i k_x x + i k_y y),$$

For  $i = x$  component from equation (6) we obtain differential equation for two-dimensional spectral component:

$$\begin{aligned}\frac{\partial \psi}{\partial z} + \frac{i}{k_x \frac{E_{0z}}{E_{0x}} - 2 k_0} \left[ k_x (k_y + k_{\perp}) \frac{E_{0y}}{E_{0x}} + k_x k_0 \frac{E_{0z}}{E_{0x}} - \right. \\ \left. - k_y (k_y + 2 k_{\perp}) \right] \psi = -i \frac{k_0^2}{k_x \frac{E_{0z}}{E_{0x}} - 2 k_0}.\end{aligned}\quad (7)$$

The relations of the mean electric field components are determined by the well-known formulae [11] ( $E_{0y} / E_{0x} = i P_j$ ,  $(E_{0z} / E_{0x}) = i \Gamma_j$ , minus sign and index  $j = 1$  correspond to the extraordinary wave, plus sign and index  $j = 2$  - to the ordinary wave; the polarization coefficients are [11]:

$$\begin{aligned}P_j &= \frac{2 \sqrt{u} (1-v) \cos \alpha}{u \sin^2 \alpha \pm \sqrt{u^2 \sin^4 \alpha + 4 u (1-v)^2 \cos^2 \alpha}} \\ \Gamma_j &= -\frac{v \sqrt{u} \sin \alpha + P_j u v \sin \alpha \cos \alpha}{1-u-v+u v \cos^2 \alpha},\end{aligned}\quad (8)$$

In general, ordinary and extraordinary waves in collisionless magnetized plasma are elliptically polarized.

Transverse correlation function of a scattered field has the following form [6]  $W_{EE^*}(\mathbf{p}) = \langle E(\mathbf{r})E^*(\mathbf{r} + \mathbf{p}) \rangle$  taking into account that the observation points are spaced apart at a small distance  $\mathbf{p} = \{\rho_x, \rho_y\}$ :

$$W_{EE^*}(\mathbf{p}, k_{\perp}) = E_0^2 \exp(-i\rho_y k_{\perp}) \exp \left\{ \text{Re} \left[ \frac{1}{2} \left( \langle \varphi_1^2(\mathbf{r}) \rangle + \langle \varphi_1^{2*}(\mathbf{r} + \mathbf{p}) \rangle + \langle \varphi_1(\mathbf{r}) \varphi_1^*(\mathbf{r} + \mathbf{p}) \rangle + 2 \langle \varphi_2 \rangle \right) \right] \right\}, \quad (9)$$

where  $E_0^2$  is the intensity of an incident radiation.

SPS of a scattered field in case of incident plane wave  $W(k', k_{\perp})$  is easily calculated by Fourier transform of the transversal correlation function [1,2].

$$W(k', k_{\perp}) = \int_{-\infty}^{\infty} d\rho_y W_{EE^*}(\rho_y, k_{\perp}) \exp(ik'\rho_y). \quad (10)$$

## 2.1 Second order statistical moments of the phase fluctuations

In these notations two-dimensional spectral component of the phase fluctuation of scattered electromagnetic field (7) in the first approximation satisfies the stochastic differential equation:

$$\frac{\partial \psi}{\partial z} + \frac{i d_1 - d_2}{\Gamma_j k_x + 2i k_0} \psi(k_x, k_y, z) = -\frac{k_0^2}{\Gamma_j k_x + 2i k_0} \left\{ \varepsilon_{xx}^{(1)}(k_x, k_y, z) - \left[ P_j \tilde{\varepsilon}_{xy}^{(1)}(k_x, k_y, z) - \Gamma_j \tilde{\varepsilon}_{xz}^{(1)}(k_x, k_y, z) \right] \right\} \quad (11)$$

where:  $d_1 = k_x(k_y + k_{\perp})P_j + k_0 k_x \Gamma_j$ ,  $d_2 = k_y(k_y + 2k_{\perp})$ .

The solution of this equation satisfying the boundary condition has the following form

$$\psi(k_x, k_y, z) = i \frac{k_0}{2} \int_{-\infty}^{\infty} dz' \left\{ \varepsilon_{xx}^{(1)}(k_x, k_y, z') - \left[ P_j \tilde{\varepsilon}_{xy}^{(1)}(k_x, k_y, z') - \Gamma_j \tilde{\varepsilon}_{xz}^{(1)}(k_x, k_y, z') \right] \right\} \cdot \exp \left[ -\frac{d_2 - i d_1}{\Gamma_j k_x + 2i k_0} (L - z') \right]. \quad (12)$$

Taking into account that:  $\langle T_{\alpha\beta}(\mathbf{k}, z') T_{\gamma\delta}(\mathbf{k}', z'') \rangle = W_{\alpha\beta, \gamma\delta}(\mathbf{k}, z' - z'') \delta(\mathbf{k} + \mathbf{k}')$  and changing the variables:

$z' - z'' = \rho_z$ ,  $z' + z'' = 2\eta$ , second order statistical moments of phase fluctuations of scattered electromagnetic waves for arbitrary correlation function of electron density fluctuations are finally expressed as:

$$\langle \varphi_1^2(\mathbf{r}) \rangle = \frac{\pi k_0^2}{2} \int_{-\infty}^{\infty} dk_x \int_{-\infty}^{\infty} dk_y \frac{G_1 + i G_2}{G_1^2 + G_2^2} \left[ V_{xx,xx} + V_{xy,xy} + V_{xz,xz} + 2(V_{xx,xz} - V_{xx,xy} - V_{xy,xz}) \right] \left\{ 1 - \exp[(G_1 - i G_2)L] \right\} \quad (13)$$

$$\langle \varphi_1^{*2}(\mathbf{r} + \mathbf{p}) \rangle = \frac{\pi k_0^2}{2} \int_{-\infty}^{\infty} dk_x \int_{-\infty}^{\infty} dk_y \frac{G_1 - i G_2}{G_1^2 + G_2^2} \left[ V'_{xx,xx} + V'_{xy,xy} + V'_{xz,xz} + 2(V'_{xx,xz} - V'_{xx,xy} - V'_{xy,xz}) \right] \cdot \left\{ 1 - \exp[(G_1 + i G_2)L] \right\}, \quad (14)$$

$$\langle \varphi_1(\mathbf{r}) \varphi_1^*(\mathbf{r} + \mathbf{p}) \rangle = \frac{\pi k_0^2 L}{2} \int_{-\infty}^{\infty} dk_x \int_{-\infty}^{\infty} dk_y \left[ V''_{xx,xx} + V''_{xy,xy} + V''_{xz,xz} + 2(V''_{xx,xz} - V''_{xx,xy} - V''_{xy,xz}) \right] \exp(-ik_x \rho_x - ik_y \rho_y) \quad (15)$$

where:  $V_{\alpha\beta, \gamma\delta} \equiv V_{\alpha\beta, \gamma\delta}(k_x, k_y, i G_3 - G_4)$ ,  $V''_{\alpha\beta, \gamma\delta} \equiv V''_{\alpha\beta, \gamma\delta}(k_x, k_y, -(d_1 \Gamma_j k_x + 2d_2 k_0) / 4k_0^2)$ ,  $V'_{\alpha\beta, \gamma\delta} \equiv V'_{\alpha\beta, \gamma\delta}(k_x, k_y, -i G_3 - G_4)$ , indices denote the product of fluctuating terms of the second rank tensors;

$$G_1 = \frac{1}{k_0^2} (\Gamma_j k_{\perp} - P_j k_0) k_x k_y, \\ G_2 = \frac{1}{2k_0^2} \left[ \Gamma_j (P_j k_{\perp} + \Gamma_j k_0) k_x^2 + 2k_0 k_y^2 \right], \\ G_3 = \frac{1}{4k_0^2} (2\Gamma_j k_0^2 + 2P_j k_0 k_{\perp} - \Gamma_j k_y^2) k_x, \\ G_4 = \frac{1}{4k_0^2} (P_j \Gamma_j k_x^2 + 4k_0 k_{\perp}) k_y.$$

For  $i = x$  component from equation (1) we obtain stochastic differential equation in the second approximation

$$i P_j \frac{\partial^2 \varphi_2}{\partial x \partial y} + i \Gamma_j \frac{\partial^2 \varphi_2}{\partial x \partial z} - (k_{\perp} P_j + k_0 \Gamma_j) \frac{\partial \varphi_2}{\partial x} - \frac{\partial^2 \varphi_2}{\partial y^2} - 2i k_{\perp} \frac{\partial \varphi_2}{\partial y} - 2i k_0 \frac{\partial \varphi_2}{\partial z} = -i P_j \frac{\partial \varphi_1}{\partial x} \frac{\partial \varphi_1}{\partial y} + \left( \frac{\partial \varphi_1}{\partial y} \right)^2, \quad (16)$$

By Fourier transform first term of the right part of equation (16) can be written as:

$$\begin{aligned} \langle \frac{\partial \varphi_1}{\partial x} \frac{\partial \varphi_1}{\partial y} \rangle &= \frac{\pi k_0^2}{4} \int_{-\infty}^{\infty} dk_x \int_{-\infty}^{\infty} dk_y \frac{k_x k_y}{G_1^2 + G_2^2} [V_{xx,xx} + \\ &+ V_{xy,xy} + V_{xz,xz} + 2(V_{xx,xz} - V_{xx,xy} - V_{xy,xz})] \\ &\cdot \{ 1 - \exp[(G_1 - i G_2)L] \}. \end{aligned} \quad (17)$$

Solution of equation (16) is expressed as:

$$\begin{aligned} \text{Re} \langle \varphi_2(\mathbf{r}) \rangle &= \text{Re} \frac{\pi k_0}{4} \int_{-\infty}^{\infty} dk_x \int_{-\infty}^{\infty} dk_y \frac{1}{G_1^2 + G_2^2} \cdot \\ &\cdot \left\{ L(A_1 + i B_1) + \frac{1}{G_1^2 + G_2^2} [(A_1 G_1 - B_1 G_2) + \right. \\ &\left. + i(B_1 G_1 + A_1 G_2)] - \frac{1}{G_1^2 + G_2^2} (A_2 + i B_2) \exp(G_1 L) \right\} \cdot \\ &[V_{xx,xx} + V_{xy,xy} + V_{xz,xz} + 2(V_{xx,xz} - V_{xx,xy} - V_{xy,xz})] \end{aligned} \quad (18)$$

where:  $A_1 = -(P_j G_1 k_x k_y + G_2 k_y^2)$ ,  $A_2 = (A_1 G_1 - B_1 G_2) \cdot \cos(G_2 L) + (B_1 G_1 + A_1 G_2) \sin(G_2 L)$ ,  $B_1 = G_1 k_y^2 - P_j G_2 k_x k_y$ ,  $B_2 = (B_1 G_1 + A_1 G_2) \cos(G_2 L) - (A_1 G_1 - B_1 G_2) \sin(G_2 L)$ .

In the absence of an external magnetic field ( $H_0 = 0$ ,  $u_0 = 0$ ), from equation (8) follows:  $P_j = \Gamma_j = 0$ ,  $d_1 = 0$ , and (12)-(15) coincide with [6].

## 2.2 Numerical calculations

In analytical and numerical calculations we will use anisotropic Gaussian correlation function of electron density fluctuation [12] for investigation of the influence of electron density and external magnetic field fluctuations on evolution of the SPS

$$\begin{aligned} W_n(k_x, k_y, k_z) &= \sigma_n^2 \frac{l_{\perp}^2 l_{\parallel}}{8\pi^{3/2}} \exp \left( -\frac{k_x^2 l_{\perp}^2}{4} - p_1 \frac{k_y^2 \bar{l}^2}{4} - \right. \\ &\left. - p_2 \frac{k_z^2 l_{\parallel}^2}{4} - p_3 k_y k_z l_{\parallel}^2 \right). \end{aligned} \quad (19)$$

This function is characterized by anisotropy factor of irregularities  $\chi = l_{\parallel} / l_{\perp}$  (ratio of longitudinal and transverse linear scales of plasma irregularities with respect to the external magnetic field) and the inclination angle of prolate

irregularities with respect to the external magnetic field  $\gamma_0$ .

$$p_1 = 1 + (1 - \chi^2)^2 \sin^2 \gamma_0 \cos^2 \gamma_0 / \chi^2, \quad p_2 = (\sin^2 \gamma_0 + \chi^2 \cos^2 \gamma_0) / \chi^2, \quad p_3 = (1 - \chi^2) \sin \gamma_0 \cos \gamma_0 / 2 \chi^2,$$

$\bar{l} = l_{\parallel} (\sin^2 \gamma_0 + \chi^2 \cos^2 \gamma_0)^{-1/2}$ . In isotropic case ( $\chi = 1$ )

we have:  $p_1 = p_2 = 1$ ,  $p_3 = 0$ ; at  $\gamma_0 = 0^0$ :  $p_1 = 1 / \chi^2$ ,

$$p_2 = 1, \quad p_3 = 0.$$

We investigate the influence of electron density fluctuations on the SPS of scattered radiation in turbulent magnetized plasma ( $H_0 \neq 0$ ). At  $T = k_0 l_{\parallel} \gg 1$  using the saddle point method, we obtain:

$$\begin{aligned} \langle \varphi_1^2(\mathbf{r}) \rangle + \langle \varphi_1^*(\mathbf{r} + \boldsymbol{\rho}) \rangle &= -\frac{\sigma_n^2 \Omega_1 T^2}{8\sqrt{2} \chi^2} \\ &\int_{-\infty}^{\infty} ds \frac{1}{s^2 \sqrt{b_1}} \sin(s^2 k_0 L) \exp(-T^2 b_2), \\ \langle \varphi_1(\mathbf{r}) \varphi_1^*(\mathbf{r} + \boldsymbol{\rho}) \rangle &= \frac{\sigma_n^2 \Omega_1 T^2 k_0 L}{16 \chi^2} \int_{-\infty}^{\infty} ds \frac{1}{\sqrt{b_6}} \\ &\exp \left\{ -\frac{T^2}{4} \left[ p_1 s^2 + \frac{1}{4} p_2 (s^2 + 2s\mu)^2 + 2p_3 (s^3 + 2s^2 \mu) \right] \right. \\ &\left. - i\eta s \right\}, \\ \text{Re} \langle \varphi_2(\mathbf{r}) \rangle &= -\frac{\sigma_n^2 \Omega_1 T^2 k_0 L}{32 \chi^2} \int_{-\infty}^{\infty} ds \frac{1}{\sqrt{b_1}} \\ &\cdot \left[ 1 - \frac{1}{k_0 L s^2} \sin(k_0 L s^2) \right] \exp(-T^2 b_2), \end{aligned} \quad (20)$$

where:  $b_1 = \frac{1}{4\chi^2} - \frac{1}{4} p_2 (m_3^2 - 2m_4 \mu s) + p_3 m_4 s$ ,  $x = \frac{k_x}{k_0}$ ,

$$\Omega_1 = \frac{v_0}{(1 - u_0)^2} [1 + u_0 - 2\sqrt{u_0} (\sin \alpha - \cos \alpha + \sqrt{u_0} \sin \alpha \cos \alpha)]$$

$$s = \frac{k_y}{k_0}, \quad b_6 = \frac{1}{4} \left( \frac{1}{\chi^2} + 2p_2 m_5 m_6 + 4p_3 m_5 s \right), \quad \eta = k_0 \rho_y,$$

$$B_0 = \sqrt{\pi} \sigma_n^2 v_0^2 T k_0 L / 4, \quad m_3 = \frac{1}{4} (2\Gamma_j + 2P_j \mu - \Gamma_j s^2),$$

$$m_4 = \frac{1}{4} P_j \Gamma_j, \quad m_5 = \frac{1}{4} [(s + \mu) P_j + \Gamma_j] \Gamma_j,$$

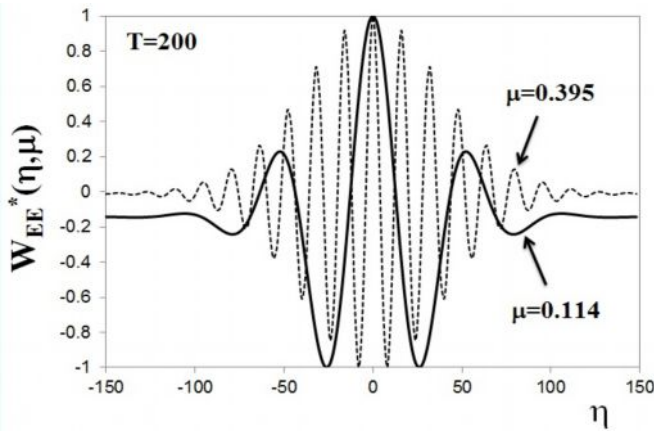
$$m_6 = \frac{1}{2} (s^2 + 2s\mu).$$



Numerical calculations were carried out for 0.1 MHz and 40 MHz at  $\alpha = 15^\circ$ . The solution of the dispersion equation (5) yields the roots:  $\mu = 0.395$ , for 0.1 MHz and  $\mu = 0.114$ , for 40 MHz.

The curves in Figure 1 illustrate the dependence of normalized correlation function of scattered electromagnetic field versus non-dimensional parameter  $\eta$  for  $T = 200$  and are normalized on their maximum value. Second maxima on the solid and dotted lines correspond  $\eta = 52$  and  $\eta = 16$ , respectively. Next maxima at 0.1 MHz appear at  $\eta = 32, 48, 64$  (periodical oscillations). Increasing parameter  $\eta$  normalized correlation functions rapidly attenuates.

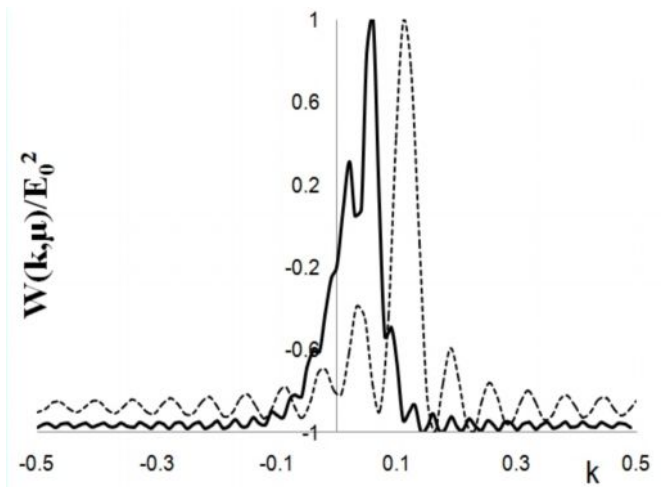
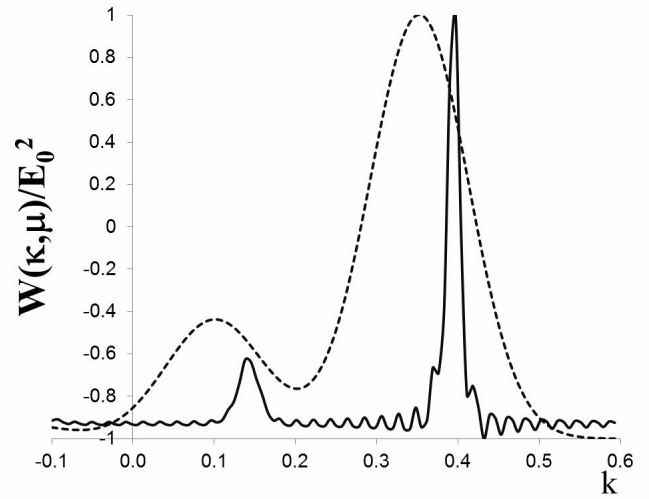
Figure 2 presents the dependence of the SPS of scattered field versus nondimensional parameter  $k$ . Numerical calculations show that for 0.1 MHz (left figure), at  $H_0 = 0$ , the gap arises due to electron density fluctuations. First and second maxima correspond  $k = 0.355$  and  $k = 0.1$ , respectively; gap appears at  $k = 0.194$ . In magnetized turbulent plasma ( $H_0 \neq 0$ ) two pronounced maxima arise at  $k = 0.38$  and  $k = 0.14$ ; and two gaps at  $k = 0.37$  and  $k = 0.42$ . For 40 MHz (right figure) at  $H_0 = 0$  first maximum arise at  $k = 0.11$ , and next two maxima at  $k = 0.033$  and  $k = 0.189$ . First two gaps appear at  $k = 0.072$  and  $k = 0.163$ ; next two gaps at  $k = 0.007$  and  $k = 0.228$ .



**Figure 1.** Dependence of normalized correlation function of scattered field  $W_{EE^*}(\eta, \mu)$  versus distance between two observation points  $\eta = k_0 \rho_y$  at different values of the parameter  $\mu$ . Dotted line corresponds 0.1 MHz, solid line 40 MHz.

At  $H_0 \neq 0$  first and other two maxima arise at  $k = 0.06$  and  $k = 0.02$ ,  $k = 0.08$ , respectively. First gap appears at  $k = 0.03$ , second one at  $k = 0.795$ .

It should be emphasized that “double-hump” shape of the SPS caused by electron density fluctuations in turbulent plasma without external magnetic field are more pronounced than in magnetized plasma at  $H_0 \neq 0$ . Numerical analyses show that neglecting diffraction effects, i.e. neglecting the term  $k_y^2 / 2k_0^2$  in the arguments of 2D spectrum (13)-(15) or in set of equations (20), “double-humping” effect in the SPS disappears.



**Figure 2.** Dependence of SPS (10) versus  $k$ . Left figure corresponds to 0.1 MHz at:  $T = 2500$ ,  $\mu = 0.395$ ,  $\chi = 130$ ,  $\gamma_0 = 15^\circ$ ,  $\xi = 1$ ,  $k_0 = 2.8 \text{ km}^{-1}$ ,  $B_0 = 6$ . Right figure corresponds 40 MHz at:  $T = 500$ ,  $\mu = 0.114$ ,  $\chi = 150$ ,  $\gamma_0 = 5^\circ$ ,  $\xi = 1$ ,  $k_0 = 840 \text{ km}^{-1}$ ,  $B_0 = 4$ . Dotted line denotes  $H_0 = 0$ , solid line  $H_0 \neq 0$ .

### 3 Conclusions

Numerical calculations show that for anisotropic Gaussian correlation function second-order statistical moments are nonlinear functions of wave vectors. For electron density fluctuations spatial power spectrum has a pronounced gap along a direction of wave propagation and a double-peaked shape. However, external magnetic field fluctuations can lead to the generation of different nonlinear effects. On the basis of the proposed theory some observable nonlinear effects of the ionospheric plasma can be interpreted and (or) predicted using different correlation functions of fluctuating magnetized plasma parameters taking into account satellite and remote sensing data.

### 4 References

- [1] Ishimaru, A. *Wave Propagation and Scattering in Random Media, Vol. 2, Multiple Scattering, Turbulence, Rough Surfaces and Remote Sensing*, IEEE Press, Piscataway, New Jersey, USA, 1997.
- [2] Rytov S.M., Kravtsov Yu.A., Tatarskii V.I., Principles of Statistical Radiophysics. vol.4. Waves Propagation Through Random Media. Berlin, New York, Springer, 1989.
- [3] Brown, G.H., Wolken J.J., Liquid Crystals and Biological Structures, New York, 1979.
- [4] Gershman, B.N., Eruxhimov L.M., Yashin Yu.Ya., Wavy Phenomena in the Ionosphere and Cosmic Plasma, Moscow, Nauka, 1984 (in Russian).
- [5] Jandieri G.V., Gavrilenko, V.G., Aistov A.V., "Some peculiarities of wave multiple scattering in a statistically anisotropic medium," *Waves Random Media*, vol. 10, pp. 435-445, 2000.
- [6] Gavrilenko, V.G., Jandieri G.V., Ishimaru A, Jandieri V.G., "Peculiarities of spatial spectrum of scattered electromagnetic waves in anisotropic inhomogeneous medium," *Progress In Electromagnetics Research, PIER B*, vol. 7, pp. 191–208, 2008.
- [7] Jandieri G.V., Gavrilenko V. G., Sarokin A. V., Jandieri V.G., "Some properties of the angular power distribution of electromagnetic waves multiply scattered in a collisional magnetized turbulent plasma," *Plasma Physics Report*, vol. 31, pp. 604–615, 2005.
- [8] Jandieri G.V., Ishimaru A., Jandieri V.G., Khantadze A.G., Diasamidze Zh.M., "Model computations of angular power spectra for anisotropic absorptive turbulent magnetized plasma," *Progress In Electromagnetics Research, PIER*, vol. 70, pp. 307–328, 2007.
- [9] Jandieri G.V., Ishimaru A., Zhukova N.N., Diasamidze M.R., "On the influence of fluctuations of the direction of an external magnetic field an phase and amplitude correlation functions of scattered radiation by magnetized plasma slab," *Progress In Electromagnetics Research B, PIER B*, vol. 22, pp. 121–143, 2010
- [10] Jandieri G.V., Ishimaru A., Jandieri V.G., "Depolarization effects of incoherently scattered electromagnetic waves by inhomogeneous magnetized plasma slab," *Journal of Electromagnetic Analysis and Application*, vol. 3, pp. 471-478, 2011.
- [11] Ginzburg V.L., *Propagation of Electromagnetic Waves in Plasma*. New York: Gordon and Beach, 1961.
- [12] Jandieri G.V., Ishimaru A., Jandieri V.G., Kotetishvili K.V., Bzhalava T.N., "A radio scintillation in the collision magnetized plasma". The 2007 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP'07, June 25-28, 2007, Las Vegas, USA, Proceedings of The International Conference on Scientific Computing, vol. CSC2007, pp. 189-195, 2007.

# THE METHOD OF SOLUTION OF OPTIMIZATION PROBLEMS

Seilkhan Boranbayev<sup>1</sup>, Askar Boranbayev<sup>2</sup>

<sup>1</sup>L.N. Gumilyov Eurasian National University, Astana, Kazakhstan

<sup>2</sup>Nazarbayev University, Astana, Kazakhstan

**Abstract** - This article describes the method of mathematical and program solution of optimization problems. An approach of choosing an effective method of solution is shown.

**Keywords:** Optimization methods, problem solving, optimality condition, criterion function

The problems of distribution of resources can be reduced to problems of conditional or unconditional optimization [1-2]. Further, it is possible to apply methods of optimization to these problems. The methods of optimization are described, in particular, in works [3-11] (see the references below).

Let's consider the following problem:

$$F(x) \rightarrow \min$$

$$x \in E^n,$$

$$c_s \leq x_s \leq d_s, \quad s = 1, \dots, n;$$

$$q_v(x) \geq 0; \quad v = 1, \dots, t_1;$$

$$r_k(x) = 0; \quad k = 1, \dots, t_2$$

Where the functions

$$F(x), q_v(x), r_k(x)$$

are convex.

When solving mathematical problems with an unexplored criterion function we face a problem of a choice of a concrete method of solution. The method chosen at random can converge very slowly to a point of a minimum of criterion function or not give a result at all. It is not always possible to receive characteristics of criterion function (convexity, gully, etc.). Therefore, when choosing a method, it is difficult to apply the heuristic reasons based on such characteristics.

Thus, a problem arises when trying to choose an effective method of solving a problem. This process can be automated.

The considered problem can be solved with a help of a computer system, which allows, depending on the information received during the search of the solution, to change not only methods of optimization, but also to replace values of parameters of a method on which the efficiency of methods also depends in many respects. Thus, it is possible to apply multiple methods to solve the problem without stopping the process of solution. The most effective algorithm is being chosen automatically at each stage of solving the problem.

The following approach of solving optimizational problems lays in the basis of a system. We shall designate

$$P_1, P_2, \dots, P_n$$

- methods of the optimization, included in the system.

Various methods, based on characteristics, are included in the system, because the criterion function can have a complex structure.

The problem-oriented list of methods

$$P_{k_1}, P_{k_2}, \dots, P_{k_s}$$

is formed from a set of methods that are in the system, if any information is known about the criterion function (for example if it is known, that it is convex, gullied, square-law, etc.). The solution of a problem consists of steps. The most effective method from the list is being revealed on each step, and then the solution of a problem with the help of this method. The certain time intervals are allocated for revealing and solving the problem. The time, which was allocated for revealing the effective method, is used for promotion to a point of a minimum, because the current point is used during solving the problem. It allows to save time for solving the problem.

At a stage of finding of an effective method to all methods from the list

$$P_{k_1}, P_{k_2}, \dots, P_{k_s}$$

it is enabled to solve a problem during the allocated time interval  $\Delta t$ .

After all methods had an opportunity to solve a problem during the allocated time, search of an effective method stops, gets out the most effective method of the list

$$P_{k_1}, P_{k_2}, \dots, P_{k_s}$$

The size  $\Delta t$  depends on time, spent by a computer on one calculation of criterion function and amount of calculation of criterion function on one iteration, and is calculated as follows:

$$\Delta t = \alpha \mu,$$

Where  $\mu$  - time spent for one calculation of criterion function,

$$\alpha = \max_r \alpha_{k_r},$$

Where  $\alpha_{k_r}$  - amount of calculations of criterion function on one iteration by a method  $P_{k_r}$ .

Thus, at this stage comes out the most effective method.

At the following stage the most effective method is used for solving the problem. The time allocated for solving a problem, is calculated as follows.

$$\tau_0 = \nu \Delta t, \quad \nu > 1,$$

$$\tau_s = q_s \tau_{s-1} + \tau_0.$$

The size  $\nu$  is taken a priori, and depends on complexity and dimension of criterion function. The size

$$q_s = 1$$

if the same one method appeared as the most effective on two consecutive steps. If not, then we take

$$q_s = 0$$

It means that the solution of a problem proceeds, if any method appeared effective on several consecutive steps.

Efficiency  $E_i$  of a method  $P_i$  is calculated based on the formula:

$$E_i = \frac{|F(x^k) - F(x^{k+1})|}{|F(x^k)| + \delta},$$

Where  $x^k, x^{k+1}$  - is an initial and final points, when using a method  $P_i$ ;

$F(x^k), F(x^{k+1})$  - values of criterion function in these points;

$\delta$  - is a small positive number. Efficiency of a method is equal to zero, if the value of a criterion function for allocated time has not decreased.

If for all methods  $P_i$  from the list of methods

$$P_{k_1}, P_{k_2}, \dots, P_{k_s}$$

it happens that  $E_i = 0$ , then it means, that the considered list of methods cannot effectively solve the given problem. In this case the further search of an effective method by the given list of methods stops. To avoid such situation, it is necessary to include various methods into the list, so the list could be oriented to solve different types of problems.

If all the methods that are included in the system have finished their search of solution then we finish the work. The system gives out the saved up information about how the search of a solution to the problem was happening (carried out methods, values of criterion function, time of search, etc.).

Due to self-training, the system enables an automatic choice of an effective method of optimization from the available list

$$P_{k_1}, P_{k_2}, \dots, P_{k_s}$$

for solving specific problems.

Now let's consider the approach incorporated in the system to solve a problem of finding a global minimum of function

$$F(x)$$

Let functions

$$F(x) \quad q_v(x), \quad r_k(x)$$

not necessarily are convex.

Usually, in practical problems there is no goal to find a global minimum with high accuracy. With methods of global optimization it is possible to find a good initial approximate solution to problem, and then to use effective local methods. Generally it is difficult to find out, whether the last found local minimum is the global solution. Except for some narrow classes of problems.

The search of a global minimum can be stopped, if in the current point we got a value of criterion function, which satisfies the requirements of a real life practice, or if we have used up the time that was dedicated to solve the problem.

The best received value of a function can be considered as a final solution of a problem.

When solving practical problems of global optimization the important value has a choice of good initial solution. In this case it is possible to find the minimum nearest to them.

In rare cases it is possible to conduct an analytical research of a criterion function and to receiving the information about the value of a global minimum, or its location.

But generally the considered problem is complex enough, though there is a big number of numerical methods and algorithms of global optimization. At the same time, there is no established classification, both methods of global optimization, and corresponding problems. We shall note, that the majority of authors adhere to classification of methods depending on the used information about the criterion function.

The process of search of a global minimum differs from the process of local optimization by that, that all methods of global optimization are not relaxational. Also, during the global optimization, each method should make a certain number of steps, only after that it is possible to make a decision on transition to another method. Thus the information received as a result of previous search, further is not always possible to use.

However, despite of these difficulties, we can control the process of finding a global minimum.

First, it is possible to change some parameters of this method, when we are looking for a global minimum.

Second, it is possible to do a transition to processes of a local search, from some current points of the process of a global search.

Third, it is possible to make the values of some variables fixed, during the search of a global minimum of the function that has the large number of variables, and to carry out global minimization on the rest of the variables.

These moments allow us to control the process of a global search.

## References

- [1] Boranbayev S.N. "Method of the first and second orders for the decision of a problem of network structure"; Yalichanov readings 4. Kokshetau, Kazakhstan 1998.
- [2] Boranbayev S.N. "Optimality condition in optimization network tasks"; Program 989 American Mathematical Society Meeting. University of Colorado. Boulder. USA. 2003.
- [3] Vasyilev F.P. "Numerical methods of the decision of extreme problems". M.: Science, 1980.
- [4] Evtushenko JU.G. "Method of the decision of extreme problems and their application in systems of optimization". M.: Science, 1982.
- [5] Zhilinskas A. "Global optimization. Axiomatics of statistical models, algorithms, applications". Vilnius, 1986.
- [6] Cea Z. Optimization. The theory and algorithms. M.: the World, 1979.
- [7] Strongin R.G. Numerical methods in multiextreme problems. M.: Science, 1978.
- [8] Fiakko A., G. Mack – Kormik G. Nonlinear programming. Methods of consecutive unconditional minimization. M.: the World, 1979.
- [9] Himmelblau D. Applied nonlinear programming. M.: the World, 1975.
- [10] Bellavia S, Macconi M., Morini B. On the Numerical Solution of Large-Scale Constrained Nonlinear Systems // High performance algorithms and software for nonlinear optimization: status and perspectives. Italy, 2004.
- [11] Zhilinskas A., Zilinskas J. Global Optimization by Means of Stochastic Interval Arithmetic // High performance algorithms and software for nonlinear optimization: status and perspectives. Italy, 2004.



## **SESSION**

# **THE 4TH CELLULAR AUTOMATA, THEORY AND APPLICATIONS WORKSHOP (Drs. Lou D'Alotto, James F. Nystrom, and William Spataro)**

## **Chair(s)**

**Dr. Lou D'Alotto**  
**Dr. James F. Nystrom**  
**Dr. William Spataro**





# Classification of two-dimensional binary cellular automata with respect to surjectivity

Henryk Fukś and Andrew Skelton  
 Department of Mathematics  
 Brock University  
 St. Catharines, ON, Canada

**Abstract**—While the surjectivity of the global map in two-dimensional cellular automata (2D CA) is undecidable in general, in specific cases one can often decide if the rule is surjective or not. We attempt to classify as many 2D CA as possible by using a sequence of tests based on the balance theorem, injectivity of the restriction to finite configurations, as well as permutivity. We introduce the notion of slice permutivity which is shown to imply surjectivity in 2D CA. The tests are applied to 2D binary CA with neighbourhoods consisting of up to five sites, considering all possible contiguous shapes of the neighbourhood. We find that if the size of the neighbourhood is less than five, complete classification of all rules is possible. Among 5-site rules, those with von Neuman neighbourhoods as well as neighbourhoods corresponding to T, V, and Z pentominos can also be completely classified.

**Keywords:** cellular automata, surjective, permutive, classification, neighbourhood

## 1. Introduction

In the theory of cellular automata (CA), the surjectivity of the global map is one of the most extensively studied properties of CA. It is only natural to ask, therefore, what are the examples of surjective CA?

In the case of one-dimensional CA, such examples are easy to construct because there exists the well-known Amoroso-Patt algorithm for determining if a given elementary cellular automaton is surjective [1]. Using this algorithm it can be shown that among the 88 minimal elementary CA rules, the only surjective rules have Wolfram code numbers 15, 30, 45, 51, 60, 90, 105, 106, 150, 154, 170 and 204.

In two dimensions, however, the situation is much different. It has been shown that the question of surjectivity of two-dimensional cellular automata is undecidable [4], which means that it is impossible to construct a single algorithm which would always decide if an arbitrary rule is surjective or not. This, of course, does not exclude a

possibility that for specific classes of 2D rules surjectivity can still be decidable – it is known, for example, that rules which are permutive with respect to the corners of the Moore neighbourhood are surjective [2].

In this paper, we attempt to classify 2D rules with respect to surjectivity using two known properties equivalent to surjectivity, namely the balance theorem and the injectivity of restrictions to finite configurations. Moreover, we introduce the concept of slice-permutivity which is then shown to imply surjectivity. We show that all 2D CA with neighbourhoods of size four (or less), no matter what shape, can be classified. For five-site CA, complete classification is still possible for certain neighbourhood shapes, notably including von Neumann neighbourhood.

## 2. Basic Definitions

Let  $\mathcal{A}$  be a finite set of symbols, to be called a *symbol set*. We define a *two-dimensional configuration*  $s$  to be a function  $s : \mathbb{Z}^2 \rightarrow \mathcal{A}$ , and  $\mathcal{A}^{\mathbb{Z}^2}$  to be the set of all two-dimensional configurations. For any vector  $\vec{x} \in \mathbb{Z}^2$ , we denote  $s_{\vec{x}} \in \mathcal{A}$  to be a symbol located at position (or site)  $\vec{x}$  in configuration  $s$ . If  $\mathcal{V} \subset \mathbb{Z}^2$ , we define  $s_{\mathcal{V}} = [s_{\vec{x}}]_{\vec{x} \in \mathcal{V}}$ .

A *neighbourhood*  $\mathcal{N}$  is a finite subset of vectors in  $\mathbb{Z}^2$ . A neighbourhood is said to be *contiguous* if, for any vector  $\vec{x} \in \mathcal{N}$ , at least one vector in the set  $\{\vec{x} \pm (1, 0), \vec{x} \pm (0, 1)\}$  is also in  $\mathcal{N}$ . For any vector  $\vec{x}$ , the *neighbourhood of  $\vec{x}$*  is defined as  $\mathcal{N}(\vec{x}) = \{\vec{u} + \vec{x} : \vec{u} \in \mathcal{N}\}$ .

We can now define the *local mapping* of a *two-dimensional cellular automata (2D CA)* to be the function  $f : \mathcal{A}^{\mathcal{N}} \rightarrow \mathcal{A}$ . The local mapping induces a *global mapping*  $F : \mathcal{A}^{\mathbb{Z}^2} \rightarrow \mathcal{A}^{\mathbb{Z}^2}$  so that  $F(s)_{\vec{x}} = f(s_{\mathcal{N}(\vec{x})})$ , for all  $s \in \mathcal{A}^{\mathbb{Z}^2}$  and all  $\vec{x} \in \mathbb{Z}^2$ .

If  $\mathcal{Z} \subset \mathbb{Z}^2$  is a finite set of vectors, then we define a *block* to be an element of  $\mathcal{A}^{\mathcal{Z}}$ .

The neighbourhood of  $\mathcal{Z}$  is defined similarly as before, so that  $\mathcal{N}(\mathcal{Z}) = \{\vec{u} + \vec{x} : \vec{u} \in \mathcal{N}, \vec{x} \in \mathcal{Z}\}$ . The *block*

evolution operator  $\mathbf{f} : \mathcal{A}^{\mathcal{N}(\mathcal{Z})} \rightarrow \mathcal{A}^{\mathcal{Z}}$  is now defined by  $\mathbf{f}(b)_{\vec{x}} = f(b_{\mathcal{N}(\vec{x})})$  for any  $\vec{x} \in \mathcal{Z}$  and  $b \in \mathcal{A}^{\mathcal{N}(\mathcal{Z})}$ .

Given a block  $b \in \mathcal{A}^{\mathcal{Z}}$ , the set of preimages of  $b$  under  $\mathbf{f}$  is the set of blocks  $b' \in \mathcal{A}^{\mathcal{N}(\mathcal{Z})}$  such that  $\mathbf{f}(b') = b$ . This preimage set will be denoted  $\mathbf{f}^{-1}(b)$ .

Sometimes, we will need to consider the neighbourhood of a neighbourhood. We will then use the notation  $\mathcal{N}^2(\mathcal{Z}) = \mathcal{N}(\mathcal{N}(\mathcal{Z}))$ , and higher powers will refer to the appropriate number of neighbourhood compositions.

Let  $\vec{v} \in \mathcal{N}$ , and let us denote  $\mathcal{M} = \mathcal{N} \setminus \vec{v}$ . Let  $b \in \mathcal{A}^{\mathcal{M}}$  and let us denote  $[x, b]$  to be an element of  $\mathcal{A}^{\mathcal{N}}$  such that its entries with indices in  $\mathcal{M}$  are the same as corresponding entries in  $b$ , while the entry with index  $\vec{v}$  is equal to  $x$ ,  $x \in \mathcal{A}$ . A 2D CA is *permutive with respect to*  $\vec{v} \in \mathcal{N}$  if, for any choice of  $b$ , the function  $x \rightarrow f([x, b])$  is one-to-one.

## 2.1 One-dimensional Binary Rules

Before we attempt to classify two-dimensional CA rules, let us discuss what happens in one dimension, as this will give us some important insight. As mentioned in the introduction, surjectivity in 1D is known to be decidable, and the algorithm for testing for surjectivity has been developed by Amoroso and Patt in early 70's [1]. We used this algorithm to find all surjective binary rules of a given neighbourhood size, for neighbourhood sizes ranging from 1 to 5. We also checked which of these rules are permutive. The results are given in Table 1. One can make two interesting observations from this table. First of all, the proportion of rules which are surjective decreases dramatically as the neighbourhood size increases. The second observation can be stated as the following proposition.

*Proposition 2.1:* Any contiguous one-dimensional binary cellular automata dependent on three or less sites is surjective if and only if it is permutive.

This means that for a binary rule to be surjective yet non-permutive a neighbourhood of at least four sites is needed. A natural question to ask, therefore, is whether this is also the case in two dimensions?

## 3. Permutivity and Surjectivity

As we will shortly see, permutivity alone is not enough to guarantee surjectivity in two dimensions. In [2], the authors considered 2D CA with Moore neighbourhood of radius  $r$ , where  $\mathcal{N} = \{(i, j) : |i|, |j| \leq r\}$ . They proved that any such rule is surjective if it is permutive with respect to sites  $(\pm r, \pm r)$ . We will prove a similar result using an arbitrary neighbourhood and any

site that can be sliced off from the neighbourhood by a straight line.

Given  $m, c \in \mathbb{Q}$ , we define a line  $\ell = \{(x, y) : y = mx + c\}$ , and the following two regions,  $\ell^+ = \{(x, y) : y > mx + c\}$  and  $\ell^- = \{(x, y) : y < mx + c\}$ . For vertical lines  $\ell = \{(x, y) : x = c\}$  we similarly define  $\ell^+ = \{(x, y) : x > c\}$  and  $\ell^- = \{(x, y) : x < c\}$ . A site  $\vec{x} \in \mathcal{N}$  can be *sliced* if there exists a set  $\ell$  such that  $\vec{x} \in \ell$  and  $\mathcal{N} \setminus \vec{x} \subset \ell^+$  (or  $\ell^-$ ). A 2D CA is *slice permutive* if it is permutive with respect to a site which can be sliced.

The main result relating slice-permutivity and surjectivity can be stated as follows.

*Theorem 3.1:* Any two-dimensional slice permutive CA is surjective.

Before we start the proof, we will need the following classical result. Let  $\mathcal{Z}_n$  be a square region in  $\mathbb{Z}^2$ , defined as  $\mathcal{Z}_n = [0, 1, \dots, n-1] \times [0, 1, \dots, n-1]$ , where  $n \in \mathbb{N}$ .

*Theorem 3.2 (Balance Theorem):* A 2D CA is surjective if and only if for all  $n \geq 1$  and all  $b, b' \in \mathcal{A}^{\mathcal{Z}_n}$ , we have  $\text{card } \mathbf{f}^{-1}(b) = \text{card } \mathbf{f}^{-1}(b')$ .

A one-dimensional version of this theorem first appeared in [3]. The proof of the two-dimensional version can be found in [5], where the authors consider a Moore neighbourhood of any radius. Since any neighbourhood can be extended to a Moore neighbourhood by adding extra sites, the Balance Theorem also holds for a CA rule with any neighbourhood shape.

*Proof:* [of Theorem 3.1] Consider an arbitrary block  $b \in \mathcal{A}^{\mathcal{Z}_n}$ . Since our CA is slice permutive, there exists a line  $\ell$  which slices the neighbourhood at some site  $\vec{u} \in \mathcal{N}$ , so that all sites in  $\mathcal{N} \setminus \vec{u}$  are either in  $\ell^-$  or in  $\ell^+$ . Without loss of generality, we assume that all sites of  $\mathcal{N} \setminus \vec{u}$  are in  $\ell^-$ . The proof of the other case can be obtained by replacing  $\ell^\pm$  with  $\ell^\mp$ . Let  $\vec{n}$  be the normal vector to  $\ell$  oriented so that it points in the direction of  $\ell^+$ . Consider  $\{\ell_1, \ell_2, \dots, \ell_k\}$  to be the family of lines with the same slope as  $\ell$  with indices increasing in the direction of  $\vec{n}$ , so that for any  $\vec{x} \in \mathcal{Z}_n$  there exists  $\ell_i$  such that  $\vec{x} \in \ell_i$ . We now construct the set of preimages of  $b$ , each element of which is of the form  $[a_{\vec{x}}]_{\vec{x} \in \mathcal{N}(\mathcal{Z}_n)}$ .

In order to illustrate this better, we will conduct the proof while simultaneously referring to an example of a

Neighbourhood Size	Total Rules	Surjective Rules		
		Permutive	Not Permutive	Total
1	2	2	0	2
2	16	6	0	6
3	256	28	0	28
4	65536	518	64	582
5	4294967296	131502	11516	143018

Table 1: One-dimensional binary rules

rule defined on the seven-site neighbourhood

$$\mathcal{N} = \{(-3, 1), (-3, 0), (-2, 0), (-1, 0), (0, 0), (0, -1), (1, -1)\}.$$

In this example, we assume that the rule is permutive with respect to the sliceable site  $(0, 0)$ , for which the corresponding line  $\ell$  has slope  $-1/2$ . The neighbourhood of  $\mathcal{Z}_3$  is shown in Figure 3(a).

We start the construction from sites of  $\mathcal{Z}_n$  which belong to the line  $\ell_1$ . Due to slice permutivity, all sites of the preimage which are below this line can take arbitrary values, and sites which are on the line can be chosen in such a way that  $b_{\vec{x}} = f(a_{\mathcal{N}(\vec{x})})$  for all  $\vec{x} \in \ell_1 \cap \mathcal{Z}_n$ . This means that sites of  $\mathcal{N}(\mathcal{Z}_n)$  which belong to  $\ell_1^-$  can take arbitrary values, and sites of  $\mathcal{Z}_n$  which belong to  $\ell_1$  are uniquely determined by those arbitrary values.

In Figure 3(b) arbitrary sites are denoted by stars, and the uniquely determined site (only one in this case) is denoted by D.

We then move to  $\ell_2$ , and again, for every possible configuration of sites from the set  $\mathcal{N}(\mathcal{Z}_n) \cap \ell_2^-$ , values of sites which lie in  $\mathcal{Z}_n \cap \ell_2$  will be uniquely determined. It may happen, as shown in Figure 3(b), that some sites of  $\mathcal{N}(\mathcal{Z}_n) \cap \ell_2^-$  have not been labeled before. These sites can also assume arbitrary values, and thus they are marked as red stars in Figure 3(c).

We repeat the above procedure until all sites of  $\mathcal{N}(\mathcal{Z}_n)$  are labeled, as shown in Figures 3(d-h). In the end, as in Figure 3(i), all sites which belong to  $\mathcal{Z}_n$  will be labeled by D, while the sites of  $\mathcal{N}(\mathcal{Z}_n) \setminus \mathcal{Z}_n$  will be labeled by stars. This means that we have  $\text{card}(\mathcal{N}(\mathcal{Z}_n) \setminus \mathcal{Z}_n)$  sites in the preimage which can assume any values from the symbol set  $\mathcal{A}$ . Thus, for each block  $b \in \mathcal{B}_r$ , we have  $\text{card } \mathbf{f}^{-1}(b) = (\text{card } \mathcal{A})^{\text{card}(\mathcal{N}(\mathcal{Z}_n) \setminus \mathcal{Z}_n)}$ . Since this is independent of  $b$ , by Theorem 3.2 we conclude that the CA is surjective. ■

## 4. Classification procedure

The result of the previous section gives us a method to determine surjectivity of a special class of rules,

namely slice-permutive rules. In classifying rules we will also need some method for checking if a rule is not surjective – although of course a *general* algorithm of this type cannot exist. One such method involves the Balance Theorem. One can simply check if all blocks of a given size have the same number of preimages - if one finds a single violation of the balance theorem, the rule is obviously not surjective. Clearly, one can perform such exhaustive check only for block of small size, and if no balance violation is found, then the test remains inconclusive. It turns out, however, that this simple method is surprisingly effective for 2D rules with small neighbourhood size.

Another test for non-surjectivity is based on the following classical result [6]. Here by a finite configuration we mean a configuration where all but a finite number of sites are in some arbitrary fixed state.

*Theorem 4.1:* A CA is surjective if and only if it is injective when restricted to finite configurations.

If one can find two finite configurations which are different but have the same image, then the rule is not surjective. One can perform this test exhaustively for small finite configurations. It turns out to be useful in cases where the balance theorem test is inconclusive.

We may now present a procedure for classification of 2D rules with a given neighbourhood shape. This procedure is illustrated in Figure 2. While it has been tailored for binary rules (that is, rules where  $\text{card } \mathcal{A} = 2$ ), it can be easily adapted for larger alphabets.

*Procedure 4.1:* We start with a list of all rules on a given neighbourhood and apply the following filters to remove known surjective and non-surjective rules.

- 1) First we check the simplest balance condition, that is,  $\text{card } \mathbf{f}^{-1}(0) = \text{card } \mathbf{f}^{-1}(1)$ . Rules which violate this condition are non-surjective.
- 2) We check if the rule truly depended on all given sites. If not, it means it can be treated as if it had

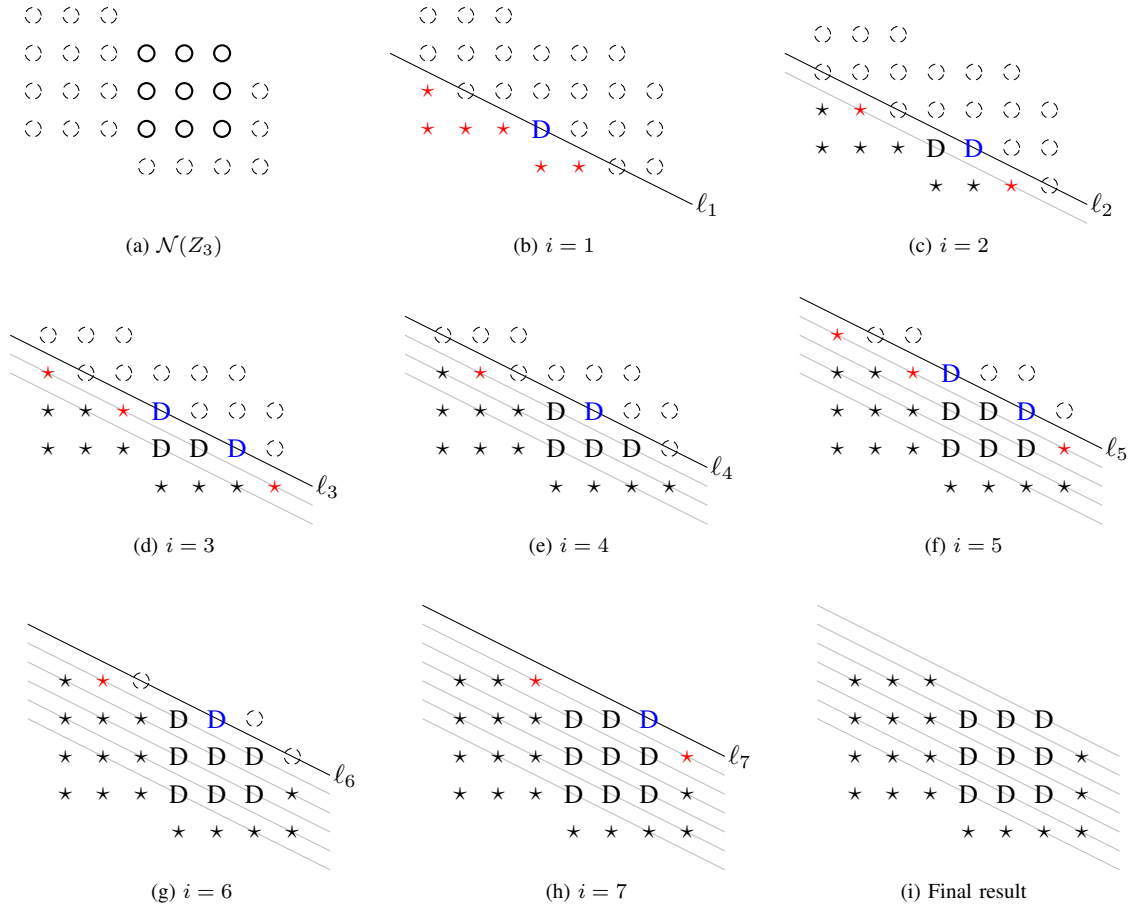


Fig. 1: Example: Iteration of the procedure in Theorem 3.1

smaller neighbourhood, thus we check how it was classified before (we were classifying rules with progressively increasing neighbourhood size).

- 3) We check if the rule is permutive with respect to at least one sliceable site. If it is, it is surjective.
- 4) The following two tests are performed simultaneously owing to the fact that they required approximately the same information.
  - We check if all blocks from the set  $\mathcal{A}^{\mathcal{N}(z_1)}$  have the same number of preimages. If not, the rule is non-surjective.
  - We search for violations of injectivity on finite configurations. Let  $\mathcal{M}$  be the set of all sites  $\vec{x} \in \mathbb{Z}^2$  for which  $(0,0) \in \mathcal{N}(\vec{x})$ . For each  $b \in \mathcal{A}^{\mathcal{M}}$  we consider a finite configuration  $s$  such that  $s_{\mathcal{M}} = b$  and the sites which do not belong to  $\mathcal{M}$  are in the state 0, as well as configuration  $s'$  obtained from  $s$  by replacing  $s_{(0,0)}$  by  $1-s_{(0,0)}$ . If  $F(s) = F(s')$ , injectivity is violated, thus the rule is not surjective.

- 5) We check if all blocks from the set  $\mathcal{A}^{\mathcal{N}^2(z_1)}$  have the same number of preimages. If not, the rule is non-surjective.

## 5. Classification Results

We applied the aforementioned procedure to two-dimensional rules with neighbourhoods of size 3, 4, and 5. It is not necessary to consider smaller neighbourhoods, as those are effectively one-dimensional, thus results of sec. 2.1 apply. In all cases we considered only truly two-dimensional contiguous neighbourhoods, meaning that linear neighbourhoods were excluded – again, these are included in results of sec. 2.1. Contiguous neighbourhoods have shapes known as polyominoes, that is, plane geometric figures formed by joining several equal squares edge to edge. Since rigid transformations of  $\mathbb{Z}^2$  preserve surjectivity, we considered only shapes which are representatives of equivalence classes with respect to the group of isometries of  $\mathbb{Z}^2$ .

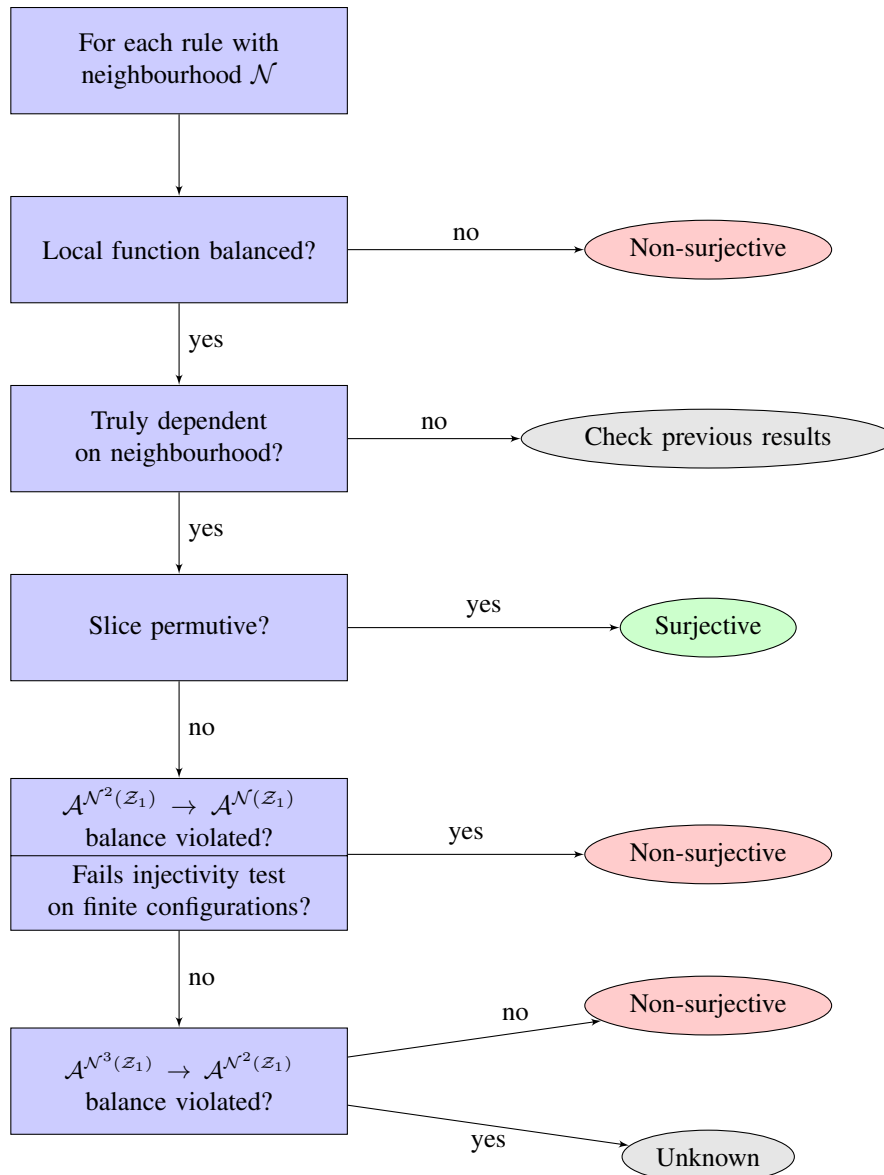


Fig. 2: Surjectivity test flowchart.

### 5.1 Three-site neighbourhoods

There is only one contiguous three-site neighbourhood which is truly two-dimensional: the L-shaped neighbourhood (Figure 3a), with 256 corresponding binary rules. Applying the procedure of Figure 2, we found that all these rules can be classified, and that among them 38 rules are surjective, all of which are slice permutive. The remaining rules are non-surjective.

*Proposition 5.1:* Any contiguous three-site binary CA is surjective if and only if it is slice permutive.

### 5.2 Four-site neighbourhoods

There are four contiguous two-dimensional four-site neighbourhoods, often referred to as *tetrominos* and named after their resemblance to the letters L, O, S and T (Figure 3b). There are 65536 total binary rules dependent on each of these neighbourhoods.

In the case of the L and T neighbourhoods, there are 724 slice permutive (and thus surjective) rules. In the case of the O and S neighbourhoods, there are 942 slice permutive and surjective rules. This difference seems to be due to the number of sliceable sites in

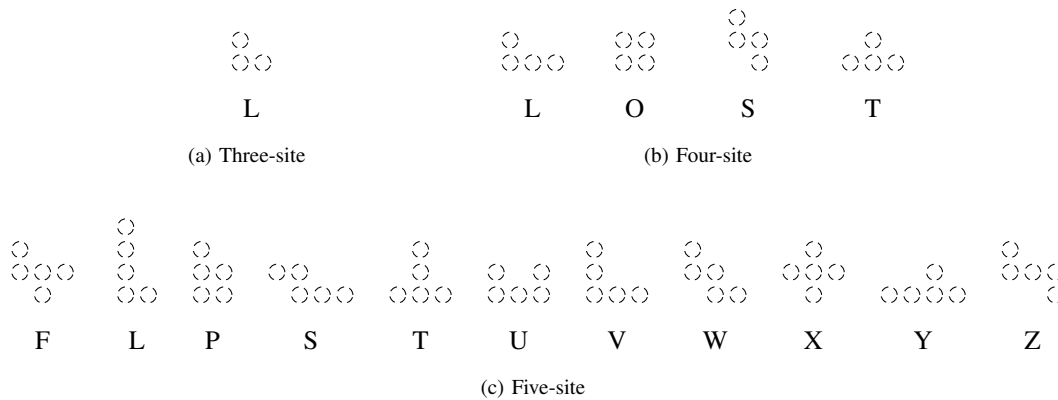


Fig. 3: Contiguous Neighbourhoods

each neighbourhood (3 and 4 respectively). In each case, following Procedure 4.1 we determined that these are the only surjective rules, thus we again obtain a complete classification of all rules truly dependent on all four sites. This can be summarized as follows.

*Proposition 5.2:* Any contiguous four-site two-dimensional binary cellular automaton is surjective if and only if it is slice permutive.

### 5.3 Five-site neighbourhoods

There are eleven contiguous two-dimensional five-site neighbourhoods, often referred to as *pentominos* and named after their resemblance to the letters F, L, P, S, T, U, V, W, Z, Y and Z. There are 4294967296 binary rules associated with each neighbourhood. The results of application of Procedure 4.1 are shown in Table 2. We can observe that the classification is complete in the case of four neighbourhood shapes, T, V, X, and Z.

*Proposition 5.3:* Any contiguous five-site two-dimensional binary cellular automaton with the neighbourhood shaped as T, V, X, or Z pentomino is surjective if and only if it is slice permutive.

Note that this includes the traditional von Neumann neighbourhood (shape X). Also note that for neighbourhood shapes for which complete classification was not possible, the vast majority of rules were nevertheless classified. In all cases the fraction of rules which were classified exceeded 99.998%.

## 6. Conclusions

We were able to obtain complete classification with respect to surjectivity of all 2D rules with contiguous neighbourhoods of size up to 4. In these neighbourhoods, all surjective rules are slice permutive. Among 5-site rules, those with von Neumann neighbourhoods as well as neighbourhoods corresponding to T, V, and Z pentominos can also be completely classified, and again, surjectivity and slice permutivity are equivalent for them. For the remaining pentomino shapes, only a very small fraction of rules defies classification. The worst case is the S pentomino, for which .0016% rules cannot be classified with our algorithm.

A number of open questions remain. First of all, can the remaining 5-site rules be classified? We suspect that at least some of them could be, if one performed balance and/or injectivity tests on larger blocks, although such tests would be computationally very expensive. A related question is whether there exist any truly two-dimensional five-site binary rule which is surjective yet not slice permutive? Since in one dimension such rules are possible even in four-site neighbourhoods, we suspect that the answer is affirmative, although currently we cannot offer any evidence of this claim.

## 7. Acknowledgements

One of the authors (HF) acknowledges financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) in the form of Discovery Grant. This work was made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET:www.sharcnet.ca) and Compute/Calcul Canada.

Table 2: Two-dimensional truly five-site binary rules

Neighbourhood	Surjective	Unknown	Neighbourhood	Surjective	Unknown
F	257106	16	V	193138	0
L	193138	1472	W	257106	3596
P	257106	3596	X	257106	0
S	192938	67764	Y	193138	1472
T	193138	0	Z	257106	0
U	192938	64264			

## References

- [1] S. Amoroso and Y. N. Patt. Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *Journal of Computer and System Sciences*, 6:448–464, 1972.
- [2] A. Dennunzio and E. Formenti. Decidable properties of 2d cellular automata. *Lecture Notes in Computer Science*, 5257:264–275, 2008.
- [3] G. Hedlund. Endomorphisms and automorphisms of shift dynamical systems. *Mathematical Systems Theory*, 3:320–375, 1969.
- [4] J. Kari. Reversibility and surjectivity problems of cellular automata. *Journal of Computer System Science*, 48:149–182, 1994.
- [5] A. Maruoka and M. Kimura. Condition for injectivity of global maps for tessellation automata. *Information and Control*, 32:158–162, 1976.
- [6] D. Richardson. Tessellation with local transformation. *Journal of Computer System Science*, 6:373–388, 1972.

# A cellular automaton model to investigate emergent behavior of heterogeneous cell populations

Carsten Mente<sup>\*,†</sup>, Andreas Deutsch<sup>\*,‡</sup>

**Abstract**—*Biological cells are never exactly identical, even cells of the same cell type usually display stochastic differences in their properties. Understanding the role of stochastic differences between individual cells in shaping the behavior of the entire cell population is crucial for understanding many biological phenomena. We are interested especially in populations of interacting cells with stochastic differences in the interaction properties of individual cells.*

*We introduce a novel extension of classical lattice-gas automata (LGCA) models called individual-based lattice-gas cellular automata (IB-LGCA) that enables the modeling of heterogeneous cell populations while preserving the advantages of classical LGCA models.*

*We show with a specific IB-LGCA model that the introduction of stochastic differences in the interaction properties of cells leads to significantly different qualitative behavior: while simulations with identical cells show behavior reminiscent of a phase transition, this behavior is lost with the introduction of stochastic differences in the interaction properties of cells.*

**Keywords:** *Lattice-gas cellular automaton, individual-based model, computational biology*

Submitted to: 4th Automata, Theory and Applications Workshop (\*A-PDPTA'12)

## 1. Introduction

Populations of biological cells of the same cell type still display stochastic differences in the properties of individual cells [1]. Mathematical models aimed at describing the behavior of cell populations often disregard these individual cell differences and assign identical properties to all cells of one cell type. However, understanding the role of stochastic differences between individual cells in shaping the behavior of the entire cell population is crucial for understanding many biological phenomena [2].

In this manuscript, we are interested especially in populations of interacting cells with stochastic differences in the interaction properties of individual cells. Possible interactions in a population of interacting cells can be divided into two mutually exclusive categories: internal interactions, i.e. interactions between cells belonging to the cell population, and external interactions, i.e. interactions between cells of the population and components of the environment. The interplay between internal and external interactions gives rise to macroscopic, observable behavior of the cell population. Typical

in many cell populations are cell-cell adhesion, an internal interaction, and cell migration along an external gradient (axis), an external interaction [3].

Lattice-gas cellular automaton (LGCA) models are appropriate models to investigate the emergent behavior of populations of interacting cells [4],[5]. However, classical LGCA models assume a population of identical cells and model different types of cells by combining different LGCA models, one for each cell type. For more than a few different cell types this approach leads to very cumbersome interaction probabilities. LGCA models are therefore not well suited to model populations of interacting cells with individual cell properties. To address this issue, we introduce individual-based lattice-gas cellular automata (IB-LGCA) which extend classical LGCA by incorporating individual cell properties. IB-LGCA allow to model heterogeneous cell populations but preserve all the advantages of classical LGCA.

Here, we investigate the behavior of a population of cells subject to an external gradient and moving in response to it (axis). We want to understand which role differences in cell properties play with regard to the behavior of the entire cell population. In the next section, we give a definition of IB-LGCA models and introduce the specific LGCA model we use here. In the third section of the manuscript, we compare simulation results for systems with identical cells and systems with various degrees of individual cell differences.

## 2. Individual-based lattice-gas cellular automata

**a) General definition:** In an individual-based lattice-gas cellular automaton (IB-LGCA), space, time, and states are discrete, while the temporal dynamics are formulated as a local transition rule. In contrast to classical lattice-gas cellular automata (LGCA) we extend the definition by introducing a set  $\mathcal{P}$  which contains additional properties. Before we can give the definition of an individual-based lattice-gas cellular automaton we need to define the terms regular lattice, neighborhood, configurations and transition rule. Here, we define a regular lattice as a countable subset of  $\mathbb{R}^d$  constructed as the linear combination with at most countable many coefficients of vectors  $c_i \in \mathbb{R}^d$ ,  $i = 0, \dots, b-1$  where  $b \in \mathbb{N}_{>0}$ .

*Definition 1:* A regular lattice of dimension  $d \in \mathbb{N}$  is a set  $\mathcal{L}$  with

$$\bullet \mathcal{L} = \left\{ r \mid r = \sum_{i=0}^{b-1} a_i c_i \text{ with } a_i \in X \right\} \subset \mathbb{R}^d \text{ with } X = \mathbb{Z} \text{ or } X = [0, a] \subset \mathbb{Z}, \text{ and a fixed } b \in \mathbb{N}_{>0}.$$

The elements  $r$  of  $\mathcal{L}$  are called lattice nodes. The vectors  $c_i \in \mathbb{R}^d$ ,  $i = 0, \dots, b-1$  are called channels. More specific,  $c_i \in \mathbb{R}^d$  is called a *velocity channel* if  $c_i \neq 0^d$  and a *rest channel* otherwise.

\*Center for Information Services and High Performance Computing, Department for Innovative Methods of Computing, Technische Universität Dresden, Nöthnitzer Straße 46, 01062 Dresden, Germany

†Email: carsten.mente@tu-dresden.de, corresponding author

‡Email: andreas.deutsch@tu-dresden.de



We also have to define what it means for two lattice nodes to be neighbors:

**Definition 2:** A finite *neighborhood*  $\mathcal{N}_r$  around  $r \in \mathcal{L}$  is a set with

- $\mathcal{N}_r = \left\{ r + e | e = \sum_{i=0}^{b-1} a_i c_i \text{ with } a_i \in Y_i \text{ and } \sum_{i=0}^{b-1} a_i \leq R \right\} \subset \mathcal{L}$  with  $Y_i \subset \mathbb{Z}$ ,  $i = 0, \dots, b-1$  and a fixed value  $R \in \mathbb{Z}$ ,
- $r' \in \mathcal{L}$  is a neighbor of  $r \in \mathcal{L}$  with regard to the neighborhood  $\mathcal{N}_r$  if  $r' \in \mathcal{N}_r$ .

We specify cells by configurations at every lattice node:

**Definition 3:** A mapping  $\eta : \mathcal{L} \rightarrow \mathcal{E}^b$  with  $\eta(r) = (\eta_0(r), \dots, \eta_{b-1}(r))$  and  $\mathcal{E} = \{0, 1\}$ , is called a *configuration*. For a fixed  $r \in \mathcal{L}$  and  $i = 0, \dots, b-1$  we call a channel  $c_i$  *occupied* at  $r \in \mathcal{L}$  if  $\eta_i(r) = 1$  and *empty* if  $\eta_i(r) = 0$ . Cells are identified with occupied channels.

The crucial difference between classical LGCAs and IB-LGCAs is the introduction of individual cell properties:

**Definition 4:** A mapping  $p : \mathcal{L} \rightarrow \mathcal{P}^b$  with  $p(r) = (p_0(r), \dots, p_{b-1}(r))$  is called a *configuration of properties* at  $r \in \mathcal{L}$ . There is no restriction on  $\mathcal{P}$ , it is sufficient that it is a set.

The last item we have to define is the transition rule.

**Definition 5:** A *local transition rule*  $\mathcal{R}$  consists of two parts: an interaction rule  $\mathcal{I}$  and a migration rule  $\mathcal{M}$

- interaction  $\mathcal{I}$ : stochastically assigns a new configuration  $\eta^{\mathcal{I}}(r) \in \mathcal{E}^b$  and a new configuration of properties  $p^{\mathcal{I}}(r) \in \mathcal{P}^b$  to node  $r \in \mathcal{L}$  given by the transition probabilities  $P((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}) \rightarrow \eta^{\mathcal{I}}(r))$  and  $P((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}, \eta^{\mathcal{I}}) \rightarrow p^{\mathcal{I}}(r))$ , with  $\eta_{\mathcal{N}_r} := \{\eta(r') | r' \in \mathcal{N}_r\}$  and  $p_{\mathcal{N}_r} = \{p(r') | r' \in \mathcal{N}_r\}$ .
- migration  $\mathcal{M}$ : redistributes cells and properties in the neighborhood  $\mathcal{N}_r$  according to  $\eta^{\mathcal{M}}(r) = (\eta_0^{\mathcal{M}}(r - c_0), \dots, \eta_{b-1}^{\mathcal{M}}(r - c_{b-1}))$  and  $p^{\mathcal{M}}(r) = (p_0^{\mathcal{M}}(r - c_0), \dots, p_{b-1}^{\mathcal{M}}(r - c_{b-1}))$ .

While the form of  $P((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}) \rightarrow \eta^{\mathcal{I}}(r))$  is problem specific,  $P((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}, \eta^{\mathcal{I}}) \rightarrow p^{\mathcal{I}}(r))$  is characterized by:

$$P((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}, \eta^{\mathcal{I}}) \rightarrow p^{\mathcal{I}}(r)) \sim \begin{cases} 1 & : \text{ if } p_i^{\mathcal{I}}(r) = p_{\pi(i)}(r) \\ & \forall i \in 0, \dots, b-1 \\ 0 & : \text{ else} \end{cases}$$

where  $\pi \in \{\text{set of all permutations on } \{0, \dots, b-1\} \text{ that leave } \eta^{\mathcal{I}}(r) \text{ invariant}\}$ . This means we only consider permutations  $\pi$  that do not switch occupied and empty channels. So, properties  $p$  are redistributed among cells of the new configuration  $\eta^{\mathcal{I}}$  with equal probability for any permutation  $\pi$  on the cells. Obviously,  $\eta^{\mathcal{I}}(r)$  has to be determined from  $P((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}) \rightarrow \eta^{\mathcal{I}}(r))$  before  $p^{\mathcal{I}}(r)$  can be determined from  $P((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}, \eta^{\mathcal{I}}) \rightarrow p^{\mathcal{I}}(r))$ . Application of interaction and migration specifies new configurations  $\eta^{\mathcal{M} \circ \mathcal{I}}(r)$  and properties  $p^{\mathcal{M} \circ \mathcal{I}}(r)$  for every  $r \in \mathcal{L}$ .

We can now give the definition of an individual-based lattice-gas cellular automaton

**Definition 6:** A individual-based lattice-gas cellular automaton (IB-LGCA) is a tuple  $(\mathcal{L}, \mathcal{E}, \mathcal{P}, \mathcal{R})$  with:

- a lattice  $\mathcal{L}$  of dimension  $d$  and channels  $c_i$ ,  $i = 0, \dots, b-1$  and  $b \in \mathbb{N}_{>0}$ ,
- a set of states  $\mathcal{E} \in \{0, 1\}$  and a set of properties  $\mathcal{P}$ ,

- a transition rule  $\mathcal{R}$ .

The transition rule  $\mathcal{R}$  is applied to every lattice node  $r$  synchronously. The repeated application of the transition rule  $\mathcal{R}$  describes the time evolution of the lattice-gas cellular automaton. Thus, the time evolution of a LGCA consists of a number of discrete points starting at  $k = 0$  and advancing in time from  $k$  to  $k + 1$ ,  $k \in \mathbb{N}$ .

**b) Specific application:** Here, we consider a simple model system of cells moving on a surface under an external stimulus. The cells are subject to two competing mechanisms: “adhesion” which keeps cells together and “taxis” which causes the cells to move along a fixed external gradient. The influence of these mechanisms depends on the properties of each individual cell, specified in the model as two parameters: adhesivity and taxis-sensitivity.

We choose the set of properties  $\mathcal{P} := \mathbb{R} \times \mathbb{R}$ . For every cell  $i$ ,  $i = 0, \dots, b-1$ , i.e. for every occupied channel of  $\eta(r)$  the adhesivity  $\alpha_i \in \mathbb{R}$  and taxis-sensitivity  $\beta_i \in \mathbb{R}$  are given by  $p(r)_i = (\alpha_i, \beta_i)$ . The interaction probability for adhesion is defined by:

$$P_{ad}((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}) \rightarrow \eta^{\mathcal{I}}(r)) \sim \exp \left( \sum_{i=0}^{b-1} \alpha_i \eta_i^{\mathcal{I}} \langle c_i, g_{ad}(r) \rangle \right),$$

where  $g_{ad}(r) = \sum_{r_j \in \mathcal{N}_r} \sum_{j=0}^{b-1} \eta_j(r'_j) (r'_j - r_i)$  is the local gradient field, which points in the direction of the highest number of cells in the neighborhood  $\mathcal{N}_r$  of  $r$ . This interaction probability specifies that cells are more likely to move in the direction of higher neighboring cell numbers than in other directions. For the taxis the interaction probability is:

$$P_{ta}((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}) \rightarrow \eta^{\mathcal{I}}(r)) \sim \exp \left( \sum_{i=0}^{b-1} \beta_i \eta_i^{\mathcal{I}} \langle c_i, g_{ta} \rangle \right),$$

where  $g_{ta} = (1, 0)^T$  is an external gradient. This interaction probability specifies that cells are more likely to move in the direction of the gradient. The full interaction probability  $P$  is the combination of  $P_{ad}$  and  $P_{ta}$ :

$$P((\eta_{\mathcal{N}_r}, p_{\mathcal{N}_r}) \rightarrow \eta^{\mathcal{I}}(r)) = \frac{1}{Z} \exp \left( \sum_{i=0}^{b-1} \eta_i^{\mathcal{I}} (\alpha_i \langle c_i, g_{ad}(r) \rangle + \beta_i \langle c_i, g_{ta} \rangle) \right) \delta(\rho(r), \rho^{\mathcal{I}}(r)),$$

with the normalization factor:

$$Z := \sum_{\eta^{\mathcal{I}}} \exp \left( \sum_{i=0}^{b-1} \eta_i^{\mathcal{I}} (\alpha_i \langle c_i, g_{ad}(r) \rangle + \beta_i \langle c_i, g_{ta} \rangle) \right) \delta(\rho(r), \rho^{\mathcal{I}}(r)).$$

$\rho(r) := \sum_{i=0}^{b-1} \eta_i(r)$  and  $\rho^{\mathcal{I}}(r) := \sum_{i=0}^{b-1} \eta_i^{\mathcal{I}}(r)$  are the number of cells (=occupied channels) at  $r$  before and after interaction, respectively.  $\delta(\cdot, \cdot)$  is the Kronecker delta. The term  $\delta(\rho(r), \rho^{\mathcal{I}}(r))$  prevents the loss or gain of cells during interaction.

For the actual simulations we utilize an IB-LGCA model on a two-dimensional square lattice of size  $101 \times 101$  with 4 velocity and 4 rest channels, i.e.  $b = 8$ . The velocity

channels are  $c_0 = (1,0)^T$ ,  $c_1 = (0,1)^T$ ,  $c_2 = (-1,0)^T$  and  $c_3 = (0,-1)^T$ . The neighborhoods are chosen as  $\mathcal{N}_r = \{r + e : e \in \{(1,0)^T, (2,0)^T, (1,1)^T, (0,1)^T, (0,2)^T, (-1,1)^T, (-1,0)^T, (-2,0)^T, (-1,-1)^T, (0,-1)^T, (0,-2)^T, (1,-1)^T\}\}$ . We start at time  $k = 0$  at the center  $r_0 = (50,50)^T$  of the lattice with a circular area of radius 10 completely filled with cells, i.e. all channels occupied. The rest of the lattice is empty. We have chosen periodic boundary conditions. The parameters  $\alpha, \beta$  are chosen randomly for every cell on the lattice with a Gaussian distribution with means  $\langle\alpha\rangle, \langle\beta\rangle$  and variances  $\sigma_\alpha^2, \sigma_\beta^2$ . We compare simulations by measuring the center of mass  $r_{ctr} = (r_{ctr,x}, r_{ctr,y})^T$  of all cells given by:

$$r_{ctr} = \frac{1}{N} \sum_{r \in \mathcal{L}} (r - r_0) \sum_{i=0}^{b-1} \eta_i(r),$$

where  $N = \sum_{r \in \mathcal{L}} \sum_{i=0}^{b-1} \eta_i(r)$  is the number of all cells on the lattice. Since  $g_{ta} = (1,0)^T$  and therefore  $r_{ctr,y} \approx 0$  only  $r_{ctr,x}$  is of interest here. In the following, we focus on the influence of heterogeneous adhesivities and will therefore fix the taxis-sensitivities by setting  $\sigma_\beta^2 = 0$ .

### 3. Results & discussion

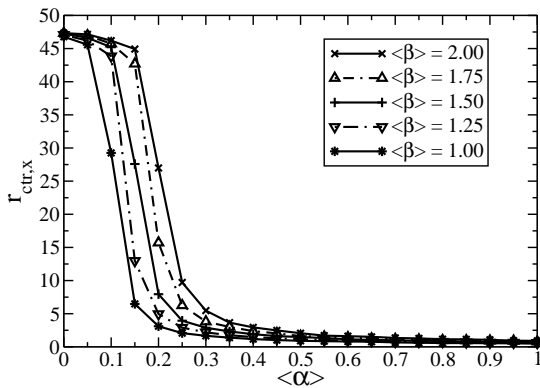


Fig. 1  
DEVELOPMENT OF  $r_{ctr,x}$  AFTER 1000 TIME STEPS FOR  
 $\langle\beta\rangle \in \{1.0, 1.25, 1.5, 1.75, 2.0\}$  AND  $\sigma_\alpha^2 = 0$

Figure 1 illustrates the dependence of  $r_{ctr,x}$  on the adhesivity  $\langle\alpha\rangle$  for different taxis-sensitivities  $\langle\beta\rangle$  for identical cells ( $\sigma_\alpha^2 = 0$  and  $\sigma_\beta^2 = 0$ ). It can be observed that increasing the adhesivity  $\langle\alpha\rangle$  leads to a decrease in  $r_{ctr,x}$ . Furthermore, the value of  $r_{ctr,x}$  decreases abruptly around a critical value  $\alpha_{crit}(\langle\beta\rangle)$  (e.g.  $\alpha_{crit}(2.0) \approx 0.2$ ). This behavior holds for all  $\langle\beta\rangle$  shown in Figure 1, lower values of  $\langle\beta\rangle$  only shift the critical value  $\alpha_{crit}$  to the left. The curves depicted in Figure 1 allow a clear separation into a fast migration phase for  $\langle\alpha\rangle < \alpha_{crit}(\langle\beta\rangle)$  and a slow migration phase for  $\langle\alpha\rangle > \alpha_{crit}(\langle\beta\rangle)$ . The shift between both phases appears similar to a phase transition.

Figure 2 illustrates the dependence of  $r_{ctr,x}$  on the adhesivity  $\alpha$  for different  $\sigma_\alpha^2$  for a fixed taxis-sensitivity of  $\langle\beta\rangle = 1.0$ . It can be seen that with increased  $\sigma_\alpha^2$  the abrupt decrease of  $r_{ctr,x}$  around  $\alpha_{crit}$  vanishes. Figure 2 shows that the introduction of individual cell differences can change the qualitative behavior. The simulations shown in Figure 2 differ only in the standard deviation  $\sigma_\alpha^2$ . Since the quantity  $r_{ctr,x}$  is already the average over all cell positions one might expect no qualitative difference between simulations with the same  $\langle\alpha\rangle$ . Figure 2, however, demonstrates that this is not the case. The clear separation between two distinct phases appears to arise from the presence of identical cells and is lost with increasing  $\sigma_\alpha^2$ .

Biological systems usually show stochastic differences in cell properties, even for cells of the same cell type. From Figure 2, it can be concluded that mathematical models which disregard stochastic differences in cell properties and consider only average values might not capture the actual behavior of biological systems.

We have introduced an extended LGCA modeling approach called individual-based lattice-gas cellular automata (IB-LGCA) that enables us to consider systems with individual cells in LGCA and to study the influence of heterogeneities in interacting cell populations

In this manuscript, we have investigated a rather simple combined adhesion/taxis model system of cell migration in two dimensions and focused on the role of adhesive heterogeneities. Our results demonstrate that models of biological systems that disregard stochastic differences in cell properties in favor of average values might not capture the actual system behavior. Investigating of LGCA models in higher dimensions and with different cell interactions might lead to further insight into the role of heterogeneity on the behavior on interacting cells.

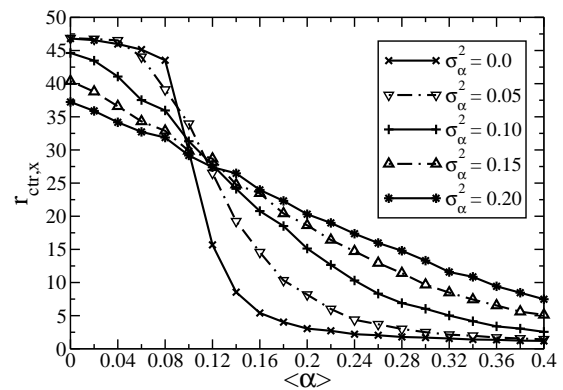


Fig. 2  
DEVELOPMENT OF  $r_{ctr,x}$  AFTER 1000 TIME STEPS FOR  
 $\sigma_\alpha^2 \in \{0.0, 0.05, 0.1, 0.15\}$  AND  $\langle\beta\rangle = 1.0$

**c) Acknowledgements:** We are grateful to Anja Voss-Boehme for fruitful discussions. We acknowledge support by the German Ministry for Education and Research (BMBF) through grant no. 0315734 and by the Human Frontier Science Program (HFSP) through grant no. RGP0016-2010. Andreas Deutsch is a member of the DFG Research Center for Regenerative Therapies Dresden Cluster of Excellence and gratefully acknowledges support by the Center.

## References

- [1] Mads Kaern, Timothy C. Elston, William J. Blake, James J. Collins, "Stochasticity in gene expression: from theories to phenotypes", *Nature Review Genetics* 6:451-464, 2005.
- [2] David G. Spiller, Christopher D. Wood, David A. Rand, Michael R. H. White, "Measurement of single-cell dynamics", *Nature* 465: 736-745, 2010.
- [3] Peter Friedl, "Prespecification and plasticity: shifting mechanisms of cell migration", *Current Opinion in Cell Biology* 16:14-23, 2004.
- [4] Bastian Chopard, Rafik Ouared, Andreas Deutsch, Haralambos Hatzikirou, Dieter Wolf-Gladrow, "Lattice-gas cellular automaton models for biology: from fluids to cells", *Acta Biotheoretica* 58(4):329-340, 2010.
- [5] Andreas Deutsch, Sabine Dormann, "Cellular automaton modeling of biological pattern formation", Birkhauser, Boston, 2005

# Elastic Dimer Automata: Discrete, Tunable Models for Complex Systems

Dustin Arendt<sup>1</sup> and Yang Cao<sup>1</sup>

<sup>1</sup>Department of Computer Science, Virginia Tech, Blacksburg, VA, USA

**Abstract**—*Cellular automata, dimer automata, and other similar models are useful tools for understanding complex phenomena using simple rules and discrete states. They have simple implementations that run efficiently without problems such as numerical stability and round off errors that are common with continuous models like PDE's. However, these discrete models, in general, lack a mechanism to tune the desired level of detail or accuracy, an important feature of PDE's. We propose elastic dimer automata as a general way to reconcile the simplicity of discrete models with the tunable nature of continuous models. Furthermore, we present a simple method for measuring self-organization in elastic dimer automata which we use to aid in an exhaustive search for interesting rules. This search revealed simple rules for several interesting phenomena including multiple different types of wave phenomena as well as a mechanism for labyrinthine patterns and dislocation repairing.*

**Keywords:** self-organization, continuous behavior, dimer automata, asynchronous cellular automata, graph

## 1. Introduction

The discrete nature of cellular automata [1], [2], boolean networks [3], and other models is both a blessing and a curse. Discreteness allows for a simpler implementation on a digital computer compared to their continuous counterparts, such as PDE's. The rules describing the dynamics of discrete systems are often easy to understand and implement, and there is no cause to worry about round off error or numerical stability. In many cases, the state of the system can be represented with 8-bit integers, using less memory and CPU resources compared to the floating point representations needed by continuous models. Unfortunately, cellular automata and other models typically lack an important feature common to numerical solutions to PDE's: a tunable level of detail. Adjusting the space and time step in PDE's allows one to maximize accuracy within the constraints of available memory and CPU resources. This is not the case when we consider classical cellular automata; there is not usually an obvious way to add states to the rule in a way that “sharpens the picture.” Normally, we can increase the length or size of the simulation, but this does not necessarily equate to a more accurate result.

So, is there a way to reconcile the stable nature of discrete models with the tunable nature of continuous ones? There

are many ad-hoc examples of this where cellular automata rules are accompanied by a parameter to determine the number of states used. This effectively tunes the model to exhibit smoother, more continuous behavior. For example, the Greenberg-Hastings [4] and cyclic cellular automata [5], [6] models for excitable media allow an arbitrary number of integer states, which affects the size of the spirals in the simulation. Other approaches have focused directly on cellular automata as a direct discretization of various PDE's. For example, a coarse discretization of the Belousov-Zhabotinsky reaction was very effective in reproducing the important qualitative aspects of that phenomena, including wave curvature and dissipation [7]. Weimar developed a technique for a quantitatively accurate cellular automaton discretization of reaction diffusion systems [8]. A similar technique based on operator splitting is used by Narbel for cellular automaton discretization of the Fitzhugh-Nagumo equation [9]. Roughly speaking, these techniques carefully transform the continuous phase space of various PDEs into a discrete map to be used as the cellular automaton rule.

These techniques map well known continuous models into cellular automata, but is it possible to accomplish the reverse? Can we start with an arbitrary, coarse-grained, discrete rule and develop it into a continuous model? This has been accomplished for elementary cellular automata using the “inverse ultradiscretization technique” [10]. However, this technique essentially maps the cellular automaton rule into a set of continuous functions that exactly reproduce the discrete behavior of the rule. A more useful and intuitive approach would be to increase the automaton's state space one state at a time so that in the infinite limit of states the rule behaves continuously, but is still modeled by discrete state transitions.

In previous work we used this approach to create rules for grain growth and spiral waves [11], [12]. We showed that the original 3-state rules could be grown in a straightforward manner preserving the topological properties of these rules. These generalized rules were engineered by studying the behavior of the original rules; we intuited a way to modify the rules to produce their appropriate behaviors. However, a general technique to produce continuous behavior from any seed rule would be very useful. For example, this would enable searching within a finite space of continuous behaviors with the potential to discover simple new explanations for complex phenomena, which is one of the main contributions

of this paper. We propose a generic tunable discrete model for complex systems called elastic dimer automata that we use to search for simple rules exhibiting self-organization.

Elastic dimer automata allow stretching (i.e. increasing of the state space) to produce smooth, continuous behavior in a generic manner, overcoming the issues discussed above. We also develop a simple statistical method to measure self-organization in dimer automaton configurations having many states. Together, elastic dimer automata and this measurement let us perform an exhaustive search of the resulting behavior space, yielding several simple interesting rules. The rules discovered included several variants of excitable media phenomena with various wave behaviors and labyrinthine patterns that repair dislocations over time.

## 1.1 Dimer Automata

For this paper we only consider dimer automata, a fully discrete, asynchronously updating model for complex systems [13]. Space is represented as a graph  $G = (V, E)$ . Each vertex in the graph has a single state from a discrete state space  $\Sigma$ . Dimer automata are iterated by choosing a random edge in the graph and updating both endpoints of that edge<sup>1</sup> according to a simple rule  $R$ . Assuming  $G$  is undirected lets us assume the effect of  $R$  is symmetric. In other words, if  $(a, b) \rightarrow (c, d)$  then  $(b, a) \rightarrow (d, c)$ , which gives  $R$  a slightly simpler form (i.e.  $R : \Sigma^2 \mapsto \Sigma$  instead of  $R : \Sigma^2 \mapsto \Sigma^2$ ), thus

$$(x, y) \rightarrow (R(x, y), R(y, x)).$$

This simplification allows  $R$  to be considered a finite state machine whose input alphabet and state space are both  $\Sigma$ . In this paper we often refer to the rule's topology, which is the directed graph defined by the state transitions of the finite state machine.

Dimer automata have several nice properties to justify their use here over traditional models such as cellular automata. Representing space as a general graph is much less restrictive than the uniform lattice often assumed (out convenience) for cellular automata. Since, by definition, all edges in a graph have exactly two endpoints, and the rule operates on these endpoints, any rule can be applied to any spatial structure without modification. This is only easily accomplished in cellular automata by assuming the rule is totalistic, or in boolean networks by assuming each vertex has the same degree. Also, the rule is a function of two states, the fewest number of states needed to model interactions, and one less state than elementary cellular automata [1]. Despite the challenges resulting from operating asynchronously over general graphs, in previous work we showed that dimer automata have efficient parallelizations assuming a low-dimensional graph [14].

<sup>1</sup>This occurs by replacing the states of the endpoints of an edge with new states, simultaneously. So,  $(a, b) \rightarrow (c, d)$  is to be read as "states  $a$  and  $b$  are replaced with states  $c$  and  $d$ ."

## 1.2 Self-Organization

Since the goal is to search for dimer automata exhibiting self-organization, it is important to have a way to quantify self-organization for a number of reasons. Presumably, in the search we will be considering a large number (thousands or more) of dimer automata. The least sophisticated search is one in which a human views each output, and manually classifies them. This was the approach initially taken and still endorsed by Wolfram in his search of elementary cellular automata [1]. However, this approach quickly becomes impractical when one must consider many outputs. So, at the very least, a measurement of self-organization can be used to organize the output to make this task easier. For example, outputs can be sorted according to this value so that the best results are presented first, or a threshold can be set so that only the images exceeding this value are presented. More sophisticated techniques can build on this by replacing the role of the human with a computer algorithm.

Over the years, there have been many approaches towards understanding and measuring self-organization [15], [16], [17], [18], [19], [20]. This is partially due to the term "self-organization" itself being overly ambiguous, and partially due to the many different disciplines interested in the phenomena (e.g., physics, mathematics, computer science) [18]. It is generally accepted that a self-organizing system transforms an initially unstructured configuration in to one with more structure over time. There is no clear consensus on what good definitions of structure are, and how its increase over time should appear. However, there is general consensus that the thermodynamic concept of entropy is a very poor choice for measuring self-organization because self-organizing systems have elements of both order and disorder [17]. Neither a purely homogenous configuration (with zero entropy) nor a purely random configuration (with maximal entropy) are very interesting. Thus, an appropriate measurement will have large values for structured configurations and small values for fully ordered as well as fully disordered configurations.

Our measurement of self-organization makes several assumptions for simplicity and utility. Almost all measurements consider the rate of change of the complexity of the system over time, however, we only consider the final configuration of the dimer automaton. We let the dimer automaton be initialized with a fully random state (which contains zero structure), so we can assume any resulting structure is a sign of self-organization. Measuring the self-organization of dimer automata should also take into account the local, spatial distribution of states. The amount of information that one part of a configuration tell us about another part should depend on how far away those parts are, motivating our information theoretical approach.

Let  $D_R$  and  $D_G$  be random variables representing the distance between two vertices' states and spatial locations respectively. If  $i$  and  $j$  are vertices chosen uniformly at

random from the graph  $G$ , then

$$\begin{aligned} D_R &= d_R(x_i^t, x_j^t), \\ D_G &= d_G(i, j). \end{aligned} \quad (1)$$

The functions  $d_R$  and  $d_G$  refer to the finite state machine and graph shortest path distances<sup>2</sup>, respectively. We propose measuring local structure by their mutual information,

$$\mathcal{L}_I = I(D_R; D_G). \quad (2)$$

This measurement of local structure agrees well with our intuition about structure in the uniform and uniformly random cases. In either of these two cases,  $D_R$  and  $D_G$  are statistically independent, so the mutual information is 0. This measurement is similar to the concept of long range mutual information [16], and is a simplified and adapted version of Shalizi's "light cone" approach [20]. Also, we note that our measurement is very general, as it applies to any configuration represented by a graph where vertices take discrete states. This means it can potentially be used for other complex systems models as long as  $d_R$  is appropriately defined.

A visualization of this distribution is shown in Figure 1 where each pixel  $(i, j)$  in the image is proportional to  $Pr[D_R = i, D_G = j]$ . This compares the joint probability distributions before and after the creation of spiral waves from an excitable media model. The first joint probability distribution shows no overall structure, which is intuitive since the initial condition is totally random. However, the second distribution has several interesting features. Most notably, there is a dark triangle in the upper left corner, implying that  $Pr[D_R > D_G] \approx 0$ . Roughly speaking, this means that in this case the topological distance is an upper bound on the distance between any pair of states. Finally, using Equation 2, the mutual information is 0.021 in the initial configuration and 0.574 in the final configuration.

## 2. Methods: Elastic Dimer Automata

Our primary goal is to formally define a technique for creating arbitrary generalizable rules; the technique we develop here we refer to as elastic dimer automata. These are rules that are created through a two step process where an initial graph is stretched and its edges are labeled to produce the finite state machine for a dimer automaton rule. In the first step, an initial graph,  $G_0$  is stretched  $s$  times through an edge rewriting process to produce a sequence of graphs  $\{G_1, G_2, \dots, G_{s-1}, G_s\}$ . Edges are rewritten according to

$$\begin{aligned} \overleftrightarrow{ij} &\rightarrow \{\overleftrightarrow{ik}, \overleftrightarrow{kj}\}, \\ \overrightarrow{ij} &\rightarrow \{\overrightarrow{ik}, \overrightarrow{kj}\}, \end{aligned} \quad (3)$$

<sup>2</sup>Since  $G$  is potentially very large, it is not wise to compute  $d_G(i, j) \forall (i, j) \in V^2$ . Instead,  $(i, j)$  pairs can be sampled by repeatedly picking a random vertex, performing a breadth first search of fixed depth, and picking a random vertex from each depth.

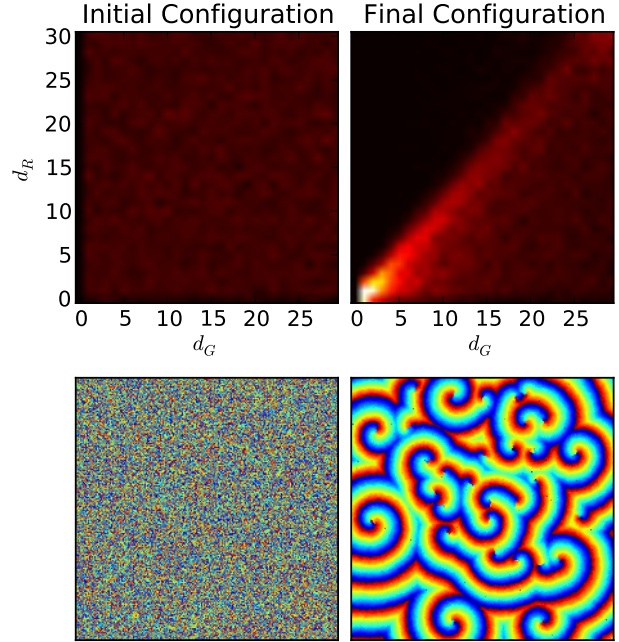


Fig. 1: Joint probability distribution (top) before and after simulation of excitable media (bottom).

and  $k$  is added to  $V$ . All edges are rewritten in parallel so that the graphs grow uniformly, and in a manner that preserves (qualitatively) the original shape. Thus, these operations result in  $G_s$  being, qualitatively speaking, a stretched version of the initial graph  $G_0$ , an example of which is shown in Figure 2.

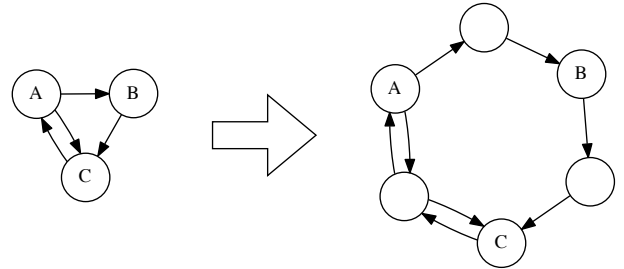


Fig. 2: One iteration of the stretching operation for an example graph.

We let  $G_s$  define the set of allowable transitions in the finite state machine for each state. Let  $N[i]$  be the inclusive neighborhood of vertex  $i$  defined by  $G_s$ . Thus  $N[i]$  is the set of states that  $i$  may transition to (including remaining as  $i$ ). Deciding what causes this transition allows the edges of the finite state machine to be labeled, creating the dimer automaton rule  $R$ . Suppose that for each  $(i, j) \in \Sigma^2$  there is a configuration energy defined by  $J(i, j)$ . The rule  $R$  is defined by picking a pair of states from the set of allowable

transitions that minimizes the configuration energy, thus

$$(R(x, y), R(y, x)) = \min_{(i, j) \in N[x] \times N[y]} J(i, j) \quad (4)$$

However, this equation creates some ambiguity if there is more than one pair of transitions tied for the minimum. There are a number of potential ways to handle this tie. One way is to simply disallow transitions when there is not a unique minimum. Another option could be to swap  $i$  and  $j$  (we will refer to this as zero-swapping). The logic is that swapping  $i$  and  $j$  neither increases nor decreases that edge's configuration energy. Swapping injects enough energy into the system to avoid totally frozen configurations, where no transitions are occurring. Yet another approach is to pick the  $(i, j)$  corresponding to the smallest unique energy state. This is analogous to a ball placed on the top of a hypothetical mountain. If both slopes of the mountain are equally steep, the ball will roll down the ridge between the two slopes, even though this route is less optimal. We use a combination of this approach and swapping, where swapping is performed if there is no better unique energy configuration.

We let  $J$  be a function of the distance between  $i$  and  $j$  in  $G_s$  such that  $J(i, j) = F(\min[d_R(i, j), d_R(j, i)])$ . Note that  $J$  is symmetric, which is important since we are still assuming space is undirected. For simplicity we let  $F(d) = d$  so that the configuration energy of  $(i, j)$  increases as  $i$  and  $j$  move farther apart in  $G_s$ . Under the above assumptions, the rule  $R$  for an elastic dimer automaton is fully defined by  $G_0$  and  $s$ .

### 3. Results: Searching for Self-Organizing Elastic Dimer Automata

The previous assumptions give us a good starting point to search for interesting elastic dimer automata, with the search space being transformed to the set of all directed graphs. However, this space grows very quickly since there are  $2^{|V|^2}$  possible adjacency matrices for a directed graph with  $|V|$  vertices. Fortunately, the search space can be further reduced by assuming that each graph contains no self loops, is strongly connected, and is isomorphically unique. It is not useful to allow graphs with self loops because a self loop would specify a state could transition to itself, which is already allowed by Equation 4, since  $N[x]$  includes  $x$ . A strongly connected graph is one in which there is a path of finite length between every pair of vertices. In other words, every vertex is recurrent, so no vertices are absorbing/transient (out/in-degree are zero). Because transient states have no incoming edges, the dimer automaton's configuration may eventually contain no such states. On the other hand, once part of the dimer automaton's configuration is absorbing, it can never change to another state. Finally, it is useful to discard rules that are isomorphically equivalent to others since they perform identical jobs.

First, we conducted an exhaustive search of the space of elastic dimer automata where  $|V_0| = 4$ , which has 89 unique rules under the above assumptions<sup>3</sup>. For each of these rules, we start with random initial conditions run a simulation long enough for the system to exhibit its long term behavior<sup>4</sup>. In each simulation, the spatial topology used was an isotropic planar graph, roughly equivalent to a  $600 \times 600$  square lattice. From this set of 89 rules, we identified four interesting and unique forms of self-organizing behavior: rules<sup>5</sup> 13, 18, 55, and 85. Figure 3 shows the information  $\mathcal{L}_I$  and energy  $\mathcal{H}$  of these four rules over time (with and without swapping) and Figure 4 shows the topology of each rule and its configuration after a long time. The configuration energy is the average energy of each edge in the graph using the above definition of  $J$ , thus

$$\mathcal{H} = \frac{1}{|E|} \sum_{\vec{ij} \in E} J(x_i, x_j). \quad (5)$$

Rule 13 produces fast moving spiral waves and swapping produces slightly larger waves in this case, but there is not a significant change in the qualitative behavior or in the information-energy curve. Rule 18 shows a rapid decrease in energy relative to the more gradual trends seen in the other rules. Furthermore, in both cases, there is a spontaneous increase in the configuration energy eventually following the initial sudden drop. This increase is then followed by yet another drop, where the system eventually settles down. With swapping, rule 18 produces a combination of quickly moving wave fronts that leave a wake of slowly decaying uniform regions. Without swapping, the behavior is characterized by rounded homogeneous regions that spontaneously transition to a different state. Rule 55 appears to be a frustrated version of a rule 13. The length scale of the waves are much larger, and spirals are less fully developed than those seen in rule 13. Swapping allows the waves to propagate more quickly, but does not appear to affect the qualitative behavior. Rule 85 is perhaps the most interesting, producing labyrinthine patterns without zero-swapping (an example of this is shown in Figure 5). Its configurations are characterized by dislocations, which gradually disappear over time, resulting in a low energy, high information configuration. Swapping allows the dislocations to be fixed more quickly, but causes the information to decrease eventually, perhaps because of the creation of large, nearly uniform regions. Once the labyrinthine pattern is formed, removing dislocations drives the system closer to a uniform state, decreasing its information content.

<sup>3</sup>Without these assumptions, we would be considering  $2^{16}$  rules.

<sup>4</sup>Experimentally, we found that  $300 \cdot |V|$  edge updates were sufficient.

<sup>5</sup>The rule's number refers to the order in which the algorithm generates each graph.



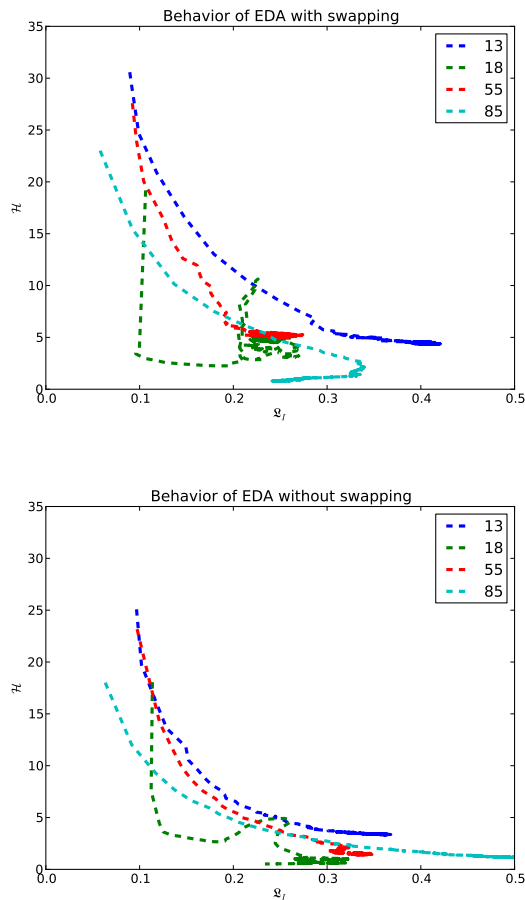


Fig. 3: Information-Energy time series of rules 13, 18, 55, and 85 with (top) and without (bottom) swapping.

#### 4. Discussion

Our intention in creating elastic dimer automata is to couple the tunable level of detail of continuous models such as PDEs with the simplicity, speed, and stability of discrete models such as cellular automata. The results thus far are promising, with elastic dimer automata producing a variety of interesting phenomena with continuum-like behavior. The stretching operations have the intended effect of tuning the level of detail in the automaton. Stretching the initial graph increases the characteristic length scale of the structures produced by the elastic dimer automaton. An example of this is shown in Figure 5 for rule 85 without swapping. Stretching causes structures take up more space, and require more time to develop. Finally, we note that random asynchronous updating is a closer approximation of continuous time than the synchronous method used by cellular automata since state changes are discrete. Synchronous updating in cellular automata moves the system forward in one large leap because state is discrete, as opposed to synchronous updating in PDE's where state is continuous, and is updated by very

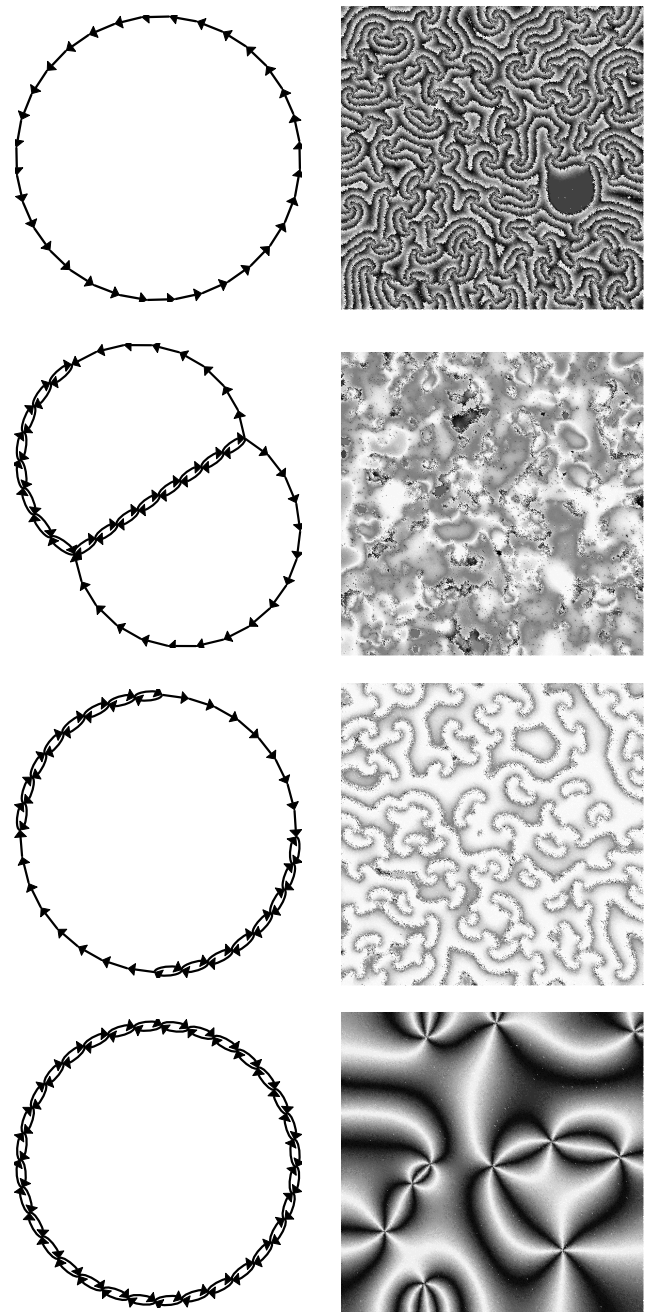


Fig. 4: Elastic dimer automaton 13, 18, 55, and 85 state transition topology (left) and resulting configurations (right) when swapping is allowed.

tiny increments. So we conclude that stretching elastic dimer automata is analogous to simultaneously tuning their space and time step, which provides the tunable level of detail we have sought in their design.



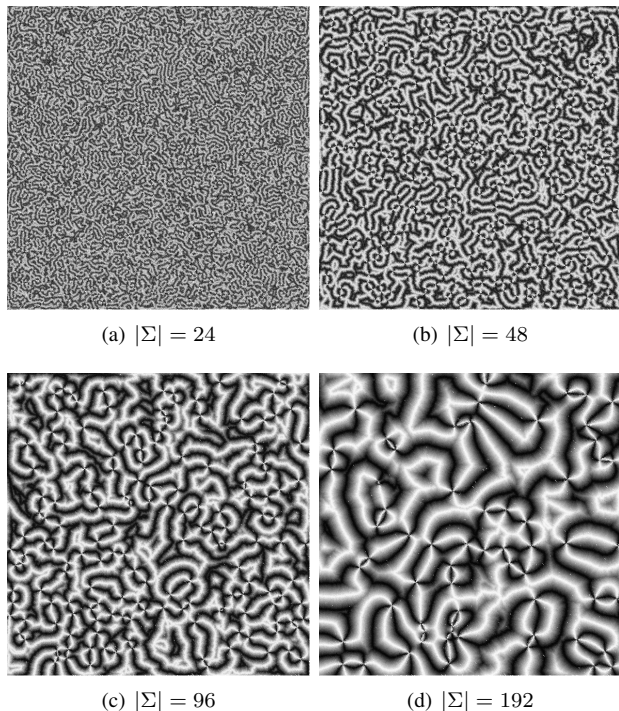


Fig. 5: Stretching increases the level of detail without qualitatively affecting its behavior (rule 85 without swapping is shown).

## 4.1 Rendering

Although the mutual information is useful in identifying configurations with structure, it is still important to render the configurations so they can be interpreted visually. The naïve approach of using a grayscale value directly proportional to each state will not work. The reason is that there may be little to no correlation between a state's value and its context in the finite state machine. So, it is necessary to develop an alternative way of rendering configurations. Recall that elastic dimer automata are constructed from an initial seed graph  $G_0$ , and that the indices of this original graph map directly to their corresponding indices in the final stretched graph  $G_s$ . Therefore, we let the color  $c(i)$  corresponding to a state  $i$  in  $G_s$  be dependent on that state's distance to one of the original vertices such that

$$c(i) = \min_{j \in V_0} d_R(i, j). \quad (6)$$

In other words, a state's color will be proportional to that state's shortest distance to one of the original vertices from  $V_0$ . The effect is to accentuate regions of near-equal state, and for smooth state transitions in space to appear as gradients. An example of the usefulness of this transformation is shown in Figure 6 compared to the naïve case where  $c(i) = i$ .

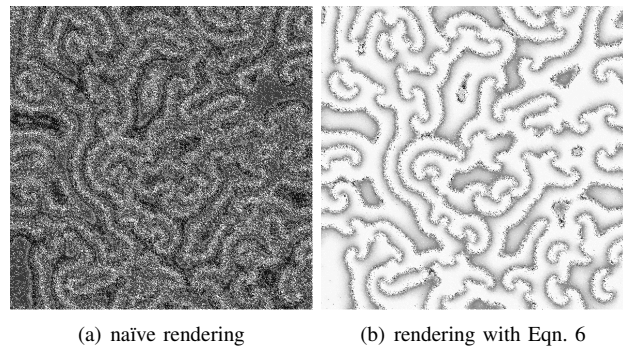


Fig. 6: A comparison of two rendering techniques for dimer automata.

## 4.2 Distribution of Behaviors

An interesting question to ask for this system is, what is the distribution of elastic dimer automata behaviors in the information-energy space? To approach this, we extended the search from the previous section by one, so that  $|V_0| \leq 5$ . This resulted in a total of 5137 suitable unique rules. The information and energy of the final configuration of each rule is plotted in Figure 7. Additionally, we used k-means clustering to pick 9 characteristic rules; the configurations of each of these rules are also shown in the plot. The information-energy distribution has several interesting features that become apparent as a result of the larger search space. First, there appears to be a clear upper bound on the ratio of information to energy, so we hypothesize that

$$\mathcal{H} = O(\mathcal{L}_I^{-1}), \quad (7)$$

as evidenced by the emptiness of the upper right portion of the plot. Furthermore, there appears to be another empty region where  $\mathcal{L}_I \in [0, 0.1]$  and  $\mathcal{H} > 1$ . This suggests, at least for this experiment, a correlation between very low information and very low energy. In other words, the configurations in this region are mostly uniform, and not mostly random. The tail of the distribution where  $\mathcal{L}_I \geq 2.5$  corresponds to low energy and high information configurations that result from spiral waves (e.g. rule 13). The remaining rules that generate high information configurations in the distribution appear to be somewhat similar to rule 18.

## 5. Conclusion

We have presented a technique, elastic dimer automata, that when given a small (e.g. granular) state transition graph, creates a dimer automaton rule in a manner that shows successively more continuous like behavior. Additionally, we have developed a measurement of self-organization tailored towards these rules based on the mutual information between state and space distance distributions. Using these tools, we performed an exhaustive search of a simple class of elastic

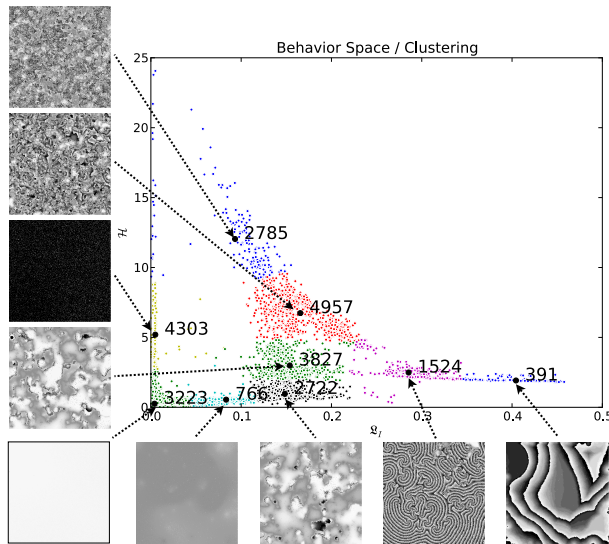


Fig. 7: Clustering of behavior resulting from the 5137 elastic dimer automata with  $|V_0| \leq 5$

dimer automata which revealed several interesting rules, and a rich behavior space.

Future work can take several directions. One question we may ask is, can we derive the PDE corresponding to a given elastic dimer automaton directly by considering the infinite limit of states? This may be simple for cases such as rules 13 and 85 where the transition graph is simply a cycle, allowing a direct mapping between the discrete state and its continuous phase angle. However, in the general case, deriving the PDE may be less trivial; perhaps other variants of elastic dimer automata would facilitate this better. Furthermore, in this paper we consider only a very simple energy function to minimize; future work may consider other classes of energy functions or even domain-specific functions to model a narrower range of physical phenomena. Finally it is useful and frequently challenging to demonstrate the connection between the abstract rules for complex systems and the corresponding physical phenomena (if such a connection exists). We encourage readers to do so for interesting phenomena revealed here or any that may be found in the future.

## Acknowledgement

The work of Yang Cao was supported by the National Science Foundation under awards CCF-0726763 and CCF-0953590, and the National Institutes of Health under award GM078989.

## References

- [1] S. Wolfram, *A New Kind of Science*. Wolfram Media, Inc., 2002.
- [2] J. L. Schiff, *Cellular Automata: a Discrete View of the World*. Hoboken, NJ: Wiley-Interscience, 2008.

- [3] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 437–467, 1969.
- [4] J. M. Greenberg and S. P. Hastings, "Spatial patterns for discrete models of diffusion in excitable media," *SIAM Journal on Applied Mathematics*, vol. 34, no. 3, pp. 515–523, 1978.
- [5] R. Fisch, "Cyclic cellular automata and related processes," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1-3, pp. 19–25, 1990.
- [6] R. Fisch, J. Gravner, and D. Griffeath, "Cyclic cellular automata in two dimensions," *Spatial Stochastic Processes: A Festschrift in Honor of Ted Harris on His Seventieth Birthday*, K. Alexander and J. Watkins, eds, pp. 171–188, 1991.
- [7] M. Gerhardt, H. Schuster, and J. J. Tyson, "A cellular automaton model of excitable media including curvature and dispersion," *Science*, vol. 247, no. 4950, pp. 1563–1566, MAR 30 1990.
- [8] J. R. Weimar, "Cellular automata for reaction-diffusion systems," *Parallel Computing*, vol. 23, no. 11, pp. 1699–1715, 1997.
- [9] P. Narbel, "Qualitative and quantitative cellular automata from differential equations," in *Cellular Automata*, ser. Lecture Notes in Computer Science, S. El Yacoubi, B. Chopard, and S. Bandini, Eds. Springer Berlin / Heidelberg, 2006, vol. 4173, pp. 112–121.
- [10] W. Kunishima, A. Nishiyama, H. Tanaka, and T. Tokihiro, "Differential equations for creating complex cellular automaton patterns," *Journal of the Physical Society of Japan*, vol. 73, p. 2033, 2004.
- [11] D. Arendt and Y. Cao, "Evolutionary motifs for the automated discovery of self-organizing dimer automata," *Advances in Complex Systems*, 2012.
- [12] D. Arendt, "In search of self-organization," Ph.D. dissertation, Virginia Tech, 2012.
- [13] B. Schönfisch and K. P. Hadeler, "Dimer automata and cellular automata," *Physica D*, vol. 94, no. 4, pp. 188–204, JUL 15 1996.
- [14] D. Arendt and Y. Cao, "Effective GPU acceleration of large scale, asynchronous simulations on graphs (doi:10.1142/s021952591250035x)," *Advances in Complex Systems*, 2012.
- [15] C. H. Bennett, "On the nature and origin of complexity in discrete, homogeneous, locally-interacting systems," *Foundations of Physics*, vol. 16, no. 6, pp. 585–592, 1986.
- [16] —, "How to define complexity in physics, and why," *Complexity, Entropy, and the Physics of Information*, vol. 8, pp. 137–148, 1990.
- [17] D. Feldman and J. Crutchfield, "Measures of statistical complexity: Why?" *Physics Letters A*, vol. 238, no. 4-5, pp. 244–252, 1998.
- [18] C. Gershenson and F. Heylighen, "When can we call a system self-organizing?" *Advances in Artificial Life*, pp. 606–614, 2003.
- [19] C. R. Shalizi, K. L. Shalizi, and R. Haslinger, "Quantifying self-organization with optimal predictors," *Physical Review Letters*, vol. 93, no. 11, p. 118701, 2004.
- [20] C. R. Shalizi and K. L. Shalizi, "Quantifying self-organization in cyclic cellular automata," *Arxiv preprint nlin/0507067*, 2005.

# GPU Acceleration of Many Independent Mid-Sized Simulations on Graphs

Dustin Arendt<sup>1</sup> and Yang Cao<sup>1</sup>

<sup>1</sup>Department of Computer Science, Virginia Tech, Blacksburg, VA, USA

**Abstract**—Many GPU parallelizations exist to speedup simulation of complex systems, but these approaches see less benefit when the simulation is not large. Simulation of many independent complex systems is useful for Monte Carlo sampling or for exploring the behavior of many different models at once. We present and evaluate an algorithm for simulating many mid-sized dimer automata (e.g., having tens of thousands of vertices) on the GPU. Our algorithm has, in the best case, a throughput of over 300 million edge updates per second, and a speedup of over 37 on modest GPU hardware. Dimer automata can also be used to design and implement useful computations on graphs. As a test case, we implement a solution to the all pairs shortest path problem using dimer automata and our GPU algorithm, but find that the structure of the graph has a significant effect on the efficiency of the algorithm.

**Keywords:** GPGPU, dimer automata, asynchronous cellular automata, graph, Monte Carlo methods

## 1. Introduction

Often it is necessary to perform many independent trials (e.g., Monte Carlo sampling) to understand systems, especially stochastic complex systems. These independent trials can be characterized by the following cases:

- 1) varying the random seed to observe a distribution of behaviors;
- 2) varying the initial condition to observe how it affects the outcome;
- 3) varying the model or its parameters to observe a variety of different behaviors; or
- 4) some reasonable combination of the above cases.

An example of the first case is in well mixed systems of reacting chemicals in low concentration, which is modeled exactly by Gillespie's algorithm [1]. In these systems sometimes the same initial condition results in two drastically different outcomes. A famous example of this is the "developmental bifurcation pathway in phage lambda-infected e. coli cells" [2]. In this case just a handful of chemical species with low population randomly determines whether the phage enters lysis or lysogeny. Lysis and lysogeny are two global attractors that are both reachable from the same initial condition. The second case, varying the initial condition, can provide useful information about a systems dependence on its initial conditions (i.e., is the system chaotic?).

Wolfram's classical search of elementary cellular automata is a famous example of the third case (varying the model) [3]. He ran a simulation for each of the 256 different rules starting from simple initial conditions and observed four classes of behaviors. Automata belonging to two of these classes exhibited surprisingly complex behaviors despite their simple formulation. Since this original work it has been popular to perform various similar types of searches for other complex systems [4], often aided by genetic algorithms [5], [6], [7], [8], [9]. In all of these cases, the simulation of many independent complex systems is crucial.

Clearly, any technology that improves the efficiency of one or more of the above cases is of general interest, and this is the contribution of this paper. We present an algorithm for the GPU-acceleration of many concurrent dimer automata simulations that handles any combination of cases 2 and 3 mentioned above. Dimer automata are discrete (in state, space, and time), stochastic, asynchronous dynamical systems that can be used to model a number of useful phenomena [10], [11], [12]. One of the useful properties of dimer automata is that they operate on arbitrary graphs, updating both endpoints of a single randomly chosen edge simultaneously. So, in addition to providing a nice modeling and simulation framework for complex systems, dimer automata can be used to design randomized, decentralized, and robust algorithms for computing on graphs. We benchmark and discuss the algorithm we present, and we also consider what circumstances this GPU acceleration of dimer automata can be effectively used for general graph computations.

### 1.1 Dimer Automata

Dimer automata assume interactions occur on an arbitrary graph  $G = (V, E)$  where each vertex in that graph has a single state from the state space  $\Sigma$ . The state of a vertex  $u$  after  $t$  edge updates is  $x_u^t$ . The system is iterated by picking an edge  $\overleftrightarrow{uv} \in E$  uniformly at random and updating the states of *both* endpoints of that edge simultaneously. The endpoints are replaced according to the rule  $R$ ; if  $G$  is undirected then we can make the simplifying assumption that  $R : \Sigma^2 \mapsto \Sigma$ . If  $x$  and  $y$  are the states of the endpoints of an edge to be updated,  $x$  and  $y$  are replaced such that

$$(x, y) \rightarrow (R(x, y), R(y, x)). \quad (1)$$

The rule  $R$  may be represented as an algebraic expression or as a matrix (lookup table). If we consider finite, discrete

$\Sigma$ , then  $R$  is a finite state machine (FSM), and can be represented as a  $|\Sigma| \times |\Sigma|$  matrix. An example of this state transition matrix is,

$$R(x, y) = \begin{array}{c|cc} x \backslash y & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 0 & 1 \end{array}, \quad (2)$$

which defines the output of  $R$  for every possible pair of inputs. This example results in the state of  $x$  and  $y$  being swapped, which is an elegant way to implement diffusion of particles. Many other simple dimer automata rules with useful applications have been found, and are shown in Table 1.

One difference in this presentation of dimer automata from the original (see [10]) is the assumption that  $G$  is undirected, allowing  $R$  to be simplified based on an argument for symmetry. However, it is reasonable in some cases to use a directed graph in combination with rules that break symmetry. The updating rule use is modified to reflect this, so if  $\vec{uv}$  was the directed edge chosen to be updated, then  $(x, y) \rightarrow R(x, y)$  but,  $R : \Sigma^2 \mapsto \Sigma^2$ . For consistency, symmetry breaking rules should always be used with the convention that the state of the source vertex is the first argument of  $R$ , and the state of the destination vertex is the second.

## 1.2 General Purpose GPU Programming

Parallel computing can often be used to improve the efficiency of simulations, but in the past relatively few people had access to the hardware necessary to see much benefit from the use of parallelism. This has changed only in the last decade for two important reasons: powerful GPU's became ubiquitous on personal computers and SDK's such as OpenCL and CUDA made general purpose programming on GPU hardware relatively straightforward. As a result there has been a rapid increase in the use of the GPU's for scientific computing within the past ten years [13]. An in depth discussion of GPU architecture and topics is readily found in the literature, so we only give a brief summary here. When designing a GPGPU algorithm, one must subdivide the problem into identical tasks in a manner that facilitates coalesced memory access. Each task uses the same code (called the kernel), but sees a different part of the data; this approach is called single instruction multiple data (SIMD). Coalescing memory access is accomplished (in very general terms) by ensuring that task  $i$  accesses memory location  $ci$  (where  $c$  is the "stride") for each task. This allows the GPU to amortize memory latency across many threads. Without designing for coalesced memory access, it is difficult to achieve significant speedup from the GPU parallelization.

Many examples of GPU accelerated simulations of complex systems can easily be found in the literature [14], [15], [16], [17]. Many of these models (including our own recent work [11]) are designed to speed up the execution

of a single large model. However, when the size of the model decreases, the speedup diminishes. There are only a few cases where researchers have focused on the GPU acceleration of a large number of independent simulations. One approach that deserves mention due to its similarity to dimer automata is the GPU acceleration of Gillespie's algorithm for stochastic chemical kinetics [18]. In this case, each thread on the GPU performs Gillespie's algorithm independently by firing randomly chosen reactions (firing a reaction is similar to an edge update in dimer automata). This algorithm is tailored towards very small simulations (e.g., one case study had just three species and four reaction channels). It works best assuming the number of chemical species, channels, etc. is low enough that all information for all threads in a thread block fits into the limited shared memory space. This assumption reduces the cost of the random memory access inherent to simulations of stochastic chemical kinetics. Instead, our algorithm is designed to operate on many mid-sized simulations. We define mid-sized as having too many vertices to fit in shared memory, but few enough that (at least) several copies fit in global memory. We discuss the implementation details of our algorithm next.

## 2. Methods

Our goal is to develop an efficient GPU parallelization of many independent, mid-sized dimer automata. An embarrassingly parallel approach would work by running many dimer automata across as many processors and collecting the result. Since GPU's are adept at such embarrassingly parallel tasks, we might try to distribute a large number of dimer automata on the GPU in a similar manner. However, one would not see much speedup because of the random memory access patterns resulting from each dimer automaton having its own unique and random order of updates. To resolve this, we assume that every dimer automaton undergoes the same order of edge updates. This allows memory access to be coalesced, which greatly improves the efficiency of the GPU parallelization. Of course, this assumption comes with a cost; our approach cannot be applied to case 1 from the introduction. Under the same rule, initial condition, and (from the assumption) the same order of updates, a dimer automaton will always produce the same outcome. This would defeat the purpose of repeated trials as any single trial would be sufficient. Fortunately, the assumption is still reasonable for the remaining cases, so we develop a simple yet efficient GPU parallelization building on this idea.

We assume a direct mapping between threads and dimer automata simulations on the GPU. Since we are also assuming that each dimer automaton will update the same edge at the same time, we can arrange the memory layout of the experiments state so that memory access is coalesced. This is accomplished by arranging each  $x_u$  for each automaton contiguously in memory. This is the transpose of the more common approach, where the states of adjacent vertices are

Table 1: Several useful dimer automaton rules in algebraic form.

name	algebraic dimer automaton rule	notes
flocculation	$R(x, y) = \begin{cases} y & \text{if } x + y = 1 \\ 2 & \text{else if } x + y = 3 \\ x & \text{else} \end{cases}$	Initialize the center seed(s) with 2 and $\rho V $ other random vertices with 1, which are free to move. Produces a dendritic fractal. See [11]
grain growth	$R(x, y) = \begin{cases} y & \text{if } x = 0 \\ 0 & \text{else if } x \neq y \wedge x, y \neq 0 \\ x & \text{else} \end{cases}$	Initialize vertices from a set of at least 3 different random integers. Creates competing domains/grains that grow over time. See [11]
Schelling segregation	$R(x, y) = \begin{cases} y & \text{if } x, y \leq 0 \\ -x & \text{else if } (x > 0 \wedge x \neq  y ) \vee x = - y  \\ x & \text{else} \end{cases}$	Similar to grain growth, but with conservation of mass. Domains are separated by 0, a buffer state. See [11]
excitable media	$R(x, y) = \begin{cases} x + 1 & \text{if } 1 \leq y - x \leq z \\ x & \text{else} \end{cases} \pmod n$	Self-organizing spiral waves. See [11]
infection	$R(x, y) = \begin{cases} x - 1 & \text{if } x > 0 \\ n & \text{else if } y > \beta \\ x & \text{else} \end{cases}$	Initialize at least one vertex with $n$ , and the remaining with 0. On a small world graph and the appropriate $\beta$ , smooth oscillation of the population of infected is observed. See [12]
component labeling	$R(x, y) = \min(x, y)$	Initialize the state each vertex with its own index (i.e., $x_i^0 = i$ ). The system will converge to a state where each vertex's state is the smallest vertex index belonging to that component. See [12]
shortest path	$R(x, y) = \min(x, y + 1)$	Computes path distance for a single source. Initialize each vertex so $x_i^0 =  V $ except the source vertex $j$ which gets 0. See [12]
vertex coloring	$R(x, y) = (x + \delta(x, y), y) \pmod k$	Vertices initialized with a random color from the set of $k$ colors. Symmetry breaking. May not converge. See [12]
topological sorting	$R(x, y) = \begin{cases} (y, x) & \text{if } y < x \\ (x, y) & \text{else} \end{cases}$	Initialize $X$ with a permutation of the vertex indices. Will not converge if the graph has cycles. Symmetry breaking. See [12]

arranged contiguously. To make the GPU algorithm as robust and widely applicable as possible, the entirety of  $E$  (the connectivity information) is not stored the GPU. Instead, a small number of edges are sampled from  $E$  by the CPU and written to the GPU's local memory in small chunks. These edges are then updated sequentially for each experiment during a single kernel invocation. There are several advantages to this technique. Not storing  $E$  on the GPU frees up memory that can be used for additional experiments, which increases throughput. Furthermore, if the graph is very dense, there may not even be memory on the GPU, since  $|E| = O(|V|^2)$ . This approach is also compatible with sampling edges from a non-uniform distribution, despite the

fact that this can require complex data structures [19]. The complex sampling algorithm can be kept on the CPU since the edges sampled are simply copied to the GPU afterwards.

However, interleaving kernel execution with edge sampling results in the CPU and GPU being idle while the other works, negatively impacting performance. Fortunately, a simple solution is found by dividing the computation into blocks. The GPU portion of a block is dependent on the edges sampled by the CPU in the previous block. The CPU portion of the current block computes and writes the edges needed by the next GPU block. This reduces the amount of time the CPU and GPU are idle, which improves performance. An illustration of this host/device scheduling

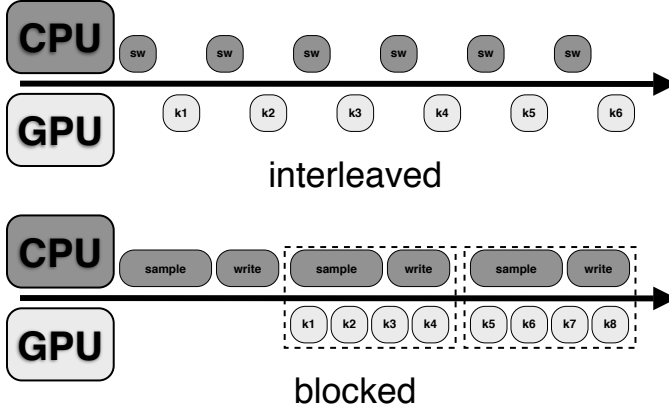


Fig. 1: Arranging kernel calls into blocks as shown allows helps the GPU to avoid becoming idle.

optimization is shown in Figure 1, and its pseudocode and OpenCL kernel code<sup>1</sup> is outlined in Algorithm 1. We note that, at best, this approach will speed up the GPU parallelization by a factor of 2. This occurs when the execution times of the GPU and CPU tasks are equal.

**Algorithm 1:** pseudocode to call the kernel in a blocked fashion to allow concurrent CPU/GPU execution

**Input:**  $n$  (number of edges to update);  $N$  (edges per block);  $block\_size$

```

1 steps =  $\lceil n/block\_size/N \rceil$ ;
2 sample edges for block 0;
3 copy block 0 to GPU;
4 sample edges for block 1;
5 for  $i=0..steps$  do
6   copy block  $i + 1$  to GPU (asynchronously);
7   wait for block  $i$  to finish copying;
8   for  $j=0..block\_size$  do
9     call DimerAutomatonKernel on block  $i$ 
      with offset  $j \cdot N$ ;
10  end
11  sample edges for block  $i + 2$ ;
12  finish command queue;
13 end

```

### 3. Results

To evaluate the effectiveness of our GPU parallelization, we consider two different cases corresponding to two common dimer automaton rule representations:

- 1) *eqn*, an algebraic representation of the rule where each simulation uses the same equation with a different

<sup>1</sup>OpenCL and CUDA are the two main GPGPU SDK's available today. Both packages offer various advantages and disadvantages, but overall one would expect roughly equivalent performance on identical hardware (anecdotally, CUDA seems to run very slightly faster).

---

**Function** `DimerAutomatonKernel` (`uint n`, `uint offset`, `__global uint* Eu`, `__global uint* Ev`, `__global T* R`, `__global T* X`, `__local uint* u_cache`, `__local uint* v_cache`, `__global int* counts`)

---

```

1 uint gsi = get_global_size(0);
2 uint gid = get_global_id(0);
3 int count = 0;
  // copy the edges to local memory
4 event_t events[2];
5 events[0] = async_work_group_copy(u_cache,
  &Eu[offset], (size_t)n, 0);
6 events[1] = async_work_group_copy(v_cache,
  &Ev[offset], (size_t)n, 0);
7 wait_group_events(2, events);
  // begin updating edges
8 for uint i = 0; i < n; i++ do
  // read the edge  $\vec{uv}$ 
9   uint u_idx = mad24(gsi, u_cache[i], gid);
10  uint v_idx = mad24(gsi, v_cache[i], gid);
  // read the state of  $\vec{uv}$ 
11  T xu = X[u_idx];
12  T xv = X[v_idx];
  // compute the next state
13  T xup = apply_rule(xu,xv,R);
14  T xvp = apply_rule(xv,xu,R);
  // write the new state of  $\vec{uv}$ 
15  X[u_idx] = xup;
16  X[v_idx] = xvp;
  // count number of changed edges
17  count += (isnotequal(xu,xup) | isnotequal(xv,xvp));
18 end
19 counts[gid] += count; // optional

```

---

initial configuration; and

- 2) *mat*, a lookup table representation of the rule where each simulation uses a different rule and, potentially, a different initial configuration.

The matrix representation results in random memory access patterns in the kernel. This is not the case with the algebraic representation, which simply requires computing an expression (whose constants can be accessed in a coalesced manner). Thus we would expect the algebraic representation to have better performance.

We carefully compare the performance of the GPU<sup>2</sup> algorithm to the CPU<sup>3</sup> algorithm in the following manner. The size of the graph  $|V|$  and the number of threads  $\tau$  are varied simultaneously so that  $\tau|V| = M$ , where  $M$  is a constant close to the memory capacity of the GPU. For simplicity, the graph's edges are chosen randomly. The CPU algorithm essentially executes the same loop as

<sup>2</sup>using a NVIDIA GeForce 9400M with 256MB RAM and 32 cores

<sup>3</sup>using an Intel 2.8GHz Core 2 Duo with 6MB L2 cache

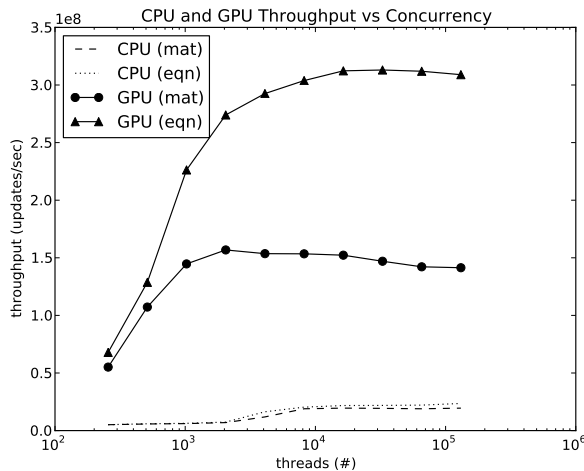


Fig. 2: Comparison of throughput for algebraic and finite state machine representations of dimer automata rules for GPU and CPU implementations.

Table 2: Effectiveness of matrix/equation GPU acceleration

	threads (#)	vertices (#)	throughput (ups/sec)	speedup	max. throughput (ups/sec)
mat	1024	32768	$1.45 \times 10^8$	23.84	$1.57 \times 10^8$
eqn	2048	16384	$2.74 \times 10^8$	37.10	$3.13 \times 10^8$

`DimerAutomatonKernel`, but simulates each dimer automaton in its entirety before moving on to the next. Figure 2 shows the comparison of the four cases (GPU/CPU and eqn/mat). For the algebraic case, the shortest path equation was used, and for the matrix case, each automaton uses a different random matrix for its rule. The GPU algorithm is clearly faster in both cases, and a summary of the results is shown in Table 2. The algebraic GPU case is the fastest, as expected, being 37 times faster than the CPU equivalent and having a maximum throughput of 313 million edge updates per second.

## 4. Discussion

Several notable effects are seen in Figure 2 from the original experiment. As the number of threads  $\tau$  increases, the throughput of the matrix GPU algorithm increases quickly, but then begins to decrease after 2048 threads. This is most likely due to the increasing number of un-coalesced memory accesses. When  $\tau$  is small, the GPU algorithm is less efficient because there is not enough inherent concurrency for the GPU to take advantage of. When  $\tau$  is large, the number of random memory accesses is large, resulting in more latency. This produces the optimal point between these extremes that we observe. The decrease towards the tail of the algebraic case is less pronounced, and is likely due to the slight overhead incurred in scheduling such a large number of threads.

A final observation worth noting here is that while the GPU algorithm's performance tends to decrease after a certain number of threads, the CPU algorithm has the opposite behavior. In our experiment, the size of the graph is inversely proportional to the number of threads so that the memory footprint remains constant. When a large number of threads are used, the graph becomes very small. There appears to be a certain threshold, (around 8192 threads and 4096 vertices—roughly 16KB) where the CPU algorithm suddenly becomes more efficient. This is most likely due the working set of the CPU algorithm becoming small enough for memory caching and paging to become optimal. For this reason, the biggest GPU speedup (in both cases) does not occur with the highest throughput (see Table 2).

### 4.1 Block and Rule Sizes

Here we examine how other aspects of our GPU algorithm affect its performance. An important aspect of the GPU algorithm is the consolidation of a series kernel invocations into blocks<sup>4</sup>. This helps prevent the CPU and GPU from remaining idle while the other is working. Figure 3 shows how the block size affects the throughput of the algorithm (the number of threads is varied in the same manner as the previous experiment). The results show that with enough concurrency, any choice of block size reaches edge peak throughput. However, it appears that increasing the number of kernels called per block causes this threshold to be reached more quickly.

We also consider the effect of adding additional states to  $\Sigma$  which increases the rule size. Previously we showed that increasing the concurrency can decrease throughput after a certain point by increasing the total number of un-coalesced memory reads. It is also the case that the total memory footprint of the rule affects the GPU performance. This is evidenced by Figure 4 where the original experiment is repeated for different sized rules. Even while keeping the number of threads constant, the throughput is negatively affected as the rule size  $|\Sigma|$  increases from 3 to 8.

### 4.2 Throughput vs Efficiency

Dimer automata can be used to implement algorithms for computations on graphs. However, their efficiency may be dependent on the topology of the graph; in other words, throughput is not the same as efficiency. Dimer automata compute on graphs via front propagation, much like a breadth first search, to compute the result. Ideally the front will cover a significant portion of the vertices at any given time and will move across the entire graph quickly, which is important to avoid unnecessary computation. This is the case for high dimensional graphs, especially small world graphs. However, for low dimensional graphs, the diameter can be large, causing the algorithm to converge more slowly.

<sup>4</sup>To clarify, the blocks we are referring to are those shown in Figure 1, and not thread blocks, which are also commonly referred to as work groups.

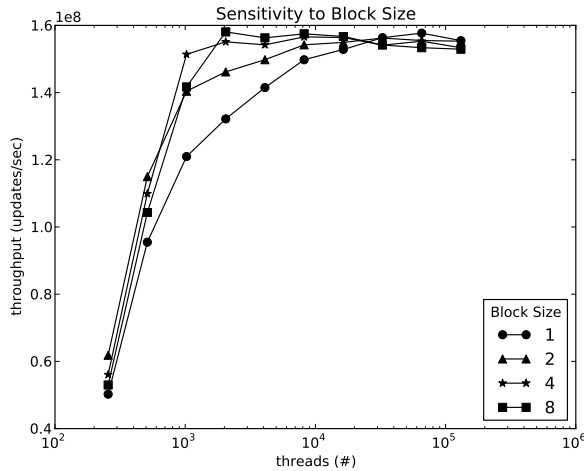


Fig. 3: Increasing the number of kernel invocations per block causes the GPU algorithm to reach maximum throughput sooner (with fewer threads).

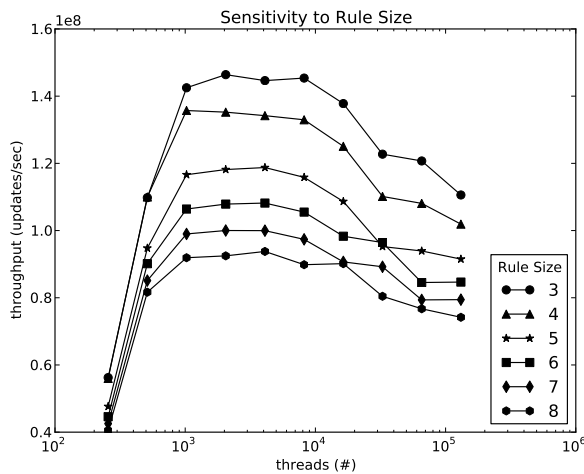


Fig. 4: Increasing the number of states (the rule size) also has a detrimental impact on performance.

The dimension of a graph is analogous to the dimension of euclidean space. A hyper-cube in this space with sides of length  $r$  will have volume of  $r^d$ . Thus, the volume follows a power law parameterized by  $d$ , which is exactly the dimension of the space, i.e.  $V(r) \sim r^d$ . This concept can be extended to graphs by defining the volume  $V(r)$  as the average number of vertices reachable in  $r$  or fewer steps [20], [21]. For example, a square lattice with Moore neighborhoods has  $d = 2$  since  $V(0) = 1, V(1) = 9, V(2) = 25, \dots, V(r) = (2r+1)^2$ . Thus a square lattice approximates 2-D space, as both have the same volume scaling exponent of 2. Random long range connections in graphs (e.g. small world and scale free properties) cause the diameter of the

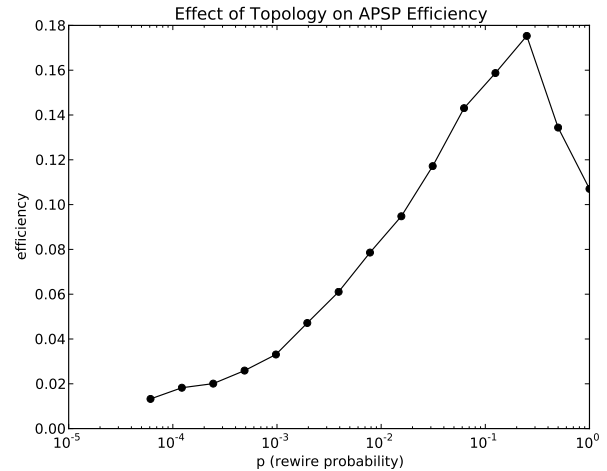


Fig. 5: As the structure of the graph transitions from uniform to random, efficiency improves.

graph to scale according to  $O(\log |V|)$  [22], giving the graph high dimension. So, small world graphs are likely the best case scenario in terms of the efficiency of the simple dimer automata rules from Table 1. The Watts-Strogatz small world model provides a simple way to interpolate between fully random and uniform graphs based only on the rewiring probability  $p$  [23].

We use this classic approach to quantify how the efficiency of our shortest path rule depends on the structure of the graph. We measure efficiency by considering the fraction of updates that do useful work by changing the system. We use the shortest path rule  $R(x, y) = \min(x, y + 1)$  with the initial configuration setup so that each simulation solves the shortest path problem for a different source vertex. Edge updates are broken up into sets where each edge is updated at least once (any order can be used). If no updates changed the system after updating a set (i.e. every edge), then the algorithm has converged. Figure 5 shows how the efficiency is affected by the structure of the graph.<sup>5</sup> As the graph transitions from ordered to random, the efficiency improves predictably, with maximum efficiency close to 20%. This is one reason why the GPU algorithm's performance was not better than a more sophisticated serial algorithm<sup>6</sup> for the same problem.

## 5. Conclusion

Many GPU parallelizations of cellular automata are designed to take advantage of the regular structure of the underlying lattice, with larger lattices often allowing more concurrency, resulting in better performance. However, there

<sup>5</sup>the graph had 16,384 vertices and 2,048 threads because this configuration produced the highest speedup in the original experiment.

<sup>6</sup>comparison based on the igrph diameter function [24]



are clear cases where simulating many (not necessarily large) systems is warranted. Furthermore, there are many cases where non-uniform topologies and/or asynchronous updating is also warranted. Each of these issues presents unique challenges. Therefore, we have designed a GPU parallelization of the dimer automaton framework tailored towards concurrent simulation of mid-sized simulations. The parallelization is made effective by assuming each automaton sees the same order of updates, but this reduces the applicability of our approach somewhat.

Fortunately, there are many cases where this assumption is acceptable, and the GPU parallelization results in significant performance gains. In the best case, our algorithm has a throughput of over 313 million edge updates per second and produces a speedup of 37 over the CPU implementation. Additionally we note that the hardware used to evaluate our algorithm was fairly modest by today's standards, having only 32 cores. High-end models can increase this number by an order of magnitude, which, ideally, would cause a proportional improvement in performance.

We also tested our GPU algorithm on a classical graph computation, the all pairs shortest path problem. However, we found that the performance was not better than more sophisticated serial algorithms due to the potential for wasted computational resources. Additionally, the GPU algorithm's performance suffered when each dimer automaton used a different rule, represented as a matrix. Future work to convert matrix representations to algebraic ones would have a significant positive impact on performance. Finally we note that our approach can be extended to asynchronous variants of complex systems (e.g., asynchronous cellular automata, random boolean networks, etc.) with only trivial modification to the kernel and algorithm.

## Acknowledgement

The work of Yang Cao was supported by the National Science Foundation under awards CCF-0726763 and CCF-0953590, and the National Institutes of Health under award GM078989.

## References

- [1] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.
- [2] A. Arkin, J. Ross, and H. H. McAdams, "Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected *escherichia coli* cells," *Genetics*, vol. 149, no. 4, pp. 1633–1648, AUG 1998.
- [3] S. Wolfram, *A New Kind of Science*. Wolfram Media, Inc., 2002.
- [4] C. G. Langton, "Computation at the edge of chaos - phase-transitions and emergent computation," *Physica D*, vol. 42, no. 1-3, pp. 12–37, JUN 1990.
- [5] M. Mitchell, J. P. Crutchfield, and R. Das, "Evolving cellular automata with genetic algorithms: A review of recent work," in *Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA'96)*. Russian Academy of Sciences, 1996.
- [6] E. A. Di Paolo, "Rhythmic and non-rhythmic attractors in asynchronous random boolean networks," *Biosystems*, vol. 59, no. 3, pp. 185–195, MAR 2001.
- [7] F. Jiménez-Morales, M. Mitchell, and J. P. Crutchfield, "Evolving one dimensional cellular automata to perform a non-trivial collective behavior task: One case study," in *Computational Science — ICCS 2002*, ser. Lecture Notes in Computer Science, P. Sloot, A. Hoekstra, C. Tan, and J. Dongarra, Eds. Springer Berlin / Heidelberg, 2002, vol. 2329, pp. 793–802.
- [8] D. Wolz and P. P. de Oliveira, "Very effective evolutionary techniques for searching cellular automata rule spaces," *Journal of Cellular Automata*, vol. 3, no. 4, pp. 289–312, 2008.
- [9] S. K. Tan and S.-U. Guan, "Evolving cellular automata to generate nonlinear sequences with desirable properties," *Applied Soft Computing*, vol. 7, no. 3, pp. 1131 – 1134, 2007.
- [10] B. Schönfisch and K. P. Haderer, "Dimer automata and cellular automata," *Physica D*, vol. 94, no. 4, pp. 188–204, JUL 15 1996.
- [11] D. Arendt and Y. Cao, "Effective GPU acceleration of large scale, asynchronous simulations on graphs (doi:10.1142/s021952591250035x)," *Advances in Complex Systems*, 2012.
- [12] D. Arendt, "In search of self-organization," Ph.D. dissertation, Virginia Tech, 2012.
- [13] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, MAY 2008.
- [14] S. Gobron, F. Devillard, and B. Heit, "Retina simulation using cellular automata and GPU programming," *Machine Vision and Applications*, vol. 18, no. 6, pp. 331–342, 2007.
- [15] C. Kauffmann and N. Piche, "Cellular automaton for ultra-fast watershed transform on GPU," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. IEEE, 2008, pp. 1–4.
- [16] S. Rybacki, J. Himmelspach, and A. Uhrmacher, "Experiments with single core, multi-core, and GPU based computation of cellular automata," in *First International Conference on Advances in System Simulation*. IEEE, 2009, pp. 62–67.
- [17] S. Gobron, H. Bonafos, and D. Mestre, "GPU accelerated computation and visualization of hexagonal cellular automata," *Cellular Automata*, pp. 512–521, 2010.
- [18] H. Li and L. Petzold, "Efficient parallelization of the stochastic simulation algorithm for chemically reacting systems on the graphics processing unit," *International Journal of High Performance Computing Applications*, vol. 24, no. 2, pp. 107–116, MAY 2010.
- [19] A. Slepoy, A. P. Thompson, and S. J. Plimpton, "A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks," *Journal of Chemical Physics*, vol. 128, no. 20, MAY 28 2008.
- [20] M. E. J. Newman and D. J. Watts, "Scaling and percolation in the small-world network model," *Physical Review E*, vol. 60, no. 6, p. 7332, 1999.
- [21] O. Shanker, "Defining dimension of a complex network," *Modern Physics Letters B*, vol. 21, no. 6, pp. 321–326, 2007.
- [22] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, no. 6825, pp. 268–276, MAR 8 2001.
- [23] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, JUN 4 1998.
- [24] G. Csárdi and T. Nepusz, "The igraph software package for complex network research," *InterJournal Complex Systems*, vol. 1695, 2006.

# Topological Mixing Derived From Glider $D_1$ of Universal ECA Rule

Fang Wang, Fangyue Chen, Lingxiao Si, and Pingping Liu

School of Science, Hangzhou Dianzi University, Hangzhou, Zhejiang, P. R. China

**Abstract**—Rule 110 is a complex cellular automaton (CA) in Wolfram's system of identification and capable of supporting universal computation. There is no doubt that the dynamical property of rule 110 is extremely complex and still not well understood. Gliders in one-dimensional cellular automata are compact groups of non- quiescent and non-ether patterns translating along automaton lattice. This paper reveals a interesting relation between the complexity and the glider in rule 110, and rigorously proves that rule 110 is topological mixing and is chaotic in the sense of both Li-Yorke and Devaney on the subsystem derived from the existing glider  $D_1$ .

**Keywords:** cellular automata; glider; topologically mixing; chaos; directed graph.

## 1. Introduction

Cellular automata (CA) was introduced by von Neumann and Ulam in the late 1940s and early 1950s [1]. In late 1960s, Conway proposed his now-famous Game of Life, which shows the great potential of CA in the simulation of complex systems [2]. Mathematical theory of CA was developed by Hedlund about two decades after Neumann's work. He studied CA in the context of symbolic dynamics as homomorphisms of the full shift [3]. In the early 1980s, Wolfram carried out a lot of research on dynamical and computational aspects of CA [4-5]. In [6], he classified the 256 elementary cellular automata (ECA) informally into four classes using dynamical concepts like periodicity, stability and chaos.

Based on Wolfram's work, Chua et al. provided a rigorous nonlinear dynamical approach to his empirical observations based on mathematical analysis [7-10]. Although there are 256 elementary cellular automata (ECA) rules, only 88 rules are globally independent from each other [7-8, 15]. These 88 global independent ECA rules are also organized into 4 groups with distinct qualitative dynamics: 40 period- $k$  ( $k = 1, 2, 3, 6$ ), 30 topologically distinct Bernoulli shift rules, 10 complex Bernoulli shift rules and 8 hyper Bernoulli shift rules [7-8].

Among infinitely many local CA rules, rule 110 in Wolfram's system of identification has received special attention. One of the first investigations about rule 110 was described by Wolfram, discovering that the rule displays complex behaviors by means of the existence of gliders—a glider is a periodic structure moving into the evolution space—from random initial conditions. Rule 110, like the Game of Life, is

named as left life by Cook. He showed new gliders with rare extensions and a pair of gliders of complicated constructions [11]. Also, he demonstrated that rule 110 was universal via simulating a cyclic tag system with well-defined blocks of gliders by means of collisions [12, 20, 21].

This paper mainly focuses the complexity of rule 110 on a subsystem which is derived from the existing glider  $D_1$ . The rest of the paper is organized as follows: Section 2 reviews the basic concepts of one-dimensional CA and symbolic dynamics. Section 3 identifies a subsystem of rule 110 which is a subshift of finite type, and discusses its complex dynamics. Section 4 concludes the present work.

## 2. Preliminaries

A bi-infinite sequence,  $x = (\dots, x_{-1}, x_0^*, x_1, \dots)$ , is called a configuration, where  $x_i \in S = \{0, 1, \dots, k-1\}$  and the star “\*” denotes the designated symbol of  $x$ . If  $I = [i, j]$  is an interval integers, put  $x_{[i,j]} = (x_i, x_{i+1}, \dots, x_j)$  ( $i < j$ ),  $x_{[i,j)} = (x_i, \dots, x_{j-1})$ . For a finite sequence  $a = (a_0, a_1, \dots, a_{n-1})$ , if there exists an  $m \in \mathbb{Z}$  such that  $x_{m+k} = a_k$  ( $k = 0, 1, \dots, n-1$ ), then it is said that  $a$  appears in  $x$ , denoted as  $a \prec x$ . The set of configurations is denoted by  $S^{\mathbb{Z}}$  and a metric “ $d$ ” on  $S^{\mathbb{Z}}$  is defined as

$$d(x, y) = \sup \left\{ \frac{\rho(x_i, y_i)}{2^{|i|}} \mid i \in \mathbb{Z} \right\}, \quad \rho(x_i, y_i) = \begin{cases} 0, & x_i = y_i, \\ 1, & x_i \neq y_i, \end{cases}$$

where  $\rho(\cdot, \cdot)$  is the metric on  $S$  and  $x, y \in S^{\mathbb{Z}}$ . It is known that  $(S^{\mathbb{Z}}, d)$  is a compact, perfect, and totally disconnected metric space [13].

Let  $f$  denote a self map on  $S^{\mathbb{Z}}$ . A subset  $X \subseteq S^{\mathbb{Z}}$  is  $f$ -invariant if  $f(X) \subseteq X$ , and strongly  $f$ -invariant if  $f(X) = X$ . If  $X$  is closed and  $f$ -invariant, then  $(X, f)$  or simply  $X$  is called a subsystem of  $(S^{\mathbb{Z}}, f)$ .

The left-shift map  $\sigma_L : S^{\mathbb{Z}} \rightarrow S^{\mathbb{Z}}$  is the homeomorphism defined by  $[\sigma_L(x)]_i = x_{i+1}, \forall x \in S^{\mathbb{Z}}, i \in \mathbb{Z}$ , where  $[\sigma_L(x)]_i$  stands for the  $i$ -th element of  $\sigma_L(x)$ . The restriction of  $\sigma_L$  to any closed and strongly  $\sigma_L$ -invariant subset  $\Lambda \subset S^{\mathbb{Z}}$  is called a subshift, denoted by  $(\Lambda, \sigma_L)$ . For instance, let  $\mathcal{A}$  denote a set of some finite sequences over  $S$ , and  $\Lambda = \Lambda_{\mathcal{A}}$  is the set of configurations made consisting of the sequences in  $\mathcal{A}$ . Then  $\Lambda_{\mathcal{A}}$  is a subsystem of  $(S^{\mathbb{Z}}, \sigma_L)$ , where  $\mathcal{A}$  is said to be the determinative block system of  $\Lambda$ . Meanwhile, the right-shift map  $\sigma_R : S^{\mathbb{Z}} \rightarrow S^{\mathbb{Z}}$  defined by  $[\sigma_R(x)]_i = x_{i-1}, \forall x \in S^{\mathbb{Z}}, i \in \mathbb{Z}$ , has similar results as  $\sigma_L$ .

A map  $f : S^{\mathbb{Z}} \rightarrow S^{\mathbb{Z}}$  is a CA if it is continuous and commutes with  $\sigma$ , where  $\sigma$  is left-shift or right-shift map.

For any CA, there exist a radius  $r > 0$  and a local map  $\hat{f} : S^{2r+1} \rightarrow S$  such that  $[f(x)]_i = \hat{f}(x_{[i-r, i+r]})$ .  $f$  is an ECA when  $r = 1$ .

Each ECA can be expressed by a 3-bit Boolean function and coded by an integer  $N$ , which is the decimal notation of the output binary sequence of the Boolean function [7]. The local expression of rule 110 is

$$\begin{aligned} \hat{f}(0, 0, 0) &= 0 & \hat{f}(1, 0, 0) &= 0 \\ \hat{f}(0, 0, 1) &= 1 & \hat{f}(1, 0, 1) &= 1 \\ \hat{f}(0, 1, 0) &= 1 & \hat{f}(1, 1, 0) &= 1 \\ \hat{f}(0, 1, 1) &= 1 & \hat{f}(1, 1, 1) &= 0 \end{aligned}$$

i.e., the evolution rule is expressed in binary notation as 01110110 (representing the decimal number  $N = 110$ , where the leftmost one is the low position and the rightmost one is the high position). The global map of rule 110 is denoted by  $f_{110}$ . The evolution of the automaton is defined starting from linear array of cells each containing one state of  $S = \{0, 1\}$ ; taking every cell  $x_i$  as a central one, to evaluate the value of its corresponding neighborhood so as to determine the new central element in the following generation:

$$\hat{f}(x_{i-1}^t, x_i^t, x_{i+1}^t) \rightarrow x_i^{t+1}.$$

Time  $t$  is discrete and there is a simultaneous evaluation of each  $x_i$  in the array, i.e., parallel mappings generate the next array, determining the evolution space  $S^Z$ .

### 3. Glider and Subsystem

#### 3.1 Glider $D_1$

A glider is a compact group of non-quiescent states traveling along CA lattice, and is a periodic structure moving in time [6, 11-12, 20-21].

It is known that the speed of glider  $D_1$  of rule 110 is  $2/10$ , the lineal volume is 11-25, and the even and odd number of periodic margin on the left (or right) border in the ether pattern are 1 and 2 respectively [21]. Under the viewpoint of symbolic dynamics, the ether pattern and glider can be defined as the evolutionary orbit starting from a special initial configuration [22]. The ether factor of the ether patterns is  $a = (1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0)$ , and one of glider factors of  $D_1$  is  $b = (1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0)$ , i.e., an ether pattern of Rule 110 is the evolutionary orbit  $Orb_{f_{110}}(a^*) = \{a^*, f_{110}(a^*), f_{110}^2(a^*), \dots\}$  and glider  $D_1$  is the evolutionary orbit  $Orb_{f_{110}}(\bar{x}) = \{\bar{x}, f_{110}(\bar{x}), f_{110}^2(\bar{x}), \dots\}$  in the CA lattice space, where  $a^* = (\dots, a, a, a, \dots)$  is a cyclic configuration and  $\bar{x} = (\dots, a, a, a, b, a, a, a, \dots)$ . That the speed of  $D_1$  is  $2/10$  implies this glider shifts to right by 2 bits in every 10 times iteration under rule 110, i.e.,  $f_{110}^{10}(\bar{x}) = \sigma_R^2(\bar{x})$ . The ether pattern and glider  $D_1$  are shown in Figure 1.

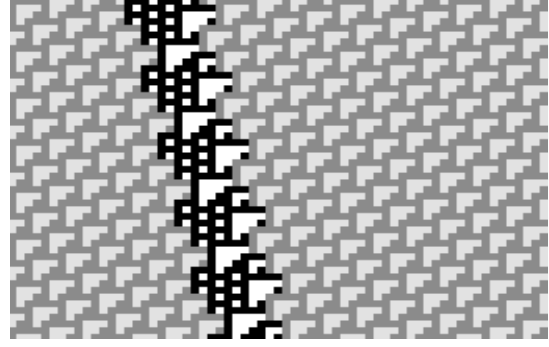


Fig. 1: Ether pattern and glider  $D_1$  of rule 110.

#### 3.2 Subsystem

One can obtain a 21-sequence set  $\mathcal{A}_1 = \{q|q = \bar{x}_{[i, i+20]}, \forall i \in Z\}$  from  $\bar{x} = (\dots, a, a, a, b, a, a, a, \dots)$ , where  $a = (1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0)$  is the ether factor and  $b = (1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0)$  is a glider factor of  $D_1$ . More specifically  $\mathcal{A}_1 =$

```
{000100110111011101101, 00010011011110001001, 001001101110111011011,
00100110111100010011, 001101110111011011111, 001101111100010011011,
010011011101110110111, 01001101111000100110, 011011101110110111110,
01101111000100110111, 01110110111100010011, 011101110110111110001,
01111000100110111011, 01111000100110111110, 100010011011101110110,
10001001101111000100, 100110111011101101111, 10011011110001001101,
10110111100010011011, 10111011011110001001, 10111011101101111000,
10111100010011011101, 10111100010011011111, 110001001101110111011,
11000100110111100010, 11011011110001001101, 110111011011111000100,
110111011101101111100, 11011110001001101110, 11011110001001101111,
111000100110111011101, 11100010011011110001, 111011011111000100110,
11101110110111100010, 111100010011011101110, 11110001001101111000,
111110001001101110111, 111110001001101111100}
```

For simplicity,  $\mathcal{A}_1$  is expressed by its decimal code  $D(\mathcal{A}_1) =$

```
{159469, 159625, 318939, 319251, 454367, 456859, 637879,
638502, 908734, 913719, 974611, 978417, 1018299, 1018302,
1128310, 1128388, 1275759, 1277005, 1505435, 1535881,
1537784, 1557725, 1557727, 1612731, 1612770, 1801293,
1816516, 1817468, 1827438, 1827439, 1854941, 1854961,
1949222, 1956834, 1976046, 1976056, 2036599, 2036604}.
```

#### Proposition 1:

(1) For rule 110, there exists a subset  $\Lambda_1 \subset S^Z$ , such that  $f_{110}^{10}(x) = \sigma_R^2(x)$  for  $x \in \Lambda_1$ , where  $\Lambda_1 = \Lambda_{\mathcal{A}_1} = \{x \in S^Z \mid x_{[i-10, i+10]} \in \mathcal{A}_1, \forall i \in Z\}$ , and  $\mathcal{A}_1$  is its deterministic block system.

(2) Let

$$\Lambda = \bigcup_{i=0}^9 f_{110}^i(\Lambda_1),$$

then,  $\Lambda$  is a  $f_{110}$ -invariant subset of  $S^Z$ , and  $f_{110}^{10}(x) = \sigma_R^2(x)$  for  $x \in \Lambda$ .

(3)  $\sigma_R : \Lambda \rightarrow \Lambda$  is a subshift of finite type (SFT).

Let  $\Lambda = \Lambda_{\mathcal{A}}$ ,  $\mathcal{A}$  is the determinative block system of  $\Lambda$ , then,  $\mathcal{A}$  is a 21-sequence set consisting of 237 elements. Due to space limitations and for simplicity, the decimal code set  $D(\mathcal{A})$  of  $\mathcal{A}$  is placed in Appendix I.

### 3.3 Finite directed graph and complexity

Since  $(\Lambda_{\mathcal{A}}, \sigma_R)$  is a SFT, then  $\Lambda_{\mathcal{A}}$  can be described by a finite directed graph  $G_{\mathcal{A}} = \{\mathcal{A}, \mathcal{E}\}$ , in which each vertex is labeled by a sequence in  $\mathcal{A}$ , and  $\mathcal{E}$  is the edge set. Two vertices  $a = (a_0, a_1, \dots, a_{n-1})$  and  $b = (b_0, b_1, \dots, b_{n-1})$  are connected by an edge of  $\mathcal{E}$  if and only if  $a_k = b_{k-1}, k = 1, 2, \dots, n-1$ . Every edge  $(a_0, a_1, \dots, a_{n-1}) \rightarrow (b_0, b_1, \dots, b_{n-1})$  of  $\mathcal{E}$  is labeled by  $b_{n-1}$ . One can think of each element of  $\Lambda_{\mathcal{A}}$  as a bi-infinite path on the graph  $G_{\mathcal{A}}$ . Whereas a directed graph corresponds to a square transition matrix  $A = (A_{ij})_{m \times m}$  with  $A_{ij} = 1$  if and only if there is an edge from vertex  $b^{(i)}$  to vertex  $b^{(j)}$ , where  $m = |\mathcal{A}|$  is the number of elements in  $\mathcal{A}$ , and  $i$  (or  $j$ ) is the code of the vertex in  $\mathcal{A}$ ,  $i, j = 0, 1, \dots, m-1$ . Thus,  $\Lambda_{\mathcal{A}}$  is precisely defined by the transition matrix  $A$ . Transition matrix  $A = (A_{ij})_{237 \times 237}$  is shown in Appendix II.

A square matrix  $A$  is irreducible if, for any  $i, j$ , there exists an  $n$  such that  $A_{ij}^n > 0$ ; aperiodic if there exists an  $n$ , such that  $A_{ij}^n > 0$ , for all  $i, j$ . Where  $A_{ij}^n$  is the  $(i, j)$  entry of  $A^n$  [13, 14].

#### Lemma 1:

- (1) If  $\Lambda_{\mathcal{A}}$  is a SFT, then  $\sigma$  ( $\sigma_R$  or  $\sigma_L$ ) is topologically transitive if and only if  $A$  is irreducible; topologically mixing if and only if  $A$  is aperiodic [13].
- (2)  $A$  is irreducible if and only if for every ordered pair of vertices  $b^{(i)}$  and  $b^{(j)}$  in  $\mathcal{A}$  there is a path in the graph  $G_{\mathcal{A}}$  starting at  $b^{(i)}$  and ending at  $b^{(j)}$ ;  $A$  is aperiodic if and only if it is irreducible and the numbers of the length of two different closed paths in the graph  $G_{\mathcal{A}}$  are coprime [13-14, 16].

#### Proposition 2:

- (1)  $\sigma_R$  is topologically mixing on  $\Lambda$ ;
- (2)  $\sigma_R^2 = f_{110}^{10}$  is topologically mixing on  $\Lambda$ ;
- (3)  $f_{110}$  is topologically mixing on  $\Lambda$ ;
- (4) The topological entropy of  $f_{110}|_{\Lambda}$  is positive.

*Proof:* (1) By computer-aided methods, it is easily found that there exist two different closed paths in the graph  $G_{\mathcal{A}} = \{\mathcal{A}, \mathcal{E}\}$ , and the numbers of their length are coprime (one is 14 and another is 25, as shown in Fig. 2), thus, the transition matrix  $A$  corresponding to the graph is aperiodic, so the shift  $\sigma_R$  is mixing;

(2) The conclusion is obvious;

(3) One only need to prove that for any nonempty open subsets  $U, V \subset \Lambda$ ,  $\exists N_0 > 0$ ,  $f_{110}^n(U) \cap V \neq \emptyset$  for any  $n \geq N_0$ . In fact,  $f_{110}^{10} = \sigma_R^2$  is mixing on  $\Lambda$ , it is easy to know that  $f_{110}^i : \Lambda \rightarrow \Lambda$  are homeomorphism,  $U_i = f_{110}^i(U)$  ( $i = 0, 1, \dots, 9$ ) are also open sets. Thus,  $\exists M_i > 0$ , such that  $(f_{110}^{10})^m(U_i) \cap V \neq \emptyset$  for any  $m \geq M_i$ . Let  $M = \max\{M_i\}$  and  $N = 11M$ , thus,  $f_{110}^n(U) \cap V \neq \emptyset$

for any  $n = 10m + i \geq N$ ;

- (4) The topological entropy of  $f_{110}|_{\Lambda}$  satisfies  $ent(f_{110}|_{\Lambda}) = \frac{1}{10}ent(f_{110}^{10}|_{\Lambda}) = \frac{1}{5}ent(\sigma_R|_{\Lambda}) = \frac{1}{5} \log(\rho(A)) \approx 0.0182312 > 0$ , where  $\rho(A)$  is the spectral radius of the transition matrix  $A$  corresponding to  $\Lambda = \Lambda_{\mathcal{A}}$ . ■

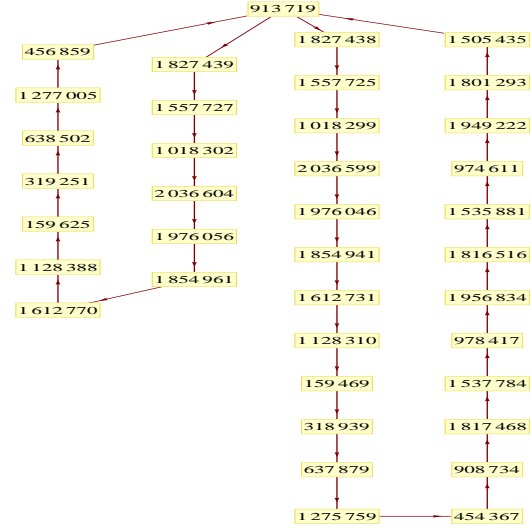


Fig. 2: Two different closed paths in the  $G_{\mathcal{A}} = \{\mathcal{A}, \mathcal{E}\}$

It is well known that positive topological entropy implies chaos in the sense of Li-Yorke [13-14, 16-19], and topologically mixing property implies many chaotic properties in different senses such as Devaney [16, 18]. Thus, one has the interesting result.

**Theorem 1:**  $f_{110}$  is chaotic in the sense of both Li-Yorke and Devaney on  $\Lambda$ .

## 4. Conclusion

In this paper, some topological dynamics of rule 110 has been discussed under the framework of symbolic dynamical systems. It is proved that rule 110 is topological mixing and possesses positive topological entropy and is chaotic in the sense of both Li-Yorke and Devaney on a subsystem derived from a existing glider of rule 110. Nevertheless, the complete symbolic dynamical properties of rule 110 are still an open problem, some new methods should be exploited to investigate its dynamical behaviors in future studies.

## Acknowledgments

This research was jointly supported by the NSFC (Grants No. 11171084, 60872093 and 10832006).

## References

- [1] von Neumann, J. *Theory of Self-reproducing Automata (edited and completed by A. W. Burks)*, University of Illinois Press, Urbana and London, 1966.
- [2] Gardner, M. *The fantastic combinations of John Conway's new solitaire game 'life'*, Scientific American, 223:120-123, 1970.
- [3] Hedlund, G. A. *Endomorphisms and automorphism of the shift dynamical system*, Theory of Computing Systems, 3:320-375, 1969.
- [4] Wolfram, S. *Statistical mechanics of cellular automata*, Rev. Mod. Phys., 3:601-644, 1983.
- [5] Wolfram, S. *Theory and Applications of Cellular Automata*, World Scientific, Singapore, 1986.
- [6] Wolfram, S. *Universality and complexity in cellular automata*, Phys. D, 10: 11-35, 1984.
- [7] Chua, L.O., Sbitnev, V. I., Yoon, S. *A nonlinear dynamics perspective of Wolfram's new kind of science. Part IV: From Bernoulli-shift to 1/f spectrum*, International Journal of Bifurcation and Chaos, 15 (4):1045-1223, 2005.
- [8] Chua, L.O., Sbitnev, V. I., Yoon, S. *A nonlinear dynamics perspective of Wolfram's new kind of science. Part VI: From time-reversible attractors to the arrows of time*, International Journal of Bifurcation and Chaos, 16 (5):1097-1373, 2006.
- [9] Chua, L.O., Guan, J. B., Valery, I. S., Shin, J. *A nonlinear dynamics perspective of Wolfram's new kind of science. Part VII: Isle of Eden*, International Journal of Bifurcation and Chaos, 17 (9):2839-3012, 2007.
- [10] Chua, L.O., Paziienza, G. E., Orzo, L., Sbitnev, V. I., Shin, J. *A nonlinear dynamics perspective of Wolfram's new kind of science. Part IX: Quasi-Ergodicity*, International Journal of Bifurcation and Chaos, 18 : 2487-2642, 2008.
- [11] Cook, M. *Introduction to the activity of rule 110*, Copyright 1994-1998 Matthew Cook, 1999.
- [12] Cook, M. *Universality in elementary cellular automata*, Complex Systems, 15(1), 1-40, 2004.
- [13] Zhou, Z.L. *Symbolic Dynamics*, Shanghai Scientific and Technological Education Publishing House, Shanghai, 1997.
- [14] Kitchens, B. *Symbolic Dynamics: One-sided, Two-sided and Countable State Markov Shifts*, Springer-Verlag, Berlin, New York, 1998.
- [15] Guan, J.B., Shen, S. W., Tang, C. B., Chen, F. Y. *Extending Chua's global equivalence theorem on Wolfram's new kind of science*, International Journal of Bifurcation and Chaos, 17 (12):4245-4259, 2007.
- [16] Xiong, J.C., Young, Z. *Chaos caused by a topologically mixing map in Dynamical Systems and Related Topics*, World Scientific, Singapore, 1992.
- [17] Blanchard, F., Glasner, E., Kolyada, S., Maass, A. *On Li-Yorke pairs*, Journal für die reine und angewandte Mathematik, 547:51-68, 2002.
- [18] Devaney, R.L. *An Introduction to Chaotic Dynamical Systems*, Addison-Wesley, 1989.
- [19] Favati, P., Lotti, G., Margara, L. *Additive one-dimensional cellular automata are chaotic according to Devaney's definition of chaos*, Theoretical Computer Science, 174:157-170, 1997.
- [20] Martinez, G.J., McIntosh, H.V. and Seck-Tuoh-Mora J.C. *Gliders in Rule 110*, International Journal of Unconventional Computing, 2:1-49, 2005.
- [21] Martinez, G.J., McIntosh, H.V., Seck-Tuoh-Mora J.C. and Chapa-Vergara S.V. *Reproducing the cyclic tag system developed by Matthew Cook with Rule 110 using the phases  $f_{i-1}$* , Journal of Cellular Automata 6:121-161, 2011.
- [22] Chen, F.Y., Shi, L., Chen, G.R. and Jin, W.F. *Chaos and gliders in periodic cellular automaton Rule 62*, Journal of Cellular Automata, in press.

## Appendix I

The determinative block system  $\mathcal{A}$  of  $\Lambda$  is a 21-sequence set consisting of 237 elements. For a sequence  $a = (a_0, a_1, \dots, a_{20}) \in \mathcal{A}$ , its decimal code is defined as

$$D(a) = \sum_{i=0}^{20} a_i \cdot 2^{20-i}$$

For simplicity,  $\mathcal{A}$  is replaced by its decimal code set  $D(\mathcal{A}) =$

{79812, 97357, 146509, 157261, 159373, 159469, 159621, 159624, 159625, 159630, 159673, 159675, 159693, 194715, 228429, 240717, 243789, 247007, 293019, 314523, 318747, 318939, 319243, 319249, 319251, 319261, 319347, 319351, 319387, 389431, 413169, 419039, 452831, 454367, 456799, 456847, 456857, 456859, 456939, 456942, 457631, 457663, 457951, 481435, 487579, 494014, 509151, 581395, 586039, 629047, 637495, 637879, 638487, 638499, 638502, 638522, 638523, 638695, 638703, 638775, 725884, 778863, 806385, 826338, 838078, 905662, 908734, 913598, 913694, 913715, 913719, 913878, 913884, 915262, 915326, 915902, 922492, 949325, 953841, 962871, 974611, 975159, 978417, 982093, 988028, 1017329, 1018097, 1018265, 1018298, 1018299, 1018302, 1018303, 1019569, 1019617, 1030641, 1031153, 1033439, 1035761, 1088482, 1097254, 1121830, 1127206, 1128262, 1128310, 1128386, 1128388, 1128391, 1128412, 1128413, 1128422, 1162790, 1168934, 1170470, 1172079, 1208201, 1255160, 1258095, 1274991, 1275759, 1276975, 1276999, 1277004, 1277005, 1277045, 1277047, 1277391, 1277407, 1277551, 1303151, 1339273, 1411518, 1451768, 1505435, 1509822, 1523238, 1525496, 1535881, 1537784, 1539622, 1557240, 1557624, 1557708, 1557725, 1557727, 1558360, 1558384, 1563896, 1564152, 1565295, 1566456,

1592817, 1597203, 1609491, 1612179, 1612707, 1612731, (197,161), (198,162), (199,163), (200,166), (201,167), (202,168),  
 1612769, 1612770, 1612771, 1612782, 1612787, 1633043, (203,169), (204,170), (205,171), (206,172), (207,174), (208,176),  
 1633811, 1652676, 1676156, 1687078, 1700151, 1718212, (209,186), (210,189), (211,190), (212,191), (213,192), (214,193),  
 1754335, 1801293, 1803487, 1810195, 1811324, 1816516, (215,194), (215,195), (216,196), (217,197), (218,198), (219,199),  
 1817468, 1818387, 1827196, 1827388, 1827430, 1827438, (220,200), (221,206), (222,208), (223,210), (224,211), (225,212),  
 1827439, 1827756, 1827768, 1830524, 1830652, 1831223, (226,213), (227,214), (228,215), (229,216), (230,217), (231,218),  
 1831804, 1844984, 1847177, 1853321, 1854665, 1854929, (232,219), (233,220), (234,221), (235,222), (236,228), (236,229),  
 1854941, 1854960, 1854961, 1854967, 1854969, 1865097, (237,233}).  
 1865481, 1892115, 1898651, 1907682, 1925743, 1949222,  
 1950319, 1953673, 1956834, 1957769, 1964187, 1972164,  
 1975236, 1975908, 1976040, 1976046, 1976056, 1976059,  
 1976060, 1981124, 1981316, 1994633, 2025412, 2027460,  
 2034658, 2036194, 2036530, 2036596, 2036599, 2036604,  
 2036605, 2036606, 2039138, 2039234, 2045892, 2061282,  
 2062306, 2066878, 2071522}.

## Appendix II

The index  $(i, j)$  set of the transition matrix  $A = (A_{ij})_{237 \times 237}$  with  $A_{ij} = 1, 1 \leq i, j \leq 237$  corresponding to the determinative block system  $\mathcal{A}$  is as follows.

{(1,8), (1,9), (2,14), (3,19), (4,20), (5,21), (6,22), (7,23), (8,24),  
 (9,25), (10,26), (11,27), (12,28), (13,29), (14,30), (15,38), (16,44),  
 (17,45), (18,46), (19,49), (20,50), (21,51), (22,52), (23,53),  
 (24,54), (25,55), (26,56), (26,57), (27,58), (28,59), (29,60), (30,62),  
 (31,64), (32,65), (33,66), (34,67), (35,68), (36,69), (37,70), (38,71),  
 (39,72), (40,73), (41,74), (42,75), (43,76), (44,80), (45,82), (46,85),  
 (47,91), (47,92), (48,111), (49,114), (50,117), (51,118), (52,119),  
 (53,120), (54,121), (55,122), (55,123), (56,124), (57,125), (58,126),  
 (59,127), (60,128), (61,132), (62,144), (63,158), (63,159), (64,164),  
 (65,165), (66,173), (67,175), (68,177), (69,178), (70,179), (71,180),  
 (71,181), (72,182), (73,183), (74,184), (75,185), (76,187), (77,188),  
 (78,201), (79,202), (80,203), (81,204), (82,205), (83,207), (84,209),  
 (85,215), (86,223), (87,224), (88,225), (89,226), (90,227), (91,228),  
 (91,229), (92,230), (93,231), (94,232), (95,234), (96,235), (97,236),  
 (98,237), (99,1), (100,2), (101,3), (102,4), (103,5), (104,6), (105,7),  
 (106,8), (106,9), (107,10), (108,11), (109,12), (110,13), (111,15),  
 (112,16), (113,17), (114,18), (115,25), (116,31), (117,32), (118,33),  
 (119,34), (120,35), (121,36), (122,37), (123,38), (124,39), (125,40),  
 (126,41), (127,42), (128,43), (129,47), (130,48), (131,61), (132,63),  
 (133,71), (134,77), (135,78), (136,79), (137,81), (138,83), (139,84),  
 (140,86), (141,87), (142,88), (143,89), (143,90), (144,91), (144,92),  
 (145,93), (146,94), (147,95), (148,96), (149,97), (150,98), (151,99),  
 (152,100), (153,101), (154,102), (155,103), (156,104), (157,105),  
 (158,106), (159,107), (160,108), (160,109), (161,110), (162,112),  
 (163,113), (164,115), (165,116), (166,122), (166,123), (167,129),  
 (168,130), (169,131), (170,133), (171,134), (172,135), (173,136),  
 (174,137), (175,138), (176,139), (177,140), (178,141), (179,142),  
 (180,143), (181,144), (182,145), (183,146), (184,147), (185,148),  
 (186,149), (187,150), (188,151), (189,152), (190,153), (191,154),  
 (192,155), (193,156), (194,157), (195,158), (195,159), (196,160),

# Chaotic Subsystem Come From Glider $E^3$ of CA Rule 110

Lingxiao Si, Fangyue Chen, Fang Wang, and Pingping Liu

School of Science, Hangzhou Dianzi University, Hangzhou, Zhejiang, P. R. China

**Abstract**—*The existence of glider in the evolution space of the one-dimensional cellular automaton rule 110, has important lines of investigation in cellular automata theory such as complex dynamical behavior, self-reproduction, universal computation and so on. This work reveals a subsystem based on the existing glider  $E^3$  under the framework of the symbolic dynamics, and proves that the global map of the rule is chaotic in the sense of both Li-Yorke and Devaney on the subsystem.*

**Keywords:** cellular automata (CA); chaos; de Bruijn diagram; glider; topologically transitive.

## 1. Introduction

Cellular automata (CA), introduced by von Neumann in the late 1940s and early 1950s, are a class of spatially and temporally discrete mathematical systems characterized by local interactions and synchronous dynamical evolution [1]. Among the 88 possible unique elementary cellular automata (ECA), rule 110 in Stephen Wolfram's system of identification [2] has been an object of special attention due to the structures or gliders which have been observed in evolution space from random initial conditions.

One of the first investigations about rule 110 was described by Wolfram [3], discovering that rule 110 displays complex behaviors by means of the existence of gliders - a glider is a periodic structure moving into the evolution space - from random initial conditions. Thus Wolfram establishes the conjecture that this rule could perform universal computation.

Lind presented the first classification of gliders in rule 110 in [3] with 13 gliders. Next the first paper dedicated to the analysis of rule 110 is made by Lindgren and Nordahl in [4], where a statistical study and some of the most common behaviors of rule 110 are considered. Cook presented his proof of the universality of rule 110 in a conference which taken place at the Santa Fe Institute in 1998 [5,7,8]. On the other hand, another perspective is reported by McIntosh in [6], analyzing rule 110 as a problem of tiles and applying de Bruijn diagrams for characterizing every glider. In this way, Wolfram presents his book *A New Kind of Science* [2] in 2002. The book explains the features of the gliders and the functionality of a cyclic tag system (CTS) to demonstrate that rule 110 is an elemental universal CA [8]. Since 2004, Martinez and his partners further investigated

the types of gliders, their properties and collisions and their representation by tiles [9-11].

With this background, the aim of this paper is to reveal a little complex nature contained in rule 110 under the framework of the symbolic dynamical systems. That is, based on the existing glider  $E^3$ , this paper find a subsystem on which the global map of rule 110 is chaotic in the sense of both Li-Yorke and Devaney.

## 2. Symbolic Dynamics and de Bruijn Diagram

### 2.1 Symbolic sequence space

Let  $a$  be a finite or infinite sequence over  $S = \{0, 1\}$  and  $I = [i, j]$  be an interval of integers on which  $a$  is defined, then denote  $a_{[i,j]} = (a_i, \dots, a_j)$  and  $a_{[i,j)} = (a_i, \dots, a_{j-1})$ . A sequence  $b$  is said to appear in  $a$ , denoted by  $b \prec a$ , if  $b = a_I$  for some interval  $I \subseteq \mathbb{Z}$ .

A bi-infinite sequence over  $S$  is called a configuration, the collection of all configurations is  $\Sigma_2 = S^{\mathbb{Z}} = \{(\dots, x_{-1}, x_0, x_1, \dots) \mid x_i \in S, i \in \mathbb{Z}\}$ , and the distance "d" is defined by

$$d(x, y) = \sum_{i=-\infty}^{\infty} \frac{|x_i - y_i|}{2^{|i|}}$$

for  $x, y \in \Sigma_2$ . It is well known that  $\Sigma_2$  is a Cantor complete metric space. The left-shift map  $\sigma_L$  and right-shift map  $\sigma_R$  are defined by  $[\sigma_L(x)]_i = x_{i+1}$  and  $[\sigma_R(x)]_i = x_{i-1}$  for any  $x \in \Sigma_2, i \in \mathbb{Z}$ , respectively, where  $[\sigma(x)]_i$  stands for the  $i$ -th symbol of  $\sigma(x)$ , and  $\sigma$  is left-shift or right-shift.

### 2.2 Truth table of rule 110

By a theorem of Hedlund [12], a map  $f : \Sigma_2 \rightarrow \Sigma_2$  is a cellular automaton iff it is continuous and commutes with  $\sigma$ . Moreover, for any CA  $f$ ,  $(\Sigma_2, f)$  defines a dynamical system  $(\Sigma_2, f)$ . A subset  $X \subseteq \Sigma_2$  is  $f$ -invariant if  $f(X) \subseteq X$ , and strongly  $f$ -invariant if  $f(X) = X$ . If  $X$  is a closed and  $f$ -invariant, then  $(X, f)$  or simply  $X$  is called a subsystem of  $(\Sigma_2, f)$  [13].

Each ECA rule can be expressed by a local function [18-22], the logical truth table of rule 110's local function  $\hat{f}_{110}$  is shown in Table 1.

Table 1: Logical truth table of rule 110's local function

$(x_{i-1}, x_i, x_{i+1})$	$f_{110}(x_{i-1}, x_i, x_{i+1})$
(0, 0, 0)	0
(0, 0, 1)	1
(0, 1, 0)	1
(0, 1, 1)	1
(1, 0, 0)	0
(1, 0, 1)	1
(1, 1, 0)	1
(1, 1, 1)	0

### 2.3 SFT and de Bruijn diagram

Let  $\mathcal{A}$  be  $n$ -sequence set over  $S = \{0, 1\}$ , and  $\Lambda_{\mathcal{A}}$  be the set of the configurations whose any  $n$ -subsequence is in  $\mathcal{A}$ . Then,  $(\Lambda_{\mathcal{A}}, \sigma)$  is a subsystem of  $(\Sigma_2, \sigma)$ , where  $\Lambda_{\mathcal{A}} = \{x \in \Sigma_2 \mid x_{[i, i+n-1]} \in \mathcal{A}, \forall i \in \mathbb{Z}\}$ , and  $\mathcal{A}$  is said to be the determinative block system of  $\Lambda_{\mathcal{A}}$ .  $(\Lambda_{\mathcal{A}}, \sigma)$  (or simply  $\Lambda_{\mathcal{A}}$ ) is also called the subshift of finite type (SFT) of  $(\Sigma_2, \sigma)$ . Furthermore,  $\Lambda_{\mathcal{A}}$  can be described by a finite directed graph,  $G_{\mathcal{A}} = \{\mathcal{A}, E\}$ , where each vertex is labeled by a sequence in  $\mathcal{A}$ , and  $E$  is the set of edges connecting the vertices in  $\mathcal{A}$ . The finite directed graph  $G_{\mathcal{A}}$  is called the de Bruijn diagram of  $\mathcal{A}$  (or  $\Lambda_{\mathcal{A}}$ ). Two vertices  $a = (a_0, \dots, a_{n-1})$  and  $b = (b_0, \dots, b_{n-1})$  are connected by an edge  $(a_0, \dots, a_{n-1}) \rightarrow (b_0, \dots, b_{n-1})$  if and only if  $(a_1, \dots, a_{n-1}) = (b_0, \dots, b_{n-2})$ . One can think of each element of  $\Lambda_{\mathcal{A}}$  as a bi-infinite path on the diagram  $G_{\mathcal{A}}$ . Whereas a de Bruijn Diagram corresponds to a square transition matrix  $A = (A_{ij})_{m \times m}$  with  $A_{ij} = 1$  if and only if there is an edge from vertex  $b^{(i)}$  to vertex  $b^{(j)}$ , where  $m = |\mathcal{A}|$  is the number of elements in  $\mathcal{A}$ , and  $i$  (or  $j$ ) is the code of the corresponding vertex in  $\mathcal{A}$ ,  $i, j = 0, 1, \dots, m-1$ . Thus,  $\Lambda_{\mathcal{A}}$  is precisely defined by the transition matrix  $A$ .

Remarkably, a  $\{0, 1\}$  square matrix  $A$  is irreducible if, for any  $i, j$ , there exists an  $n$  such that  $A_{ij}^n > 0$ ; aperiodic if there exists an  $n$  such that  $A_{ij}^n > 0$  for all  $i, j$ , where  $A_{ij}^n$  is the  $(i, j)$  entry of the power matrix  $A^n$ . If  $\Lambda_{\mathcal{A}}$  is a SFT of  $(\Sigma_2, \sigma)$ , then  $\sigma$  is transitive on  $\Lambda_{\mathcal{A}}$  if and only if  $A$  is irreducible;  $\sigma$  is mixing if and only if  $A$  is aperiodic. Equivalently,  $A$  is irreducible if and only if for every ordered pair of vertices  $b^{(i)}$  and  $b^{(j)}$  there is a path in  $G_{\mathcal{A}}$  starting at  $b^{(i)}$  and ending at  $b^{(j)}$  [13-15].

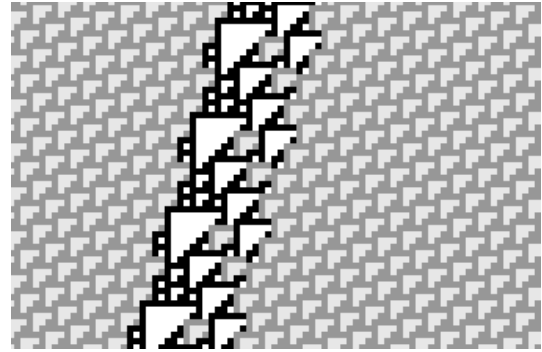
## 3. Glider $E^3$ and The Corresponding Subsystem

### 3.1 Glider $E^3$ of rule 110

A glider is a compact group of non-quiescent states traveling along CA lattice, and is a periodic structure moving in time [9, 10]. From the viewpoint of symbolic dynamics, a glider can be defined as the evolutionary orbit starting from a

special initial configuration in the bi-infinite sequence space  $\Sigma_2$  [16].

It is known that the speed of glider  $E^3$  of rule 110 is  $-4/15$ , the lineal volume is 19, and the even and odd number of periodic margin on the left (or right) border in the ether pattern are 3 and 1 respectively [10]. From the viewpoint of symbolic dynamics, the ether pattern and glider can be defined as the evolutionary orbit starting from a special initial configuration [16]. The ether factor of the ether patterns  $e_r$  (or  $e_l$ ) is  $a = (1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0)$ , and one of glider factors of  $E^3$  is  $b = (1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0)$ , i.e., an ether pattern of rule 110 is the evolutionary orbit  $Orb_{f_{110}}(a^*) = \{a^*, f_{110}(a^*), f_{110}^2(a^*), \dots\}$  and glider  $E^3$  is the evolutionary orbit  $Orb_{f_{110}}(\bar{x}) = \{\bar{x}, f_{110}(\bar{x}), f_{110}^2(\bar{x}), \dots\}$  in the CA lattice space, where  $a^* = (\dots, a, a, a, \dots)$  is a cyclic configuration and  $\bar{x} = (\dots, a, a, a, b, a, a, a, \dots)$ . That the speed of  $E^3$  is  $-4/15$  implies this glider shifts to left by 4 bits in every 15 iterations under rule 110, i.e.,  $f_{110}^{15}(\bar{x}) = \sigma_L^4(\bar{x})$ . The ether pattern and glider  $E^3$  are shown in Fig. 1.

Fig. 1: Ether pattern and glider  $E^3$  of rule 110.

### 3.2 Subsystem of rule 110

First, one can obtain a 31-sequence set  $\mathcal{B} = \{q \mid q = \bar{x}_{[i, i+30]}, \forall i \in \mathbb{Z}\}$  from  $\bar{x} = (\dots, a, a, a, b, a, a, a, \dots)$ , where  $a = (1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0)$  is the ether factor and  $b = (1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0)$  is a glider factor of  $E^3$ , as  $\mathcal{B} =$

```

11111100010011011111000100110111,    1111000100110111110001001101111,
1110001001101111100010011011111,    1100010011011111000100110111110,
1000100110111110001001101111100,    00010011011110001001101111000,
00100110111100010011011110001,    01001101111000100110111100010,
10011011110001001101111000100,    00110111100010011011110001001,
01101111000100110111100010011,    11011110001001101111000100110,
10111100010011011110001001101,    01111000100110111100010011010,
01111000100110111100010011010,    11110001001101111000100110100,
11100010011011110001001101001,    111000100110111100010011010011,
11000100110111100010011010011,    10001001101111000100110100111,

```



```

00010011011111000100110100111111,    00100110111110001001101001111111,
01001101111100010011010011111111,    10011011111000100110100111111110,
00110111110001001101001111111101,    0110111110001001101001111111010,
11011111000100110100111111110101,    1011111000100110100111111101011,
01111100010011010011111111010111,    111110001001101001111111010110,
11110001001101001111111101011100,    111000100110100111111101011001,
11000100110100111111110101110011,    1000100110100111111101011100110,
00010011010011111111010111001101,    0010011010011111110101110011011,
01001101001111111101011100110111,    1001101001111111010111001101111,
00110100111111110101110011011111,    0110100111111101011100110111110,
110100111111110101110011011111000,    1010011111110101110011011111000,
0100111111010111001101111100001,    1001111111010111001101111100010,
0011111101011100110111110000100,    01111111010111001101111100001001,
1111110101110011011111000010011,    11111101011100110111110000100110,
1111010111001101111100001001101,    1111010111001101111100001001011,
1110101110011011110000100110111,    11010111001101111100001001101111,
1010111001101111000010011011111,    01011100110111110000100110111110,
1011100110111100001001101111100,    0111001101111100001001101111000,
1110011011110000100110111100001,    11001101111000010011011111000010}.

```

Let  $\Lambda_0 = \Lambda_{\mathcal{B}} = \{x \in \Sigma_2 \mid x_{[i, i+30]} \in \mathcal{B}, i \in \mathbb{Z}\}$ . Since the 15 times iteration of the local function  $\hat{f}_{110}$  is a map  $\hat{f}_{110}^{15} : S^{31} \rightarrow S$ , and obviously,  $\hat{f}_{110}^{15}(q) = q_{i+4}$  for any  $q = (q_{i-15}, \dots, q_i, \dots, q_{i+15}) \in \mathcal{B}$ . Thus, it follows that  $f_{110}^{15}(x) = \sigma_L^4(x)$  for  $x \in \Lambda_0$ . Furthermore, let

$$\Lambda = \bigcup_{i=0}^{14} f_{110}^i(\Lambda_0).$$

The following propositions can be easily verified.

**Proposition 1:**  $\Lambda$  is closed  $f_{110}$ -invariant set, and  $f_{110}^{15}(x) = \sigma_L^4(x)$  for  $x \in \Lambda$ .

**Proposition 2:**  $\Lambda$  is a subshift of finite type (SFT) of  $\sigma_L$ .

Let  $\mathcal{A}$  be a determinative block system of  $\Lambda$ , then,  $\Lambda = \Lambda_{\mathcal{A}}$ , where  $\mathcal{A}$  is a 31-sequence set consisting of 595 elements. Due to space limitation and simplicity, the decimal code set  $D(\mathcal{A})$  of  $\mathcal{A}$  is placed in Appendix.

### 3.3 Chaoticity of rule 110

In the subsection, the chaoticity of rule 110 on  $\Lambda_{\mathcal{A}}$  will be revealed.

**Proposition 3:**

- (1)  $\sigma_L$  is topologically transitive on  $\Lambda$ ;
- (2)  $f_{110}$  is topologically transitive on  $\Lambda$ .

*Proof:* (1) In fact, it can be verified that for every ordered pair of vertices  $b^{(i)}$  and  $b^{(j)}$  in  $\mathcal{A}$  there is a path in the de Bruijn Diagram  $G_{\mathcal{A}} = \{\mathcal{A}, E\}$  starting at  $b^{(i)}$  and ending at  $b^{(j)}$ , thus, the transition matrix  $A = (A_{ij})_{595 \times 595}$  corresponding to  $G_{\mathcal{A}}$  is irreducible, so  $\sigma_L$  is topologically transitive on  $\Lambda$  [13, 14].

(2) Similar to [17], the topological transitivity of  $f_{110}$  on  $\Lambda$  can be proved. ■

**Proposition 4:** The set of periodic points of  $f_{110}$ ,  $P(f) = \{y \in \Lambda \mid \exists n > 0, f^n(y) = y\}$ , is dense in  $\Lambda$ .

*Proof:* For any  $x \in \Lambda$  and  $\epsilon > 0$ , there exists a positive integer  $M (> 15)$  such that  $\sum_{i=M+1}^{\infty} \frac{1}{2^i} < \epsilon/2$ , and for  $(a_0, \dots, a_{2M}) = x_{[-M, M]} \prec x \in \Lambda$ , it follows that  $(a_{2M-30}, \dots, a_{2M}), (a_0, \dots, a_{30}) \in \mathcal{A}$ . Since  $\sigma$  is transitive on  $\Lambda$ , there exists a path from  $(a_{2M-30}, \dots, a_{2M})$  to  $(a_0, \dots, a_{30})$  in  $G_{\mathcal{A}} = \{\mathcal{A}, E\}$ . Let  $\tilde{b} = (a_{2M-30}, \dots, a_{2M}, b_0, \dots, b_{k_0}, a_0, \dots, a_{30})$  be the sequence corresponding to this path. Then, its any 31-subsequences belong to  $\mathcal{A}$ .

Now, construct a cyclic configuration  $y = c^* = (\dots, c, c, c, \dots)$ , where  $c = (a_0, \dots, a_{2M}, b_0, \dots, b_{k_0})$ . Obviously,  $y \in \Lambda$  and  $\sigma^m(y) = y$ , where  $m = |c|$  is the length of  $c$ . Thus,  $f^{15m}(y) = \sigma^{4m}(y) = y$  and  $y_{[-M, M]} = x_{[-M, M]}$ , i.e.,  $y$  is a periodic point of  $f_{110}$  and  $d(x, y) < \epsilon$ . Therefore, the set of periodic points  $P(f)$  is dense in  $\Lambda$ . ■

**Proposition 5:** The topological entropy of  $f_{110}$  is positive.

*Proof:* The topological entropy of  $f_{110}|_{\Lambda}$  satisfies  $ent(f_{110}) \geq ent(f_{110}|_{\Lambda}) = \frac{4}{15} ent(\sigma_L|_{\Lambda}) = \frac{4}{15} \log(\rho(A)) \approx 0.0179 > 0$ , where  $\rho(A)$  is the spectral radius of the transition matrix  $A$  corresponding to  $\mathcal{A}$ . ■

It is well known that positive topological entropy implies chaos in the sense of Li-Yorke [14, 15], and topological transitivity and density of periodic points imply chaos in the sense of Devaney [23, 24]. Thus, one has the interesting result.

**Theorem 1:**  $f_{110}$  is chaotic in the sense of both Li-Yorke and Devaney on  $\Lambda$ .

## 4. Conclusion

In this paper, it is shown that rule 110 defines a chaotic subsystem on which its global map is topologically transitive, dense periodic points and has positive topological entropy. Thus, it is chaotic in the sense of both Li-Yorke and Devaney. Nevertheless, the complete symbolic dynamical properties of rule 110 are still an open problem. We need to find new ways to uncover its rich and complex dynamics. Conclusively, the result obtained in this paper provides intriguing and valuable clues for researching CA as rule 110.

## Acknowledgments

This research was jointly supported by the NSFC (Grants No. 11171084, 60872093 and 10832006).

## References

- [1] von Neumann, J. *Theory of self-reproducing automata* (edited and completed by A. W. Burks), University of Illinois Press, Urbana and London, 1966.
- [2] Wolfram, S. *A New kind of science*, Champaign Illinois: Wolfram Media, 2002.
- [3] Wolfram, S. *Theory and applications of cellular automata*, World Scientific, Singapore, 1986.

- [4] Lindgren, K. and Nordahl, M. *Universal computation in simple one-dimensional cellular automata*, Complex Systems, 4:229-318, 1990.
- [5] Griffeath, D. and Moore, C. (eds.) *New constructions in cellular automata*, Oxford University Press, 2003.
- [6] McIntosh, H.V. *Rule 110 as it relates to the presence of gliders*, 1999; <http://delta.cs.cinvestav.mx/mcintosh/oldweb/pautomata.html>
- [7] Giles, J. *What kind of science is this?* Nature, 417:216-218, 2002.
- [8] Cook, M. *Universality in elementary cellular automata*, Complex Systems, 15:1-40, 2004.
- [9] Martinez, G.J., McIntosh, H.V. and Seck-Tuoh-Mora J.C. *Gliders in rule 110*, International Journal of Unconventional Computing, 2:1-49, 2005.
- [10] Martinez, G.J., McIntosh, H.V., Seck-Tuoh-Mora J.C. and Chapa-Vergara S.V. *Reproducing the cyclic tag system developed by Matthew Cook with rule 110 using the phases  $f_{i-1}$* , Journal of Cellular Automata, 6:121-161, 2011.
- [11] Martinez, G.J., Adamatzky, A., Stephens, C.R. and Hoeich, A.F. *Cellular automaton supercolliders*, International Journal of Modern Physics C, 22:419-439, 2011.
- [12] Hedlund, G.A. *Endomorphisms and automorphism of the shift dynamical system*, Theory of Computing Systems, 3:320-375, 1969.
- [13] Kitchens, B. *Symbolic dynamics: one-sided, two-sided and countable state markov shifts*, Springer-Verlag, Berlin, New York, 1998.
- [14] Zhou, Z.L. *Symbolic dynamics*, Shanghai Scientific and Technological Education Publishing House, Shanghai, 1997.
- [15] Xiong, J.C., Young, Z. *Chaos caused by a topologically mixing map in dynamical systems and related topics*, World Scientific, Singapore, 1992.
- [16] Chen, F.Y., Shi, L., Chen, G.R. and Jin, W.F. *Chaos and gliders in periodic cellular automaton Rule 62*, Journal of Cellular Automata, in press.
- [17] Chen, F.Y., Chen, G.R., Jin, W.F. *Transitivity and chaoticity in 1-d cellular automata*, Dynamical Systems, (submitted).
- [18] Chua, L.O., Yoon, S., Dogaru, R. *A nonlinear dynamics perspective of Wolfram's new kind of science. Part I: Threshold of complexity*, International Journal of Bifurcation and Chaos, 12:2655-2766, 2002.
- [19] Chua, L.O., Sbitnev, V.I., Yoon, S. *A nonlinear dynamics perspective of Wolfram's new kind of science. Part IV: From bernoulli-shift to 1/f spectrum*, International Journal of Bifurcation and Chaos, 15:1045-1223, 2005.
- [20] Chua, L.O., Sbitnev, V.I., Yoon, S. *A nonlinear dynamics perspective of Wolfram's new kind of science. Part VI: From time-reversible attractors to the arrows of time*, International Journal of Bifurcation and Chaos, 16:1097-1373, 2006.
- [21] Chua, L.O., Guan, J.B., Valery, I.S., Shin, J. *A nonlinear dynamics perspective of Wolfram's new kind of science. Part VII: Isle of eden*, International Journal of Bifurcation and Chaos, 17:2839-3012, 2007.
- [22] Guan, J.B., Shen, S.W., Tang, C.B., Chen, F.Y. *Extending Chua's global equivalence theorem on Wolfram's new kind of science*. International Journal of Bifurcation and Chaos, 17 :4245-4259, 2007.
- [23] Devaney, R.L. *An introduction to chaotic dynamical systems*, Addison-Wesley, 1989.
- [24] Banks, J., Brooks, J., Cairns, G., Davis, G. and Stacey, P. *On the Devaney's definition of chaos*, American Mathematical Monthly, 99:332-334, 1992.

## Appendix

The determinative block system  $\mathcal{A}$  of  $\Lambda$  is a 31-sequence set consisting of 595 elements. For a sequence  $a = (a_0, a_1, \dots, a_{30}) \in \mathcal{A}$ , its decimal code is defined as

$$D(a) = \sum_{i=0}^{30} a_i \cdot 2^{30-i}$$

For simplicity,  $\mathcal{A}$  is replaced by its decimal code set  $D(\mathcal{A})$ .

$D(\mathcal{A}) = \{5078093, 10156187, 14664927, 20312375, 29329854, 32525809, 40624751, 40864190, 51437331, 58659708, 65051618, 65171337, 65790575, 81249503, 81728380, 102874662, 103217307, 104736504, 116956130, 117319416, 120713527, 130103236, 130342675, 131581151, 135134285, 141527486, 159510605, 160549965, 162002381, 162335821, 162499006, 163066125, 163263949, 163447093, 163455031, 163456519, 163456583, 163456671, 163456692, 163456736, 163456748, 163456760, 163456762, 163456763, 163456767, 163461335, 163461569, 163462193, 163481393, 163496437, 163510261, 163574221, 205749325, 206434615, 209473009, 225645436, 233912260, 234638833, 238406733, 241332105, 241427055, 242190782, 242252877, 252476493, 260206473, 260685350, 263162302, 270268571, 283054972, 298601969, 319021211, 321099931, 324004763, 324671643, 324998012, 326132251, 326527899, 326894187, 326910063, 326913039, 326913166, 326913343, 326913384, 326913473, 326913497, 326913520, 326913521, 326913524, 326913526, 326913527, 326913535, 326922670, 326923139, 326924387, 326962787, 326992875, 327020523, 327148443, 404686047, 411498651, 412889231, 418946018, 421315807, 444554463, 449889503, 451290872, 451344539, 452500450, 461574367, 462439551, 467669855, 467796863, 467820668, 467821683, 467823101, 467823427, 467824140, 467824333, 467824512, 467824520, 467824521, 467824526, 467824527, 467824545, 467824560, 467824573, 467824636, 467897715, 467911471, 467911455, 468218655, 468459359, 468680543, 469277666, 469703903, 471798652, 476813467, 482664211, 482854111, 484381564, 484505755, 504952987, 520412947, 521370701, 526324604, 534198212, 540537143, 562675238, 566109944, 567049293, 597203938, 638042423, 642199863, 648009527, 649343287, 649996024, 652264503, 653055799, 653788375, 653820127, 653826079, 653826332, 653826687, 653826768, 653826947, 653826995, 653827040, 653827042, 653827043, 653827048, 653827052, 653827055, 653827071, 653845340, 653846279, 653848775, 653925575, 653985751, 654041047, 654296887, 670420465, 755781105, 778203017, 779061694, 797556262, 809372094, 813530189, 822997303, 825738463, 837892036, 842631614, 861724435, 875640772, 889108926, 899779006, 902581745, 902689079, 905000900, 923148734, 929479102, 935339710, 935593726, 935641336, 935643366, 935646202, 935646854, 935648280, 935648666, 935649024, 935649040, 935649043, 935649052, 935649054, 935649091, 935649120, 935649147, 935649272, 935795430, 935802942, 935822910, 936437310, 936918718, 937361086, 938555332, 939407806, 942738929, 943597304, 953626935, 965328422, 965708222, 968763128, 969011511, 993381873, 1009905975, 1040022605, 1040266737, 1040825894, 1042298865, 1042679745, 1042695985, 1042718679, 1042723889, 1042735300, 1042738391, 1042741252, 1042741376, 1042741400, 1042741401, 1042741402, 1042741403, 1042741403, 1042741475, 1042741479, 1042741488, 1042741788, 1042742023, 1042742239, 1042743239, 1043912497, 1043972593, 1044132337, 1049047537, 1052649208, 1052898801, 1056437745, 1065991718, 1068396425, 1068468670, 1072811505, 1076280870, 1081074287, 1090004728, 1094173919, 1099460489, 1106327492, 1106637111, 1125350477, 1126110076, 1132219889, 1134098587, 1141308966, 1144505567, 1153497126, 1154016806, 1154743014, 1154909734, 1155274886, 1155373798, 1155465370, 1155469339, 1155470083, 1155470115, 1155470159, 1155470170, 1155470192, 1155470198, 1155470204, 1155470205, 1155470207, 1155472491, 1155472608, 1155472920, 1155482520, 1155490042, 1155496954, 1155528934, 1186564542, 1192945190, 1194407876,$

1194837215, 1194868262, 1199980070, 1204084499, 1223042808, 1276084847,  
1280176439, 1283214833, 1284399727, 1296019055, 1298686575, 1299414093,  
1299992049, 1304529007, 1306111599, 1307576751, 1307640255, 1307652158,  
1307652665, 1307653374, 1307653537, 1307653894, 1307653990, 1307654080,  
1307654084, 1307654087, 1307654096, 1307654104, 1307654110, 1307654142,  
1307690681, 1307692559, 1307697551, 1307851151, 1307971503, 1308082095,  
1308593775, 1309641150, 1340840930, 1355079443, 1357266470, 1408952056,  
1451632376, 1462843332, 1463272671, 1472519955, 1480506918, 1504604041,  
1511562210, 1545111288, 1556406035, 1558123388, 1570432760, 1593753126,  
1593875192, 1594891256, 1595081696, 1595089816, 1595101163, 1595103768,  
1595109474, 1595111019, 1595112450, 1595112512, 1595112524, 1595112525,  
1595112561, 1595112563, 1595112568, 1595112718, 1595112835, 1595112943,  
1595113443, 1595698072, 1595728120, 1595807992, 1598265592, 1600191224,  
1601960696, 1606737683, 1607976159, 1610147576, 1611882259, 1618744188,  
1620828783, 1623472068, 1626905570, 1627060379, 1636796862, 1644396307,  
1645994607, 1650490387, 1650750227, 1651113331, 1651196691, 1651379267,  
1651428723, 1651474509, 1651476493, 1651476865, 1651476881, 1651476903,  
1651476909, 1651476920, 1651476923, 1651476926, 1651476927, 1651478069,  
1651478128, 1651478284, 1651483084, 1651486845, 1651490301, 1651506291,  
1667024095, 1670214419, 1671160431, 1671175955, 1673731859, 1675784073,  
1685263228, 1713830043, 1715349240, 1723448870, 1727568866, 1728562399,  
1751281545, 1752375059, 1778217852, 1799558012, 1805163490, 1805378159,  
1810001801, 1813995283, 1826043844, 1846297468, 1858958204, 1870618387,  
1870679420, 1871187452, 1871282672, 1871286732, 1871292405, 1871293708,  
1871296561, 1871297333, 1871298049, 1871298080, 1871298086, 1871298104,  
1871298105, 1871298108, 1871298183, 1871298241, 1871298295, 1871298545,  
1871590860, 1871605884, 1871645820, 1872874620, 1873837436, 1874722172,  
1877110665, 1877729903, 1878815612, 1879682953, 1884156215, 1885477858,  
1887194609, 1892140255, 1895939977, 1898987017, 1899116937, 1899298489,  
1899340169, 1899431457, 1899456185, 1899479078, 1899480070, 1899480256,  
1899480264, 1899480275, 1899480278, 1899480284, 1899480285, 1899480287,  
1899480858, 1899480888, 1899480966, 1899483366, 1899485246, 1899486974,  
1899494969, 1907253871, 1908849033, 1909322039, 1909329801, 1910607753,  
1930656845, 1931416444, 1937526257, 1938023023, 1949929353, 1976430903,  
1978742724, 1980739465, 1986763746, 2009051017, 2012606775, 2013583300,  
2015819931, 2019811951, 2021711812, 2023235332, 2023300292, 2023391068,  
2023411908, 2023457552, 2023469916, 2023481363, 2023481859, 2023481952,  
2023481956, 2023481961, 2023481963, 2023481966, 2023481967, 2023482253,  
2023482268, 2023482307, 2023483507, 2023484447, 2023485311, 2023489308,  
2028166340, 2028402843, 2028406724, 2029045700, 2048706500, 2061957275,  
2063113186, 2064111556, 2078267332, 2080045211, 2080533474, 2081651789,  
2084597730, 2085359490, 2085391970, 2085437358, 2085447778, 2085470600,  
2085476782, 2085482505, 2085482753, 2085482800, 2085482802, 2085482804,  
2085482805, 2085482807, 2085482950, 2085482958, 2085482977, 2085483577,  
2085484047, 2085484479, 2085486478, 2087824994, 2087943245, 2087945186,  
2088264674, 2098095074, 2104720461, 2105298417, 2105797602, 2112875490,  
2116483227, 2117713446, 2126102054, 2126391032, 2131983437, 2132598547,  
2136792851, 2136937340, 2140041097, 2143762372, 2145623010}.

# From Glider to Chaos: A Transitive Subsystem Derived From Glider $\bar{B}$ of CA Rule 110

Pingping Liu, Fangyue Chen, Lingxiao Si, and Fang Wang

School of Science, Hangzhou Dianzi University, Hangzhou, Zhejiang, P. R. China

**Abstract**—Rule 110, a member of Wolfram's class IV and Chua's hyper Bernoulli-shift rules, has been proved to be capable of supporting varieties of mobile self-localizations referred as gliders. This paper is devoted to a careful study for glider  $\bar{B}$  of rule 110 from the viewpoint of symbolic dynamics. A transitive subsystem is revealed based on existing glider  $\bar{B}$ , and its complex dynamics such as having positive topological entropy and density of periodic points are proved. These results therefore suggest that rule 110 is chaotic in the sense of both Li-Yorke and Devaney.

**Keywords:** cellular automata; glider; directed graph; symbolic dynamics; topologically transitive; chaos

## 1. Introduction

Cellular automata (CA) are a class of spatially and temporally discrete, deterministic mathematical systems characterized by local interactions and an inherently parallel form of evolution, and able to produce complex dynamical phenomena by means of designing simple local rules [1]. The study of topological dynamics of CA began with Hedlund in 1969, who viewed one-dimensional CA (1-D CA) in the context of symbolic dynamics as endomorphisms of the shift dynamical system [2], where the main results are the characterizations of surjective and open CA. In 1970, Conway proposed his now-famous "Game of Life" [3], which received widespread interests among researchers in different fields. In the early 1980's, Wolfram introduced space-time representations of 1-D CA and informally classified them into four classes by using dynamical concepts like periodicity, stability and chaos [4, 5]. In 2002, he introduced his monumental work *A New Kind of Science* [6]. Based on this work, Chua *et al.* provided a nonlinear dynamics perspective to Wolfram's empirical observations via the concepts like characteristic function, forward time- $\tau$  map, basin tree diagram and Isle-of-Eden digraph [7-10].

Rule 110 in Stephen Wolfram's system of identification [6] has been an object of special attention due to the structures or gliders which have been observed in evolution space from random initial conditions. A list of gliders is presented in [12]. Wolfram established the conjecture that rule 110 could perform universal computation [13]. Lindgren and Nordahl around 1992 studied the transitional role of rule 110 and its relation with class IV rules sitting between Wolfram's classes II and III [14]. In 1999, Cook gave a brief

introduction about the complex activity of gliders [12], and made a comparison between rule 110 and "Game of Life", finding some similarities and suggesting to call it as left life. Further, he demonstrated that rule 110 is universal via simulating a novel cyclic tag system (CTS) [12, 15] with well-defined blocks of gliders by means of collisions.

Gratefully, the research of CA has drawn more and more scientists' attention in the last 20 years. Many concepts of topological dynamics have been used to describe and classify them [11, 16-19]. The dynamical properties of some robust Bernoulli-shift rules with distinct parameters have been studied in the bi-infinite symbolic sequence space [20-22]. Based on Chua's classes, Bernoulli-shift pattern for rule 110 changes with length, and Bernoulli-shift dynamics of some gliders parameters are relatively large, so it's difficult to investigate their invariant subsystems according to references before. And some gliders with small parameters may not have chaotic subsystems. Thus, this work extends the investigation of Bernoulli-shift dynamics of glider  $\bar{B}$  of rule 110, and reveals its complex dynamics under the framework of bi-infinite symbolic sequence space.

The rest of the paper is organized as follows: Section 2 presents the basic concepts of 1-D CA, symbolic dynamics and glider. Section 3 identifies the subsystem of glider  $\bar{B}$ , and shows the chaotic dynamics of the subsystem. It is indeed remarkably that rule 110 is topologically transitive, possesses positive topological entropy and has a dense periodic set. Therefore, it is chaotic in the sense of both Li-Yorke and Devaney. Finally, Section 4 highlights the main results of this work.

## 2. Symbolic Dynamics and Glider

### 2.1 Symbolic sequence space and CA

Let  $S = \{0, 1\}$ , and  $\Sigma_2 = \{x = (\cdots, x_{-1}, x_0^*, x_1, \cdots) \mid x_i \in S, i \in Z\}$  with distance "d":  $d(x, y) = \sup \left\{ \frac{\rho(x_i, y_i)}{2^{|i|}} \mid i \in Z \right\}$ , where

$$\rho(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i \end{cases} \quad (1)$$

It is known that  $\Sigma_2$  is a compact, perfect, and totally disconnected metric space.

If  $x \in \Sigma_2$  and  $I = [i, j]$  is an interval of integers, put  $x_{[i, j]} = (x_i, x_{i+1}, \cdots, x_j)$  ( $i < j$ ),  $x_{[i, j]} = (x_i, \cdots, x_{j-1})$ .

Table 1: Truth table of local function of rule 110

$(x_{i-1}, x_i, x_{i+1})$	$\hat{f}_{110}(x_{i-1}, x_i, x_{i+1})$
(0, 0, 0)	0
(0, 0, 1)	1
(0, 1, 0)	1
(0, 1, 1)	1
(1, 0, 0)	0
(1, 0, 1)	1
(1, 1, 0)	1
(1, 1, 1)	0

Let  $x_{(-\infty, i]} = (\dots, x_{i-1}, x_i)$  and  $x_{[j, +\infty)} = (x_j, x_{j+1}, \dots)$  denote the left and right half infinite sequence, respectively. For a finite sequence  $a = (a_0, \dots, a_{n-1})$ , if there exists an  $m \in \mathbb{Z}$  such that  $x_{m+k} = a_k$  ( $k = 0, 1, \dots, n-1$ ), then  $a$  is said to be a subsequence of  $x$ , denoted by  $a \prec x$ . The left-shift map  $\sigma_L$  and right-shift map  $\sigma_R$  are defined by  $\forall x \in \Sigma_2$ ,  $[\sigma_L(x)]_i = x_{i+1}$  and  $[\sigma_R(x)]_i = x_{i-1}$ , respectively, where  $[\sigma_L(x)]_i$  ( $[\sigma_R(x)]_i$ ) stands for the  $i$ -th element of  $\sigma_L(x)$  ( $\sigma_R(x)$ ).

By a theorem of Hedlund [2], a map  $f : \Sigma_2 \rightarrow \Sigma_2$  is a CA iff it is continuous and commutes with  $\sigma$ , where  $\sigma$  is left-shift map  $\sigma_L$  or right-shift map  $\sigma_R$ . Furthermore, if  $f$  is a CA, then there exists a radius  $r \geq 1$  and a local function  $\hat{f} : S^{2r+1} \rightarrow S$  such that  $[f(x)]_i = \hat{f}(x_{[i-r, i+r]})$ . If  $r = 1$ , then  $f$  is an elementary CA (ECA). Any CA  $f$  defines a dynamical system  $(\Sigma_2, f)$ . A subset  $X \subseteq \Sigma_2$  is  $f$ -invariant if  $f(X) \subseteq X$ , and strongly  $f$ -invariant if  $f(X) = X$ . If  $X$  is a closed and  $f$ -invariant, then  $(X, f)$  or simply  $X$  is called a subsystem of  $(\Sigma_2, f)$ .

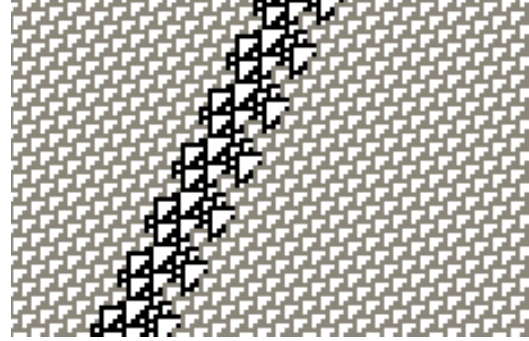
Each ECA rule can be expressed by a local function, the logical truth table of rule 110's local function  $\hat{f}_{110}$  is shown in Table 1. Obviously, the output binary sequence of the rule is 01101110, its decimal number is  $N = 110$ .

## 2.2 Glider $\bar{B}$

A glider is a compact group of non-quiescent states traveling along CA lattice, and is a periodic structure moving in time [15, 25-27]. From the viewpoint of symbolic dynamics, a glider can be defined as the evolutionary orbit starting from a special initial configuration in the bi-infinite sequence space  $\Sigma_2$  [28].

It was known that the speed of glider  $\bar{B}$  of rule 110 is  $-6/12$ , the lineal volume is 22, and the even number of periodic margin on the left (or right) border in the ether pattern are 3 [26]. In symbolic sequence space, the ether pattern and glider can be defined as the evolutionary orbit starting from a special initial configuration [28]. The ether factor of the ether patterns  $e_r$  (or  $e_l$ ) is  $a = (1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0)$ , and one of glider factors of  $\bar{B}$  is  $b = (1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0)$ , i.e., an ether pattern of rule 110 is the evolutionary orbit  $Orb_{f_{110}}(a^*) = \{a^*, f_{110}(a^*), f_{110}^2(a^*), \dots\}$  and glider  $\bar{B}$  is the evolutionary orbit  $Orb_{f_{110}}(\bar{x}) = \{\bar{x}, f_{110}(\bar{x}), f_{110}^2(\bar{x}), \dots\}$  in the

CA lattice space, where  $a^* = (\dots, a, a, a, a, \dots)$  is a cyclic configuration and  $\bar{x} = (\dots, a, a, a, b, a, a, a, \dots)$ . That the speed of  $\bar{B}$  is  $-6/12$  implies this glider shifts to left by 6 bits in every 12 iterations under rule 110, i.e.,  $f_{110}^{12}(\bar{x}) = \sigma_L^6(\bar{x})$ . The ether pattern and glider  $\bar{B}$  are shown in Figure 1.

Fig. 1: Ether pattern and glider  $\bar{B}$  of rule 110.

## 3. Subsystem Derived From Glider $\bar{B}$

### 3.1 Shift of finite type

Based on glider  $\bar{B}$ , one can obtain a 25-sequence set  $\mathcal{B} = \{q | q = \bar{x}_{[i, i+24]}, \forall i \in \mathbb{Z}\}$  from  $\bar{x} = (\dots, a, a, a, b, a, a, a, \dots)$ , where  $a = (1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0)$  is the ether factor and  $b = (1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0)$  is a glider factor of  $\bar{B}$ . More specifically,  $\mathcal{B} =$

```
{0000111100100110111110001, 0001001101111100010011011,
000100110111110000111100, 000111100100110111100010,
001001101111000100110111, 001001101111100001111001,
0011011111000100110111110, 001101111100010011011111,
001101111100001111001001, 001111001001101111000100,
010011011111000100110111, 010011011111000011110010,
011011110001001101111100, 01101111000100110111110,
01101111100001110010011, 011110010011011110001001,
011111000100110111100010, 01111100010011011110000,
011111000011110010011011, 100001111001001101111000,
100010011011110001001101, 100010011011111000011110,
100100110111100010011011, 100110111100010011011111,
10011011111000011100100, 101111100010011011110001,
10111100010011011111000, 110000111100100110111100,
110001001101111100001111, 110010011011110001001101,
11011110001001101111000, 11011110001001101111100,
110111110000111100100110, 11100001110010011011110,
111000100110111100010011, 111000100110111100001111,
111001001101111000100110, 111100001111001001101111,
111100010011011110001001, 11110001001101111000011,
111100100110111100010011, 111110000111001001101111,
111110001001101111000100, 11111000100110111100001,
111111000011100100110111}.
```

The decimal code set  $D(\mathcal{B})$  of  $\mathcal{B}$  is  $D(\mathcal{B}) = \{1986033, 2554011, 2554940, 3972066, 5108023, 5109881, 7309758, 7309759, 7324617, 7944132, 10216047, 10219762, 14619516, 14619518, 14649235, 15888265, 16292834, 16292848, 16530587, 17770232, 18054221, 18054686, 19331227, 20432095, 20439524, 24923633, 24923640, 25042509, 25662332, 25804326, 25804559, 26442829, 29239032, 29239036, 29298470, 29608382, 29679379, 29679495, 29998630, 31581407, 31616905, 31616963, 31776531, 32567919, 32585668, 32585697, 33061175\}$ .

Let  $\Lambda_0 = \Lambda_{\mathcal{B}} = \{x \in \Sigma_2 \mid x_{[i, i+24]} \in \mathcal{B}, i \in \mathbb{Z}\}$ . Since the 12 times iteration of the local function  $\hat{f}_{110}$  is a map  $\hat{f}_{110}^{12} : S^{25} \rightarrow S$ , and obviously,  $\hat{f}_{110}^{12}(q) = q_{i+6}$  for any  $q = (q_{i-12}, \dots, q_i, \dots, q_{i+12}) \in \mathcal{B}$ . Thus, it follows that  $f_{110}^{12}(x) = \sigma_L^6(x)$  for  $x \in \Lambda_0$ . Furthermore, let

$$\Lambda = \bigcup_{i=0}^{11} f_{110}^i(\Lambda_0) \quad (2)$$

The following propositions can be easily verified.

**Proposition 1:**  $\Lambda$  is closed  $f_{110}$ -invariant set, and  $f_{110}^{12}(x) = \sigma_L^6(x)$  for  $x \in \Lambda$ .

**Proposition 2:**  $\Lambda$  is a subshift of finite type of  $\sigma_L$  (SFT).

Let  $\mathcal{A}$  is a determinative block system of  $\Lambda$ , then,  $\Lambda = \Lambda_{\mathcal{A}}$ , where  $\mathcal{A}$  is a 25-sequence set consisting of 435 elements. Due to space limitations and for simplicity, the decimal code set  $D(\mathcal{A})$  of  $\mathcal{A}$  is placed in Appendix.

### 3.2 de Bruijn diagram

The invariant set  $\Lambda = \Lambda_{\mathcal{A}}$  in Propositions 1 and 2 can be described by a finite directed graph,  $G_{\mathcal{A}} = \{\mathcal{A}, E\}$ , where each vertex is labeled by a sequence in  $\mathcal{A}$ , and  $E$  is the set of edges connecting the vertices in  $\mathcal{A}$ . The finite directed graph  $G_{\mathcal{A}}$  is called the de Bruijn diagram of  $\mathcal{A}$  (or  $\Lambda_{\mathcal{A}}$ ). Two vertices  $a = (a_0, \dots, a_{24})$  and  $b = (b_0, \dots, b_{24})$  are connected by an edge  $(a_0, \dots, a_{24}) \rightarrow (b_0, \dots, b_{24})$  if and only if  $(a_1, \dots, a_{24}) = (b_0, \dots, b_{23})$ . One can think of each element of  $\Lambda_{\mathcal{A}}$  as a bi-infinite path on the diagram  $G_{\mathcal{A}}$ . Whereas a de Bruijn diagram corresponds to a square transition matrix  $A = (A_{ij})_{435 \times 435}$  with  $A_{ij} = 1$  if and only if there is an edge from vertex  $b^{(i)}$  to vertex  $b^{(j)}$ , where  $|\mathcal{A}| = 435$  is the number of elements in  $\mathcal{A}$ , and  $i$  (or  $j$ ) is the code of the corresponding vertex in  $\mathcal{A}$ ,  $i, j = 1, 2, \dots, 435$ . Thus,  $\Lambda_{\mathcal{A}}$  is precisely defined by the transition matrix  $A$ . The de Bruijn diagram  $G_{\mathcal{A}}$  associated with  $\mathcal{A}$  is shown in Fig. 2, where the code of vertex represents the location of corresponding element in  $\mathcal{A}$ .

Remarkably, a 0-1 square matrix  $A$  is irreducible if for any  $i, j$ , there exists a positive integer  $n$  such that  $A_{ij}^n > 0$ ; aperiodic if there exists an  $n > 0$ , such that  $A_{ij}^n > 0$  for all  $i, j$ , where  $A_{ij}^n$  is the  $(i, j)$  entry of the power matrix  $A^n$ . If  $\Lambda_{\mathcal{A}}$  is a SFT of  $(\Sigma_2, \sigma)$ , then  $\sigma$  is topologically transitive on  $\Lambda_{\mathcal{A}}$  if and only if  $A$  is irreducible;  $\sigma$  is topologically mixing if and only if  $A$  is aperiodic. Equivalently,  $A$  is irreducible if and only if for every ordered pair of vertices  $b^{(i)}$  and  $b^{(j)}$

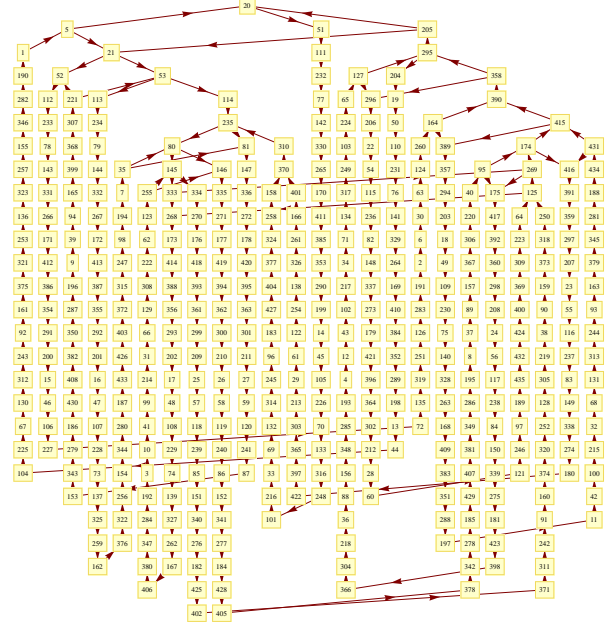


Fig. 2: The de Bruijn diagram  $G_{\mathcal{A}}$  associated with glider  $\bar{\mathcal{B}}$ .

there is a path in  $G_{\mathcal{A}}$  starting at  $b^{(i)}$  and ending at  $b^{(j)}$  [23, 24].

### 3.3 Chaoticity

In the subsection, the chaoticity of rule 110 on  $\Lambda_{\mathcal{A}}$  will be revealed.

**Proposition 3:**

- (1)  $\sigma_L$  is topologically transitive on  $\Lambda$ ;
- (2)  $f_{110}$  is topologically transitive on  $\Lambda$ .

*Proof:* (1) In fact, it can be verified that for every ordered pair of vertices  $b^{(i)}$  and  $b^{(j)}$  in  $\mathcal{A}$  there is a path in the de Bruijn diagram  $G_{\mathcal{A}} = \{\mathcal{A}, E\}$  starting at  $b^{(i)}$  and ending at  $b^{(j)}$ , thus, the transition matrix  $A = (A_{ij})_{435 \times 435}$  corresponding to  $G_{\mathcal{A}}$  is irreducible, so  $\sigma_L$  is topologically transitive on  $\Lambda$  [23, 24].

(2) Similar to [29], the topologically transitive of  $f_{110}$  on  $\Lambda$  can be proved. ■

**Proposition 4:** The set of periodic points of  $f_{110}$ ,  $P(f) = \{y \in \Lambda \mid \exists n > 0, f^n(y) = y\}$ , is dense in  $\Lambda$ .

*Proof:* For any  $x \in \Lambda$  and  $\epsilon > 0$ , there exists an positive integer  $M (> 12)$  such that  $\sum_{i=M+1}^{\infty} \frac{1}{2^i} < \epsilon/2$ , and for  $(a_0, \dots, a_{2M}) = x_{[-M, M]} \prec x \in \Lambda$ , it follows that  $(a_{2M-24}, \dots, a_{2M}), (a_0, \dots, a_{24}) \in \mathcal{A}$ . Since  $\sigma$  is topologically transitive on  $\Lambda$ , there exists a path from  $(a_{2M-24}, \dots, a_{2M})$  to  $(a_0, \dots, a_{24})$  in  $G_{\mathcal{A}} = \{\mathcal{A}, E\}$ . Let  $\tilde{b} = (a_{2M-24}, \dots, a_{2M}, b_0, \dots, b_{k_0}, a_0, \dots, a_{24}) \in \Lambda$

the sequence corresponding to this path. Then, its any 25-subsequence belongs to  $\mathcal{A}$ .

Now, construct a cyclic configuration  $y = c^* = (\dots, c, c, c, \dots)$ , where  $c = (a_0, \dots, a_{2M}, b_0, \dots, b_{k_0})$ . Obviously,  $y \in \Lambda$  and  $\sigma^m(y) = y$ , where  $m = |c|$  is the length of  $c$ . Thus,  $f^{12m}(y) = \sigma^{6m}(y) = y$  and  $y_{[-M, M]} = x_{[-M, M]}$ , i. e.,  $y$  is a periodic point of  $f_{110}$  and  $d(x, y) < \epsilon$ . Therefore, the set of periodic points  $P(f)$  is dense in  $\Lambda$ . ■

**Proposition 5:** The topological entropy of  $f_{110}$  is positive.

*Proof:* The topological entropy of  $f_{110}|_{\Lambda}$  satisfies  $ent(f_{110}) \geq ent(f_{110}|_{\Lambda}) = \frac{6}{12} ent(\sigma_L|_{\Lambda}) = \frac{1}{2} \log((A)) = \frac{1}{2} \log(1.07466) \approx 0.0360 > 0$ , where  $(A)$  is the spectral radius of the transition matrix  $A$  corresponding to  $\mathcal{A}$ . ■

It is well known that positive topological entropy implies chaos in the sense of Li-Yorke [24], and topological transitivity and density of periodic points imply chaos in the sense of Devaney [30, 31]. Thus, one has the interesting result.

**Theorem 1:**  $f_{110}$  is chaotic in the sense of both Li-Yorke and Devaney on  $\Lambda$ .

## 4. Conclusion

In this work, we uncover some dynamic properties of glider  $\bar{B}$  of rule 110. That is, glider  $\bar{B}$  defines one subsystem, on which it is topologically transitive and possesses positive topological entropy and the density of periodic points. Hence, the rule is chaotic in the sense of both Li-Yorke and Devaney on the subsystem. One important problem is how to find the biggest subsystem associating with an existing glider and how to find the relation between universal computation and dynamics of CA, which are important topics for further research in the near future.

## Acknowledgments

This research was jointly supported by the NSFC (Grants No. 11171084, 60872093 and 10832006).

## References

- [1] J. von Neumann, *Theory of self-reproducing automata* (edited and completed by A. W. Burks), University of Illinois Press, Urbana and London, 1966.
- [2] G. A. Hedlund, *Endomorphisms and automorphisms of the shift dynamical system*, *Mathematical Systems Theory*, 3:320-375, 1969.
- [3] M. Gardner, *The fantastic combinations of John Conway's new solitaire game life*, *Scientific American*, 223:120-123, 1970.
- [4] S. Wolfram, *Computation theory of cellular automata*, *Communications in Mathematical Physics*, 96:15-57, 1984.
- [5] S. Wolfram, *Theory and application of cellular automata*, World Scientific, Singapore, 1986.
- [6] S. Wolfram, *A new kind of science*, Wolfram Media, Inc., Champaign, Illinois, 2002.
- [7] L. O. Chua, V. I. Sbitnev and S. Yoon, *A nonlinear dynamics perspective of Wolfram's new kind of science. Part IV: From bernoulli shift to 1/f spectrum*, *International Journal of Bifurcation and Chaos*, 15:1045-1183, 2005.

- [8] L. O. Chua, V. I. Sbitnev and S. Yoon, *A nonlinear dynamics perspective of Wolfram's new kind of science. Part VI: From time-reversible attractors to the arrows of time*, *International Journal of Bifurcation and Chaos*, 16:1097-1373, 2006.
- [9] L. O. Chua, J. B. Guan, I. S. Valery and J. Shin, *A nonlinear dynamics perspective of Wolfram's new kind of science. Part VII: Isle of Eden*, *International Journal of Bifurcation and Chaos*, 17:2839-3012, 2007.
- [10] L. O. Chua, G. E. Paziienza, L. Orzo, V. I. Sbitnev and J. Shin, *A nonlinear dynamics perspective of Wolfram's new kind of science. Part IX: Quasi-Ergodicity*, *International Journal of Bifurcation and Chaos*, 18:2487-2642, 2008.
- [11] J. B. Guan, S. W. Shen, C. B. Tang and F. Y. Chen, *Extending Chua's global equivalence theorem on Wolfram's new kind of science*, *International Journal of Bifurcation and Chaos*, 17:4245-4259, 2007.
- [12] M. Cook, *Introduction to the activity of rule 110* (copyright 1994-1998 Matthew Cook), 1999.
- [13] S. Wolfram, *Cellular automata and complexity: collected papers*, Addison-Wesley Publishing Company, 1994.
- [14] K. Lindgren and M. Nordahl, *Universal computation in simple one-dimensional cellular automata*, *Complex Systems*, 4:229-318, 1990.
- [15] M. Cook, *Universality in elementary cellular automata*, *Complex Systems*, 15:1-40, 2004.
- [16] F. Ohi, *Chaotic properties of the elementary cellular automata rule 168 in Wolfram's class I*, *In AUTOMATA-2008: Theory and applications of cellular automata*, United Kingdom: Luniver Press, 2008.
- [17] J. P. Allouche and G. Skordev, *Remarks on permutive cellular automata*, *Journal of Computer and System Sciences*, 67:174-182, 2007.
- [18] P. Favati, G. Lotti and L. Margara, *Additive one-dimensional cellular automata are chaotic according to Devaney's definition of chaos*, *Theoretical Computer Science*, 174:157-170, 1997.
- [19] J. B. Guan, S. W. Shen, C. B. Tang and F. Y. Chen, *Extending Chua's global equivalence theorem on Wolfram's new kind of science*, *International Journal of Bifurcation and Chaos*, 17:4245-4259, 2007.
- [20] W. F. Jin, F. Y. Chen, G. R. Chen, L. Chen and F. F. Chen, *Extending the symbolic dynamics of Chua's Bernoulli-shift rule 56*, *Journal of Cellular Automata*, 5:121-138, 2010.
- [21] F. Y. Chen, W. F. Jin, G. R. Chen, F. F. Chen and L. Chen, *Chaos of elementary cellular automata rule 42 of Wolfram's class II*, *CHAOS*, 19:013140 1-6, 2009.
- [22] W. F. Jin, F. Y. Chen, G. R. Chen, L. Chen and F. F. Chen, *Complex symbolic dynamics of Chua's period-2 rule 37*, *Journal of Cellular Automata*, 5:315-331, 2010.
- [23] B. Kitchens, *Symbolic dynamics: one-sided, two-sided and countable state Markov shifts*, Springer-Verlag, Berlin, NY, 1998.
- [24] Z. L. Zhou, *Symbolic dynamics*, Shanghai Scientific and Technological Education Publishing House, Shanghai, 1997.
- [25] G. J. Martinez, H. V. McIntosh and J. C. Seck-Tuoh-Mora, *Gliders in rule 110*, *International Journal of Unconventional Computing*, 2:1-49, 2005.
- [26] G. J. Martinez, H. V. McIntosh, J. C. Seck-Tuoh-Mora and S. V. Chapa-Vergara, *Reproducing the cyclic tag system developed by Matthew Cook with rule 110 using the phases  $f_i - 1$* , *Journal of Cellular Automata*, 6:121-161, 2011.
- [27] G. J. Martinez, A. Adamatzky, A. Stephens and A. F. Hoeich, *Cellular automaton supercolliders*, *International Journal of Modern Physics C*, 22:419-439, 2011.
- [28] F. Y. Chen, L. Shi, G. R. Chen and W. F. Jin, *Chaos and gliders in periodic cellular automaton rule 62*, *Journal of Cellular Automata*, in press.
- [29] F. Y. Chen, G. R. Chen, W. F. Jin, *Transitivity and chaoticity in 1-D cellular automata*, *Dynamic Systems*. (submitted)
- [30] R. L. Devaney, *An introduction to chaotic dynamical systems*, Addison-Wesley, 1989.
- [31] J. Banks, J. Brooks, G. Cairns, G. Davis and P. Stacey, *On the Devaney's definition of chaos*, *American Mathematical Monthly*, 99:332-334, 1992.

## Appendix

The determinative block system  $\mathcal{A}$  of  $\Lambda$  is a 25-sequence set consisting of 435 elements. For a sequence

$a = (a_0, a_1, \dots, a_{24}) \in \mathcal{A}$ , its decimal code is defined as

$$D(a) = \sum_{i=0}^{24} a_i \cdot 2^{24-i}. \quad (3)$$

For simplicity,  $\mathcal{A}$  is replaced by its decimal code set  $D(\mathcal{A})$ .

$D(\mathcal{A}) =$

{638502, 771295, 1152760, 1156646, 1277005, 1542590, 1827439, 1866829, 1986033, 2305521, 2309112, 2313293, 2520800, 2526459, 2539896, 2551290, 2553926, 2553996, 2554009, 2554010, 2554011, 2554015, 2554039, 2554081, 2554328, 2554864, 2554940, 2618104, 3008248, 3085180, 3333907, 3391369, 3403375, 3458271, 3654879, 3688088, 3733659, 3864731, 3972066, 4073208, 4611042, 4618225, 4626587, 5041601, 5052919, 5079793, 5102580, 5107852, 5107993, 5108019, 5108020, 5108022, 5108023, 5108031, 5108079, 5108162, 5108657, 5109729, 5109881, 5236209, 6016497, 6021743, 6170360, 6230908, 6451081, 6667814, 6778377, 6782739, 6806751, 6868921, 6916542, 7083913, 7266209, 7308389, 7309518, 7309725, 7309731, 7309744, 7309755, 7309758, 7309759, 7309820, 7310201, 7310865, 7314830, 7323406, 7324617, 7376177, 7467319, 7729463, 7748344, 7790747, 7860535, 7944132, 8146417, 8244721, 8335241, 8845467, 9222084, 9236450, 9239451, 9253175, 10001378, 10083202, 10105838, 10159586, 10205160, 10215705, 10215987, 10216039, 10216040, 10216044, 10216046, 10216047, 10216063, 10216158, 10216324, 10217315, 10219459, 10219762, 10472418, 12032994, 12043487, 12340721, 12461816, 12679323, 12902163, 13066003, 13335629, 13556755, 13565478, 13613502, 13737843, 13833084, 14167827, 14422908, 14532419, 14577545, 14616779, 14619037, 14619451, 14619462, 14619488, 14619510, 14619516, 14619517, 14619518, 14619641, 14620403, 14621731, 14629661, 14646812, 14649235, 14655934, 14719970, 14752354, 14934639, 15136923, 15458927, 15496689, 15581495, 15596059, 15721071, 15808452, 15888265, 15957069, 16270939, 16289007, 16292315, 16292403, 16292613, 16292791, 16292833, 16292834, 16292835, 16292839, 16292847, 16292848, 16293835, 16299931, 16310554, 16373995, 16489442, 16511203, 16529656, 16530587, 16584177, 16656120, 16670483, 17096467, 17162863, 17353596, 17355539, 17690935, 17710630, 17770232, 17931772, 18037616, 18040445, 18047164, 18052861, 18054179, 18054214, 18054220, 18054221, 18054223, 18054235, 18054256, 18054380, 18054648, 18054686, 18086268, 18281340, 18444169, 18472900, 18478903, 18506351, 18621260, 18709581, 18813820, 19331227, 19788087, 19892670, 20002756, 20166404, 20211676, 20319172, 20410320, 20431410, 20431975, 20432078, 20432081, 20432088, 20432093, 20432095, 20432126, 20432316, 20432648, 20434631, 20438919, 20439524, 20651388, 20672589, 20707483, 20899576, 20944836, 21199949, 21396941, 21777905, 23008124, 23116877, 23310217, 23988670, 24065988, 24086974, 24105183, 24137201, 24345677, 24575245, 24681442, 24755750, 24912685, 24921719, 24923373, 24923417, 24923522, 24923611, 24923632, 24923633, 24923635, 24923639, 24923640, 24924133, 24927181, 24932493, 24964213, 25032817, 25042044, 25042509, 25069304, 25105276, 25325449, 25358647, 25454014, 25454985, 25632531, 25662332, 25743102, 25796024, 25797438, 25800798, 25803646, 25804305, 25804323, 25804326, 25804327, 25804333, 25804344, 25804406, 25804540, 25804559, 25820350, 25917886, 26087846, 26132006, 26184126, 26442829, 26671259, 26723551, 26993263, 27102910, 27113510, 27130957, 27227004, 27377190, 27475686, 27666168, 28281278, 28335654, 28432324, 28771551, 28829807, 28845816, 28950054, 29064838, 29155091, 29233558, 29238075, 29238902, 29238924, 29238977, 29239021, 29239032, 29239033, 29239035, 29239036, 29239282, 29240806, 29243462, 29259322, 29293624, 29298238, 29298470, 29311868, 29329854, 29439940, 29504223, 29504708, 29593481,

29608382, 29648767, 29675228, 29675935, 29677615, 29679039, 29679368, 29679377, 29679379, 29679382, 29679388, 29679419, 29679486, 29679495, 29687391, 29736159, 29821139, 29869279, 29998630, 30138991, 30273847, 30328671, 30465811, 30917855, 30993378, 31162991, 31192119, 31252243, 31426335, 31442143, 31529327, 31573956, 31581407, 31601599, 31614830, 31615183, 31616023, 31616735, 31616900, 31616904, 31616905, 31616907, 31616910, 31616925, 31616959, 31616963, 31620911, 31645295, 31687785, 31776531, 31846711, 31914139, 31941551, 32010121, 32403337, 32490383, 32541879, 32564194, 32567919, 32578015, 32584631, 32584807, 32585227, 32585583, 32585666, 32585668, 32585669, 32585671, 32585678, 32585695, 32585697, 32587671, 32599863, 32621108, 32700571, 32747991, 32782276, 32978884, 33022407, 33059313, 33061175, 33070050, 33127501, 33168354, 33312241, 33340966}.



# Parallel Programming of Cellular Automata on Multi-core and Many-core Computers

John-Thones Amenyo

Department of Mathematics & Computer Science, York College – CUNY, New York, NY, USA

**Abstract** - Cellular Automata (CA) models are important for many applications in physics, chemistry, biology and engineering. Thus, programming such CA models to run as parallel applications on modern computer architectures is of great interest. The approach advocated and studied in this paper is to consider CA models as intermediate formalisms that can be mapped into other (better or well-researched) computational paradigms, such as BSP, Multi-BSP, MapReduce, and Resource-Oblivious algorithms. With this approach, the application-to-architecture mapping results that have been obtained can be leveraged to ultimately create CA frameworks that will lead to improvements in (end-user, non-professional) programmer productivity in specifying, developing, maintaining, migrating and evolving CA applications, as the technologies of computer architecture undergo rapid advances in the next decade.

**Keywords:** Cellular Automata; BSP, Multi-BSP; MapReduce; Parallelization; Multi-Core, Many-core; Resource-oblivious, cache-oblivious, network-oblivious.

## 1 Introduction

The Cellular Automata (CA) model is an important intermediate computational formalism (ICF) that lies between many applications and algorithms from many domains, and their implementations on computer architectures and platforms. Although CA simulations can exhibit complex phenomena such as self-organization, emergent behavior, non-linear dynamics, CA models themselves are quite simple and easy to describe and explain. Many computational models in STEM (science: (biology, physics, chemistry, geology), technology, engineering, mathematics) have been studied using CA and related models, such as Lattice Gas [17, 20, 31, 32, 8]. In these models, one uses discrete sets of values for space, time and states of entities in collections. Furthermore, the entity ensembles are organized according to specific (linear or non-linear) rule of interactions and co-dependencies.

It is thus an important issue to determine how CA models and applications can be programmed to execute on modern multi-core and many-core computer systems. These systems include, shared-memory (UMA or NUMA-CCA) multi-processors; heterogeneous systems consisting of CPU multi-core + GPU / GPGPU / Co-processor / Hardware

Accelerator multi-cores and many-cores; as well as clusters, supercomputers, data centers, distributed and inter-networked grid computing, inter-grid computing and cloud computing infrastructures and fabrics, built using the chip level multi-processor (CMP) multi-cores. The mapping of CA applications and algorithms into computer architectures is subject to the constraints or requirements of (end-user, domain expert) programmer productivity, execution performance efficiency, scalability, and fault tolerance. For owners of computational resources, one may further add the requirement of maximizing resource utilization (throughput) at all times. The latter requirement constitutes an important context for parallelizing single tasks (jobs, applications) like a CA model or application.

The current paper discusses using a leveraging technique to accomplish the application to architecture mapping. The mapping of several computational models, paradigms and formalisms onto (homogeneous and heterogeneous) computer architectures, has already been studied quite extensively. Thus, it is useful to determine how the mapping of CA applications and models for execution on modern multi-core and many-core computer architectures can exploit the pre-existing body of knowledge. Furthermore, leveraging such knowledge allows end-users, (non-professional programmers, albeit domain experts and specialists), to concentrate on being productive in devising enhanced parallel algorithms, as well as providing hints about automatic parallelization to compilers and code generators, instead of focusing on creating parallel versions of CA applications from scratch, especially as computer platforms and configurations undergo rapid evolution.

The rest of the paper presents results about (re-)formulating CA models to fit several frameworks, including: BSP (bulk synchronous parallel) [29]; multi-BSP [30]; (Google) MapReduce [11, 22] and variants Hadoop (Apache 2011) [2], Dryad [21], and Twister [12].

## 2 CA Model / Performance Cost Modeling

A CA array consists of  $N$  cells. The cells may be organized into 1-dimensional, 2-dimensional or multi-dimensional (spatial) arrays, according to the coordinate-based naming, addressing, labeling or tagging scheme used to refer to the cells in the array, (Fig. 1). Of practical interest is

to consider the value of  $N$  reaching  $O(10^{10})$  to  $O(10^{12})$ , for STEM simulations. The value of  $N$  is also allowed to vary over time. Each cell in a CA array has associated with it a set of States (S); a Nearest Neighbor or Neighborhood relation (H); and a State Transition Function, Operation or Transformation (R). Each cell may also have its own local, private input and output (IO) data streams associated with it, interfacing with its external, ambient environment. If the  $\langle S, H, R \rangle$ -tuples associated with all the cells are the same, the CA array is termed *homogeneous*, which corresponds to “SIMD” (single instruction multiple data stream) style, (and/or “SPMD” single program multiple data stream style), in parallel computing; otherwise, the CA array is *heterogeneous*, with this case corresponding to the “MIMD” (multiple instruction multiple data stream) style. In classical CA models, the size of  $H$ ,  $|H|$ , is typically  $O(1)$ , with small values, for example, for von Neumann or Moore neighbor template specifications. In computational PDE (partial differential equations) and numerical analysis, templates of neighborhood relations are called *stencils*. Neighborhood templates can also be modeled using polyominoes (as well as variants and related poly-forms, such as pseudo-polyominoes, polyhexes, poly-cubes, etc.). Visualized as graphs, such CA arrays have associated with them sparse adjacency matrices. One can also contemplate the other extreme case of  $|H| = O(N)$ , whereby each cell has all other cells as 1-hop nearest neighbors, the associated graph being a complete graph.

A CA system is an organized collection of CA arrays, [3, 4], where various combinatorial techniques can be used to structure the collection.

In the ensuing discussion, one wishes to distinguish between *synchronous* CA array, subject to unison synchrony, and *multi-synchronous* CA, whereby any cell can operate on its own schedule, without being synchronized with other cells. In the literature, there is also the concept of an *asynchronous* CA [5], whereby, only one randomly chosen cell is allowed to operate at any time.

The starting point for insights about mapping CA applications into architectures is to create a *computational graph* for a CA application (model, computation, simulation). A CA *computation* (simulation run or session) consists of a sequence of  $K$ , ( $K$  can be up to  $O(10^6)$ ), CA simulation computation *supersteps*. Each superstep occurs in each cell, is multi-phase or multi-stage, and consists of a *state transition and update* phase, followed by a *communication-exchange-coordination* phase. Thus, each superstep can be visualized as a (multi-stage) interconnection network with three layers of nodes, forming an orchestration chart, or message sequence chart [UML] or Quipu chart [1]. The first layer corresponds to the  $N$  cells executing state transition updates, and the next 2 layers are devoted to the send and receive portions of the communication-exchange-coordination signal (data and control message) flows with nearest neighbors in  $H$ . Hence, the computational graph of a superstep (and thus of a CA simulation) is a DAG (directed acyclic graph), whose

topology is a multi-stage interconnection network or switching network, (a composition of bipartite graphs), [18, 33].

The *work* that occurs in a superstep consists of computation work, (*P-work*), (of state transition updates), and interoperation work, (*C-work*), (of communication, exchange, coordination, and  $n$  operations), which results in  $O(N)$  of *P-work* and  $O(2N)$  of *C-work*. Let  $T_p$  be the cost of *P-work*, measured in a computer's cycle of instruction executions. Let  $T_c$  be the cost of *C-work*. Then, the cost of a superstep work is  $O(NT_p) + O(2NT_c)$ . Thus, the work in a total simulation computation is  $O(NK)$  of *P-work* and  $O(2NK)$  of *C-work*, with the cost  $O(NKT_p) + O(2NKT_c)$ .

The *span*, *depth* or *critical path length* of a CA simulation application is  $O(K)$  of *P-span* and  $O(2K)$  of *C-span*, with the cost  $O(KT_p) + O(2KT_c)$ . The span is related to the influence of Amdahl's law for sequential limitations or constraints on achievable parallelization.

Define the new concept of the *embrace*, *grasp* or *caliber* of a computational graph to be the amount of parallel operations that can be obtained at any time or stage, (imagine embracing, encompassing, hugging or grasping a tree or a cylindrical object). Then the caliber of a CA superstep (and simulation) is  $O(N)$  during a *P-work* phase and  $O(2N)$  during a *C-work* phase, (see Fig 2).

Let  $T_1^S$  ( $T_1^t$ ) be the cost (in time) for using a single processor in executing a superstep (total simulation). Let  $T_P^S$  ( $T_P^t$ ) be the *cost* (in time) for using a  $P$  processors in executing a superstep (total simulation). Similarly, Let  $T_\infty^S$  ( $T_\infty^t$ ) be the cost (in time) for using an ideal number processors, as well as ideal usage of other resources, in executing a superstep (total simulation). Then the relativity of the cost measures are:  $T_\infty^S < T_P^S < T_1^S$ ;  $T_\infty^t < T_P^t < T_1^t$ . The *speed up* (parallel efficiency) measures are:  $T_1^S / T_P^S$ ;  $T_1^t / T_P^t$ ;  $T_1^S / T_\infty^S$ ;  $T_1^t / T_\infty^t$ . From the CA computational graph, it is easy to determine that  $T_1^S = O(NT_p) + O(2NT_c)$ , and  $T_1^t = O(NKT_p) + O(2NKT_c)$ .

The parallelization costs are more involved and depend on the cost model adopted about targeted computer architecture platforms and configurations. In the *uniform cost* model, all *P-work* unit costs are the same, just as all *C-work* unit costs are the same, irrespective of the (mostly) hierarchical organization of the computer architecture target, (this view corresponds to the flat PRAM and UMA models in parallel computer architecture analysis). In this case,  $T_P^S = O([N/P]T_p) + O(2[N/P]T_c) + O(N T^{Q(P)})$ , and  $T_P^t = O([N/P]KT_p) + O(2[N/P]KT_c)$ , with  $[X]$  being the appropriate

floor or ceiling operator. Here,  $T_c^{Q(P)}$  is a cost adjustment factor that is necessary, if it happens to be the case that the neighborhood sizes  $|H|$  are so large that in some cases, the coordination of the information about all the neighbors of a cell cannot be accommodated in one assignment to a processor.

### 3 Mapping into BSP/Multi-BSP Models

A BSP computation [29] consists of a *sequence of supersteps*, with *barrier synchronization* steps interposed between consecutive supersteps. A superstep consists of a computation step, followed by a communication-exchange-coordination step. The target architecture is characterized and parameterized by a  $\langle P, g, L, h \rangle$ -tuple, (with  $P$ : number of processors,  $g$ : cost for bandwidth,  $L$ : periodicity – a latency cost for synchronization at regular intervals, and  $h$ :  $h$ -relation – maximum number of data messages a processor can send and/or receive during one BSP superstep). In visualizing CA computations via computational graphs, as presented above, it is clear that a synchronous CA simulation is a BSP computation, and thus the results from the latter theory and practice can be directly applied to gain insights about CA computations running on modern architectures. For example, the CA cost unit  $T_c$  can be expressed in terms of the  $g$  and  $h$  parameters in the BSP model. Additionally,  $T_p$  is the cost charge for basic processing of instructions by processes or threads.

The multi-synchronous CA case can also be reformulated in the BSP framework. The neighborhood inter-relations of the cells of a CA array are such that the cells cannot be partitioned into isolated sub-groups. Operationally, after each CA simulation superstep each cell has to necessarily wait for the eventual superstep work completion (and quiescence) of every other cell in the CA array; otherwise, the results of the simulation will not be consistent. Thus, there is an implicit (virtual) global (barrier) synchronization imposed on the whole CA array, even if there is no explicit centralized mechanism being used. This corresponds exactly to the *periodicity* parameter ( $L$ ) in the BSP model.

The BSP model was generalized to the Multi-BSP model [30] to handle the mapping of parallel applications into computer architectures targets which are organized as hierarchical tree-like structures, with (multi-core and many-core) processors at the leaves, and memory storage units with various performance characteristics co-located at different levels of the architectural hierarchy. The new parameterizations for each hierarchical level ( $k$ ) consist of a  $\langle P_k, g_k, L_k, h_k, m_k \rangle$ -tuple. The  $(P, g, L, h)$  parameters have the same interpretations as in the classic BSP model. The most significant aspect embraced by the multi-BSP model is the role of latency costs due to caches, caching and memory hierarchies. For this purpose,  $m_k$  is the amount of memory

(storage capacity) available at level  $k$ , and is accessible to all lower level units.

A 1-level multi-BSP, (1-BSP), is the same as the classic BSP model. Thus,  $T_p^S = O([N/P] T_p) + O(2[N/P] T_c) + O(N T_c^{Q(P)})$ , and  $T_p^t = O([N/P] K T_p) + O(2[N/P] K T_c)$ .

A 2-level multi-BSP, (2-BSP), model has to consider the fact that  $T_c$  becomes very significant, and for some architecture targets can come to dominate and swamp  $T_p$ . It is also the case that  $T_c$  should be decomposed into contributions from intra-grouping ( $T_c^{ip2}$ ) and inter-grouping ( $T_c^{ep2}$ ) communication-exchange-coordination (C-work) efforts and costs, (with  $T_c^{ip2} \ll T_c^{ep2}$ ). In this case,  $T_p^S = O([N/P] T_p) + O(2[N/P] T_c^{ip2}) + O(2([N/P] - 1)[N/P] T_c^{ep2})$ , and  $T_p^t = O([N/P] K T_p) + O(2[N/P] K T_c^{ip2}) + O(2([N/P] - 1)[N/P] T_c^{ep2})$ .

Each higher level, ( $k$ -BSP),  $k > 2$ , in the hierarchy, leads to a nested bracketing or grouping of the processor (and other computational) resources, and thus requires further decompositions into contributions from (nested) inter-grouping communication-exchange-coordination (C-work) costs. In each case, the new contributions easily come to dominate all cost considerations.

### 4 Mapping into MapReduce Models

The MapReduce parallel computing paradigm for (very) large-scale processing was introduced by Google [11], and now has inspired several variations and extensions, such as Apache's Hadoop [2, 10], Microsoft's Dryad [21], and Twister [12] and Amazon's Azure Twister [15]. The paradigm has been used to explore many applications [22]. MapReduce is built on a rich programming tradition, with contributions from LISP programming [MacCarthy]; functional programming [Backus: FP]; function style programming [Haskell, ML]; algorithmic skeletons [Cole]; applicative programming [K. Iverson: APL, J], and nested array programming [T. More]. Originally, the computer architecture targets of MapReduce included clusters; distributed and networked data centers, followed by implementations for grid computing and cloud computing platforms, fabrics and infrastructures, using MPI. Later, there were attempts to implement MapReduce on (shared memory) multi-processors and multi-core computers (GPUs, GPGPUs and heterogeneous multi-core / many-core machines), (UMA and NUMA-CC machines), using OpenMP and CUDA, for example, Mars [16], GPMR [27], Phoenix [23], and StreamR [13].

In the MapReduce model, the computation progression (computational graph) of an application consists of a *Map* computation phase; followed by a data exchange (shuffle) *Intermediate* stage; then followed by a *Reduce* computation phase; (then possibly followed by a single, final result *Merge* stage). In each stage or phase, there are multiple processes

(and processors) executing or running in parallel, simultaneously or concurrently. This composition structure can be regarded as a *MapReduce superstep*. An *Iterative MapReduce* computation consists of a sequence of MapReduce supersteps. Implicit barrier synchronization can be assumed to be interposed between the MapReduce supersteps.

In architectural terms, the Map computation can be regarded as a form of data parallelism inherent in SIMD, MIMD, SPMD styles of computing. Also, both the Map and Reduce phases can be considered, (and in fact, typically implemented), as Master-Worker (Leader-Follower) architecture organizational patterns. The Master divides and allocates the work (broadcast, multicast, scatter), and also receives the (final) results (gather, broad-capture, multi-capture, assembly); while the Workers, (as agents, servants, services, bots), perform the actual computation (work).

A major practical advantage of the MapReduce paradigm concerns programmer productivity. In any of the available and operational MapReduce frameworks, the (end-user, non-professional) programmer provides a) the data set or data stream to be processed, as well as its data structure, such as (key, value)-pairs, b) the function to be used in executing the Map operations, and c) the function to be used in executing the Reduce function. The MapReduce framework then handles (automates) all aspects of parallelization and distribution of both data and computation: data and computation partitioning; inter-process, inter-task, inter-thread, inter-processor communication; data sharing, concurrency management and synchronization; scheduling, work or load balancing, code migration and relocation, code mobility, data-to-code and code-to-data movements; data locality management; IO data access management, fault tolerance, etc.

A CA application (simulation computation) can be regarded as a MapReduce application in the following manner. A CA superstep is identified with a MapReduce superstep as follows. The P-work of computing state transition updates is to be performed by the Map function(s). The C-work of communication-exchange-coordination is divided into two sub-phases of a) sending and receiving state (transition, change) updates, and b) processing the state update information from neighbors (local context, environment, ambience), so that it can be used to support the next (local, self) state transition update. The first C-work sub-phase is identified with the Intermediate Processing stage in MapReduce; and the second sub-phase is identified with the Reduce phase in MapReduce. Thus, a CA simulation is an Iterative MapReduce computation.

In the ideal case of the availability of unlimited computational (processor) resources, (the architecture-ignored or architecture-ignorance case), there are at least N processors that are successively used in a superstep, first as Map workers, then as Intermediate Processing workers, and then as

Reduce workers; or there could even be 3N workers, with N of them devoted to each MapReduce superstep phase. In the practical and realistic case of limited resources  $P \ll N$ , one can analyze the MapReduce superstep for insights about the performance efficiency of (CA / MapReduce) application to architecture mapping, using 1- BSP, 2-BSP, k-BSP, as discussed above, earlier in the paper.

## 5 Related Work & Further Research

The implementation of asynchronous CA systems on multi-core computers has been studied in [3]. The emphasis is on making sure that the decomposition and execution ordering for parallel processors does not result in inconsistent results.

Before the era of modern multi-core based computer architectures, CA applications were implemented on the massively parallel Connection Machine [6, 7]. In addition, there has been research to build dedicated (reprogrammable) hardware for CA application execution [28]. Several software packages had also been built and targeted to PC, workstation and HPC environments, for example, CAMEL [9] and CARPET [25]. Most of these earlier works are likely to be superseded by CA frameworks built to run on modern platforms involving multi-core multi-processors, heterogeneous many-core machines, clusters, HPC supercomputers, grid computing and cloud computing. See for example, libAuToti [26] and CAOS [14]. The latter is a CA programming language which supports both MPI and OpenMP. There have also been several attempts to implement CA applications on GPU multi-cores, using OpenGL and DirectX. Due to the march of technology, these approaches can now be deprecated. What is needed are techniques that migrate easily with the rapid changes in the technologies of using multi-cores and many-cores for heterogeneous computing, co-processors and hardware accelerators. For newer implementations on GPGPUs using CUDA, Direct Compute and OpenCL, see for example [24].

The work discussed here is part of the EUPP (End-User Parallel Programming) Project. The ultimate goal of the EUPP Project, just like similar ones, is to provide many tools that allow (end-user) programmers to expose and express as much potential parallelism that is possible in an application or algorithm, at first, without regard to available machine or computer architecture target resources, a kind of architecture ignorance, or architecture-ignored programming. Then, there are other tools available that the programmers can use to provide parallelization hints, recommendations and suggestions to compilers, code generators, auto-tuners, runtime systems, and OS work-schedulers, which actually do the detailed work of automating the parallel mapping of applications onto specific target machines and architectures.

For example, for CA models and applications, just as with MapReduce applications, in order to maximize programmer productivity, the (end-user) programmer should

be able to indicate in a suitable specification and description language that a) CA is an intended application, b) the spatial array or dimensional organization, as well as compositional and nested organizations into CA systems and complexes, c) state alphabet, d) number of CA cells, and indications about variation over time, e) cell state transition functions, f) hints about target platforms, and g) suggestions or advice about parallelization. A provided CA computational framework will then take care of, (automate), the details of mapping the application into the target (virtual or physical) machine environment: data and computation partitioning and sub-grouping into tasks, processes, threads and thread blocks; synchronization; data sharing management; communication; scheduling; and use of specific parallelization techniques for achieving and maintaining data locality; IO access, data access and communication access / exchange latency hiding; maximization of throughput and resource utilization; and fault tolerance. There is ongoing work that is attempting to implement EUPP frameworks, such as the CA Framework discussed here. Other related existing implementations making transitions into the parallel computing era include both Mathematica and MatLab.

It is also of some interest to revisit the CA application to architecture mapping problem using other bridging models such as LogP, as well as other memory organization models, such as Hierarchical Memory Model, Memory Hierarchy Model and Block Transfer Model. Actual implementations of MapReduce frameworks using OpenCL and DirectCompute, Cilk++, and TBB are also currently missing in the published literature.

An important direction for future research is the mapping of CA models and applications into frameworks and paradigms based on resource-oblivious, (cache-oblivious, processor-oblivious, IO-oblivious, network-oblivious, etc.), algorithms and techniques. These approaches rely on tools such as resource container fit, divide-and-conquer, and work stealing scheduling, in order to support scalability in the face of resource environment changes, dynamics, evolution, fault tolerance and autonomic systems. To these techniques one may add data locality management via (predictive) anticipatory presentation and availability of data in lower level caches, using the knowledge of application computational graphs and their runtime execution, scan, sweep and traversal.

## 6 Summary and Conclusions

Cellular Automata (CA) models are significant paradigms for modeling (the kinetics and dynamics of) several phenomena in STEM research. Therefore, it is worthwhile to find semi-automatic, systematic ways of mapping CA models and applications into modern computer architectures, machines, platforms and fabrics built from both CPU and hardware accelerators, co-processors (GPU,

GPGPU, DSP, FPGA) multi-core and many-core multiprocessors.

It has been shown that CA models can be first mapped into well-studied computational paradigms such as BSP, Multi-BSP, MapReduce, and Resource-oblivious techniques. In this way, extant research results on parallelization mapping of applications to architectures can be leveraged for relatively narrow and specialized domains such as CA.

## 7 References

- [1] Amenyó, J-T (2010), Automatic Programming with Combinatorial Topology of Modular Cellular Automata, Distributed Cellular Automata and Multi-Scale Cellular Automata. 2010 International Conference on Scientific Computing CSC'2010, pp. 33 - 39.
- [2] Apache Hadoop, (2011), <http://hadoop.apache.org/>
- [3] Bandman, O. (2011), Using MultiCore Computers for Implementing Cellular Automata Systems. In (ed.) Malyshkin, V., Parallel Computing Technologies. Proceedings of the 11th Conference. PaCT 2011, pp. 140 – 151.
- [4] Bandman, O. (2007), Coarse-Grained Parallelization of Cellular Automata Simulations. In (ed.) Malyshkin, V., Parallel Computing Technologies. Proceedings of the 7th Conference. PaCT 2007. Lecture Notes in Computer Science (LNCS 4671), Springer-Verlag, Berlin. pp. 370 – 384.
- [5] Bandman, O. (2006), Parallel Simulation of Asynchronous Cellular Automata Evolution. In (eds.) El Yacoubi, S., Chopard, B. and Bandini, S., ACRI 2006. Lecture Notes in Computer Science (LNCS 4173), Springer-Verlag, Berlin. pp. 41 – 47.
- [6] Boghosian, B. (1990a), Computational Physics on the Connection Machine. Computers in Physics. Volume 4, Issue 1 (Jan/Feb), pp. 14 – 33.
- [7] Boghosian, B. (1990b), Data Parallel Computation and Connection Machine. In (ed.) Jen, E., 1989 Lectures in Complex Systems: Proceedings of the 1989 Complex Systems Summer School, Santa Fe, NM, (June 1989), Addison-Wesley, Wesley, MA, pp. 325 – 370.
- [8] Boghosian, B., (1990c). Lattice Gases. In (ed.) Jen, E., 1989 Lectures in Complex Systems: Proceedings of the 1989 Complex Systems Summer School, Santa Fe, NM, (June 1989), Addison-Wesley, Wesley, MA, pp. 293 – 324.
- [9] Cannataro, M., Di Gregorio, S., Rongo, R., Spataro, W., Spenazzo, F, and Talia, D. (1995), A Parallel Cellular Automata Environment on Multicomputers for Computational Science. Parallel Computing. Volume 21, Issue 5, pp. 803 – 823.

- [10] Cloudera (2011), CDH: A Free, Stable Hadoop Distribution Offering RPM, Debian, AWS and Automatic Configuration Options. <http://www.cloudera.com/hadoop>.
- [11] Dean, J. and Ghemawat, S. (2008), MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*. Volume 51 Issue 1, (January 2008), pp. 107 – 113.
- [12] Ekanayake, J., Li, H., Zhang, B., et al. (2010), Twister: A Runtime for Iterative MapReduce. *Proceedings of the First International Workshop on MapReduce and Its Applications, ACM HPDC Conference*
- [13] Elteir, M., Lin, H., Feng, W-C and Scogland, T. (2011), StreamMR: An Optimized MapReduce Framework for AMD GPUs 2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS). pp
- [14] Grelck, C., Penczek F. and Trojahner, K. (2007), CAOS: A Domain Specific Language for the Parallel Simulation of Cellular Automata. In (ed.) Malyskin, V., *PACT 2007. Lecture Notes in Computer Science (LNCS 4671)*, Springer-Verlag, pp. 410 – 417.
- [15] Gunarathne, J., Wu, T-L., Qiu, J. and Fox, G. (2010), MapReduce in the Clouds for Science. *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science, CLOUDCOM'10*. pp. 565 – 572.
- [16] He, B., Fang, W., Luo, B. et al. (2008), Mars: A MapReduce Framework on Graphics Processors. *ACM PACT 08*. pp. 260 – 269.
- [17] Hoekstra, A.G., Kroc, J. and Sloot, P.M.A. (eds.) (2010), *Simulating Complex Systems by Cellular Automata*. Springer-Verlag, New York, NY.
- [18] Hsu, L-H. and Lin, C-K (2009), *Graph Theory and Interconnection Networks*. CRC Press. Boca Baton, FL.
- [19] Hwang, K., Fox, G.C. and Dongarra, J.J. (2012), *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Morgan Kaufman Publ., Waltham, MA.
- [20] Ilachinski, A. (2001), *Cellular Automata: A Discrete Universe*. World Scientific. River Edge, NJ.
- [21] Isard, M., Budi, M., Yu, Y., Birrell, A. and Fetterly (2007), DRYAD: Distributed Data-Parallel Programs from Sequential Building Blocks. *ACM SIGOPS Operating Systems Review*. pp. 59 –72.
- [22] Lee, K-H., Lee, Y-J., Choi, H., Chung, Y.D. and Moon, B. (2010), Parallel Data Processing with MapReduce: A Survey. *ACM SIGMOD Record*. Volume 40, Issue 4 (Jan. 2010), pp. 11-20.
- [23] Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G, and Kozyrakis, C. (2007), Evaluating MapReduce for Multi-core and Multiprocessor Systems. *Proceedings of the 13th Intl. Symposium on High-Performance Computer Architecture (HPCA)*. pp. 13 – 24.
- [24] Rybacki, S., Himmelspach, J., and Uhrmacher, A.M (2009), Experiments with Single Core, Multi-core, and GPU Based Computation of Cellular Automata. *First International Conference on Advances in System Simulation, SIMUL '09*, pp. 62 – 67.
- [25] Spezzano, G. and Talia, D. (2002), Experiences Using High Level Programming for Parallel Cellular Computation. *High Performance Computing Systems and Applications*. Kluwer Publ., pp. 516 – 528.
- [26] Spingola S., D'Ambrosio, D., Spataro, W., Rongo, R. and Zito, G. (2008), Modeling Complex Natural Phenomena with the libAuToti Cellular Automata Library: An example of application to Lava Flows Simulation. *Proceedings of the 2008 International Conference on Parallel and Distributed Processing Techniques and Applications (July 14 – 17, 2008)*, pp. 44 – 50
- [27] Stuart, J. and Owens, J.D. (2011), Multi-GPU MapReduce on GPU Clusters. *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*. pp. 1068 – 1079.
- [28] Toffoli, T. and Margolus, N. (1987), *Cellular Automata Machines: A New Environment for Modeling*. MIT Press. Cambridge, MA.
- [29] Valiant, L.G. (1990), A Bridging Model for Parallel Computation. *Communications of the ACM*. Volume 33 Issue 8, (August 1990) pp. 103 – 111.
- [30] Valiant, L.G. (2011), A Bridging Model for Multi-core Computing. *Journal of Computer and System Sciences*. 77 Volume Issue 1, (January, 2011)
- [31] Wolfram, S. (2002), *A New Kind of Science*. Wolfram Media. Champaign, IL
- [32] Wolfram, S., (1994), *Cellular Automata and Complexity Collected Papers*.
- [33] Xu, J., *Topological Structure and Analysis of Interconnection Networks (2001)*. Kluwer Academic Academic Publishers. Norwell, MA.

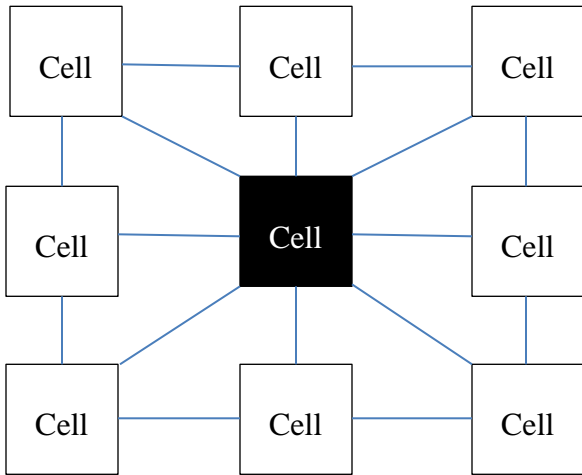


Fig. 1:Portion of CA model showing H-neighborhood of a cell

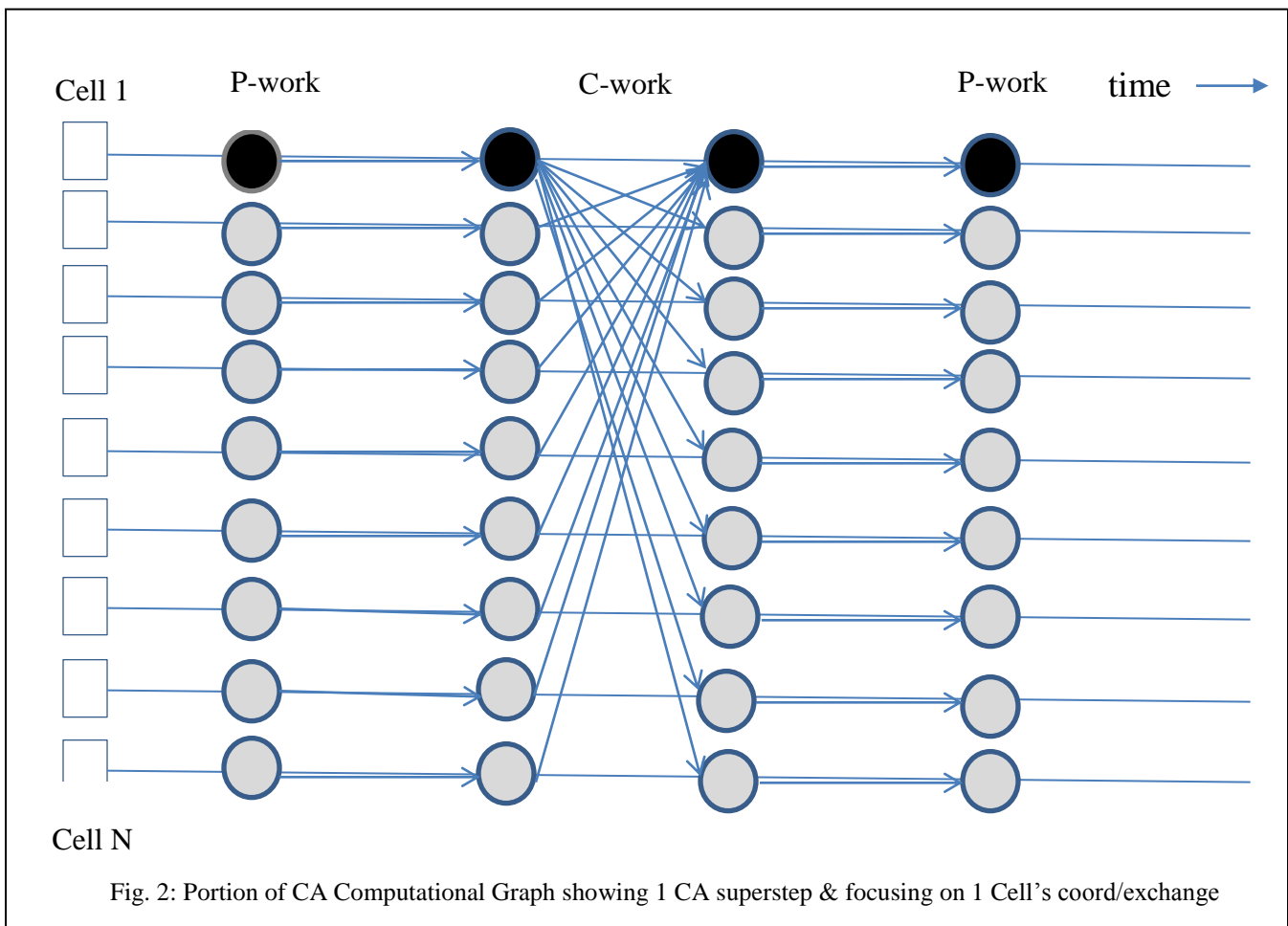


Fig. 2: Portion of CA Computational Graph showing 1 CA superstep & focusing on 1 Cell's coord/exchange

# SCIDDICA-SS<sub>3</sub>: A New Cellular Automata Model for Simulating Fast Moving Landslides

M.V. Avolio<sup>1</sup>, S. Di Gregorio<sup>1</sup>, V. Lupiano<sup>2</sup>, P. Mazzanti<sup>3</sup>, and W. Spataro<sup>1</sup>

<sup>1</sup>Department of Mathematics, University of Calabria, Rende (CS), Italy

<sup>2</sup>Department of Earth Sciences, University of Calabria, Rende (CS), Italy

<sup>3</sup>NHAZCA S.r.l. and Department of Earth Sciences, University of Rome "La Sapienza", Rome, Italy

**Abstract** - Cellular Automata (CA) are discrete and parallel computational models useful for simulating dynamic systems that evolve on the basis on local interactions. Some natural events, such as some types of landslides, fall into this type of phenomena and lend themselves well to be simulated with this approach. This paper describes the latest version of the SCIDDICA CA family models, specifically developed to simulate debris-flows type landslides. The latest model of the family, named SCIDDICA-SS<sub>3</sub>, inherits all the features of its predecessor, SCIDDICA-SS<sub>2</sub>, with the addition of a particular strategy to manage momentum. The introduction of the latter permits a better approximation of inertial effects that characterize some rapid debris flows. First simulations attempts of real landslides with SCIDDICA-SS<sub>3</sub> have produced quite satisfactory results, comparable with the previous model.

**Keywords:** Cellular Automata, Modelling, Debris Flows, SCIDDICA.

## 1 Introduction

Many natural phenomena, like some complex fluid-dynamical phenomena, are difficult to be modelled through standard approaches, such as differential equations [1]. As a consequence, innovative numerical methods emerged from alternative computational paradigms such as Cellular Automata (CA), Neuronal Nets, Genetic Algorithms, etc. (cf. [2], [3], [4]).

It is worth to note that some natural events are difficult to be simulated by valid existing models at a "microscopic" or "mesoscopic" level since they generally evolve on very large areas, thus needing a "macroscopic" level of description. In this case, Macroscopic Cellular Automata (MCA) [5] can represent a valid choice for modelling and simulating these complex dynamical systems like fluid-dynamical natural phenomena. MCA are an extension of classical CA, and were developed in order to model many natural macroscopic events that seem difficult to be modelled in other CA frames, e.g. the Lattice Boltzmann method ([6],[7]), just because they take place on a large space scale. Debris flows, for example, fall in the category of surface flows that evolve on large-scales, and are natural candidates to be modelled through two-dimensional MCA.

In the last years, CA proved to be a valid alternative to differential equations in simulating some complex natural phenomena [8], [5]. In particular, attempts of simulating flow-type landslides have recently been carried out by several authors, also through CA models, with satisfactory results (e.g. [9], [10], [11], [12], [13]). Among these efforts, the SCIDDICA MCA model family was developed for simulating landslides of debris-flows type and were first applied for simulating the Tessina slow-moving earth flow [14]. MCA are also adopted for simulating other phenomena, such as different types of lava flows [15], pyroclastic flows [16], avalanches [17] and, in their latest application, to combined subaerial-subaqueous landslides [18].

SCIDDICA is a family of deterministic MCA models [19] for simulating the behaviour of landslides that can be typologically defined as "flows" [20]. This assertion allows us to exploit on one hand the fact that MCA of the SCIDDICA family are based on the equivalent fluid principle, formalized by Hungr [21], and on the other permits to consider an intrinsic property of MCA, that is that they are considered in terms of a-centric system, i.e. systems whose evolution can be described by considering mainly local interactions among their constituent "elementary" parts [22], a typical characteristic of flows.

In the present paper, the latest SCIDDICA-SS<sub>3</sub> hexagonal release of the model is described. This latest version derives from the need to improve the previous model, SCIDDICA-SS<sub>2</sub> [18], in order to better manage inertial effects. In SCIDDICA-SS<sub>3</sub>, by adopting an empirical strategy, the inertial character of the flowing mass is translated into MCA terms by means of local rules. In general, all SS<sub>x</sub> releases of the SCIDDICA family are an extension to combined subaerial-subaqueous flow-type landslides, with a new flows characterization by their mass centre position and velocity [18]. These characteristics have allowed for a more appropriate characterization of momentum, allowing even for the description of its components along the direction of motion.

While still undergoing a preliminary calibration phase, first simulation attempts by SCIDDICA-SS<sub>3</sub> were performed by taking into account a real case of debris flows, namely the subaerial-subaqueous event which took place near the lake of Albano (Central Italy) in 1997. In the following sections, the model description and produced results will be presented.



## 2 MCA for Surface Flows

For MCA modelling purposes [5], landslides can be viewed as a dynamical system that evolves within a limited portion of the space, tessellated into regular cells (e.g. square, hexagonal). A *state* is defined for each cell that describes the physical characteristics of the corresponding portion of space; in particular, in the MCA framework the *states* of the cell are decomposed in *substates*, or rather the *state* of each cell can be expressed by the Cartesian product of all the considered *substates*, where each *substate* represents a particular feature of the phenomenon to be modelled (e.g., the altitude, depth of soil cover, thickness of landslide debris, landslide energy). *Elementary processes* constitute the *transition function* ( $\tau$ ) of the model: this is composed of a set of rules which describe *local processes* constituting the overall phenomenon. In addition, some *parameters* (e.g. the temporal MCA clock, cell dimension, etc) are generally considered, which allow to “tune” the model for reproducing different dynamical behaviours of the phenomenon of interest, by taking into consideration their physical/empirical meaning. At the beginning of the simulation, cell *states* are initialized by means of input values (e.g., through matrixes). Model *parameters* have also to be assigned in this phase. By simultaneously applying the *transition function*,  $\tau$ , to all cells and at discrete steps, *states* are changed and the evolution of the phenomenon can be simulated.

Natural macroscopic phenomena, which evolve by generating flows of material and involving surface-flows can be modelled through two-dimensional MCA, because the third dimension (i.e., the height) can be managed as a property of the cell (i.e. a *substate*). Thus, it is possible to consider characteristics of the cell (i.e. *substates*), typically expressed in terms of volumes (e.g. debris volume), here in terms of thickness. This simple assumption permits to adopt an efficacious strategy, by means of the *Minimization Algorithm of the Differences* [5] (*Minimization Algorithm* in the rest of the text), based on the hydrostatic equilibrium principle, in order to compute outflows of material (e.g. debris in the case of landslides) from a central cell to the neighbouring ones.

## 3 The SCIDDICA-SS<sub>3</sub> MCA Model

As already mentioned, SCIDDICA is a family of deterministic MCA models, with hexagonal cells, specifically developed for simulating flow-type landslides. The SCIDDICA family includes many versions developed in previous years, from the first release [14], named *T*, to the latest SCIDDICA-SS<sub>2</sub> [18]. This development is also due to a continuous refinement of the adopted approach, which has furthermore given rise to a more physical modelling framework with the development of the SCIDDICA-SS<sub>3</sub> model. The model's transition function latest improvements include a better management procedure for inertial effects which characterise rapid debris flows. In fact, a first attempt of the introduction of momentum was made in an earlier

version, named SCIDDICA-S<sub>4</sub> [23], but the lack of explication of the mass centre did not allow to exploit the full potentiality of the model and was soon abandoned. As a matter of fact, the introduction of the mass centre (or barycentre) in the SS<sub>x</sub> models has allowed a better approximation of the phenomenon from the physical point of view, so to allow to compare the SCIDDICA-SS<sub>x</sub> model with other well-known debris flows models [24]. For instance, a test has regarded the comparison of simulation results of the 1997 Lake Albano (Lazio Region, Italy) debris flow carried out by SCIDDICA-SS<sub>2</sub> and the well-known DAN3D [25] software. Results achieved by DAN3D and SCIDDICA-SS<sub>2</sub> were surprisingly similar in terms of areal debris distribution, velocity and propagation time [24].

The SS<sub>x</sub> releases have been implemented both for the need to simulate combined subareal-subaqueous landslides and to make velocity explicit, a methodological approach firstly applied to lava flows [26], [27]. The precedent release of the SS<sub>x</sub> SCIDDICA family overcomes a typical restriction of many CA models, including Lattice Boltzmann [6] ones: a fluid amount moves from a cell to another one in a CA step, which corresponds usually to a constant time. This implies a constant local “velocity” in the CA context of discrete space/time. Nevertheless, velocities can be deduced by analyzing the global behaviour of the system in time and space. In such models, the flow velocity emerges by averaging on the cell space (i.e. considering clusters of cells) or by averaging on time (e.g. considering the average velocity of the advancing flow front in a sequence of CA steps). Therefore, in SCIDDICA-SS<sub>x</sub> models, with the introduction of the coordinates of mass center of flows and the calculation of their movement, the velocity has been made explicit.

Moreover, in the SS<sub>3</sub> version, a new empirical strategy has been introduced for the determination of outflows from a cell towards its adjacent cells (the other cells of the neighbouring); such a strategy, intuitively discussed later on before its formalisation in section 3.1, has permitted to improve significantly the precision in the complicated computation of momentum for the flowing masses in the MCA context, where a macroscopic view was adopted.

The starting point is the *Minimization Algorithm*, where outflows from a cell are computed in order to obtain hydrostatic equilibrium in the neighbouring. In a cell, inflows and mass inside are composed as quantity, mass centre, kinetic energy and momentum. Momentum introduces an alteration of hydrostatic equilibrium, which is translated in terms of the *Minimization Algorithm* by modifying fictitiously altitudes of adjacent cells: altitude is opportunely lowered/raised according to the module and direction of momentum.

Then, the computation proceeds by two stages:

- a) outflows toward the neighbouring cells are computed in the condition of different altitude alterations: this represents the situation of flows that just can overcome obstacles, neglecting other interactions;
- b) the part of mass that cannot overcome obstacles is considered to interact strongly with such obstacles, in a

complicated play of hitting and bouncing, where momentum disappears and part of kinetic energy is lost. Because of lack of directionality and by considering the remaining kinetic energy, the *Minimization Algorithm* is again applied with identical altitude variation (proportional to residual kinetic energy) in the adjacent cells. This permits to compute outflows toward cells penalized by the direction of momentum in the previous stage.

Substate	Meaning
$Q_a$	Altitude (in meters)
$Q_{th}$	Thickness of landslide debris (in meters)
$Q_d$	Maximum depth of detrital cover that can be transformed by erosion in landslide debris, it depends on the type of detrital cover (in meters)
$Q_o (Q_i)$	Debris outflow (inflow) (in meters)
$Q_E$	Total Energy of landslide debris (in joule)
$Q_{px}, Q_{py}$	Represent indicators of the momentum of the landslide debris, along the outflow velocity directions (in $kg\ m/s$ )
$Q_x, Q_y$	Coordinates of the debris barycentre with reference to the cell centre (in meters)
$Q_{kh}$	Debris kinetic head (in meters)
$Q_{oe} (Q_{oi})$	Part of debris flow, the so called "external flow" ("internal flow"), normalised to a thickness, that penetrates the adjacent cell from central cell (that remains inside the central cell) (in meters)
$Q_{XE}, Q_{YE}$ ( $Q_{Xb}, Q_{Yb}$ )	Coordinates of the external flow barycentre (internal flow barycentre) with reference to the adjacent cell centre (in meters)

**Table 1.** SCIDDICA-SS<sub>3</sub>: list of considered substates.

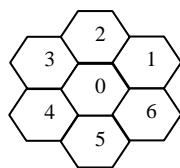
### 3.1 Formal definition of SCIDDICA-SS<sub>3</sub>

The release SS<sub>3</sub> of SCIDDICA is formally defined by the quintuple:

$$\text{SCIDDICA-SS}_3 = \langle R; X; Q; P; \tau \rangle$$

where:

- $R = \{(x, y) \in \mathbb{Z}^2 \mid -l_x \leq x \leq l_x, -l_y \leq y \leq l_y\}$  identifies the hexagonal cellular space where the phenomenon evolves;  $\mathbb{Z}$  is the set of the integer numbers;
- $X = \{0, 0\}, (1, 0), (0, 1), (0, -1), (-1, 0), (-1, 1), (1, -1)\}$  is the geometrical pattern of the neighbourhood of the cell, Fig.1, given by the "central" cell and its six adjacent cells;



**Fig. 1:** The neighbourhood adopted in SCIDDICA-SS<sub>3</sub>. Key: the central cell is individuated by the index "0"; indexes 1–6 identify the cells of the neighbourhood.

- $Q = Q_{x1} \times Q_{x2} \times \dots \times Q_{xn}$  is the finite set of states given by the Cartesian product of the sets of the considered

substates (Table 1). The value of the substate  $x$  in the cell is expressed by  $q_x \in Q_x$ . In the following,  $q_x$  indicates the values of a substate  $Q_x$ . When substates need the specification of the neighbourhood cell, the neighbour index is indicated between square brackets;

- $P$  is the set of the global parameters [18] (e.g., the side of the cell, the temporal correspondence of a step of SCIDDICA-SS<sub>3</sub>, etc) which account for the general frame of the model and the physical characteristics of the phenomenon (e.g., activation threshold of the mobilization, energy dissipation parameter);
- $\tau : Q^7 \rightarrow Q$  is the deterministic transition function for the cells in  $R$ .

At the beginning of the simulation (for  $t=0$ ), the states of the cells in  $R$  must be specified, defining the initial configuration of the MCA. Initial values of the substates (Table 1) are initialized as follows:

- $Q_a$  is set to the cell altitude a.s.l. (bedrock elevation plus depth of soil cover); in the landslide source, the thickness of the landslide debris is subtracted from the morphology;
- $Q_{th}$  is zero everywhere – except for the source area, where the landslide debris thickness is specified;
- $Q_E$  is zero everywhere – except for the source area, where it is equal to the potential energy of the landslide (with reference to the cell altitude);
- $Q_d$  is the depth of the soil cover, which can be eroded by the landslide along the path;
- All remaining substates are set to 0 everywhere.

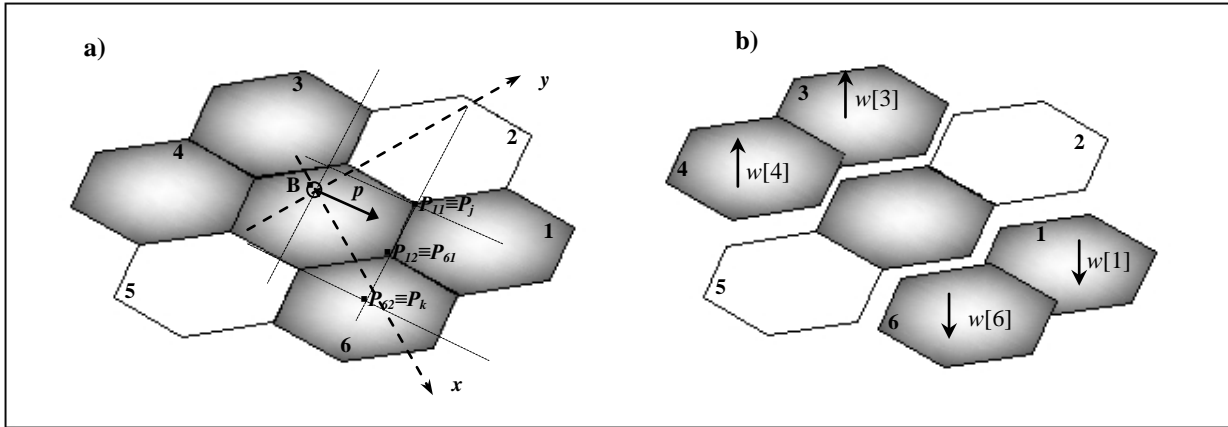
The transition function  $\tau$  is applied, step by step, to all the cells in  $R$ , and the MCA configuration changes obtaining the evolution of the simulation.

### 3.2 Main characteristic of transition function $\tau$ of SCIDDICA-SS<sub>3</sub>

Since SCIDDICA-SS<sub>3</sub> inherits all the features of the transition function of the previous model, this paper briefly describes the elementary processes of SS<sub>3</sub>,  $\tau_{SS3}$ , that are unchanged compared to SS<sub>2</sub>,  $\tau_{SS2}$ . For more details on SS<sub>2</sub>, see [18], while new features are better herein described. In the following,  $(\tau_{SS2}, \tau_{SS3})$  indicates the SS<sub>3</sub> elementary processes that have undergone slight changes (or none) with respect to SS<sub>2</sub>, whereas  $(\tau_{SS3})$  indicates the new SS<sub>3</sub> elementary processes.

**Mobilisation effects** ( $\tau_{SS2}, \tau_{SS3}$ ): An empirical strategy has been developed to better simulate the partial erosion of the regolith along the path of the landslide. In practice, when the kinetic energy overcomes an opportune threshold, then a mobilisation of the detrital cover occurs proportionally to the quantity overcoming the threshold. All heights  $q_a, q_r$ , and  $q_{th}$  and relative energies  $q_{kh}$  and  $q_E$  are updated as soil is eroded.

**Computation of debris outflows** ( $\tau_{SS3}$ ): Outflows from the central cell towards the neighbouring ones are computed by applying the *Minimization Algorithm of the Differences* ([5], [26]). The algorithm is based on the following assumptions:



**Fig. 2:** An example of cell elevation alteration for emphasizing inertial effects. Shaded coloring indicate cells (numbered 1 through 6) affected by the procedure. (a) B is the barycenter of the mass inside a generic central cell with momentum module  $p$ . Segments  $|P_{i(n)}P_{i(n+1)}|$  are determined by  $|P_{11}P_{12}|$  and  $|P_{61}P_{62}|$  for cells 1 and 6, respectively; (b) In the first phase of distribution step, the elevation of cells 1 and 6 are decreased of a quantity  $w[1]$  and  $w[2]$  respectively, while elevations of cells 3 and 4 are increased by the same quantities.

– In the central cell, an unmovable amount,  $u(0)$ , and another one which can be distributed to the neighbours,  $m$ , are in general considered, which is computed as:

$$m = \sum_{i=0}^6 q_o[i]$$

where  $q_o(i)$  is the flow towards the cell  $i$  and  $q_o(0)$  is the part of  $m$  which remains in the central cell

– The content of the adjacent cells,  $u(i)$  ( $i = 1, 2, \dots, 6$ ), is considered unmovable.

Aiming at emphasising inertial effects of rapid debris flows, in the release  $SS_3$ , some directions are privileged according to the momentum of the debris inside the cell. In this regard we consider  $(d_x, d_y)$  the directional components of the velocity of outgoing flows from one cell and applied at the corresponding center of mass (Fig. 2a). Along the directions  $(d_x, d_y)$  it is possible to compute the two indicators  $(q_{px}, q_{py})$  of the momentum given in module as:

$$p = \rho A q_{th} \sqrt{2gq_{kh}}$$

where  $\rho$  is the density of the material of the landslide,  $A$  is the area of the cell,  $q_{th}$  the thickness of debris in a cell,  $g$  the gravity acceleration and  $q_{kh}$  the kinetic head of the landslide. By means of simple conventional geometric processes (cf.

Fig. 2a), the segments  $\overline{P_{i(n)}P_{i(n+1)}}$  are identified, where  $i$  refers to the neighboring cell  $i$  and proportionally to these segments, the amount of motion is “distributed”, in module, to the neighboring cells according to the formula :

$$p_i = \frac{p \times \overline{P_{i(n)}P_{i(n+1)}}}{\overline{P_j P_k}}$$

where  $p_i$  indicates the part of the momentum which is “transferred” in the cell  $i$ ,  $\overline{P_{i(n)}P_{i(n+1)}}$  the length of the

segment corresponding to the cell  $i$  and, at last,  $\overline{P_j P_k}$  is the total length of the segment that goes from  $P_j$  to  $P_k$  (Fig. 2a). The cells towards which  $p_i \neq 0$  are considered as *privileged* ones, as mentioned previously. Within the context of the hexagonal cellular space, along these latter the effect of momentum has to be stronger in order to account for the inertial properties of the debris flow. By definition, if for instance  $p_i > p_j$ , the direction towards cell  $i$  is the most privileged.

With the aim of privileging the directions defined by the various  $p_i$ , in SCIDDICA- $SS_3$  the heights (i.e., cell altitude plus landslide thickness) of the neighbouring cells are increased by an amount  $w[i]$ , ( $i=1, 2, \dots, 6$ ), computed as follows:

- $w[i] = -\alpha p_i$ , for neighbour cells  $i$  which have  $p_i \neq 0$ , where  $\alpha$  is a proportionality coefficient of kinetic energy (Fig 2b);
- $w[i] = \alpha p_i$ , for cells opposite to cells with  $p_i \neq 0$ ;
- $w[i] = 0$  for cells not involved in the motion (Fig. 2b).

In such a way, the *Minimization Algorithm* will privilege, in the distribution of the landslide debris, those cells whose heights are most decremented.

According to the previous considerations, an empirical double-phase strategy was developed in  $SS_3$  to account for the inertial effects of rapid debris flows. In the first phase, inertial outflows  $(q_o[i])$  are first computed, introducing a sort of “weakening” of privileged senses. The *Minimization Algorithm* is applied in the following form:

$$\begin{aligned} m &= q_{th}[0] - p_{adh} \\ u[0] &= q_a[0] + p_{adh} + q_{kh} \\ u[i] &= q_a[i] + q_{th}[i] + w[i] \end{aligned}$$

where  $p_{adh}$  represents the water/air parameter of the adherence value (i.e. unmovable amount of debris in a cell).

The debris thickness remaining into the central cell is accordingly reduced to the value:

$$new\_q_{th}[0] = q_{th}[0] - \sum_{i=1}^6 q_0'[i]$$

while for the receiving cell  $i$ :

$$new\_q_{th}[i] = q_{th}[i] + q_0'[i]$$

This phase obviously involves also the update of the energy  $new\_q_E$  and  $new\_q_{kh}$ , for both the central and neighbor cells  $i$  receiving a flow.

In the second phase, by considering the eventual residual debris,  $new\_q_{th}$ , (i.e. the debris not distributed during the first phase) and kinetic energy,  $new\_q'_{kh}$ , (i.e. the kinetic energy decreased by the only outflows contribution), not-inertial outflows ( $q_0''[i]$ ) in the cell are computed, assuming the inertial effect to be negligible. The *Minimization Algorithm* is now applied in the following form:

$$m = \left[ q_{th}[0] - \sum_{i=1}^6 q_0'[i] \right] - p_{adh}$$

$$u[0] = q_a[0] + p_{adh} + new\_q'_{kh}$$

$$u[i] = q_a[i] + (q_{th}[i] + q_0'[i])$$

At the end of the two phases, the total outflows  $q_0[i]$  for each cell  $i$  of the neighbourhood can be determined as follows:

$$q_0[i] = q_0'[i] + q_0''[i]$$

while the final debris in the cell 0 is:

$$new\_q_{th} = q_{th} - \sum_{i=1}^6 q_0'[i] - \sum_{i=1}^6 q_0''[i]$$

increased by the sum of the incoming flows, in the central cell 0, of the two kinds of flows determined in the previous two phases.

**Shift of the Outflows** ( $\tau_{SS2}$ ,  $\tau_{SS3}$ ): The shift of the outflows is computed for both loops of the same computational step, according to a simple kinetic formula which depends whether the flow is subaerial or subaqueous (as for SS<sub>2</sub>) [18], [26], [27]. In particular, the shift formula for subaqueous debris considers also the water resistance, using modified Stokes equations with a form factor that is proportional to mass.

The substantial difference with SCIDDICA-SS<sub>2</sub> is the calculation of the displacement in the first loop; in fact, at this stage the shift is calculated considering a variation of the height given by the values of  $w[i]$ , previously described.

The movement of flows affect the determination of the computation of the new co-ordinates of the barycentre, that is calculated as the weighted average of  $q_x$  and  $q_y$ , considering the remaining debris in the central cell, the internal flows and the inflows, at the end of the two loop phases.

**Turbulence Effect** ( $\tau_{SS2}$ ,  $\tau_{SS3}$ ): A total energy reduction, besides the computation of the new total energy amount ( $new\_q_E$ ) in a cell, is considered by loss of flows, while an increase is given by inflows; moreover, the new value of the kinetic head ( $new\_q_{kh}$ ) is deduced from the computed kinetic energy and energy dissipation computed over the kinetic head, was considered as a turbulence effect

Obviously, updates of the total and kinetic energy are carried out at the end of the two phases of a step, that is, after the double distribution of the debris.

**Air-Water Interface** ( $\tau_{SS2}$ ,  $\tau_{SS3}$ ): Air-water interface is managed only for external flows from air to water and not vice versa. An external flow from an air cell (altitude higher than water level) to water cell (altitude lower than water level) implies always a loss of matter (water inside debris and components are lighter than water) proportional to debris mass, specified by an opportune parameter, implying a correspondent loss of kinetic energy, determined by kinetic head decrease.

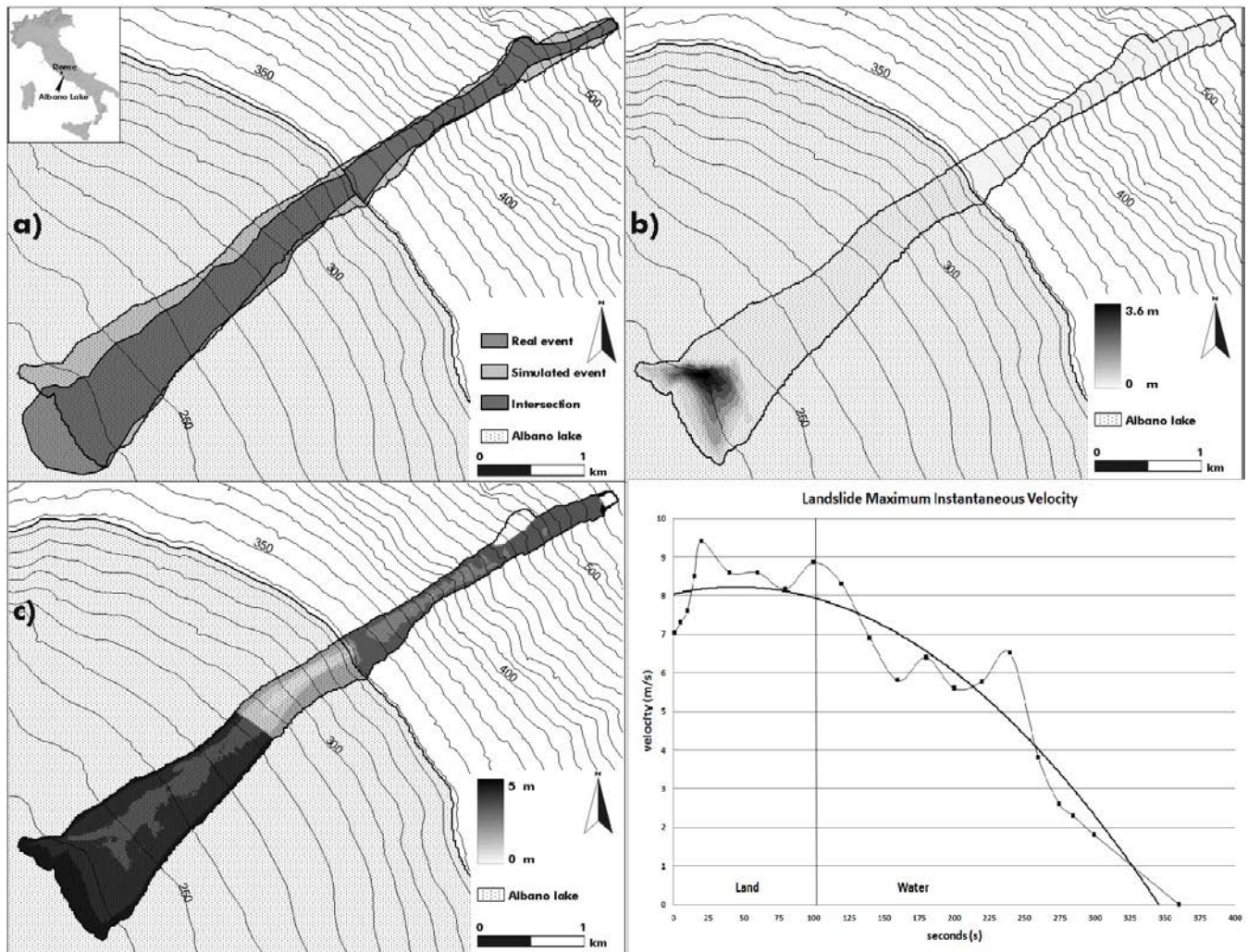
## 4 SCIDDICA-SS<sub>3</sub> Applications

SCIDDICA-SS<sub>3</sub> was calibrated against the 1997 Albano lake (near Rome, Italy) event (Fig. 3a), which is a case of combined subaerial-subaqueous debris-flow [28]. This landslide occurred in the eastern slope of the Albano lake on the 7th of November 1997 after an intense rainfall event (128 mm in 24 hours), and began as a soil slide, mobilizing about 300 m<sup>3</sup> of fluvial material. The mobilized mass was channeled within a steeply dipping impluvium (about 40°) and thus evolved as a debris flow which entrained a large amount of debris material along the bottom of the channel and reached an estimated volume of some thousands of cubic meters at the coastline. A few amount of material was deposited at the coastline, while a greater quantity entered in water generating a little tsunami wave. Simulations permitted to verify the general model and to calibrate adequately its parameters.

In order quantitatively evaluate the simulation outcomes, experiments are compared with real cases by considering the following indicator  $e_1$ :

$$e_1 = \sqrt{\frac{R \cap S}{R \cup S}}$$

where  $R$  is the set of cells affected by the landslide in the real event and  $S$  the set of cells affected by the landslide in the simulation. The fitness function,  $e_1$ , considers a normalised value between 0 (complete failure) and 1 (perfect simulation). Simulations are judged "acceptable" only when the indicators show values not exceeding pre-fixed thresholds of acceptability, fixed on the base of empirical considerations (as commonly performed in statistical analysis). In the case of the Albano landslide, such value is considered as 0.7. The simulation carried out with the SS<sub>3</sub> model, and here presented (Fig. 3a, 3b, 3c) has a value of  $e_1$  equal to 0.82, which represents a very good, though preliminary, result.



**Fig. 3:** The 1997 Albano lake subaerial-subaqueous debris flow. (a) Intersection between real and simulated event; (b) Deposit thickness; (c) Erosion depth; (d) Maximum instantaneous velocity. Results refer to the best performed simulation. The thick contour line represents the water level.

The simulation made with SCIDDICA-SS<sub>3</sub> is quite similar to that made with the previous model [18], both with regard to the areal extent (Fig. 3a) and concerning the mass of mobilized material, which has decreasing thickness values from the release area until the beginning of accumulation zone (Fig. 3b) and, finally, even as regards the thickness of eroded soil during the passage of the landslide (Fig. 3c).

The substantial difference that emerges comparing the simulation carried out with SS<sub>3</sub> with the one made with the SS<sub>2</sub> model is the management of the landslide detachment area. In fact, in the SS<sub>3</sub> model the emptying effect of the detachment area is rather quick, contrarily to what occurred in the SS<sub>2</sub> model, where this area was unable to completely empty until the end of the simulation, showing as a result less compact landslide front. Another difference, emerged from the comparison, is the speed of the landslide which decreases in SS<sub>3</sub>. In fact, the velocity values that emerge from the simulation made with SS<sub>3</sub> (Fig. 3d), though slightly different from those obtained with the model SS<sub>2</sub> [18], can be considered quite consistent with the real case.

Further refinements of the parameters could lead, therefore, for a better fitness as well as to speed up the propagation times.

## 5 Conclusions and future outlooks

SCIDDICA-SS<sub>3</sub> seems to capture fundamental instances of modeling surface flow with regards to conservation laws of physics according to a less empirical approach. Results of first simulations are satisfying, even if more controls and applications to different cases need and parameters must be better tuned.

Future research work is planned: coupling of different elementary processes must be improved, parameters have to be introduced independently from CA time step and cell dimension by an opportune formulation of the transition function.

Such future research will permit to extend such a methodology to other phenomena involving surface flows as snow avalanche, pyroclastic flows, soil erosion and coastal erosion.



**Acknowledgments** - This research was funded by the Italian Instruction, University and Research Ministry (MIUR), PON Project n. 01\_01503 "Integrated Systems for Hydrogeological Risk Monitoring, Early Warning and Mitigation Along the Main Lifelines", CUP B31H11000370005.

## References

- [1] D. L. Stein. "Lectures in the Sciences of Complexity", Addison Wesley, Redwood City, CA, USA, 1989.
- [2] John von Neumann. "Theory of Self Reproducing Automata"; University of Illinois Press, Urbana, 1966.
- [3] T. Kohonen. "Self-organization and associative memory"; Springer Verlag, Heidelberg, 1984.
- [4] J. H. Holland. "Adaptation in natural and artificial systems"; Un. of Michigan Press, Ann Arbor, 1975.
- [5] S. Di Gregorio, R. Serra. "An empirical method for modelling and simulating some complex macroscopic phenomena by cellular automata"; Future Generation Computer Systems, Vol. 16, 259–271, 1999.
- [6] G.R. McNamara, G. Zanetti. "Use of the Boltzmann equation to simulate lattice-gas automata"; Physical Review Letters 61, 2332e2335, 1988.
- [7] S. Succi, R. Benzi, F. Higuera. "The lattice Boltzmann equation: a new tool for computational fluid dynamics"; Physica 47 (D), 219-230, 1991.
- [8] T. Toffoli. "Cellular Automata as an alternative to (rather than an approximation of) differential equations in modeling physics"; Physica 10(D), 117– 127, 1984.
- [9] D. Barca, S. Di Gregorio, F.P. Nicoletta, M. Sorriso-Valvo. "A cellular space model for flow type landslides"; In: Computers and their Application for Development. Proc. Int. Symp. IASTED. Taormina, Italy. Acta Press, Calgary, 30–32, 1986.
- [10] E. Segre, C. Deangeli. "Cellular automaton for realistic modeling of landslides"; Nonlinear Processes in Geophysics 2 (1), 1 – 15, 1995.
- [11] B.D. Malamud, D.L. Turcotte. "Cellular Automata models applied to natural hazards"; IEEE Computing in Science and Engineering 2 (3), 42– 51, 2000.
- [12] A. Clerici, S. Perego. "Simulation of the Parma River blockage by the Corniglio landslide (Northern Italy)"; Geomorphology 33, 1e23, 2000.
- [13] T. Salles, S. Lopez, M.C. Cacas, T Mulder. "Cellular automata model of density currents"; Geomorphology, 88, 1-20, 2007.
- [14] M.V. Avolio, S. Di Gregorio, F. Mantovani, A. Pasuto, R. Rongo, S. Silvano, W. Spataro. "Simulation of the 1992 Tessina landslide by a Cellular Automata model and future hazard scenarios"; Journal of Applied Earth Observation and Geoinformation 2 (1), 41-50, 2000.
- [15] G.M. Crisci, R. Rongo, S. Di Gregorio, W. Spataro. "The simulation model SCIARA: the 1991 and 2001 lava flows at Mount Etna"; Journal of Volcanology and Geothermal Research, Vol. 132, 253–267, 2004.
- [16] M.V. Avolio, G.M. Crisci, S. Di Gregorio, R. Rongo, W. Spataro, D. D'Ambrosio. "Pyroclastic flows modelling using cellular automata"; Computers and Geosciences-UK, Vol. 32, 897–911, 2006.
- [17] M. V. Avolio, A. Errera, V. Lupiano, P. Mazzanti, S. Di Gregorio. A Cellular Automata Model for Snow Avalanches. To appear in Journal of Cellular Automata (2012)
- [18] M.V. Avolio, V. Lupiano, P. Mazzanti, S. Di Gregorio. Modelling combined subaerial-subaqueous flow-like landslides by Cellular Automata. H. Umeo, B. Chopard, and S. Bandini (Eds.): ACRI 2008, LNCS 5191, 329–336, 2008.
- [19] M.V. Avolio, S. Di Gregorio., V. Lupiano, P. Mazzanti, and W. Spataro. Application context of the SCIDDICA model family for simulations of flow-like landslides. Proceedings of The 2010 International Conference on Scientific Computing, Las Vegas (USA), 40-46, CSREA Press 2010 (2010).
- [20] D.M Cruden., D.J. Varnes. Landslide Types and Processes. In: Investigation and Mitigation. Special Report 247, Transportation Research Board, National Research Council, National Academy Press, Washington D.C., 36–75. 1996.
- [21] O. Hungr O. A model for the runout analysis of rapid flow slides, debris flows, and avalanches, Can Geotech J 32: 610–623. 1995.
- [22] J. Petitot. Centrato/a-centrato. Enciclopedia Einaudi, vol. 2 Einaudi, Torino, Italy, 894–954. In Italian. 1977.
- [23] D. D'Ambrosio, G. Iovine, W. Spataro, H. Miyamoto. "A macroscopic collisional model for debris-flows simulation"; Environmental Modelling and Software, Volume 22, 10, 1417-1436, 2007.
- [24] P. Mazzanti, F. Bozzano, M.V Avolio, V. Lupiano, S Di Gregorio. "3D numerical modelling of submerged and coastal landslides propagation"; In: Submarine Mass Movements and Their Consequences IV; Advances in Natural and Technological Hazards Research, Vol 28, The Netherlands, 127-139, 2009.
- [25] S. McDougall, O. Hungr. A model for the analysis of rapid landslide motion across three-dimensional terrain. Canadian Geotechnical Journal 41: 1084-1097. 2004
- [26] M.V. Avolio. "Esplicitazione della velocità per la modellizzazione e simulazione di flussi di superficie macroscopici con automi cellulari ed applicazioni alle colate di lava di tipo etneo". Ph. D. Thesis (in Italian). Dept. of Mathematics, University of Calabria, 2004.
- [27] M.V. Avolio, G.M. Crisci, S. Di Gregorio, R. Rongo, W. Spataro, G.A. Trunfio. "SCIARA  $\gamma$ 2: an improved Cellular Automata model for Lava Flows and Applications to the 2002 Etnean crisis"; Computers & Geosciences, 32, 897-911, 2006.
- [28] Mazzanti, P., Bozzano F., Esposito, C.: Submerged Landslides Morphologies in the Albano Lake (Rome, Italy). In: Lykousis V., Sakellariou, D., Locat, J. (eds), Proc. of 3rd Intern. Symp. "Submarine Mass Movements and Their Consequences", Series: Advances in Natural and Technological Hazards Research , 27, 243-250. Springer, Heidelberg, (2007).

## Cellular automata model for a specific traffic problem

R. Thieberger

*Department of Physics, Ben Gurion University, Beer Sheva 84105, Israel*

(Dated: May 14, 2012)

*We examine the possibility of using alternative roads when confronted with traffic congestion. One is the main road, the second alternative is through a secondary road into which additional cars are coming. To keep the total number of vehicles constant, we subtract on the main road the same number of cars as where added to the bypass. We check the Fourier transform of the average on each traffic light cycles of the velocity on the main road and bypass. In the jammed region we obtained different results for the main road and for the bypass. Whereas for the main road we obtained  $1/f$ , for the bypass we obtained "red noise", i.e.  $1/f^2$ .*

### 1. INTRODUCTION

We wish to examine a specific traffic problem. Entering a city we have two possible routes to take and we wish to examine which one will be faster. Our decision will depend on the specific traffic situation. We will use elementary cellular automata for our purpose.

Cellular automata were first studied by Ulam and von Neumann ([2]). An important contribution to the field was in the work of S. Wolfram [1] who introduced classifications, used in the present study. The elementary cellular automaton is a collection of cells arranged on a one dimensional array. Each cell can obtain just two possible numbers: one and zero. The "time" is discreet and at each time step all the cell values are updated synchronously. The value of each cell depends just on the values in the previous step of that cell and it's two neighbours. Wolfram names each elementary cellular automaton with a binary numeral, which he calls: "rule". This value results from reading the output when the inputs are lexicographically ordered. This will become clearer when we will explain the rules which we use. The rules we used are taken from the cellular automata model as proposed by Gershenson and Rosenblueth[3].

An interesting aspect of our results is the  $1/f^\alpha$  noise, called sometimes pink noise, where the exact "colour" depends on the value of  $\alpha$ . To understand better this term see Procaccia and Schuster[4]. We checked the  $1/f$  noise of the changes in average

velocity for the two regions, using the ideas of Takayasu and Takayasu[5].

### 2. THE MODEL

Empirical observations of traffic show that at high enough densities the behaviour of traffic becomes quite complex. Therefore many traffic models appear in the literature. We will deal here only with the "microscopic" models where we consider each individual vehicle. Our highways are represented by an array of cells, each cell has the values zero or one. one represents a vehicle and zero an empty portion of the highway. We assume that the magnitude of a cell corresponds to the average length of a vehicle. In figure 1, we show the layout of our model. At a certain point we have a bifurcation: there are two different ways to proceed and at a later point they merge again. This model represents in a simplistic way the possibility of using two alternative routes (the main route and the "bypass") when approaching a city from a certain direction of suburbs. We add the possibility that additional cars are coming from the "bypass" and are removed when approaching the city. So that overall the number of vehicles is preserved. The rules, which are the same as used by Gershenson and Rosenblueth [3], are given in Table 1. In figure 2 we give the rules at different locations along our array.

In our analysis we distinguish between three re-

TABLE I: Wolfram rules used in this model

$t-1$	$t_{184}$	$t_{252}$	$t_{136}$
000	0	0	0
001	0	0	0
010	0	1	0
011	1	1	1
100	1	1	0
101	1	1	0
110	0	1	0
111	1	1	1

gions:

- i. The "bypass region" (denoted by iq).
- ii. The region on the main road between the entrance and exit of the "bypass" (denoted by ipe).
- iii. The whole of the main road (denoted by ip).

### 2.1. Measures.

The density,  $\rho$ , is given by the number of 'ones' (i.e. vehicles) divided by the general number of cells. Initially we take this value to be the same for the three sections. We check how this value changes in the different regions. Here we are interested only in the equilibrium values. The velocities,  $v$ , denoted by  $v_p$ ,  $v_q$ ,  $v_{pe}$  are given by the number of cells which change in one step from 0 to 1. The flux of the system, denoted by  $J$ , is given by the product of the density and the velocity.

Another measure which interests us in this study is the ratio between the average time it takes to traverse the "bypass" to the average time it takes on the main road between the two merging points. We will denote this value by 'tq'.

Another value which interests us in this study is the Fourier transform of the velocities  $v_p$  and  $v_q$ . To perform our Fourier analysis we take the averages over each light change cycle and study the frequencies of these averages over all the cycles taken in our calculation. We compare the results to the  $1/f^\alpha$  by least square test.

In our calculation space and time are just ab-

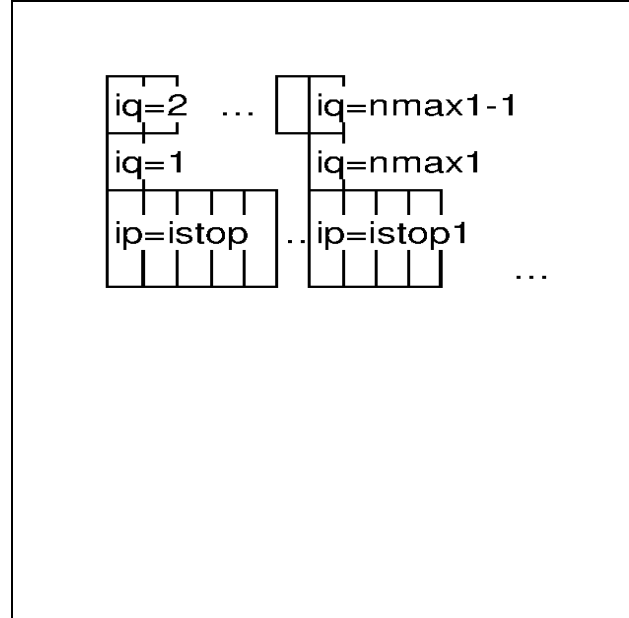


FIG. 1: The grid

stract quantities. Still if somebody of our readers wishes concrete numbers, one can quote[3] were one cell represents five meters, and a time step represents a third of a second, which gives us about 50 km/second, roughly the speed limit within a city.

### 2.2. The grid

A general view of the grid is given in fig 1.

The schematic car movement is given in fig 2.

As we see, we denoted the cells on the main route by ip and the cells on the bypass by iq. The cells between  $ip = istop$  and  $ip = istop1$  we denote by ipe. At  $ip = istop$  the vehicles move on the main road or on the bypass according to the 'lights'. Actually when a car arrives at the 'light' he has the option of going by the bypass or to wait for the change of lights and continue on the main road. Our calculation was partly to check if it is always worth waiting or if one should take the bypass. In reality many drivers when they see the



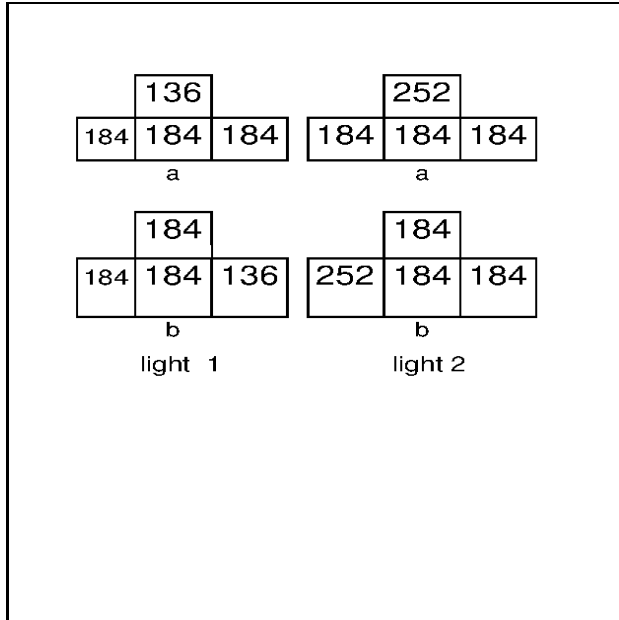


FIG. 2: The movement of vehicles

congestion on the main road decide to take the bypass.

In fig2 we show schematically the movements of the vehicles. We have two stop lights (denoted by '1' and '2' on the diagram). When the movement is in the "main road" diagram 'a' gives us the movement. When we enter or exit the "by pass" then 'b' gives us the rules.

We have a parameter telling us the amount of "cars" added to the "by pass": this same amount is deducted from the "main road" to preserve the total number of vehicles.

### 3. RESULTS AND DISCUSSION

For the most part of our calculations we used a fixed grid: The main road was comprised of 1200 cells, the "by pass" 300 cells and the distance between the two lights was 120 cells. We used the "green wave" regime. As we have just two lights it was shown by Gershenson and Rosenblueth [3]

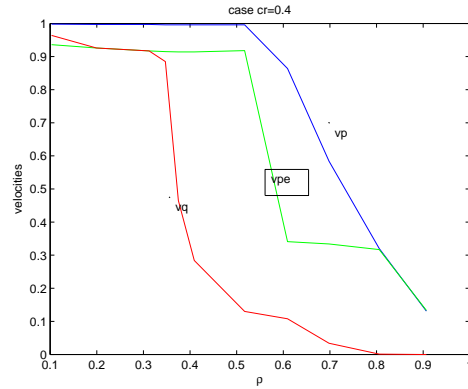


FIG. 3: The case cr=0.4

,that in this case one does not get different results using the "self-organizing" regime.

We checked a number of cases of additions and subtraction of 'vehicles' as described in the last section. We denote by cr the part of cases in which we introduce a vehicle on the "by pass" per unit time. We eliminate the same number of vehicles on the last point of our main route, again per unit time. The cases we checked are:  $cr = 0.1, 0.4$ .

In Fig 3 the case were in 40% of the cycles a vehicle is added to the 'bypass' is shown. The transition to the traffic jam appears is much sooner in the bypass than on the main road. The situation is just a little better when only in 10% of cases (see Fig 4) the jam happens.

In the next figure, Figure 5, we show the relation between the flux, J, and the density for two cases of cr. when no vehicles start out from the bypass i.e.  $cr=0$ , and when 40 % per cycle are introduced. We see a dip in J around  $\rho = 0.4$  for the  $cr=0.4$  case. It is the same effect as in vq versus  $\rho$ , where there is a change in slope.

We wanted also to examine the relation between the time going via the bypass compared to the time it took on the main road as a function of cr. We took a density corresponding to the beginning of the jammed traffic,  $\rho = 0.7$ . The results are shown in Figure 6.

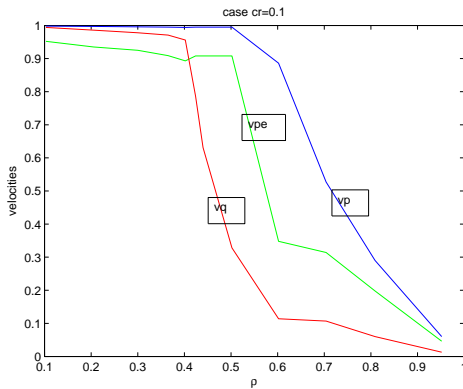


FIG. 4: The case  $cr=0.1$

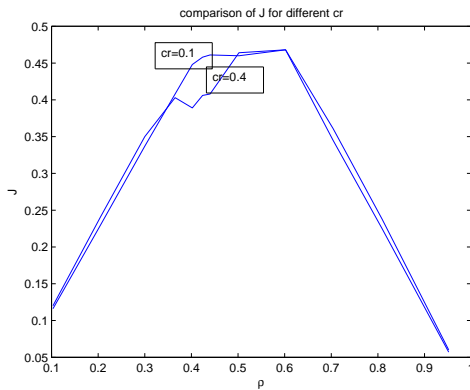


FIG. 5: The flux as a function of density.

Obviously in the case that additional vehicles enter in the bypass the time it takes to go by the bypass is lengthened. But it seems to us interesting that the difference is so great. We can therefore conclude that even if the main road is jammed, it is still better policy to stop and wait for the next green light, and go by the main road. We averaged the velocities over a traffic lights cycle and studied the power spectrum. In fig7 we show the results for  $v_p$ , the same situation we obtained also for  $v_g$ . We see that the noise is white, i.e.  $\alpha = 0$ . In that figure we drew also the  $1/f$  line just to make the result clearer. It was explained by Takayasu and

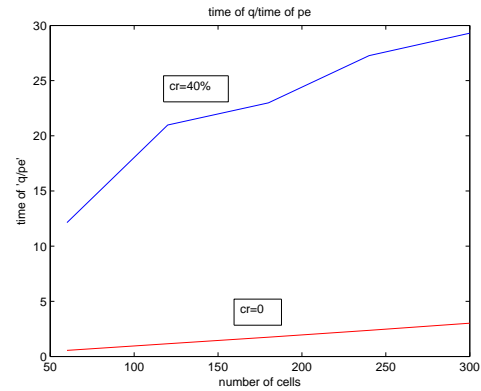


FIG. 6: The relation between the time needed to go by the two routes.

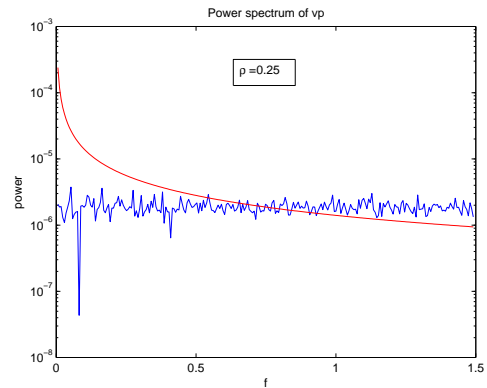


FIG. 7: The white noise region

Takayasu[5], that when the flow is free we obtain a white noise, just as we saw here.

In the jammed region we obtained different results for the main road and for the bypass. whereas for the main road we obtained  $1/f$  just as in[5], For the bypass we obtained "red noise", i.e.  $\alpha = 2$ .

On the bypass the results are different. For the free flow region we also obtain random behaviour (i.e.  $1/f^\alpha$  where  $\alpha = 0$ .) but for the jammed region we obtain a Brownian movement (i.e.  $\alpha = 2$ ).

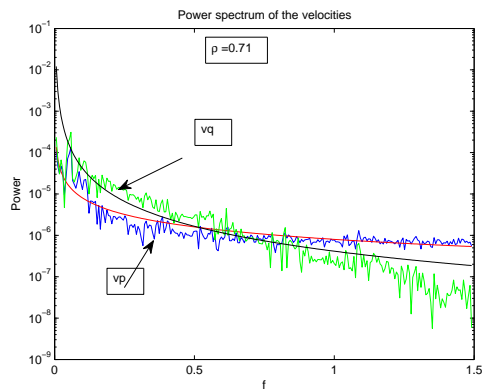


FIG. 8: The jammed region

- 
- [1] S. Wolphram, 1986, *Theory and Application of Cellular Automata.*, World Scientific.
  - [2] J. von Neumann, 1966, *Theory of Self-Reproducing Automata* (Edited and completed by Arthur Burks), University of Illinois Press.
  - [3] Carlos Gershenson and David A. Rosenbluth, 2009, *arXiv:0907.1925v1*.
  - [4] I. Procaccia and H.G. Schuster, 1983, *Phys. Rev.* 28A, 1210-12.
  - [5] Misako Takayasu and Hideki Takayasu, 1993, *Fractals*, vol 1, 860-866.

# Stochastic cellular automata for HIV infection with effects of cell-mediated immunity

M. Precharattana<sup>1,2</sup>, and W. Triampo<sup>1,2</sup>

<sup>1</sup> Insititute for Innovative Learning, Mahidol University, Thailand

<sup>2</sup> R&D Group of Biological and Environmental Physics (BIOPHYSICS), Department of Physics, Faculty of Science, Mahidol University, Thailand

**Abstract** - Ever since the lymphoid tissue is known as HIV's major reservoirs, cellular automaton (CA) models have been introduced, as alternative computational tools, to understand those infections within the infected host. Many years past, several CA models have been introduced which, together, could represent the entire dynamics of HIV infection. However, none of them has taken into account the effects of other immune cells except the changing states of  $CD4^+$  T cells. Therefore, in order to develop a more realistic model, we present a CA model for HIV infection by incorporating the effects of other types of immune cells, in addition to the  $CD4^+$  T cells, such as the  $CD8^+$  T cells and the dendritic cells, called cell-mediated immunity, into the model. Our preliminary results show that the CA model could reproduce the acute phase of HIV infection and could present the rebounding of healthy  $CD4^+$  T cells level by the effect of cell-mediated immunity parameters.

**Keywords:** AIDS; stochastic process; cellular automata; spatial structured model; spatiotemporal pattern formation

## 1 Introduction

Ever since it was reported that lymphoid tissue is the major source of HIV infection [1], cellular automaton (CA) models [2], taking into account the local interactions, have been introduced to understand those infections and discover more in-depth knowledge of the disease within the infected host.

Many years ago, to explain the dynamics of HIV infection within the host's lymphoid tissue, several CA models [3-5] have been adopted. However, those models described the dynamics by focusing on the changing states of only one kind of immune cell, i.e.,  $CD4^+$  T cells. Therefore, when closely look into the computational spatiotemporal pattern formation, it was observed that the results of those simulations are artifacts of the spatial properties inherent in CA models [6] and do not realistically reflect the immune system's response to viral attack, and their counter-response.

Therefore, it becomes our primary objective in this paper to construct a CA model for the investigation of HIV infection spreading over the lymphoid tissue, i.e., the lymph node with parameter values appropriate to the case in which the virus spreads among several kinds of immune cells. Our model is different from those in earlier works in that, in addition to the role of  $CD4^+$  T cells, the role of cell-mediated immunity, namely the dendritic cells and the  $CD8^+$  T cells, among the basic immune cell types associated with HIV infection and generally present in the lymphoid organ, are also incorporated into the system.

Our results demonstrate that our CA model can reproduce the dynamics of immune cells during the acute phase of HIV infection, and in particular the rebounding of  $CD4^+$  T cells which reflects the initiation of HIV-specific immune response to the antigen.

## 2 CA model for HIV infection

### 2.1 Cell states

Using MATLAB, we utilize a 2D cellular automaton (CA) model with a square-shaped grid of size  $L \times L$  to represent a patch of lymphoid tissue within a lymph node. Each grid in the lattice is either an empty site ( $E$ ) or a site randomly occupied by a single state of an immune cell. The kind and state of those immune cells could be: naïve  $CD4^+$  T cell, effector  $CD4^+$  T cell, infected  $CD4^+$  T cell, naïve  $CD8^+$  T cell, effector  $CD8^+$  T cell, mature dendritic cell, and dead. The symbols and meanings of the immune cells used in the model are (see Table 1):

**Table 1** State of immune cells and its meaning in our model.

Name (Symbol)	Meaning
Naïve CD4 <sup>+</sup> T cell ( $N_{T4}$ )	a mature CD4 <sup>+</sup> T cell that has never been in contact with an $M_{DC}$ cell or an $I_{T4}$ cell.
Effector CD4 <sup>+</sup> T cell ( $E_{T4}$ )	a CD4 <sup>+</sup> T cell which stays in an activated state after it had contacted an $M_{DC}$ cell.
Infected CD4 <sup>+</sup> T cell ( $I_{T4}$ )	a CD4 <sup>+</sup> T cell which has been infected after it had contacted with an $M_{DC}$ cell or an $I_{T4}$ cell.
Naïve CD8 <sup>+</sup> T cell ( $N_{T8}$ )	a mature CD8 <sup>+</sup> T cell that has never been in contact with an $M_{DC}$ cell or an $E_{T4}$ cell.
Effector CD8 <sup>+</sup> T cell ( $E_{T8}$ )	a CD8 <sup>+</sup> T cell which stays in an activated state after it had contacted with an $M_{DC}$ cell or an $E_{T4}$ cell. A cell in this state can kill an $M_{DC}$ and an $I_{T4}$ cell. Such a cell is referred to as a cytotoxic T lymphocyte; CTL in real life.
Mature dendritic cell ( $M_{DC}$ )	a dendritic cell that arrests HIV at the area of infection and then brings the HIV to the lymph node to present HIV to the other immune cells locating in the lymphoid organ.
Dead cell ( $D$ )	a state of immune cell that dies because it is out of date or faces with HIV-specific immune response.

## 2.2 Initial condition and data collection

The initial configuration is depicted as a lymph node randomly consists of the three fundamental states of immune cells, i.e., the  $N_{T4}$  cells, the  $N_{T8}$  cells, and the  $M_{DC}$  cells with the probability  $P_{N_{T4}}$ ,  $P_{N_{T8}}$ , and  $P_{M_{DC}}$ , respectively. To process each configuration, all states of immune cells, except the dead state, are randomly moved. This means that our model represents a part of the lymph node as an empty site ( $E$ ) that any immune cell could randomly move to and temporary occupy, or a site occupied by a specific state of an immune cell. Then, the state of each cell is updated according to the CA rules listed below. The number of each cell type is counted and noted. Following, the configuration is announced as one time step, being taken in the unit of four hours in real life (see also Figure 1). Table 2 contains the values of all parameters used in our model.

### CA Rules

#### (1) Update rule for $N_{T4}$ cell

If an  $N_{T4}$  cell comes into contact with  $k$   $I_{T4}$  cell, it becomes an  $I_{T4}$  cell with the probability  $1 - (1 - P_{I_{T4}})^k$ . Otherwise, it stays unchanged. If the  $N_{T4}$  cell comes into contact with  $k$   $M_{DC}$  cell, it becomes an  $I_{T4}$  cell with the probability  $1 - (1 - P_{I_{T4}})^k$  or an  $E_{T4}$  cell with the probability  $1 - (1 - (1 - P_{I_{T4}})^k)$ . Moreover, the  $N_{T4}$  cell becomes a  $D$  cell after  $\tau_{N_{T4}}$  time steps.

#### (2) Update rule for $E_{T4}$ cell

If an  $E_{T4}$  cell comes into contact with  $k$   $I_{T4}$  or  $M_{DC}$  cells, it becomes an  $I_{T4}$  cell with the probability  $1 - (1 - P_{I_{T4}})^k$ . Otherwise, it stays unchanged. Moreover, the  $E_{T4}$  cell becomes a  $D$  cell after  $\tau_{E_{T4}}$  time steps.

#### (3) Update rule for $I_{T4}$ cell

An  $I_{T4}$  cell becomes a  $D$  cell after  $\tau_{I_{T4}}$  time steps, or if it comes into contact with at least one  $E_{T8}$  cell.

#### (4) Update rule for $N_{T8}$ cell

An  $N_{T8}$  cell becomes an  $E_{T8}$  cell if it comes into contact with at least one  $M_{DC}$  cell or one  $E_{T4}$  cell, or becomes a  $D$  cell after  $\tau_{N_{T8}}$  time steps.

#### (5) Update rule for $E_{T8}$ cell

An  $E_{T8}$  cell becomes a  $D$  cell after  $\tau_{E_{T8}}$  time steps. Otherwise, it stays unchanged.

#### (6) Update rule for $M_{DC}$ cell

An  $M_{DC}$  cell becomes a  $D$  cell after  $\tau_{M_{DC}}$  time steps, or if it comes into contact with at least one  $E_{T8}$  cell.

#### (7) Update rule for $D$ cell

A  $D$  cell is replenished by either an  $M_{DC}$  cell with the probability  $P_{repl\_M_{DC}}$ , an  $N_{T4}$  cell with the probability  $P_{repl\_N_{T4}}$ , or an  $N_{T8}$  cell with the probability  $P_{repl\_N_{T8}}$  in the next time step.

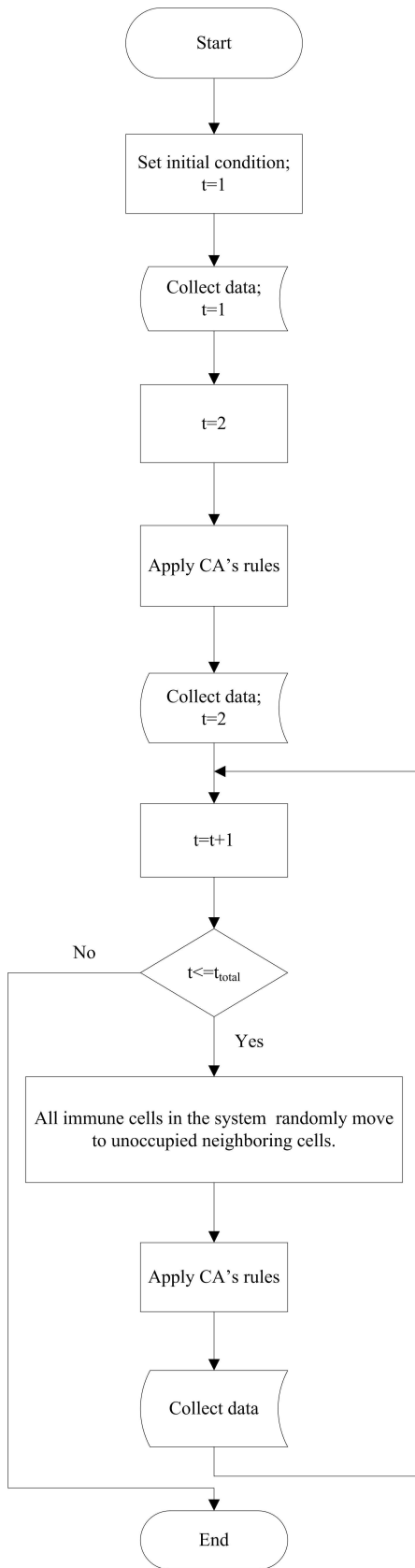
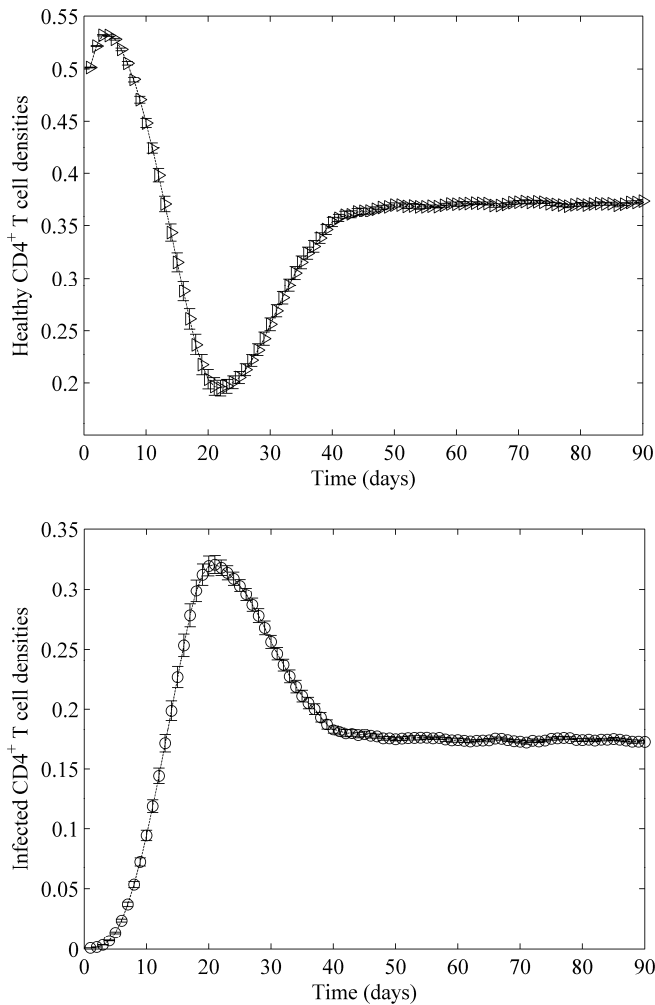


Figure 1 Flowchart of our CA algorithm.

Table 2 Used simulation parameters.

Parameters	Describes
$L = 300$	Lattice side. Therefore $L^2$ means lattice area.
$N = 8$	Neighboring cells. Moore's Neighborhood condition.
$P_{N_{T4}} = 0.5$	Probability of initial $N_{T4}$ cells.
$P_{N_{T8}} = 0.25$	Probability of initial $N_{T8}$ cells.
$P_{M_{DC}} = 0.005$	Probability of initial $M_{DC}$ cells that carry HIV to the system. It mimics the amount of initial virus entering into human body.
$P_{I_{T4}} = 0.1$	The severity of infected cells. It mimics the severity of HIV to the host cells. Therefore, $1 - (1 - P_{I_{T4}})^k$ [7] is the infectivity of an $I_{T4}$ cell and an $M_{DC}$ cell to the healthy $(N_{T4}, E_{T4})$ neighbors. $k$ is a number of $I_{T4}$ cell or $M_{DC}$ cell among the healthy.
$P_{repl\_M_{DC}} = \left( \frac{I_{T4}}{L^2 - E} \right)$	Probability that a $D$ cell is replenished by an $M_{DC}$ cell.
$P_{repl\_N_{T4}} = \frac{2}{3} \times (1 - (P_{repl\_M_{DC}}))$	Probability that a $D$ cell is replenished by an $N_{T4}$ cell.
$P_{repl\_N_{T8}} = \frac{1}{3} \times (1 - (P_{repl\_M_{DC}}))$	Probability that a $D$ cell is replenished by an $N_{T8}$ cell.
$\tau_{M_{DC}} = 180$	Time delay for an $M_{DC}$ cell.
$\tau_{N_{T4}} = 540$	Time delay for an $N_{T4}$ cell.
$\tau_{N_{T8}} = 540$	Time delay for an $N_{T8}$ cell.
$\tau_{E_{T4}} = 18$	Time delay for an $E_{T4}$ cell.
$\tau_{E_{T8}} = 6$	Time delay for an $E_{T8}$ cell.
$\tau_{I_{T4}} = 90$	Time delay for an $I_{T4}$ cell.

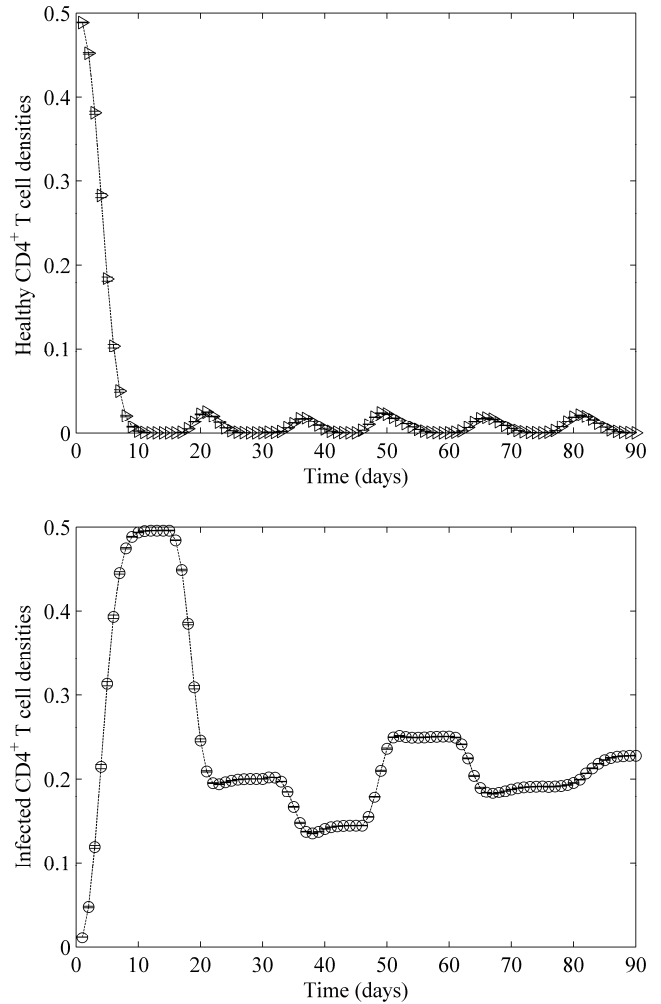
### 3 Computational results



**Figure 2** (a) and (b) are the plot of healthy  $CD4^+$  T cell densities and infected  $CD4^+$  T cell densities during an acute phase of HIV infection, respectively. The cell density is derived from the ratio of the number of each cell type to the total number of cells in the system.

Our simulation results of healthy  $CD4^+$  T cell ( $N_{T4} + E_{T4}$ ) densities and infected  $CD4^+$  T cell ( $I_{T4}$ ) densities, averaged over 20 runs, are shown in Figure 2 along with the simulation time. The results represent an acute phase of HIV infection (corresponding to 90 days in real life). From the simulation results, it is found that after the introduction of  $M_{DC}$  cells to the system, the level of infected cells rapidly increases and reaches the peak at day 21 (week 3), (see Figure 2b). In contrast, the level of healthy cells slightly increases, and then rapidly drops due to the infection (see Figure 2a). After that, the level of infected cells drops from the peak which reflects the initiation of HIV-specific immune response. Consequently, the level of healthy cells rebounds.

Then, the level of healthy cells and the level of infected cells gradually approach the steady state, which marks the end of the acute phase of HIV infection at approximately day 40 (~week 6).



**Figure 3** (a) and (b) are the plot of healthy  $CD4^+$  T cell densities and infected  $CD4^+$  T cell densities during an acute phase of HIV infection when  $P_{N_{T8}} = 0$  and  $P_{repl\_N_{T8}} = 0$ .

Figure 3 shows our simulation results when the probability of initial  $N_{T8}$  cells and the probability that a  $D$  cell is replenished by an  $N_{T8}$  cell equal to zeros, which reflects the phenomena when there is no HIV-specific immune response in the system. It is found that the level of healthy cells rapidly drops to nearly zero without any sign of rebounding. In contrast, the features of infected cell dynamics are similar to those of the original condition. Note that although there is no effect of HIV-specific immune response in this condition, the decline features of infected cell level still occur. This might be because there is a scanty level of healthy cells, the host cells to be infected, in the system.

## 4 Conclusions

We present a stochastic model for the dynamics of HIV infection based on cellular automaton method. It was found that our model could present the acute phase of immune cell dynamics after the infection. Moreover, it shows the effects of cell-mediated immunity, in particular the cells referred to as DC cells, CD8<sup>+</sup> T cells, and CTL cells which cause the rebounding of the healthy CD4<sup>+</sup> T cells in real life.

## 5 Acknowledgments

We wish to thank Parames Laosinchai for English grammar checking, and Angkana Boondirek, Paisan Kanthang and Charin Modchang for invaluable discussions.

## 6 References

- [1] G. Pantaleo, *et al.*, "Role of lymphoid organs in the pathogenesis of human immunodeficiency virus (HIV) infection"; *Immunological reviews*, vol. 140, Issue 1, pp. 105-130, Aug 1994.
- [2] S. Wolfram, "Statistical mechanics of cellular automata"; *Reviews of modern physics*, vol. 55, Issue 3, pp. 601-644, 1983.
- [3] R. Zorzenon dos Santos and S. Coutinho, "Dynamics of HIV infection: A cellular automata approach"; *Physical review letters*, vol. 87, Issue 16, p. 168102, Oct 2001.
- [4] V. Shi, *et al.*, "A viral load-based cellular automata approach to modeling HIV dynamics and drug treatment"; *Journal of theoretical biology*, vol. 253, Issue 1, pp. 24-35, Jul 2008.
- [5] M. Precharattana, *et al.*, "Stochastic cellular automata model and Monte Carlo simulations of CD4<sup>+</sup> T cell dynamics with a proposed alternative leukapheresis treatment for HIV/AIDS"; *Computers in biology and medicine*, vol. 41, Issue 7, pp. 546-558, Jul 2011.
- [6] M. Precharattana, *et al.*, "Investigation of Spatial Pattern Formation Involving CD4<sup>+</sup> T Cells in HIV/AIDS Dynamics by a Stochastic Cellular Automata Model"; *International Journal of Mathematics and Computers in Simulation*, vol. 4, Issue 4, 2010.
- [7] S. Zhang and J. Liu, "A massively multi-agent system for discovering HIV-immune interaction dynamics"; *Massively Multi-Agent Systems I*, pp. 572-573, 2005.



# Visual Simulation of a Multi-Species Coloured Lattice Gas Model

T.S. Lyes, M.G.B. Johnson and K.A. Hawick

Computer Science, Institute for Information and Mathematical Sciences,

Massey University, North Shore 102-904, Auckland, New Zealand

email: { t.s.lyes, m.johnson, k.a.hawick }@massey.ac.nz

Tel: +64 9 414 0800 Fax: +64 9 441 8181

March 2012

## ABSTRACT

Understanding complex fluid flow patterns is still a computationally challenging problem and visualising computer simulations in this area is an important tool in investigating emergent complexity in fluid systems. Mixing and unmixing of complex multi-species fluid systems is particularly difficult to tackle using conventional field equational methods. We describe our multi-species lattice gas software for simulating and visualising multi-species fluid systems. We describe how a coloured lattice gas model was developed and run on a graphical processing unit using NVIDIA's compute unified device architecture (CUDA) to yield a speed up over a typical CPU performance of one hundred fold. This then supports simulation of system sizes large enough to reveal interesting emergent complexity, and we present some initial scientific observations.

## KEY WORDS

lattice gas; coloured; FHP-3; CFHP; OpenGL; cellular automata; simulation;

## 1 Introduction

Simulating the dynamics of fluid flow can often be a very complicated task. Lattice gas cellular automata (LGCA's) [1–3] can be used to approximate the behavior of fluid flows with a certain degree of accuracy - they may fail to model all physical dynamics of fluids at a microscopic level, however they can produce satisfactory results on a macroscopic level. Many variations of LGCA approaches, including using a lattice Boltzmann approach [4–6], have already been developed and improved upon to produce faster, more complex and more accurate results. More complex geometric arrangements such as reflecting boundaries, barriers, monolayer deposition [7] and the imposition of gravitational bias [8] are also possible.

Simulation work with large scale lattice gas models

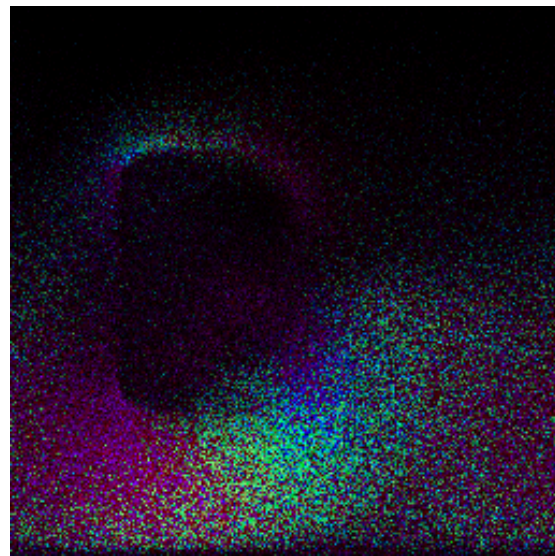


Figure 1: A coloured lattice gas visualization

[9, 10] has important implications for studies of complexity and emergence on logarithmic length scales. Work is reported in the literature on the stability limits of lattice gas models [11] and lattice gas models are also of use in modeling crowd dynamics [12]. Our particular interest is in modelling multi-species systems lattice gas systems [13–15] where a fluid or some other modelled microscopic constituents either separate out from a random initial state or mix and co-dissolve under the right parametric circumstances.

In a previous study [16], a LGCA was used to simulate a lattice gas on a graphics processing unit (GPU), as LGCA models are highly parallelisable [17–21]. This allowed for better levels of optimization and execution speed of the LGCA, as well enabling the model to be simulated at a larger scale. A method of visualizing the lattice gas behavior on the fly, however, was not developed, and

thus has become one of the goals of this paper.

Most standard LGCA models apply to a lattice containing only one type of gas (commonly referred to as a 'species'). There do exist, however, some models which take into account the option of having multiple species present in the same lattice. Such models are good for simulating diffusion effects between multiple gases, and can produce some interesting results depending on the properties of each species of gas. Models which deal with multiple species of gases are often called "coloured" models, as different particles can be represented visually using different colours. While accounting for a second species of particle will affect the overall performance of the model, there are ways to reduce this impact.

In this article we present a coloured LGCA based on the code from [16], which may be visually monitored on-the-fly while also being able to be parallelized easily for performance investigation. Firstly, a more in-depth explanation of the LGCA models used will be given in Section 2 as well as a description of the OpenGL [22] and Compute Unified Device Architecture (CUDA) [23] technologies used in Section 3. In Section 4 some screenshots of the resulting lattice gases will be shown, along with descriptions and additional information. In Section 5 some of the performance results are put forth, and some coding issues and difficulties will also be discussed. Finally, conclusions and future work will be offered in Section 6.

## 2 The LGCA Models

While many models exist for using LGCA to simulate fluid flow, the FHP-3 model [1] (named after Frisch, Hasslacher and Poumeau, who first introduced the model) allows for superior efficiency in simulation. The model's defining characteristics are the triangular lattice structure, allowing for 6 channel directions (right, right-up, right-down, left, left-up and left-down), as well as a stationary particle channel, for each site in the lattice. The state of each site in the lattice at any given time will be determined by the state of the site particle itself, as well as the velocity vectors propagating from each of the site's 6 surrounding neighbours, based on a certain set of defined collision rules (see Figure 2 for an example). A single site particle may have up to eight inputs - six neighbor velocity channels, one stationary channel, and one barrier channel. Therefore, each site can be neatly stored as one byte - one bit representing each channel. To further aid in computational efficiency, four 8-bit sites can be bit-packed into one 32-bit integer, rather than using one integer for each site.

The CFHP model [1], or known simply as a "coloured" lattice gas, is a variation of the regular FHP-3 LGCA, in which particles may belong to two different species

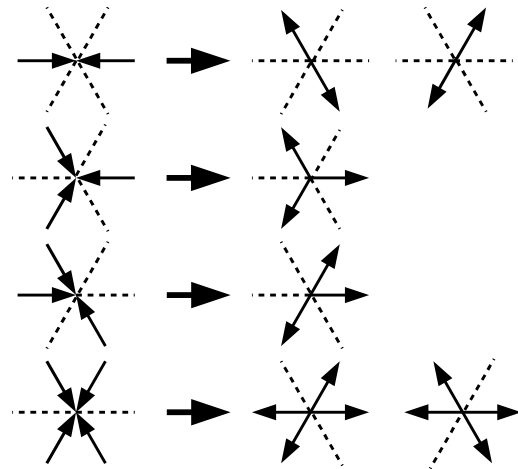


Figure 2: An example of some FHP-3 model collision rules

(colours). In a standard LGCA, every particle is designed to be mechanically identical. This remains true in a coloured LGCA - all particles remain identical in all ways except colour. This means the CFHP model need not create a larger, more complex set of collision rules to compensate for two different particles in the lattice. Indeed, the CFHP model uses exactly the same set of rules as the FHP-3 model. How, then, does one keep track of the colour of each particle in the lattice? The number of coloured particles need only be conserved in each collision. Before any collision rules are applied to each site, the number  $N$  of each coloured particle can be counted. Once the collision rules have been applied using the standard FHP-3 collision ruleset, the colour attribute can be randomly assigned to  $N$  number of channels using a lookup table. This need only be done with one of the two colours, as the second colour will automatically be assigned to those channels not randomly assigned the first colour attribute. Put simply, if the particle is not red, it is blue.

In working with the code for the coloured lattice gas, it was decided that we would develop an OpenGL rendering system that would allow us to visualize the lattice gas in real time, with the ability to monitor the different colours separate from each other. The resulting program allowed for visualization of the lattice in five different rendering methods - original single-colour lattice, red lattice only, blue lattice only, combined red and blue lattice, and a combined red and blue lattice using arrows to visualize the velocity vectors. The program was intended to investigate how a particles of two different species might behave, given they follow the same rules. Additionally, it was decided that an interesting add-on to the project would be to introduce some directional bias into the species (for ex-

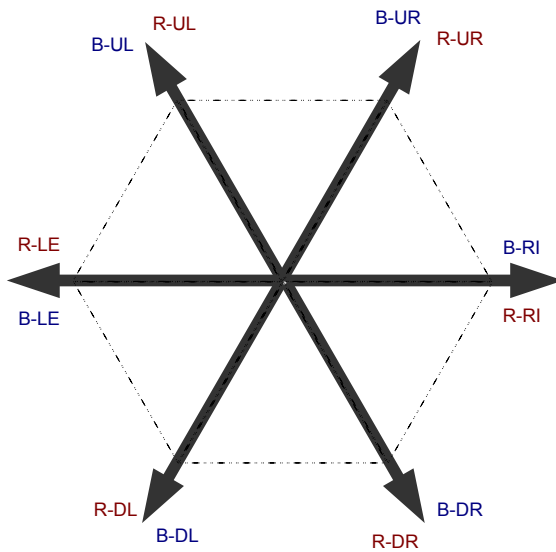


Figure 3: An CFHP model lattice setup - notice each direction has two channels, one for red particles and one for blue particles

ample, one species might be inclined to move downward as it could be heavier). Without changing any of the rule-sets, the bias could simply be applied during the lookup table generation for the randomization of the colour assignment.

### 3 OpenGL Lattice Rendering and CUDA

The code to simulate the coloured lattice gas was written both as a sequential OpenGL program for visualization purposes, and in CUDA for performance monitoring. The coloured lattice would be expected to run slower than the single-species lattice, as there is extra work in counting coloured channels and assigning random channels coloured attributes.

As opposed to the single-species lattice, the coloured lattice used three different arrays to store the lattice data - a "master" array on which the collision rule set was applied, and two arrays containing the information needed for the red and the blue lattices. This allowed for easy access by OpenGL when rendering only the red or only the blue lattice, while additionally removing the need to add extra rules to the rule set to be applied to the master lattice in order to account for the different coloured particles.

The lattice itself was displayed in a 256x256 OpenGL window, primarily using OpenGL points to represent the particles in the lattice. To visualize the movement of the particles, each OpenGL point was coloured according to

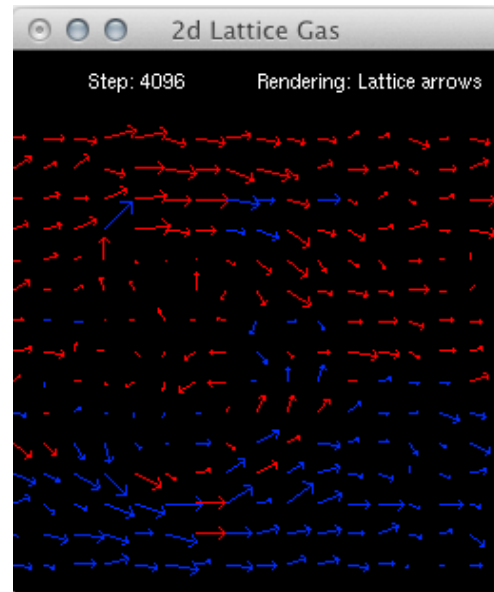


Figure 4: Sample screen dump of the lattice rendered using arrows

the represented point's velocity vector. This velocity vector was computed as an average vector over a block of 4x4 neighbouring lattice points. These coloured points were not always easy to interpret, so it was decided to allow the lattice to be rendered using arrows as well. The arrows would point in the direction of the velocity vector, while the magnitude of the vector determined the length of the arrow. To avoid clutter in the OpenGL window, the amount of arrows in the display was reduced from 256x256 to 16x16 with the vectors averaged over a larger area, while keeping the original lattice precision the same. Rather than colour the arrows the same as their OpenGL point counterparts, they were coloured a flat red or blue, depending on which species the particle belonged to at that particular point on the lattice. Figure 4 shows an example of what the lattice would look like rendered as arrows.

The bulk of the code for the lattice gas was contained in the Kernel function, which was used to perform both the collision and propagation phases of the lattice gas. It was used in OpenGL as the update function after every display call. This function was converted nicely into CUDA code, as each cell in the lattice operated independently for each time step. Cells in the lattice were also bit-packed to improve performance. The CUDA version of the code was used solely for performance checking, as any rendering would have slowed it down. The kernel itself followed an algorithm roughly resembling the following algorithm (Algorithm 1).

The coloured lattice was simulated twice, once using

**Algorithm 1** a typical coloured lattice gas kernel

---

```

for all cells in lattice do
  count red particles
  determine input channels
  perform bit-packing (unpack)
  apply collision rule
  randomize colour channels
  conserve red particles
  apply colour channels
  perform bit-packing (pack)
end for
update main lat, red lat, blue lat

```

---

the standard rule set, and once using a biased rule set. Similar to [16], a barrier was added to allow for observation of the behavior around impassible cells. The barrier was set up as a vertical line of cells centered horizontally in the lattice, roughly one fifth of the width from the left hand side. To allow for the most efficient demonstration of the gas interaction with the barrier, initial particle velocities were set to be perpendicular to the barrier, towards the right. Additionally, the lattice particles would wrap around right-to-left.

## 4 Visualisation Results

The particles in each screenshot are coloured in relation to their velocity. Darker areas of the lattice indicate an absence or lower concentration of particles in that particular area - this usually occurs around barriers. For each of the red and blue lattices a separate colour spectrum was used to represent the velocities of the particles. While visualisation of each coloured lattice on its own presented no problems, rendering both red and blue lattices together presented some visualisation problems as it is difficult in some places to distinguish the red lattice's green particles from the blue lattice's teal particles (and several other similar colour match ups as well). For reference and comparison, a visualization of the single-species lattice gas was also shown.

Each lattice configuration was simulated for 4096 time steps, while being observed and captured at the start, middle and end points of the simulation. The resulting screenshots were produced.

Both the standard and biased versions of the lattice were initialized exactly the same. Figure 5 shows this initial configuration. All particles above half the height of the lattice were initialized red, and all below were initialized as blue. In the early stages of the project, the red and blue particles were assigned random initial positions in the lattice - this configuration, however, produced no meaningful results, as interactions between red and blue particles were almost imperceptible, and diffusion behav-

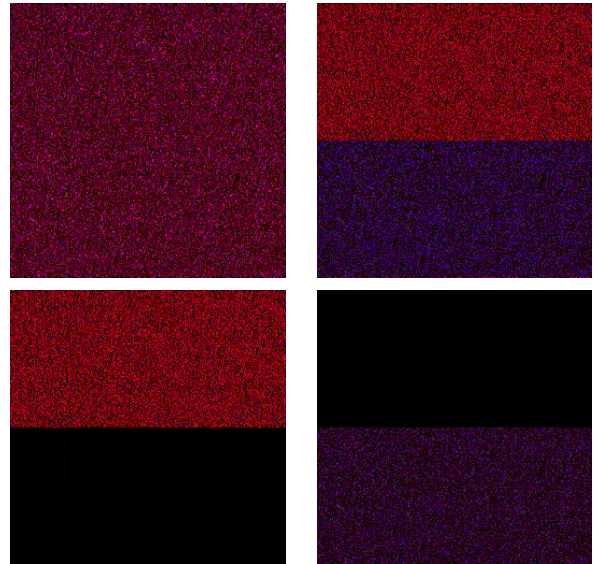


Figure 5: Initial configuration of the lattice. From top-left to bottom-right: the standard single colour lattice, both coloured lattices, red only and blue only

ior could not be observed (as both species were already fully diffused). The barrier cannot be seen yet in this example, as initial particle velocities have yet to take effect.

Figure 6 shows the state of the lattice at the half-way point of the simulation. This version uses the standard rule set. In these figures the barrier to the left has become clearly visible and the particles have begun to move around it. It is interesting to watch the behavior of the red and the blue lattices on their own. Although the majority of each species' fellow particles remain concentrated at their respective sides, some particles are still forced in the opposite direction by the barrier and other incoming particles from the left. This can be shown by a "hook" like shape created as the particles are pushed down and around the edge of the barrier (this is more clear in the red lattice as the colour is lighter). This type of behavior occurs because the particles have no cohesion between others of the same species, and instead operate independently of each other. Both lattices display some slight diffusion as some blue particles begin to slowly move upward, and in turn some red particles slowly move downward.

Figure 7 shows the final product of the lattice gas simulation after 4096 steps. The flow of the gas has now become much more apparent due to the colour change of the particles and the shape of the gas flow behind (to the right) of the barrier. The behavior of the red and blue lattices is also more apparent, as they form stronger curves around the barrier and the diffusion between the two species has increased.

The biased rule set was set up to increase the chance



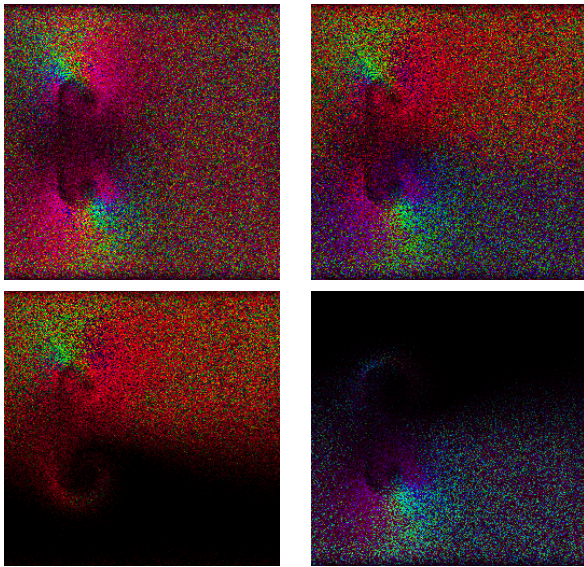


Figure 6: State of the lattice at time step 2048 (half - way point of the simulation) using standard rules. From top-left to bottom-right: the standard single colour lattice, both coloured lattices, red only and blue only

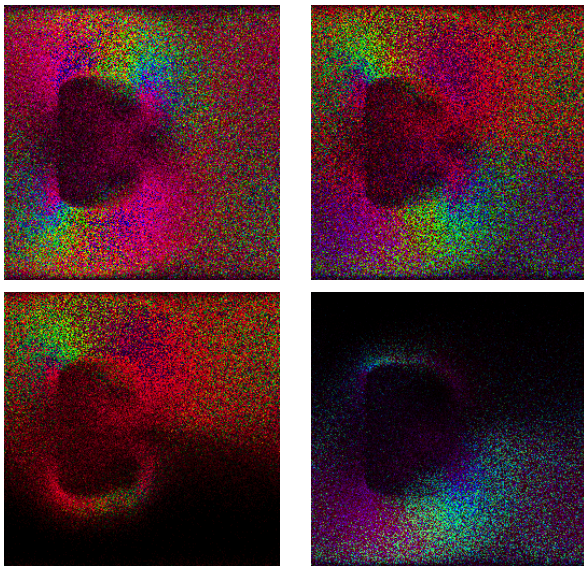


Figure 7: Final state of the lattice (after 4096 time steps) using standard rules. From top-left to bottom-right: the standard single colour lattice, both coloured lattices, red only and blue only

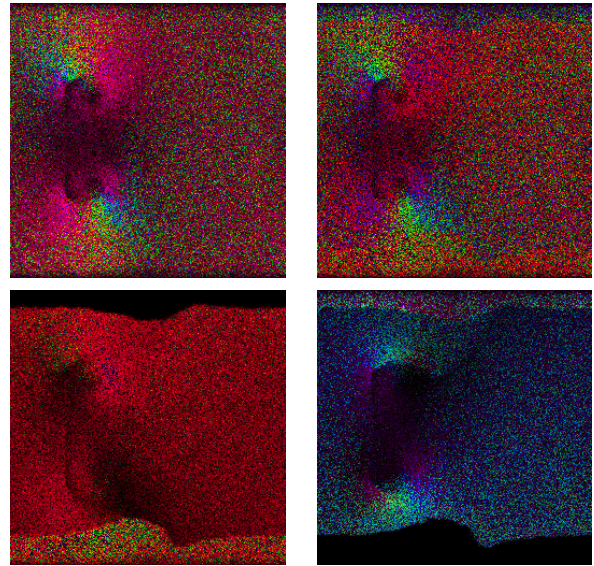


Figure 8: Half-way state (time step 2048) of the lattice using a biased rule set. From top-left to bottom-right: the standard single colour lattice, both coloured lattices, red only and blue only

that a red particle would move in a downward direction rather than any other. In the lattice configuration, this meant that both the left-down and right-down channels would have an increased chance to be assigned the colour red, although the chances of assigning either left-down or right-down still remained equal. This bias was chosen over the other directional channels because it would give more obvious results - a bias towards the right or left might simply appear to increase or decrease particle velocity in that direction. The downward bias was applied only to the red particle species.

Figure 8 shows the half-way state of the biased lattice. Notice how the shape of the single-species lattice is identical to that in Figure 6, however the shapes of the red and blue lattices are radically different. Red particles, while initially starting at the top (see Figure 5) have moved towards the bottom of the lattice, being completely replaced by blue particles at the extreme top of the lattice. Consequently, blue particles appear to have been "pushed" upwards by the red - a large area at the bottom of the lattice now contains solely red particles. This area has a curious "hump" shape, as while the red particles are biased downwards, they are still effected by the right-ward velocity and barrier and are thus collecting mostly at the bottom just to the right of the barrier. Both these behaviors form what seems to look like "layers" when both lattices are rendered together.

Finally, Figure 9 shows the final result of the biased coloured lattice gas simulation. The red particles have

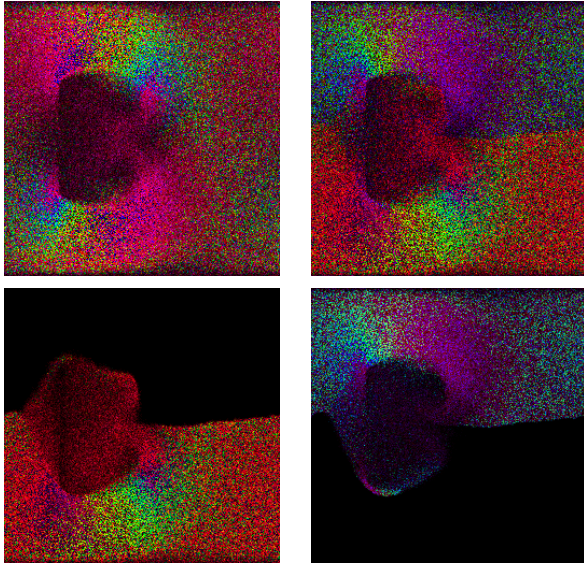


Figure 9: Final state of the lattice (after 4096 time steps) using a biased rule set. From top-left to bottom-right: the standard single colour lattice, both coloured lattices, red only and blue only

almost completely migrated to the bottom of the lattice, while the blue have almost completely moved to the top - essentially the positions of the two species have been inverted. The effect of the velocity and barrier is still apparent, however the particles seem to be less inclined to separate - for example, the curve of the red particles moving up over the barrier is much sharper than the lattice using non-biased rules (Figure 7) and once the particles move over the edge of the barrier, they curve sharply back down towards the bottom.

## 5 Discussion

The program running the coloured lattice when noticeably slower than the single-species lattice when they were both run sequentially using OpenGL to display the lattice after every time step. Due to the fact that the OpenGL rendering only used simple GL points with no 3d graphics or costly effects, the slowdown was probably due to the extra work being performed by the kernel during the update phase (as opposed to the display phase). To mitigate the overhead of copying to, and displaying the lattice on the CPU we could implement the OpenGL rendering directly on the GPU.

Any attempt to increase the lattice size would make the simulation run too slow to be viewed in real-time speed - while running nicely at a lattice size of 256x256, an increase in lattice size to 512x512 would display at around two frames (time steps) per second which is watchable but

slow, while increasing the size to 1024x1024 would display one frame every five seconds. It is clear that increasing the size of the lattice needs parallelization to make the simulation run smoothly in real time.

The following table shows the results of the performance testing using the CUDA version of the coloured lattice gas. The graphics card used for the performance test was the NVidia Quadro 4000. It was tested using different sizes of the lattice (256x256 up to 4096x4096), recording the average time taken in seconds to execute each kernel once. As expected, the coloured lattice took longer than the standard lattice in all lattice sizes, although it is almost unnoticeable in the smaller sizes (256 and 512). As the lattice gets larger, the time gap between the two lattices becomes exponentially larger until the coloured lattice is almost half a second slower than the single species lattice for each kernel execution.

Lattice size	Standard Model seconds per frame	Coloured Model seconds per frame
256	0.0084	0.0101
512	0.0131	0.0189
1024	0.0315	0.0537
2048	0.1044	0.1963
3072	0.2181	0.4290
4096	0.3996	0.7895

Table 1: Performance results of standard lattice gas vs. coloured lattice gas models, parallelized using CUDA with global memory. Times accurate to approximately  $\pm 0.00005$ .

From these results we can also see the power and speed-up gained by using CUDA and the graphics card - as mentioned earlier, if we assume negligible time cost from rendering, a single kernel execution on a 1024x1024 lattice run sequentially would take approximately five seconds. Compared to the 0.0537 seconds for a 1024x1024 lattice on the GPU, the parallelized lattice is almost 100 times faster. The coloured model adds only a manageable overhead cost.

## 6 Conclusions

We have developed a two-dimensional coloured lattice gas model, using OpenGL to render the model and observe it's behaviour in real-time. The model is able to easily be parallelized in CUDA for improved performance and scaling. For additional performance improvement, four cells in the lattice have been bit-packed into one 32-bit integer. The program allows up to two species of particle to exist in the lattice and each species may be viewed



separately as needed. The behaviours of different species can be biased as required by generating a separate colour randomization lookup table with an increased chance to assign the colour to the desired direction channel or channels.

This encoding approach to the multi species lattice gas makes good use of memory and hence aids cache-locality. Preliminary observations suggest the model is large enough to support measurements of thermodynamic mixing properties and experimental comparisons with realistic complex fluids.

We plan on extending this work into creating a three-dimensional lattice gas model and visualizing it in the same way, as well as models with varied densities. This would be expected to work well on a GPU architecture as well, although it is unsure whether the memory constraints which may arise in a three dimensional model will be manageable.

## References

- [1] Rivet, J.P., Boon, J.P.: Lattice Gas Hydrodynamics. Number ISBN 0-521-019710. Cambridge (2001)
- [2] Advisory Group for Aerospace Research and Development: Special Course on Modern Theoretical and Experimental approaches to Turbulent Flow Structure and its Modelling, 7 Rue Ancelle, 92200 Neuilly Sur Seine, France, Advisory Group for Aerospace Research and Development, NATO (1987) AGARD Report No 755.
- [3] Wylie, B.J.: Application of Two-Dimensional Cellular Automaton Lattice-Gas Models to the Simulation of Hydrodynamics. PhD thesis, Physics Department, Edinburgh University (1990)
- [4] R.Benzi, S.Succi, M.Vergassola: The lattice boltzmann equation: theory and applications. Rome Preprint ROM 2F-92-10 (1992)
- [5] D'Ortona, U., Salin, D., Cieplak, M., Rybka, R.B., Banavar, J.R.: Two-color nonlinear boltzmann cellular automata: Surface tension and wetting. *Phys. Rev. E* **51** (1995) 3718–3728
- [6] Gonnella, G., Lamura, A., Sofonea, V.: Lattice boltzmann simulation of thermal nonideal fluids. *Phys. Rev. E* **76** (2007) 036703
- [7] Hocker, T., Aranovich, G.L., Donohue, M.D.: Monolayer adsorption for the subcritical lattice gas and partially miscible binary mixtures. *Journal of Colloid and Interface Science* **211** (1999) 61–80
- [8] Hawick, K.: Visualising multi-phase lattice gas fluid layering simulations. In: Proc. International Conference on Modeling, Simulation and Visualization Methods (MSV'11), Las Vegas, USA (2011)
- [9] Wolf-Gladrow, D.A.: Lattice-Gas Cellular Automata and Lattice Boltzmann Models - An Introduction. Springer (2005)
- [10] Hernandez-Machado, A., Sancho, J.M.: From lattice-gas models to nonlinear diffusion models: A derivation of macroscopic equations and fluctuations. *Phys. Rev. A* **42** (1990) 6234–6237
- [11] Bernardin, D., Sero-Guillaume, O.E.: Exact stability results in stochastic lattice gas cellular automata. *J. Stat. Phys.* **81** (1995) 409–443
- [12] Moore, B., Ali, S., Mehran, R., Shah, M.: Visual crowd surveillance through a hydrodynamics lens. *Communications of the ACM* **54** (2011) 64–73
- [13] Blaak, R., Dubbeldam, D.: Coupling of thermal and mass diffusion in regular binary thermal lattice gases. *Phys. Rev. E* **64** (2001) 062102–1–4
- [14] Blaak, R., Dubbeldam, D.: Regular binary thermal lattice-gases. *J. Stat. Phys.* **108** (2002) 283–315
- [15] Nagl, C., Schuster, R., Renisch, S., Ertl, G.: Regular mixing in a two-dimensional lattice system: The coadsorption of n and o on ru(0001). *Phys. Rev. Lett.* **81** (1998) 3483–3486
- [16] Johnson, M., Playne, D., Hawick, K.: Data-parallelism and gpus for lattice gas fluid simulations. In: Proc. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'10). Number CSTN-109, Las Vegas, USA (2010) PDP4521.
- [17] Fan, Z., Qiu, F., Kaufman, A., Yoakum-Stover, S.: Gpu cluster for high performance computing. In: SC '04 Proceedings of the 2004 ACM/IEEE conference on Supercomputing. Number ISBN 0-7695-2153-3 (2004)
- [18] Kaufman, A., Fan, Z., Petkov, K.: Implementing the lattice boltzmann model on commodity graphics hardware. *J. Stat. Mech: Theory and Experiment* **P06016** (2009) 1–27
- [19] Zhou, J., Zhong, C., Xie, J., Yin, S.: Multiple-gpus algorithm for lattice boltzmann method. In: Proc. Int. Symp. on Information Science and Engineering (ISISE'08), Shanghai (2008) 793–796
- [20] Obrecht, C., Kuznik, F., Tourancheau, B., Roux, J.J.: Multi-gpu implementation of the lattice boltzmann method. *Computers and Mathematics with Applications* **In Press** (2011) 1–10
- [21] Bernaschi, M., Fatica, M., Melchionna, S., Succi, S., Kaxiras, E.: A flexible high-performance lattice boltzmann gpu code for the simulations of fluid flows in complex geometries. *Concurrency and Computations: Practice and Experience* **22** (2010) 1–14
- [22] Wright, R.S., Haemel, N., Sellers, G., Lipchak, B.: OpenGL Superbible. Fifth edn. Number ISBN 978-0-321-71261-5. Pearson (2011)
- [23] NVIDIA® Corporation: CUDA™ 3.1 Programming Guide. (2010) Last accessed August 2010.





**SESSION**  
**COMPUTATIONAL SIMULATION AND**  
**MODELING**

**Chair(s)**

**TBA**



## Uncertainty Analysis of Spillway Erosion Parameters

**Mitchell L. Neilsen**

Dept. of Computing and Info. Sciences  
Kansas State University  
234 Nichols Hall  
Manhattan, KS, USA

**Gregory J. Hanson, Darrel M. Temple (ret.)**

Hydraulic Engineering Research Unit  
USDA Agricultural Research Service  
1301 N. Western  
Stillwater, OK, USA

### Abstract

*Windows™ Dam Analysis Modules (WinDAM) is a set of modular software components that can be used to analyze overtopped earthen embankments and internal erosion of dams. WinDAM is being developed in stages. The initial computational modules address routing of floods through the reservoir with dam overtopping and evaluation of the potential for vegetation or riprap to delay or prevent failure of the embankment. The next module, WinDAM B, incorporates dam breach analysis; i.e., analysis of the breach failure of a homogeneous embankment through overtopping and drainage of stored water. Current work is underway to include analysis of internal erosion, non-homogeneous, zoned embankments, and analysis of various other forms of embankment protection. The focus of this paper is on the overall software architecture, interfaces between legacy software and the existing modules, and the integration with Sandia National Laboratories' DAKOTA software suite to conduct parameter studies and uncertainty analysis on a range of input parameters.*

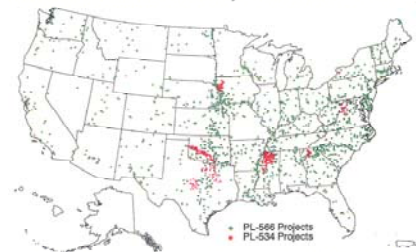
**Keywords:** Erosion, hydraulic modeling, sensitivity analysis, simulation, uncertainty quantification.

### 1. Introduction

Windows™ Dam Analysis Modules (WinDAM) is a set of modular software components that can be used to analyze overtopped earthen embankments and internal dam erosion. The development of WinDAM is staged. The initial computational model addresses routing of the flood through the reservoir with dam overtopping and evaluation of the potential for vegetation or riprap to delay or prevent failure of the embankment. The first model to be widely deployed, WinDAM A+, also incorporates the auxiliary spillway erosion technology used in SITES. However, unlike SITES, it allows a user to analyze up to three auxiliary spillways and embankment erosion on the dam. The next computational model, WinDAM B, incorporates dam breach analysis; i.e., the breach failure of a homogeneous embankment through overtopping and drainage of stored water in the reservoir. In addition, work is currently underway to include analysis of internal erosion, analysis of non-homogeneous embankments, and analysis of other forms of embankment protection.

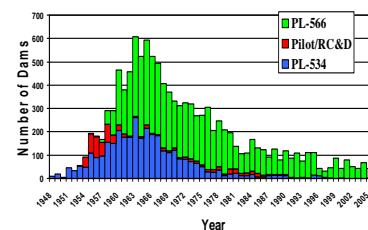
WinDAM is designed to address the dam safety concerns facing the national legacy infrastructure of over 11,000 small watershed dams constructed with US Federal involvement over a seventy-year period. The US Department of Agriculture -Agricultural Research Service (USDA-ARS), US Department of Agriculture-Natural Resources Conservation Service (USDA-NRCS), and Kansas State University (KSU) are working jointly to develop and refine this software.

Public Law 78-534 – Flood Control Act of 1944 started the small watershed program, and it was followed by Public Law 83-566 – Watershed Protection and Flood Prevention Act of 1954. The map of watersheds (Fig. 1) illustrates the impact of Public Laws 566 and 534.



**Figure 1.** Map of small watershed projects.

Starting in 1958, an average of one dam per day was constructed over a period of twenty years. The number of dams constructed per year is shown below in Fig. 2 [15].



**Figure 2.** Number of dams constructed per year.

Most flood routing of dams before the middle 1960's was computed manually. Then, routing software on mainframe computers began to replace manual methods. Design procedure from 1973 to the mid 1990's utilized the bulk length concept to determine the auxiliary spillway crest length and geometry. In 1983, the USDA-SCS-ARS Emergency Spillway Flow Study Task Group (ESFSTG) was formed to develop better technology for earth

spillway analysis. The ESFSTG collected data on dams that experienced either emergency spillway flow at least three feet deep or significant damage during a storm event. Approximately one hundred sites were selected for more in-depth evaluation and data collection, and data analysis began in 1990 from the field spillway data initially collected. Tests were conducted in the USDA-ARS outdoor Hydraulic Engineering Research Unit (HERU) Laboratory near Stillwater, Oklahoma, during this time to further understand spillway performance processes such as flow concentration, vegetal cover failure, surface detachment, and headcut migration. These findings were incorporated into the DAMS2 software, and then into Stability and Integrity Technology for Earth Spillways (SITES) software in 1994. The bulk length concept was replaced by SITES spillway erosion modeling technology in other USDA-NRCS references [15]. Although SITES may be used for analyses of existing dams and spillways, it was developed primarily for design and was developed over a period in which computational capability was much more limited than today. The legacy infrastructure of aging structures means a transition from design of new structures to analysis of existing structures. For example, existing structures may overtop as a result of watershed changes or sediment deposition within the flood pool leading to inadequate spillway capacity. WinDAM builds on and extends the existing technology in SITES to provide the needed capability for these types of analyses.

Windows™ Dam Analysis Modules (WinDAM) is a collection of modular software components that can be used to design and analyze the performance of earthen dams. The focus of the initial collection of computational modules is to evaluate earth dams subjected to flooding that may result in overtopping of the dam embankment and auxiliary spillway(s) [1]. The reservoir routing model incorporated into the software includes outflow from a principal spillway, up to three auxiliary spillways, and over the top of the dam embankment. For conditions where overtopping of the embankment is predicted, the hydraulic attack on the downstream face can also be evaluated using the initial software modules in WinDAM A+. The downstream face of a dam is typically protected using vegetation or riprap. WinDAM A+ has been extended to include erosion and breach computations for conditions where the hydraulic attack exceeds that which can be withstood by the vegetal or riprap lining, the resulting modules are in WinDAM B. The next version, WinDAM C, will incorporate analysis of failures caused by internal erosion or piping failures. To evaluate erosion in each auxiliary spillway, the SITES Spillway Erosion Analysis module with Latin Hypercube Sampling (SSEA+LHS) is integrated with WinDAM B. The Embankment Erosion Module is extended to include a Breach Analysis Module. The current model assumes the dam has a homogeneous embankment. It is most applicable to analyze or design embankments constructed from cohesive soil materials. It is anticipated that the model will be expanded to handle zoned embankments in

WinDAM D. The breach technology enabling this expansion is currently under development. Inputs to WinDAM include a description of the reservoir inflow hydrograph, reservoir storage capacity, all spillway properties, the dam cross section and profile, properties of the embankment, and input parameters for the breach analysis module. Inflow hydrographs can also be obtained automatically from other reach routing software, such as SITES 2005.1.4 [2,3], SSEA+LHS [4], HEC-HMS [5], HEC-RAS, or WinTR-20 [6] as shown below in Figure 3.

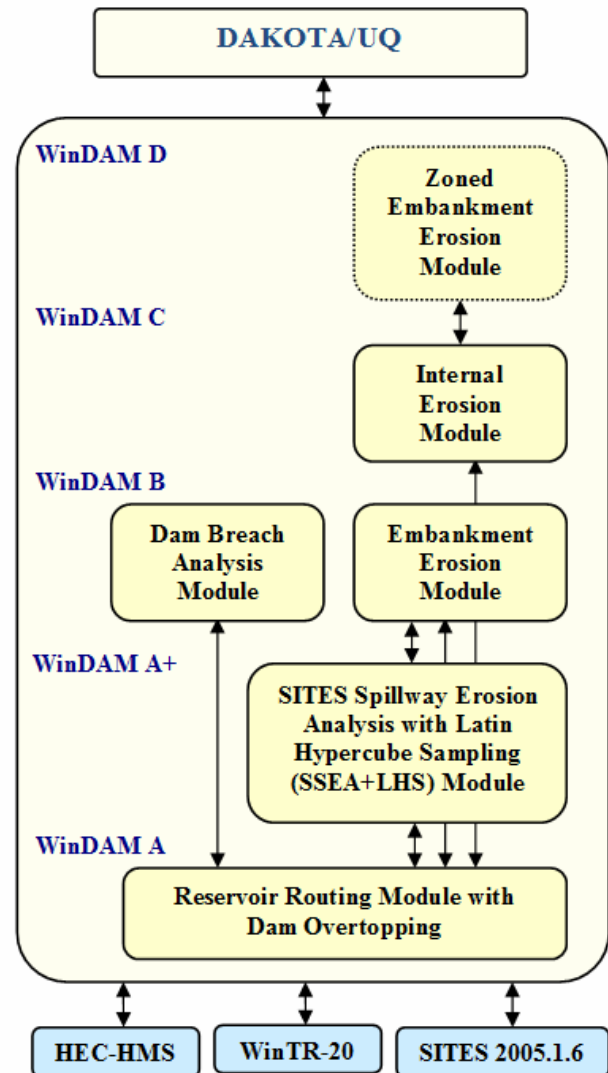


Figure 3. WinDAM software architecture

Outputs include a description of the reservoir water surface variation with time, the hydrographs associated with outflow through each of the spillways and over the top of the embankment, and a description of the attack on the dam embankment and downstream embankment face. Output hydrographs can be directed to external reach routing software. Output information is generated in both text and graphical format. The software generates ASCII text and/or XML control files for the model simulator which performs the model calculations. Output from the simulator is written to intermediate XML and/or fixed-

format ASCII text files that can be read by a Graphical User Interface (GUI) to display results in both text and graphical format. Due to the well-defined interfaces that automatically convert data to and from different forms, it is easy for software developers to interface the system with existing analysis software and with software under development.

In the DAKOTA system, the strategy creates and manages iterators and models. A model contains a set of variables, an interface, and a set of responses, and an iterator operates on the model to map the variables into responses using the interface [17]. The WinDAM system is used to automatically create a DAKOTA input file in which the user indirectly specifies these components through strategy, method, model, variables, interface, and responses keywords. Then, DAKOTA is invoked to iterate on the WinDAM simulation models as needed to generate outputs. Finally, the user returns to WinDAM.

In what follows, Section 2 covers the interfaces between existing reach routing software and WinDAM. Section 3 briefly covers the integration of WinDAM with SSEA+LHS and the Dam Breach Module to analyze auxiliary spillway erosion and dam breach, respectively. The resulting WinDAM model is called WinDAM B – WinDAM with breach analysis. Finally, Section 4 covers integration of WinDAM with DAKOTA/UQ to perform a simple parameter study.

## 2. Interfacing with Existing Reach Routing Software

Flow is routed through the reservoir by balancing inflow, outflow, and storage under the assumptions of a level reservoir surface with all outflow being a function of reservoir water surface elevation. Stage-storage properties of the reservoir are entered in tabular format with elevation in feet and the corresponding surface area in acres or storage volume in acre-feet. Reservoir inflow hydrographs are entered into WinDAM as series of time-discharge pairs with time in hours and flow in cubic feet per second (cfs).

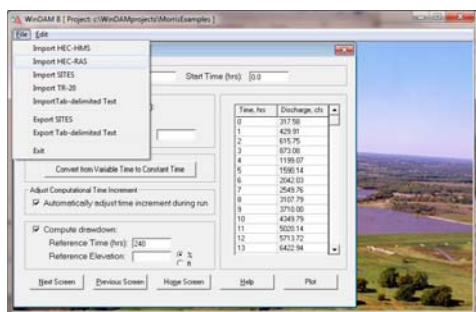


Figure 4. Hydrograph data import

Inflow hydrographs are normally computed using other software that is capable of generating a rainfall-runoff hydrograph [2-6]. The time increment used for entry of the hydrograph is normally used in performing the routing and erosive attack computations. As shown in Figure 4, an option is provided to import hydrographs from other sources and an option is provided to automatically adjust

the time increment until the change in computed maximum gross stress is less than one percent, or when the maximum effective stress is less than one percent, or when the reservoir water surface elevation changes by less than one percent, evaluated in that order when the time interval is halved (effectively doubling the number of hydrograph data points used on each run). For this example, the data is imported from a HEC-RAS public sample dataset for Beaver Creek as shown below in Figure 5, and after import as shown in Figure 4.

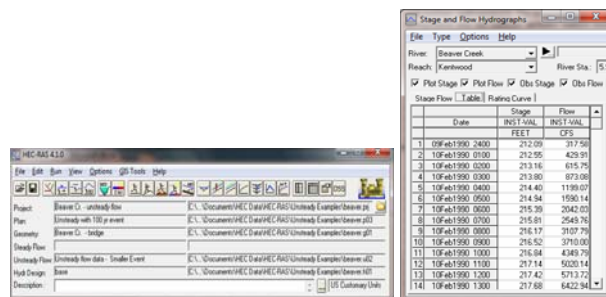


Figure 5. HEC-RAS dataset to import

Note that the date/time format is automatically converted to hours with a default start time of 0.0 hours. The user can specify a different start time as shown in Figure 4.

Since the simulator currently evaluates simulator runs in sequential order, several opportunities exist to enhance overall performance by parallelizing the application or by running the application on a cluster or grid. If no breach analysis is to be performed, then the user can also input 10-day drawdown parameters as shown in Figure 4.

The computational model incorporated into the WinDAM software assumes stepwise steady-state flow and a level water surface in the reservoir. The mass balance equation governing flow through the reservoir for any given time step may be obtained by averaging conditions over the time step. The inflow to the reservoir is a known function of time only, and is obtained through application of appropriate hydrologic models such as SITES 2005.1.5 [2], HEC-HMS [5], or WinTR-20 [6]. The outflow from the reservoir is the sum of the outflow from all spillways and the outflow over the top of the dam. Using the assumptions of a level water surface in the reservoir and stepwise steady flow, each of the individual outflows may be treated as a unique function of the reservoir water surface elevation. Likewise, the storage volume in the reservoir becomes a unique function of the reservoir water surface elevation.

## 3. Spillway Erosion and Breach Analysis

The primary purpose of WinDAM B is threefold:

- Hydraulically route one input hydrograph through, around, and over a single earthen dam.
- Estimate auxiliary spillway erosion in up to three earthen or vegetated auxiliary spillways.
- Estimate erosion of the earthen embankment caused by overtopping of the dam embankment.



Since WinDAM B does not include any specific hydrology component, the user must create the input hydrograph using other software. This allows the user the flexibility to choose the hydrologic software most suitable for analysis of site conditions; e.g., HEC-HMS, etc.

WinDAM B assumes the embankment of the dam is a homogenous earthen material. Many USDA-NRCS dams are homogenous earthen fill, so the WinDAM B model applies. Future versions of WinDAM will address zoned embankments where each zone exhibits different erosion resistance from other zones.

Most existing USDA-NRCS dams are built with a single earthen auxiliary spillway. In rehabilitation of old USDA-NRCS-designed dams, it is more common to also utilize additional auxiliary spillways. As a result, WinDAM B allows the user to input up to three auxiliary spillways, each spillway with a zoned embankment and different physical characteristics.

### Breach Erosion

Computation of the discharge through the area of the breach, if any, is unit discharge based on the effective width. If breach is to be evaluated, the associated erosion is assumed to be initiated in an area corresponding to maximum unit discharge over the top of the dam. Therefore, for any time step, the breach area unit discharge must be greater than or equal to the maximum overtopping unit discharge. During the breach initiation phase of erosion prior to the headcut entering the reservoir, the location of the headcut is tracked and used in determination of the breach area unit discharge.

Following breach initiation, the unit discharge is computed assuming negligible energy loss from the reservoir to the hydraulic control and critical flow conditions with hydrostatic pressure at the hydraulic control. The processes that determine the erosion during embankment breach are dependent on the breach geometry and the breach area discharge.

When the surface protection on the slope is failed, the underlying material is exposed to erosive attack. The location of the surface failure is taken to be the upstream point on the slope evaluated. For a simple dam embankment cross section, this also represents the downstream edge of the dam crest. The location of the surface failure along the length of the dam is taken to be the point of minimum crest elevation which is the maximum distance from either end of the embankment. This location represents the point where the resulting headcut must move a minimum distance to result in breach of the embankment and the breach area may experience a maximum amount of widening before encountering an abutment. Erosive attack on the exposed embankment material following failure of any surface slope protection will result in the eroded area (referred to herein as the breach area) deepening, widening, and advancing upstream through the embankment. The way in which the erosion will progress depends on the local geometry and discharge. Initially, the headcut (local vertical) may not be sufficiently high to generate the

plunging action that is associated with typical headcut advance. Likewise, during latter stages of the process, the headcut may become submerged.

### Submerged Headcut Erosion

The headcut is considered to be submerged for purposes of computing erosion whenever the downstream tailwater elevation is greater than or the elevation of the crest of the headcut, or the height of the headcut is less than the critical depth of the flow in the breach area. The latter implies that the minimum depth of water at the base of the headcut is the critical flow depth based on the breach area unit discharge. When the headcut is submerged, the headcut is considered not to advance or deepen from plunging action of the flow over the crest of the headcut. If elevation of the downstream tailwater computed from total flow through the reservoir is below the elevation of the base of the headcut and the base of the headcut is within the embankment, the headcut may continue to deepen as a result of flow on the face of the dam downstream of the headcut. The rate of deepening that is associated with this flow is approximated using a normal flow depth model consistent with that used in evaluating surface protection. The erosion rate resulting in deepening of the headcut is computed by:

$$\varepsilon_r = k_d (\tau_e - \tau_c) \quad (1)$$

where

$\varepsilon_r$  = the soil detachment rate in volume per unit area per unit time,

$k_d$  = a detachment rate coefficient that is a property of the embankment material,

$\tau_e$  = the erosionally effective stress (in lb/ft<sup>2</sup>), and

$\tau_c$  = the critical soil stress (in lb/ft<sup>2</sup>).

As applied in WinDAM,  $k_d$  is expressed in (ft/h)/(lb/ft<sup>2</sup>) and is provided as input to the model (see Figure 6).

The appropriate value for input may be obtained from soil tests as described by Hanson and Cook [14].

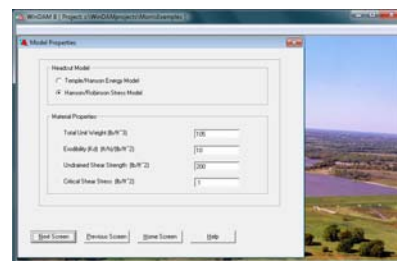


Figure 6. Breach model input

### Unsubmerged Headcut Erosion

When the tailwater is below the crest of the headcut and the height of the headcut is greater than the critical depth of flow, the flow will tend to plunge over the crest of the headcut. Stresses associated with this plunging flow may govern the rate of downward erosion at the base of the headcut, the rate of headcut advance, or both. In WinDAM, users may select an energy-based or a stress-based advance rate model. The energy-based model, designated the Temple/Hanson model, is described by

Temple et al. [8]. The model is a variation on the semi-empirical model used in the SITES spillway erosion computations [2]. The stress-based model, designated the Hanson/Robinson model, is an adaptation of the model described by Hanson, et al. [13]. These advance rate models reflect different degrees of simplification of the complex process and have different input requirements.

#### Temple/Hanson Energy-driven Model

The energy-driven headcut advance rate model computes headcut advance rate using the relation:

$$\frac{dX}{dt} = C(q_u H_h)^{1/3} \quad (2)$$

where

$dX/dt$  = the rate of headcut advance ,  
 $C$  = an advance rate coefficient,  
 $q_u$  = unit discharge, and  $H_h$  = headcut height.

#### Hanson/Robinson Stress-driven Model

The stress-driven headcut advance rate model has the advantage of being a more detailed representation of the physical processes governing headcut advance through a homogeneous soil material. However, the increased complexity makes it more difficult to utilize experience with similar conditions to calibrate the model for application to a specific site. The model is fundamentally that presented by Hanson et al. [13]. The headcut is assumed to advance through a series of mass failures driven by erosion at the base as indicated by the sketch shown below in Figure 7. Hydraulic shear stresses are assumed to erode the base increasing the distance  $E_v$  and shortening distance  $L$  until gravitational forces exceed the soil strength resulting in failure. Soil tensile strength is assumed to be negligible. The hydraulic shear stress driving the erosion on the headcut to increase  $E_v$  is computed by the relations developed by Robinson (1992).

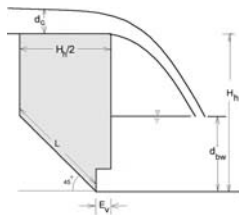


Figure 7. Sketch of headcut failure form

When headcut advance is governed by the plunging action of unsubmerged flow, the widening of the breach (headcut) area is assumed to be governed by the same action and the widening rate is taken to be equal to the headcut advance rate computed by the selected model.

#### 4. Uncertainty Analysis in WinDAM/UQ

The goal of uncertainty analysis is to obtain a better understanding of the probable range of outputs given that there is a certain amount of uncertainty in the input. In particular, based on uncertain inputs (UQ), determine the distribution function (uncertainty) of the outputs and probabilities of failure (reliability metrics); identify the

statistical measures (mean, variance, etc.) of the outputs; and identify the inputs whose variance contribute most to variance in the outputs (global sensitivity analysis) [17].

Spillway designs are compared by determining both the stability and integrity of the spillway when it is subjected to a given design storm. In a typical design, three types of hydrographs are used: principal spillway hydrographs, stability design hydrographs, and freeboard hydrographs. A *principal spillway hydrograph* is used to size the principal spillway and set the elevation of the crest of the emergency or auxiliary spillway. The principal spillway is typically a conduit through the dam used to pass low flows, whereas the auxiliary spillway is often an open channel capable of passing infrequent large flows. Earth auxiliary spillways are typically wide trapezoidal channels vegetated as appropriate for the local area. A *stability design hydrograph*, when routed through a reservoir, generates the maximum auxiliary spillway outflow that the reservoir will be expected to pass without erosion damage. For the design to be stable, erosion thresholds must bound hydraulic stresses that lead to the initiation of erosion. For flows larger than the stability design hydrograph, spillway erosion may occur, and the spillway may require maintenance (see Fig. 8). A *freeboard hydrograph* represents the maximum flow for which the structure is designed. The integrity of the auxiliary spillway, as represented by its resistance to breach, is evaluated for the spillway outflow associated with this hydrograph. Naturally, this is the most important consideration in designing an earth (soil, rock, or both) spillway. Even though extremely large discharges may cause significant erosion, the spillway must not breach during passage of the *freeboard hydrograph*. A spillway is considered breached if the spillway crest is degraded by erosion and floodwater is released through the spillway a fixed depth below the crest elevation.



Figure 8. Eroded auxiliary spillway.

Breach potential is a function of the spillway system, the characteristics of the spillway outflow hydrograph, the erodibility of the earth materials, the spillway layout, bottom width, and maintenance. Integrity analysis is based on the idea that some erosion is allowable if its occurrence is infrequent, maintenance is provided, and the spillway will not breach during passage of the freeboard hydrograph [10].

The integrated development system for water resource site analysis WinDAM/UQ – WinDAM with Uncertainty Quantification – is designed to fully integrate the simulation models in WinDAM with the uncertainty

quantification, sensitivity analysis, and parameter studies capabilities in DAKOTA. This novel, new development environment interactively guides user input, invokes the sampling and simulation models in the background, and parses the results to automatically generate output hydrographs, summary tables, and graphs.

This version of the software also allows a user to conduct parameter studies and specify the inputs to be analyzed based on a list of probability distribution functions. Twenty-five different types of distributions can be specified [7]. For example, a user could specify a Normal Distribution for hydrograph peak discharge with a mean of 50,000 cfs and a standard deviation of 10,000 cfs, or a user could specify a Uniform Distribution for a material property such as headcut index, Kh, for multiple materials or a single material as shown in Figure 9.

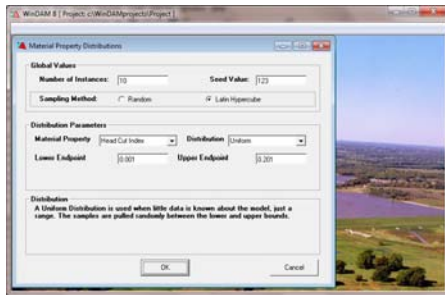


Figure 9. Uniform distribution input.

Random samples are generated using the Latin Hypercube Sampling (LHS) library routines found in Dakota 5.2. In addition to specifying the types of distributions to be used to generate samples, the user can also specify the number of instances to be generated and the algorithm to be used to generate those samples. In particular, the user can select between Monte Carlo and Latin Hypercube Sampling. With Monte Carlo Sampling, the samples are generated at random. The user can specify a random number seed to generate the same sequence of random samples. With Latin Hypercube Sampling, the samples are more evenly distributed across the search space, resulting in better coverage and fewer samples required [16]. As shown in Figure 9, a user could request 10 instances (samples) to be generated for a given material's headcut index using a Uniform Distribution from 0.001 to 0.201. Then, one sample would be randomly generated for each interval of length 0.02 from 0.001 to 0.201. For this input, the generated samples are shown in Figure 10.

```

@UNCERTAINTY
@OBSERVATIONS      10
@VARIABLES          1
  KH(1):
@SAMPLEDATA
1 1 0.178143860112386
2 1 0.114162037013804
3 1 0.153554977141378
4 1 0.184678807170631      <- min erosion
5 1 0.140484162236030
6 1 7.696084019646307E-003  <- max erosion
7 1 6.790786373412117E-002  <- mean erosion
8 1 2.112528697716014E-002
9 1 5.711419140612775E-002
10 1 9.154328306156087E-002
    
```

Figure 10. Random samples generated.

The *Build Interface* is used to generate instances for a given run based on the random variables generated, and then to invoke the simulator for each instance. The Build Interface also parses output to extract summary data for each run including instances that generate the maximum, mean, and minimum erosion.

Type	Max	Mean	Min
Eroded Area (ft <sup>2</sup> )	4395.73	4135.87	3994.64
Percent Eroded	46.18	43.45	41.97
Max. Headcut Depth Change (ft)	2.87	2.87	2.87
Max. Headcut Position Change (ft)	7.22	3.62	2.43
Total Area (ft <sup>2</sup> )	9518.26	9518.26	9518.26

Figure 11. Aux. spillway summary table.

The summary data is presented in a two-level table. The top-level table, shown in Figure 11, only displays instances resulting in maximum, mean (actually the run closest to the mean), and minimum erosion, whereas the second-level table, see Figure 12, displays all instances.

	C2012example1_0	C2012example1_1	C2012example1_2	C2012example1_3
Eroded Area (ft <sup>2</sup> )	4000.64	4004.57	4025.31	3994.64
Percent Eroded	42.03	42.70	42.29	41.97
Max. Headcut Depth Change (ft)	2.87	2.87	2.87	2.87
Max. Headcut Position Change (ft)	2.47	2.98	2.63	2.43
Total Area (ft <sup>2</sup> )	9518.26	9518.26	9518.26	9518.26
Number of Errors	0	0	0	0

Figure 12. View All runs.

After one or more related runs have been processed, they can be analyzed by using the *Output Interface*. The user can quickly compare differences between runs and instances by viewing the Summary Tables and Summary Graphs. By varying these input parameters, a user can quickly determine how changes in each will potentially impact erosion. The output graph for Spillway Erosion includes the option to display the currently selected run with the maximum erosion (shown in orange), the mean erosion (shown in red), and the minimum erosion (shown in green). The erosion for the current run is shown in blue, below in Figure 13.

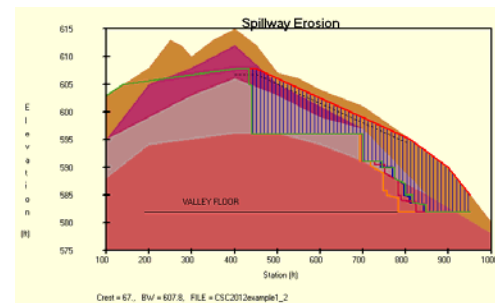


Figure 13. Spillway erosion graph.

Finally, the output can be used for a parameter study to determine how changing the value of an input parameter, in our example the headcut index, Kh, will impact the amount of erosion that results. A scatter plot of the results



is shown in Figure 14. As expected, the stronger materials result in less erosion. Note that one sample is selected from each interval due to Latin Hypercube Sampling.

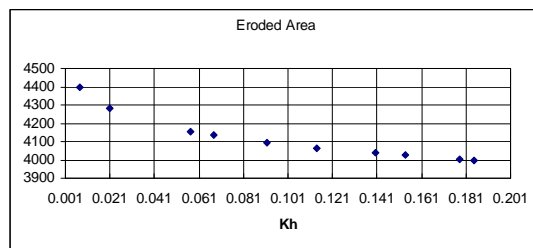


Figure 14. Scatter plot for all runs.

## 5. Conclusions

WinDAM is being developed in stages to evaluate the performance of earth dams. Existing modules with well-defined interfaces enable efficient integration of existing legacy software and future enhancements. The system provides tools that can be used to better understand the structure, function, and dynamics of such structures. Uncertainty quantification and sensitivity analysis can be incorporated by linking the development environment with DAKOTA/UQ. The paper also provides an example of how the system can be used to conduct a simple parameter study.

## Acknowledgements

The authors gratefully acknowledge the US Department of Agriculture and the US Army Corps of Engineers for providing the data and the images used in this paper.

## References

- [1] D.M. Temple, G.J. Hanson, and M.L. Neilsen, "WinDAM -- Analysis of overtopped earth embankment dams", In *Proc. of the ASABE Annual Conference*, Paper Number 062105, 2006.
- [2] D.M. Temple, G. Hanson, M. Neilsen, et al., "SITES 2005 Water Resource Site Analysis Computer Program - User Guide", T.R. No. 210-728.5, USDA, 2007.
- [3] D.M. Temple and M.L. Neilsen, "SITES integrated development environment for watersheds with multiple reservoirs", *Watershed Management 2000*, ASCE, June 2000.
- [4] M.L. Neilsen, D.M. Temple, and J.L. Wibowo, "A distributed hydrologic simulation environment with latin hypercube sampling", In *Proc. of the IASTED Intl. Conf. on Env. Modelling and Simulation*, No. 432-032, St. Thomas, USVI, Nov. 22-24, 2004.
- [5] United States Army Corps of Engineers, "Hydrologic modeling system HEC-HMS User's Manual", CPD-74A, Ver. 3.5, USACE, HEC, 2010.
- [6] USDA, Natural Resources Conservation Service, "WinTR-20 User Guide", draft, July 23, 2004.
- [7] M.L. Neilsen, D.M. Temple, and J.L. Wibowo, "A distributed hydrologic simulation environment with latin hypercube sampling", In *Proc. of the Intl. Conference on Environmental Modelling and Simulation (EMS 2004)*, No. 432-032, 2004.
- [8] D.M. Temple and G. J. Hanson, "Earth dam overtopping and breach outflow", In *Proc. of the World Water and Environmental Resources Congress*, Anchorage, Alaska, ASCE, 8 pp., 2005.
- [9] V.T. Chow, "*Open-channel hydraulics*", McGraw Hill Book Company, New York, 680 pgs., 1959.
- [10] United States Department of Agriculture, Natural Resources Conservation Service, "Earth spillway erosion model", Ch. 51, Part 628, Dams, *National Engineering Handbook*, 210-VI-NEH, 1997.
- [11] D.M. Temple, J. Wibowo, M.L. Neilsen, "Erosion of earth spillways", In *Proc. of 23<sup>rd</sup> United States Society on Dams (USSD) Annual Meeting and Conference*, pp. 331-339, 2003.
- [12] M.L. Neilsen and D.M. Temple, "A concurrent simulation model for analysis of water control structures at the watershed scale", In *Proc. of the Intl. Conf. on Par. and Dist. Proc. Tech. and Apps.*, (PDPTA 2010), pp. 1565-1570, June 26-29, 2000.
- [13] G.J. Hanson, K.M. Robinson, and K.R. Cook, "Prediction of headcut migration using a deterministic approach. Trans. ASAE, 44(3): pp. 525-531, 2001.
- [14] G.J. Hanson and K.R. Cook, "Apparatus, test procedures, and analytical methods to measure soil erodibility in-situ", in *Applied Engineering in Agriculture*, ASABE, 20(4):455-462, 2004.
- [15] K. Visser, G. Hanson, D. Temple, M. Lobrecht, M. Neilsen, T. Funderburk, and H. Moody, "WinDAM B earthen embankment erosion overtopping analysis software", in JFIC conference, 2010.
- [16] M. D. McKay, W. J. Conover, and R. J. Beckman, "A comparison of three methods for selecting values in the analysis of output from a computer code", *Technometrics*, 21(2):239-245, 1979.
- [17] M.S. Eldred, A.A. Giunta, L.P. Swiler, S.F. Wojtkiewicz, Jr., W.E. Hart, J.P. Watson, D.M. Gay, and S. L. Brown, "DAKOTA: A multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis, Version 3.2 Users Manual", Sandia Tech. Report SAND2001-3796, updated July 2004.
- [18] D.M. Temple and J.S. Moore, "Headcut advance prediction for earth spillways", *Transactions of the American Society of Agricultural Engineers (ASAE)*, 40: 557-562, 1997.
- [19] M.L. Neilsen and D.M. Temple, "A concurrent simulation model for analysis of water control structures at the watershed scale", In *Proc. of the Intl. Conf. on Parallel and Dist. Proc. Techniques and Applications*, pp. 1565-1570, June 26-29, 2000.
- [20] United States Army Corps of Engineers, "Tuttle Creek Lake, Big Blue River, Kansas", *Design Memorandum No. 5, Spillway Erosion Assessment*, USACE, Kansas City, MO, 1995.

# Application of SolidWorks<sup>®</sup> and LabVIEW<sup>®</sup>-based Simulation Technique to Gain Tuning of a 6-axis Articulated Robot

Chang Doo Jung<sup>1</sup>, Won Jee Chung<sup>1</sup> and Dong Sun Lee<sup>1</sup>

<sup>1</sup>School of Mechatronics, Changwon National University, Changwon-si, Gyeongsangnam-do, South Korea

**Abstract** - Nowadays the applications of industrial robots are spreading to a great extent so that various demands for industrial manipulators are increasing. While industrial robots are coming into wide use, the control techniques of the robots are being developed as their performance is being enhanced. In this paper, for accurate gain tuning of the lab-manufactured 6-axis articulated robot (called as "RS2") with less noise, a program routine of DSA (Dynamic Signal Analyzer) for frequency response method will be programmed using LabVIEW<sup>®</sup>. Then robot transfer functions can be obtained experimentally using frequency response method with DSA program. Data resulted from the robot transfer functions are transformed into Bode plots, based on which an optimal gain tuning will be executed. Also another contribution of this paper is the proposal of SolidWorks<sup>®</sup> and LabVIEW<sup>®</sup>-based simulation technique for the gain tuning of a 6-axis articulated robot. To realize the simulation, the LabVIEW<sup>®</sup> program used in the experimental gain tuning is incorporated with SolidWorks<sup>®</sup>. The 3-D modeling of RS2 in SolidWorks<sup>®</sup> is loaded into LabVIEW<sup>®</sup>, instead of the physical robot of RS2. Moreover virtual 6 drivers are generated on the LabVIEW<sup>®</sup>, making the joint axes of 3-D model coincident with the ones of actual RS2. Then the LabVIEW<sup>®</sup> program used in the experimental gain tuning is loaded and connected to the virtual drivers. Finally SolidWorks<sup>®</sup> and LabVIEW<sup>®</sup>-based simulation is performed from axis 6 through 1 in the same manner as in the experimental gain tuning. The comparison of the simulation-based gain tuning with the experimental gain tuning can be shown to be almost the same as those of experimental gain tuning within 5% error bound. Based on the comparison, it can be suggested that the simulation-based technique of gain tuning can be applied to a 6-axis articulated robot through interlocking SolidWorks<sup>®</sup> and LabVIEW<sup>®</sup>, instead of the experimental gain tuning.

**Keywords:** 6-axis Articulated Robot, SolidWorks<sup>®</sup>, LabVIEW<sup>®</sup>, Simulation-based Gain Tuning, Experimental Gain Tuning

## 1 Introduction

Nowadays the applications of industrial robots are spreading to a great extent so that various demands for industrial manipulators are increasing. While industrial robots are coming into wide use, the control techniques of the robots are being developed as their performance is being enhanced. For example, in order to reduce both manufacturing process and factory area, the transfer of car bodies are investigated by using 6-axis articulated robots with 6 Degrees of Freedom (DOF), instead of conveyor lines. This leads to small-scale assembly stations using 6-axis articulated robots with which can replace mass production using conveyors to meet the need of manufacturing various kinds of car in small lot. Usually 6-axis robots which are widely used for welding, spray painting and so on, have payloads from 10 kg to 300kg. In order to enhance both the control accuracy and the reliability of 6-axis articulated robot with payload of 500 kg over, the synthetic technology including design, prototyping and gain tuning should be accompanied.

As one of previous studies, Kim *et al.* [1] has presented "Prototyping and Visualization Techniques of 3-axis SCARA Robot Using DOE (Design Of Experiment) and LabVIEW<sup>®</sup>." In this paper, gain tuning using LabVIEW<sup>®</sup> has been performed for a 3-axis SCARA robot. Ahn *et al.* [2] has investigated "On Design, Prototype and Gain Optimization for Heavy Duty Handling Articulated Manipulator (HDHAM) with 6 DOF." This paper has dealt with the design of a 6-axis articulated (1/4-size lab-manufactured model for an original heavy duty handling robot [2]) robot (hereinafter called as "RS2" as shown in Fig. 1) as well as its gain tuning. But an older version of LabVIEW<sup>®</sup> programming has been used so that graphical data has been shown to be with much noise.

In this paper, for *accurate* gain tuning of RS2 with *less noise*, a program routine of DSA (Dynamic Signal Analyzer) for frequency response method will be programmed using LabVIEW<sup>®</sup>. Then robot transfer functions can be obtained experimentally using frequency response method with DSA program. Data resulted from the robot transfer functions are transformed into Bode plots, based on which an optimal gain tuning will be executed. Also another contribution of this paper is that simulation will be conducted by interlocking SolidWorks<sup>®</sup> (6-axis robot modeling) with LabVIEW<sup>®</sup>, in

order to verify the experimental results of gain tuning by being compared with the simulation results of gain tuning.

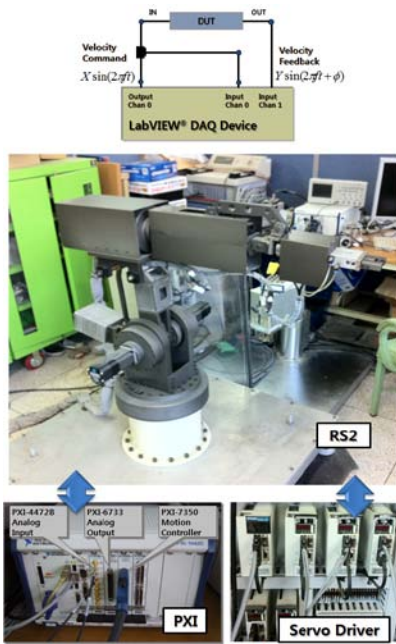


Fig.1 Configuration of RS2(6-axis articulated robot)

## 2 LabVIEW®-based Experimental Gain Tuning

For the higher control system of RS2, the Motion Controller of NI PXI-7350 equipment (see Fig. 1) has been used with the universal control and measurement software of LabVIEW®. Figure 2 represents a program controlling the axis of a robot. Here a value of motor encoder is received as an output robot signal for the applied voltage of an input value so that the robot signal is plotted by LabVIEW®. Upon the execution of DSA program, a robot transfer function can be obtained as a plot as shown in Fig. 3.

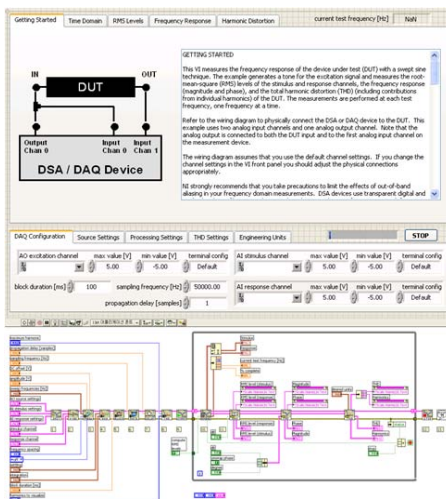


Fig. 2 LabVIEW® programming

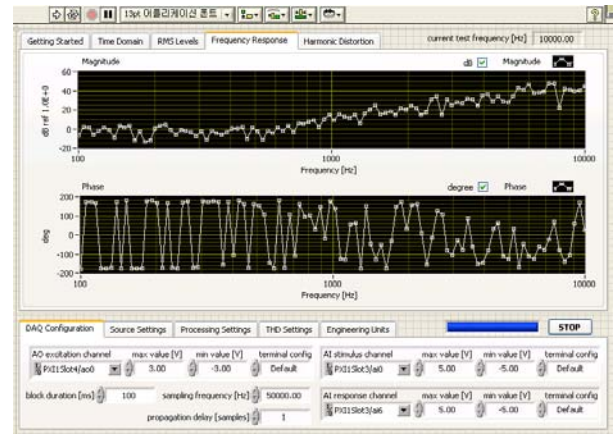


Fig. 3 Robot signal response

The faster joint velocity in robot control, the more deviation from the designated path of end-effector. In turn, this results in the vibration of robot mechanism, depending on changes in joint speed. To cope with this problem, the robot's response can be greatly improved by setting PID gain values so as to be suited to robot characteristics.

For tuning of a proportional gain ( $K_p$ ), the LabVIEW® DAQ (Data AcQuisition) equipment is connected with the 6-th (*i.e.*, the last) axis motor driver nearby the end-effector of robot. First, an arbitrary value of proportional gain has been set for the motor driver. Then an appropriate value of the sine wave amplitude  $X$  is selected according to ref. [3]. At this time, an integration effect has been eliminated by setting the integration time constant at 1000 [4]. Finally frequency response test is conducted as follows: A sine wave of  $0.5 V_{rms}$  (root mean square of voltage) from 2Hz through 500Hz is applied to the speed command pin of a servo driver as a source wave form from PXI-6733 of LabVIEW® DAQ; a Bode plot ( $G_c(s)$ ) of a closed loop can be extracted using the programmed DSA; the closed loop transfer function  $G_o(s)$  can be obtained by using the open loop transfer function  $G_c(s)$  in Eq. (1).

$$G_o(s) = \frac{G_c(s)}{1 - G_c(s)} \quad (1)$$

Then the open loop transfer function,  $G_o(s)$ , can be converted to the Bode plot by using LabVIEW® DAQ.

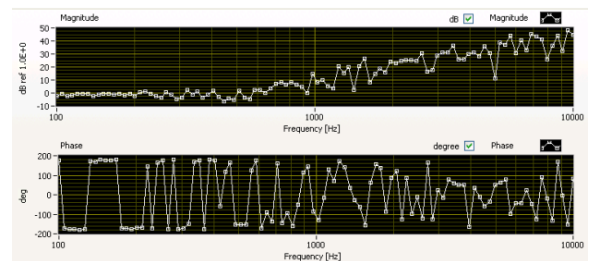


Fig. 4 Bode plot of open loop transfer function

In Fig. 4, the extracted Bode plot of open loop transfer function leads to the gain margin of -15dB and the phase margin of -91.3 degree. In general, according to Nyquist stability[5], gain margin should be -6dB ~ -20dB, while phase margin should be larger than 45 degree. In this case, if we put the gain margin to be -6dB, we can obtain the new proportional gain of velocity control loop  $K'_v$  as follows

$$20\log x = (-6\text{dB}) - (-15\text{dB})$$

$$x = 10^{\frac{-6+15}{20}} \quad (2)$$

$$\therefore K'_v = 2.8 \times 50 = 141$$

Figure 5 shows a newly extracted Bode plot of open loop transfer function,  $G_0(s)$ , when the new proportional gain of velocity control loop,  $K'_v = 141$  has been applied to the motor driver.

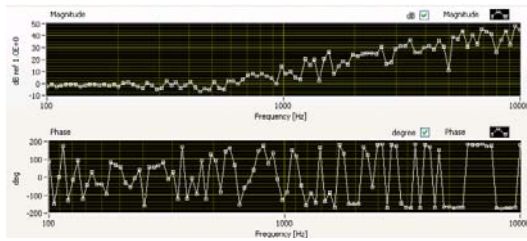


Fig. 5 Bode plot of open loop transfer function

The integral gain of velocity control loop,  $K_i$ , can be obtained from an integration time constant. When the integration time constant is applied to an integrator, it can be determined as the reciprocal of frequency whose is 10 times of gain crossover frequency. As shown in Fig. 6, the gain crossover frequency is 71.315Hz.

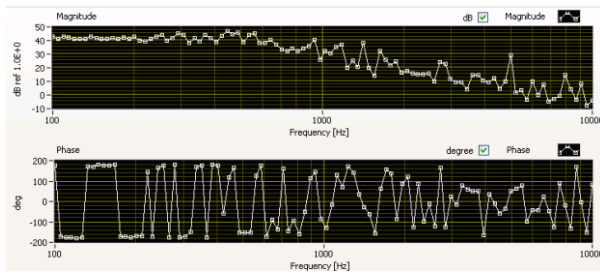


Fig. 6 New Bode plot of open loop ( $K'_v = 141$ )

Thus the integration time constant can be obtained as 14ms, *i.e.*, the reciprocal of 10 times of gain crossover frequency. Now the integral gain  $K_i$  can be resulted from Eq. (3) as follows:

$$K_i = K_v / T_i \quad (3)$$

which leads to  $K_i = 1014$ .

The proportional gain of position control loop,  $K_p$ , can be obtained by

$$K_p = \frac{\pi f_c}{2\zeta^2} \quad (4)$$

where  $\zeta$  is the damping ratio;  $f_c$  is cut-off frequency. The value of  $\zeta$  is given experimentally by 0.707 in this paper. It can be noticed that the transfer function of velocity control loop has been used for figuring out the closed loop bandwidth (*i.e.*, cut-off frequency) at the frequency whose magnitude in dB amounts to -3dB as shown in the Bode plot of Fig. 7. In this figure,  $f_c$  can be figured out as 220Hz. Therefore Eq. (4) gives  $K_p = 700$  for the 6th joint servo.

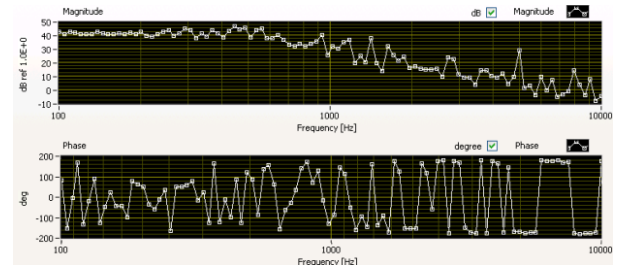


Fig. 7 Bode plot of close loop apply  $K'_v$

Through the process mentioned above, we have performed the gain optimization for the remained 5 axes joint servo controllers of 6-axis articulated robot. Consequently, Fig. 8 and Table 1 show the experimental results of the optimized gains for all 6 joint servo controllers.

Axis	Bode plot of close loop apply $K'_v$
1	
2	



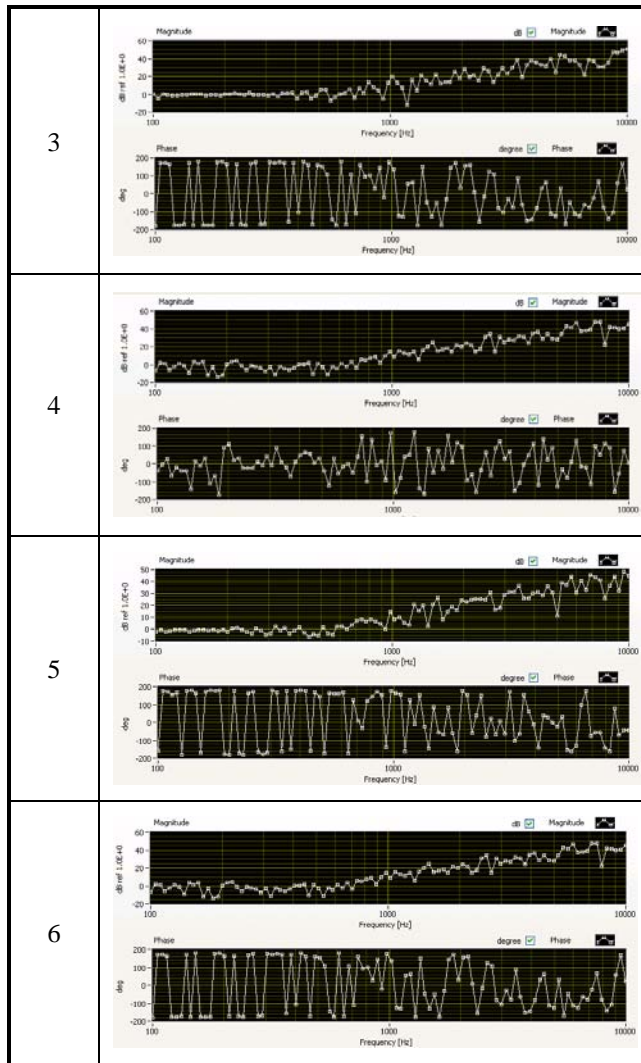


Fig. 8 Gain optimize results

Table. 1 Optimized Gains

	1Axis	2Axis	3Axis	4Axis	5Axis	6Axis
$K_v$	212	144	182	377	225	141
$K_i$	315	884	683	605	315	1014
$K_p$	465	562	561	53	47	700

### 3 Application of SolidWorks® and LabVIEW®-based Simulation Technique to Gain Tuning

To prove the test results of experimental gain tuning, a simulation technique using Solidworks® interlocked with LabVIEW® is proposed. For the simulation, 3 dimensional (or 3-D) modeling was first conducted by using the dimensions, material kinds and servo motor specifications of RS2 with SolidWorks®. Figure 9 represents the 3-D modeling of SolidWorks® for RS2. In the SolidWorks® 3-D modeling, each joint axis moved by RS2 was set from axis 1 through 6 as shown in Fig. 10.

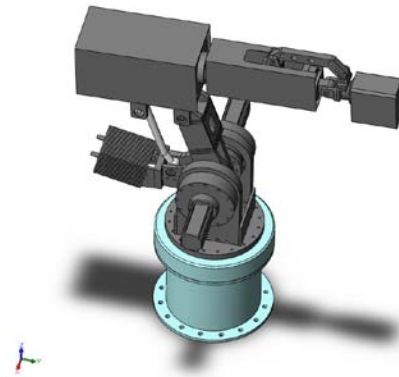
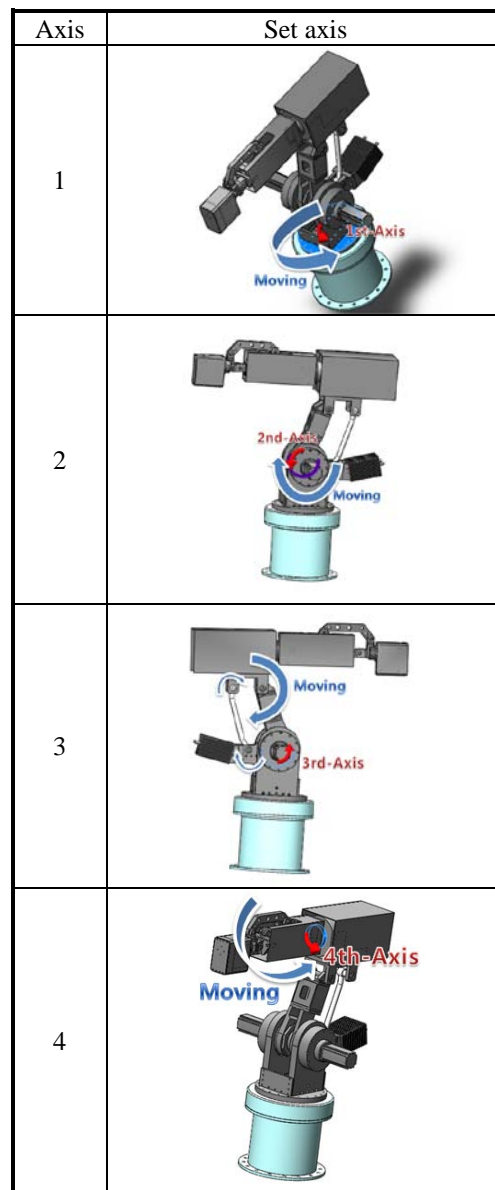


Fig. 9 SolidWorks® 3-D modeling of RS2 Using SolidWorks®



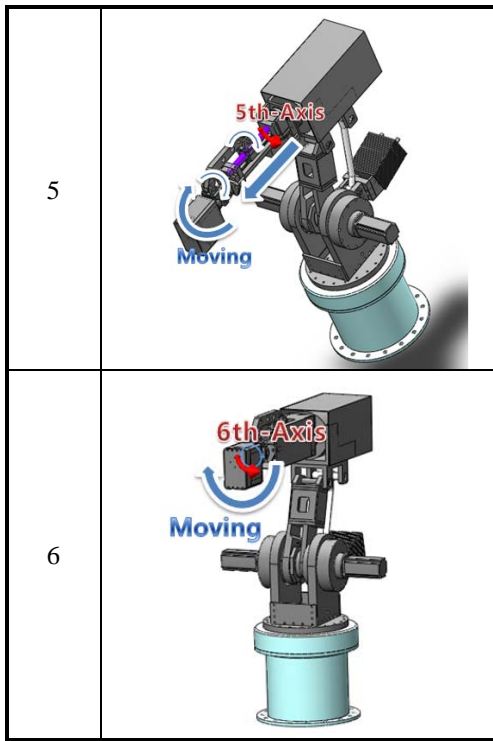


Fig. 10 Set axes of 6-axis articulated robot

To realize the simulation, the LabVIEW® program used in the previous section of experimental gain tuning was incorporated with SolidWorks®. The 3-D modeling of RS2 in SolidWorks® was loaded into LabVIEW®, instead of the physical robot of RS2. Moreover virtual 6 drivers were generated on the LabVIEW®, making the joint axes of 3-D model coincident with the ones of actual RS2. Then the LabVIEW® program used in the experimental gain tuning was loaded and connected to the virtual drivers. Finally SolidWorks® and LabVIEW®-based simulation is performed as shown in the interlocking program configuration of Fig. 11. More specific, the simulation has been executed from axis 6 through 1 in the same manner as in the experimental gain tuning. Figure 12 gives the simulation result of proportional gain of velocity control loop,  $K'_v$ .

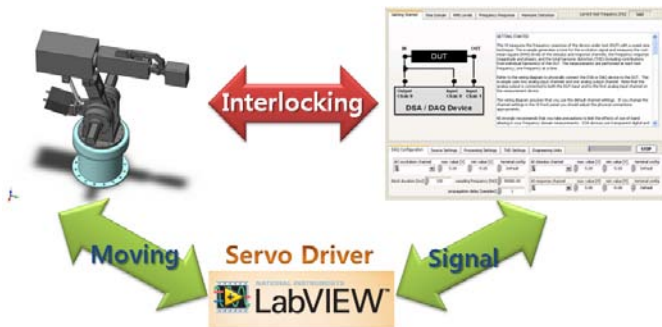


Fig. 11 Interlocking program configuration

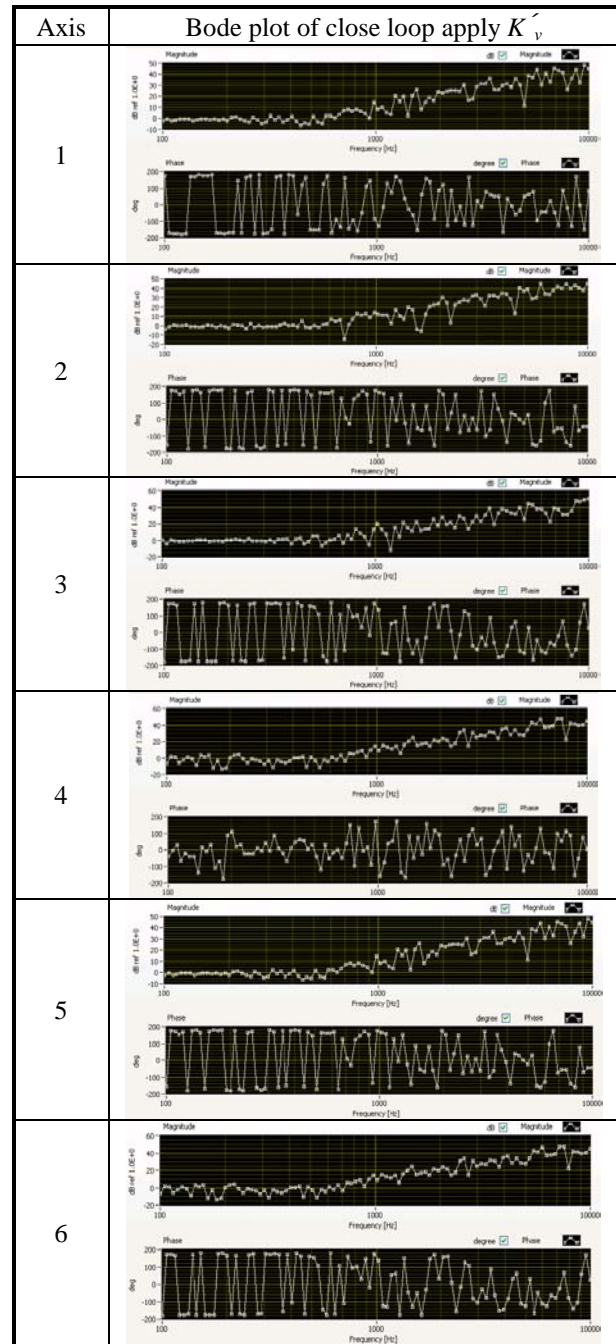


Fig. 12 Gain tuning results of simulation

Based on the above data, the optimal gain values of  $K_v$ ,  $K_i$  and  $K_p$  are calculated for each axis, using Eqs. (1) to (4) in the same way as in the experimental gain test. Table 2 below shows the simulation results of optimal gains for each axis.

Table. 2 Optimized Gains of Simulation

	1Axis	2Axis	3Axis	4Axis	5Axis	6Axis
$K_v$	210	151	180	364	239	139
$K_i$	309	879	679	615	327	1007
$K_p$	471	580	568	57	49	700

Table 3 illustrates the comparison of these simulation-based gain tuning with the previous experimental gain tuning. It can be noticed from Table 3 that the values of experimental gain tuning are almost the same as those of experimental gain tuning within 5% error bound. This verifies the effectiveness of SolidWorks® and LabVIEW®-based simulation technique for the gain tuning of a 6-axis articulated Robot. Moreover this simulation-based gain tuning is also compared with the experimental gain tuning in terms of velocity response at an applied velocity of  $0.5V_{rms}$ . Figure 13 shows both response levels for each axis. In Fig. 13, the experimental gain tuning is a little slower than the simulation-based gain tuning. Here the red line corresponds to a response velocity of the experimental gain tuning while the blue line corresponds to the simulation-based gain tuning.

Table. 3 Comparison of Experimental and Simulation-based Gain Tuning

Axis	Experimental			Simulation		
	$K_v$	$K_i$	$K_p$	$K_v$	$K_i$	$K_p$
1	212	315	465	210	309	471
2	144	884	562	151	879	580
3	182	683	561	180	679	568
4	377	605	53	364	615	57
5	225	315	47	239	327	49
6	141	1014	700	139	1007	700

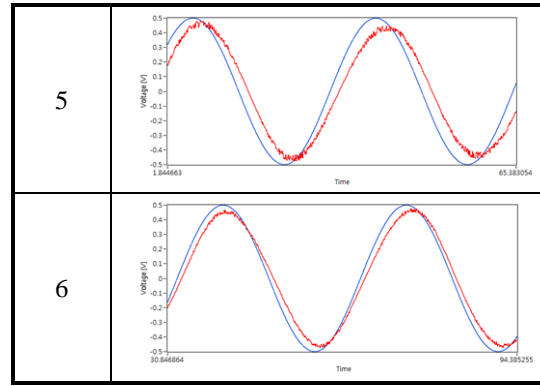
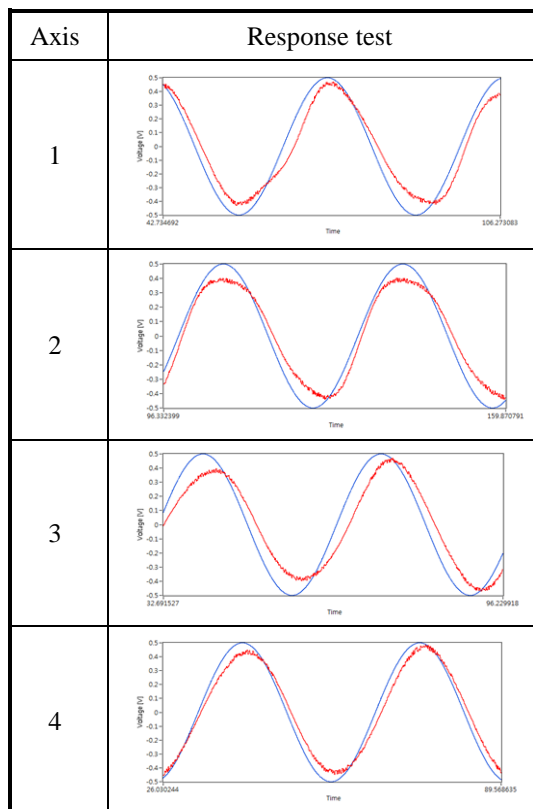


Fig. 13 Response test of experimental and simulation

### 4 Conclusion

In this paper, for accurate gain tuning of the lab-manufactured 6-axis articulated robot (called as “RS2”) with less noise, an experimental gain tuning technique has been presented based on ref. [2]. The major contribution of this paper is the proposal of SolidWorks® and LabVIEW®-based simulation technique for the gain tuning of a 6-axis articulated robot. To realize the simulation, the LabVIEW® program used in the experimental gain tuning has been incorporated with SolidWorks®. The 3-D modeling of RS2 in SolidWorks® is loaded into LabVIEW®, instead of the physical robot of RS2. Moreover virtual 6 drivers are generated on the LabVIEW®, making the joint axes of 3-D model coincident with the ones of actual RS2. Then the LabVIEW® program used in the experimental gain tuning is loaded and connected to the virtual drivers. Finally SolidWorks® and LabVIEW®-based simulation was performed from axis 6 through 1 in the same manner as in the experimental gain tuning. The comparison of the simulation-based gain tuning with the experimental gain tuning was shown to be almost the same as those of experimental gain tuning within 5% error bound. Based on this comparisons, it can be concluded that the simulation-based technique of gain tuning can be applied to a 6-axis articulated robot through interlocking SolidWorks® and LabVIEW®, in lieu of the experimental gain tuning.

### 5 Acknowledgement

The authors of this paper were partly supported by the Second Stage of Brain Korea 21 Projects. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2011-0013902).

### 6 References

[1] Jung Hyun Kim, Won Jee Chung, Hyo Gon Kim, "Prototyping and Visualization Techniques of 3-axis SCARA Robot Using DOE and LabVIEW®", MSV07, 2007

[2] Jin Su Ahn, Won Jee Chung, Koo Hong Kwon, "On Design, Prototype and Gain Optimization for Heavy Duty Handling Articulated Manipulator (HDHAM) with 6 DOF", WMSCI 2010, 2010

[3] Ogata, K., 1990, **MODERN CONTROL ENGINEERING**, Prectice-Hall, Inc. pp. 448~467.

[4] Haugen, F., 2004, **PID CONTROL OF DYNAMIC SYSTEMS**, Intl specialized book service inc. pp.342~349.

[5] Kuo, B. C., 1991, **AUTOMATIC CONTROL SYSTEMS**, Prentice-Hall, Inc. pp. 747~762.



# The effectiveness of PIES comparing to FEM and BEM for 3D elasticity problems

A. Boltuc, E. Zieniuk, K. Szerszen

Faculty of Mathematics and Computer Science, University of Bialystok,  
Sosnowa 64, 15-887 Bialystok, Poland

**Abstract** - *The paper presents the results of research on the effectiveness of the PIES method developed by the authors, applied to modeling and solving complex 3D problems of elasticity. The results were compared with those obtained using classical element methods FEM and BEM. For the calculation in mentioned methods we have used professional programs ANSYS (FEM), BEASY (BEM) and our own software in the case of PIES. Several examples were solved, but we have included only three of them, which seemed to be enough complex. The effectiveness of methods was compared taking into account: the number of input data required to define the geometry, the way of modeling and imposing the boundary conditions, the number of solved algebraic equations and the accuracy and reliability of the results.*

**Keywords** - *computer modeling and simulation, boundary problems, numerical methods, PIES, FEM, BEM*

## 1 Introduction

Computer simulations in engineering problems are widespread, and even necessary. For modeling and solving boundary problems are used so-called computer methods, among which the most popular are the finite element method (FEM) [1,2,3] and the boundary element method (BEM) [2,4,5]. Both methods are used for many years, thus is available a commercial software implementing them. The main idea of these methods is modeling of any areas by the discretization of the boundary (BEM) or the boundary and domain (FEM). This causes great possibilities taking into account the complexity of considered geometries, because any shape can be modeled by dividing it into smaller elements. However, on the other hand, the number of input data, the number of algebraic equations in the system, which is solved to obtain final solutions, the time and effort we have to invested increase. In addition, we receive a number of redundant solutions at initially defined boundary nodes (and area nodes), and in the case of insufficient accuracy of the results we have to repeat the process of discretization.

These reasons were encouraged to searching and eventually developing a method, which would at least partially eliminated mentioned drawbacks. Our previous research resulted in the creation of the parametric integral equation system (PIES) [6], in which the boundary geometry is analytically included and can be defined using curves (2D) or surfaces (3D) known from computer graphics [7]. The

effectiveness of this approach lies in the separation of the simultaneous approximation of the boundary geometry from the function which is the solution of the equation on the boundary. It gives a possibility for modeling areas without interfering in the approximation of the solution and vice versa.

Till now, the proposed approach was tested taking into account 2D [8,9] and 3D [10,11] problems modeled by Laplace's, Poisson, Helmholtz and Navier-Lame equations. In 3D problems modeled by the last mentioned equation, however, verification was referred only to the comparison of obtained results with analytical solutions for fairly elementary examples [12]. However, the main aim of the authors was to create an alternative to FEM and BEM, hence the necessity of the comparison of results obtained using PIES with these methods. The first studies on the comparison with BEM on the very elementary shape (the cube) was carried out in [13].

The main aim of this paper is to examine the effectiveness of the numerical implementation of PIES for solving complex 3D problems of elasticity in comparison with classical methods. To implement FEM software ANSYS was used, in the case of BEM - BEASY, whilst in PIES authors own software. Following subjects were compared: the number and type of input data required to define the boundary geometry, the number of solved algebraic equations and the accuracy of calculations.

## 2 Comparative analysis of FEM, BEM and PIES

In order to demonstrate the reliability and effectiveness of the proposed and tested in the paper PIES method, it is necessary to compare it with known numerical methods, especially with FEM and BEM. History and the mathematical basis of these methods are widely described in the literature. In the case of FEM all about its formulation and its computer aspects can be found among others in [1,2,3]. Also BEM, despite the fact that it is a new computer method has a number of papers on it. Thus, information about the following stages of solving boundary problems by BEM and its applications are available for example in [2,4,5]. The PIES approach has been developed as an alternative to mentioned methods, and the first results of research were published in 2001 [6]. In a series of subsequent papers we have included and described: the new technique of global modeling of the boundary in a parametric way (without the discretization), the obtained PIES for various differential equations and the way of its numerical solving

[10,8,9,11,12,13].

Due to the fact that each of mentioned methods has been already repeatedly characterized, the paper is limited only to a brief summary of their advantages and drawbacks, and also potential possibilities. This summary is given in Table 1.

Table 1  
The comparison of considered methods

Feature	FEM	BEM	PIES
generality	high	possibility of application only if a fundamental solution exists	as in BEM
discretization	boundary and domain by finite elements	only boundary by boundary elements, domain is discretized only in some cases (e.g. material nonlinearity, Poisson equation, etc.)	none, modeling as in computer graphics (by curves or surfaces)
workload / computing resources / time	high, tedious thickening of a mesh, especially in multiple discretization	smaller, reduces the dimensionality of the problem by one	least, the minimum number of data for modeling and numerical solving
accuracy	high at a high level of discretization, the accuracy of solutions and their derivatives is different	high, higher than in FEM at the same level of discretization, the same accuracy of solutions and their derivatives, less accurate solutions in the vicinity of the boundary	high, without discretization, the rest as in BEM
solutions	discrete, at boundary and area nodes, a lot of unnecessary information	discrete on the boundary, continuous in the area, redundant information at boundary nodes	continuous, obtained at any points
mathematical aspects	matrix is sparse, symmetrical and well-conditioned	difficult to calculate singular integrals, full, asymmetrical, not positive definite and worse than in FEM conditioned matrix	as in BEM
effectiveness of applications	high, difficulties with infinite areas	infinite areas are considered in natural way, not very effective for thin areas	as in BEM
modification of the boundary geometry	requires re-discretization of the modified area and its boundary	requires re-discretization of the modified boundary and dividing of the area into cells in the case of integration over area	automatically adapts to the modified shape, modification by small number of points
software	widely available	less available than in FEM	only authors software

As shown in Table I, the main differences between methods lies in the way of modeling the area and the resultant benefits or disadvantages. Therefore, later in the paper this aspect is characterized more precisely, and modeling approaches used in different methods are presented on the example of the ball (Fig. 1).

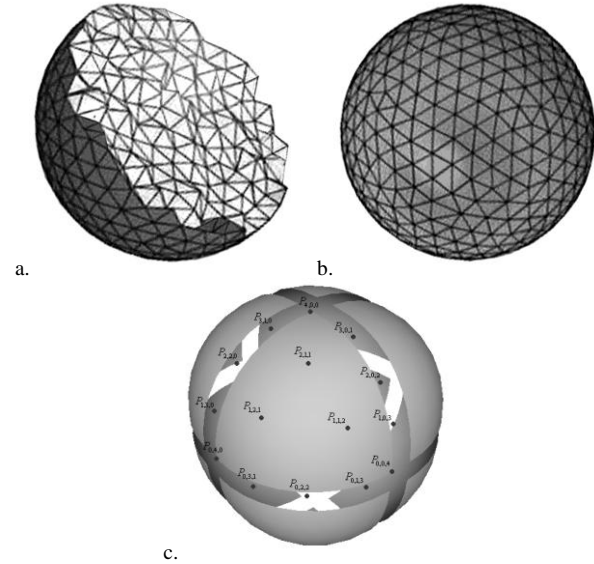


Fig. 1. Modeling of the ball by: a) FEM, b) BEM and c) PIES

In the case of FEM modeling of the area is done using so-called finite elements and is reduced to divide the area and the boundary into smaller 3D sub-areas. Finite elements in this method serves two functions. They are used to model the considered area and to facilitate the approximation of problem solutions on individual elements. Thus, is realized the simultaneous approximation of both the area (through its division into finite elements) and the solutions on each of finite elements by local shape functions. Existing in FEM opportunity for using different shape functions makes it possible to improve the accuracy and convergence of the method. The most frequently used elements in 3D problems of elasticity are tetrahedral and hexahedral elements of various degrees, which involves posing of a different number of nodes to define them. Combining finite element leads to rising an element mesh representing the 3D area. Generation of the finite element mesh can be performed using various automated techniques (e.g. technology of primitives, superelements or triangulation) and many other programs that use these techniques [1,14]. It should be noted that in case of too low accuracy of obtained by FEM solutions it is necessary to make a re-discretization: into a greater number of elements, another kind or degree or just a different arrangement. This is a tedious process and often totally unnecessary in terms of the accuracy of the area definition. As mentioned earlier example of such modeling in FEM using tetrahedral finite elements is shown in Fig.1a, where discretized was the whole area of the ball.

Another method BEM is also based on the simultaneous approximation of the boundary geometry and boundary functions. From a practical point of view, the considered boundary should be divided (discretized) into small segments called boundary elements and on each of them one should assume the type of boundary function, in the form of known in advance shape function. Boundary elements usually take the form of triangles or quadrangles declared by the different number of nodes (e.g. triangles by 3, 6 or 9 nodes).

Used strategy of the simultaneous approximation of the boundary shape and boundary functions is one of the key disadvantages of BEM. In order to improve the accuracy of boundary solutions it is necessary to concentrate the number of nodes. There are two ways (which result in increased cost of calculations): by introducing elements of the higher degree (declared by the higher number of nodes) or the division of the boundary into the large number of boundary elements. In both cases, however, we still have to deal with the discrete form of the boundary function. The advantage in comparison to FEM is reduction the dimensionality of the problem by one, due to only boundary discretization, not the boundary and the area. An example of modeling the ball by triangular boundary elements is presented in Fig.1b, where only the sphere is discretized.

The method proposed by the authors is characterized by the elimination of the discretization of both the area and the boundary. The way of modeling the boundary for 3D issues is taken directly from computer graphics and is integrated in PIES. We are talking about the use of the various types of parametric surfaces, extremely popular in computer visualization of three-dimensional geometric objects, and also widely used in the design of mechanical structures in CAD systems.

The simplest surfaces are rectangular Coons surfaces of the first degree [15]. These patches are characterized by the fact that to their practical definition is required, as shown in Fig.2a, imposing only four corner points. Equally simple in the declaration are triangular surfaces [15], where it is necessary to define three vertices of this triangle (Fig.2b). However, these surfaces are flat, so they allow the modeling of only polyhedral areas. To model areas with curved boundary we should take into account surfaces of higher degrees, the most versatile and popular are Bézier rectangular and triangular patches of the third degree [16] (Fig.2c,d).

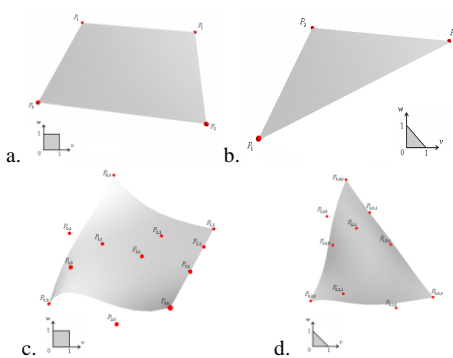


Fig. 2. Flat surfaces: a) rectangular, b) triangular and curvilinear: c) Bézier rectangular of the 3 degree, d) Bézier triangular of the 3 degree.

In the case of the surfaces of higher degrees (than first) to define their shape it is necessary to pose so-called control points. Considering a rectangular surface of the third degree we should define coordinates of 16 control points, and 10 for triangular one. In the case of increasing the degree of the surface increases also the number of control points. An

example of modeling in PIES using triangular Bézier surfaces of the fourth degree is presented in Fig. 1c, where a ball, or rather as in BEM its sphere, is defined by only 8 such surfaces (for each of the surfaces we have to define 15 control points).

Mentioned patches can imitate any large and complex surfaces, so long as we are able to obtain the shape of considered part using one surface. Complex geometries can be modeled in a similar manner using the combination of multiple surfaces. A very important advantage of this technique is that the number of surfaces is associated only with the possibility of modeling the chosen physical geometry. The accuracy of solutions depends on the completely other elements, what comes from the main idea of the PIES method - the separation of the approximation of the boundary shape from boundary functions. Therefore, the shape of the boundary in PIES is defined using the minimum number of surfaces needed for the accurate modeling, which eliminates the necessity for tedious mesh generation, and thus the process of discretization. In addition, modeling using surfaces is more effective, because it helps ensure the class of continuity at borders between surfaces.

Equally simple in PIES is the modification of the declared geometry. It requires changing single corner or control points. It is worth to emphasized, that after modification PIES automatically adjusts to the new shape, because such are the mathematical basis of the method. Changing the location of even a single point results in a substantial change in the shape, which can be seen in Fig. 3.

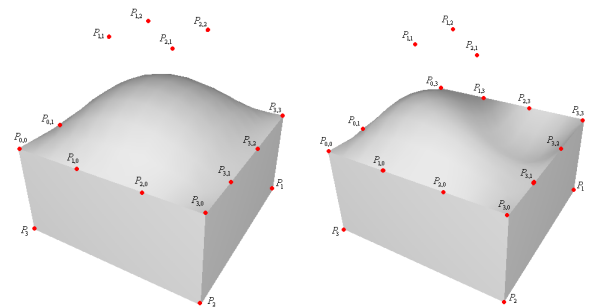


Fig. 3. Modification of the 3D area by moving one control point

It should be noted that in FEM or BEM making a modification of the shape we are always dealing with the re-discretisation and generating a new mesh of elements. This is particularly disadvantageous and inefficient in the problems of optimization or identification of the boundary geometry, where revised geometry modeling is done many times in subsequent steps of an iterative process.

### 3 Analysis of results

In order to show the effectiveness and confirm the reliability of PIES paper presents three examples. To solve problems using all considered methods we have used a software mentioned in the introduction of the paper. Compared were: the number of input data needed to define the boundary and boundary conditions, the number of solved algebraic

equations and the accuracy and convergence of the results.

### 3.1 Example 1

The first example concerns a polyhedron presented in Fig. 4a. The considered rectangular cuboid, which is a bracket, is firmly fixed at the left end and subjected to a uniform normal load  $p = 5MPa$  acting along the upper side. Selected for the calculation values of material constants are Young's modulus  $E = 1MPa$  and Poisson's ratio  $\nu = 0.3$ . The same constants were adopted in all the examples presented in the paper.

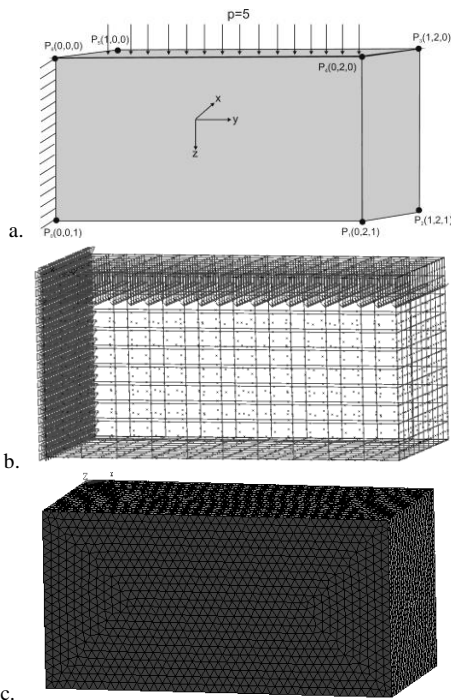


Fig. 4. A bracket with boundary conditions modeled in PIES (a) and meshes from: BEASY (b) and ANSYS (c)

Taking into account tested in the paper method PIES the considered shape is very simple to model, because its faces are rectangles. For this reason, defining the geometry is limited to the use of the simplest flat surfaces – rectangular Coons surfaces of the first degree. For each of them we have to define only corner points, resulting in a six surfaces and 8 corner points. Thus, the considered shape was modeled using the minimum amount of data that allows its accurate projection. For the numerical solution of the problem we have assumed a uniform distribution of 25 collocation points on each surface. As a result, the system of 450 algebraic equations was solved.

The aim of this study is to examine the effectiveness of PIES compared to classical element methods, such as FEM and BEM. Particular emphasis is placed on the comparison of the way of modeling the shape and the accuracy of obtained results. For this reason, the same problem was solved using the ANSYS software and BEASY. In both cases, we have searched for the mesh for which the minimum number of

elements gives stable solutions. Finally, in ANSYS we have used 135626 8-node hexahedral finite elements type SOLID45 (defined using 24784 nodes) and the system of 74352 algebraic equations was solved. In the case of BEASY 600 quadrilateral quadratic elements were used, whilst in order to define them 2046 boundary nodes were imposed. Finally, in order to obtain the final results the system of 6138 algebraic equations was solved. A detailed analysis of the effectiveness of modeling the considered polyhedron for each of the three considered above methods is shown in the table below.

Table 2  
The comparison of considered methods in terms of the number of data

	PIES	BEM	FEM
<i>boundary modeling</i>	6 rectangular Coons surfaces	600 quadratic quadrilateral boundary elements	135626 8-node hexahedral finite elements
<i>number and type of data</i>	8 corner points	2046 boundary nodes	24784 nodes in the boundary and area
<i>number of algebraic equations</i>	450	6138	74352
<i>imposing the boundary conditions</i>	continuous on each surface	discrete	discrete
<i>obtaining of solutions</i>	at any point of the boundary and area	at 2046 boundary nodes and any point of the area	at 24784 nodes in the area and boundary

The first comparison of methods in terms of the number of data for modeling, the number of solved algebraic equations, and the way of obtaining the results came out favorably for PIES. Next, we should check whether this reduction in the number of data will not reduce the accuracy of solutions. Therefore, the values of displacements in all directions at 13 points of the horizontal cross-section passing through the center of the area were examined. The results obtained by PIES, BEM and FEM are presented in Table 3.

Table 3  
The comparison of displacements obtained by considered methods

point	$u_x$			$u_y$			$u_z$		
	FEM	BEM	PIES	FEM	BEM	PIES	FEM	BEM	PIES
1	-0.108	-0.118	-0.126	0.036	0.036	0.048	10.084	10.268	10.479
2	0.046	0.042	0.044	0.127	0.130	0.148	15.923	16.188	16.478
3	0.034	0.030	0.030	0.399	0.402	0.426	37.774	38.242	38.712
4	0.032	0.029	0.029	0.342	0.348	0.370	47.478	48.026	48.561
5	0.008	-0.001	-0.001	0.539	0.541	0.567	54.143	54.743	55.314
6	0.000	-0.004	-0.004	0.516	0.527	0.548	58.124	58.755	59.351
7	0.009	0.002	0.002	0.499	0.508	0.531	67.524	68.235	68.879
8	0.009	0.003	0.003	0.681	0.689	0.715	76.707	77.496	78.185
9	-0.059	-0.063	-0.064	0.859	0.871	0.897	106.250	107.280	108.099
10	0.241	0.235	0.239	0.941	0.948	0.986	121.500	122.654	123.553
11	0.181	0.174	0.177	1.004	1.011	1.048	122.510	123.665	124.555
12	0.119	0.115	0.116	1.070	1.080	1.111	128.540	129.753	130.655
13	-0.134	-0.140	-0.144	1.185	1.195	1.367	143.860	145.201	146.379

As shown in Table 3 obtained by each method solutions are characterized by a similar level of accuracy. This allows to confirm the reliability of proposed and tested by the authors

method, but also to emphasize the fact that these results were obtained in a much more efficient way than in classical methods FEM and BEM. In their case, the mesh was generated consisting of a large number of finite or boundary elements, much more numerous system of algebraic equations was solved and was obtained a number of redundant information at the nodes of the boundary or area.

### 3.2 Example 2

In Example 2, we have considered a polyhedron of more complex shape shown in Fig. 5a, firmly fixed and subjected to a uniform normal load  $p = 10MPa$ .

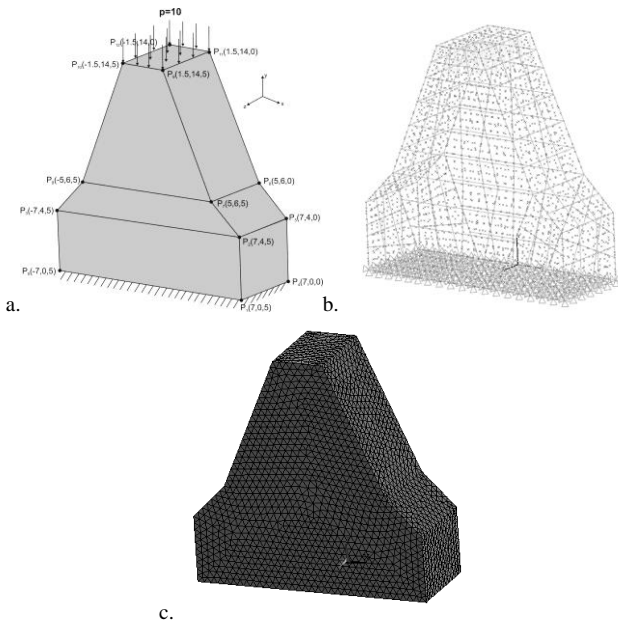


Fig.5. Considered geometry and boundary conditions a) PIES, b) BEASY and c) ANSYS

In the PIES software the analyzed polyhedral shape was modeled only by rectangular Coons surfaces (Fig.5a). Should be emphasized flexibility in the shape of surfaces (rectangular and trapezoidal) and their size. Generated in such way boundary was modeled by the minimum number of input data – after posing 16 corner points and defining 14 rectangular Coons surfaces. For calculations we have consider 25 collocation points on each surface arranged uniformly. Finally, this involves the solution of 1050 equations required to obtain the final results.

The same shape was modeled in BEM (with the help of BEASY) using 205 quadratic quadrilateral elements, for which definition 903 nodes were used (Fig. 5b), and thus the system of 2709 algebraic equations was solved.

The last stage concerned the modeling of the problem using the FEM method (ANSYS). Fig. 5c presents the area defined by 71467 8-node hexahedral finite elements, for which definition we require 13536 nodes. Finally the system consisting of 40608 algebraic equations was solved.

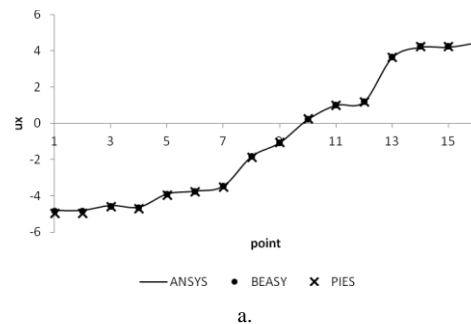
It should be noted that the number of nodes in FEM and BEM is several times higher compared with 16 corner points in PIES. The advantage of PIES is also the number of solved algebraic equations, very important for the accuracy of the final solutions. Numerical error in fact may increase with the increase in the number of equations in the solved system. The summary of the type and number of data used for modeling is presented in Table 4.

Table 4  
The comparison of considered methods in terms of the number of data

	<i>PIES</i>	<i>BEM</i>	<i>FEM</i>
<i>boundary modeling</i>	14 rectangular Coons surfaces	205 quadratic quadrilateral elements	71467 8-node hexahedral finite elements
<i>number and type of data</i>	16 corner points	903 boundary nodes	13536 boundary and area nodes
<i>number of algebraic equations</i>	1050	2709	40608

The effectiveness of the proposed method was shown taking into account the way of the shape modeling, the number of necessary input data and the number of equations solved. But equally important question to be answered is the level of the accuracy of obtained by PIES solutions compared to the results obtained using ANSYS and BEASY. Displacement values  $u_x, u_y, u_z$  were examined at 16 selected points of the domain, and the results of the comparison are given in Fig. 6.

As can be seen in Fig.6 results obtained by BEASY, ANSYS and PIES are similar taking into account the values of displacements, as well as their distribution. In order to accurately analyze the results, Table 5 contains the exact solutions at few selected points. As seen, at most points the solutions are very similar, and occurred differences will be explained in the further stages of the development of the method. It should be stressed again that solutions through PIES were obtained at 2.5 times less numerous system of equations than in BEM. Much, much more data and unknowns are in the FEM method. Thus, with much less effort on modeling and much smaller number of input data we have obtained reliable solutions by PIES.



a.

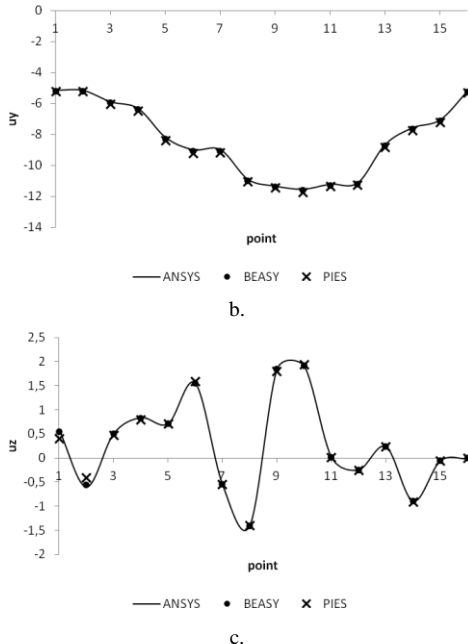


Fig. 6. Displacements a)  $u_x$ , b)  $u_y$ , c)  $u_z$  at considered points obtained by PIES, BEASY and ANSYS

Table 5

The comparison of displacements obtained by FEM, BEM and PIES

x	y	z		ANSYS	BEASY	PIES
-5.759	4.281	3.749	$u_x$	-4.5242	-4.56024	-4.58501
			$u_y$	-5.8975	-5.95087	-6.02625
			$u_z$	0.50818	0.500922	0.472321
-4.284	4.281	3.570	$u_x$	-3.8865	-3.91187	-3.94027
			$u_y$	-8.1940	-8.27438	-8.36344
			$u_z$	0.71588	0.713073	0.718234
-0.985	4.283	4.542	$u_x$	-1.0648	-1.06938	-1.04913
			$u_y$	-11.324	-11.4028	-11.4352
			$u_z$	1.8480	1.849352	1.79832
1.157	4.285	2.234	$u_x$	1.1902	1.200081	1.20594
			$u_y$	-11.112	-11.1938	-11.2561
			$u_z$	-0.24293	-0.24408	-0.2438
5.001	4.290	2.397	$u_x$	4.2094	4.244336	4.26464
			$u_y$	-7.0602	-7.14224	-7.21802
			$u_z$	-0.05546	-0.05693	-0.05714

### 3.3 Example 3

In practical applications, except boundary problems defined in polyhedral areas, very important are also issues defined in areas with curvilinear boundaries. These boundaries in PIES can be described using Bézier surface patches. An example of the geometry modeled by 7 rectangular Bézier surfaces of the third degree (curved boundary fragments) and 5

flat rectangular Coons surfaces with boundary conditions (uniform normal load  $p = 1MPa$ ) is shown in Fig. 7a.

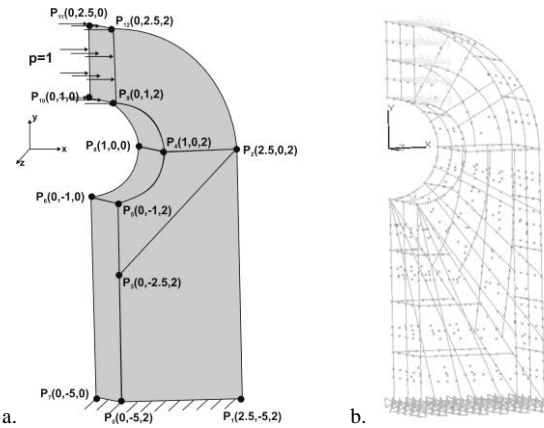


Fig. 7. Considered geometry modeled in a) PIES, b) BEM (BEASY)

A complete declaration of the boundary defined in PIES by 14 surfaces requires 112 points (control points of Bézier and corner points of Coons surfaces). In Fig. 7a, for clarity, only four main points of each surface are marked. On each surface we have defined 25 uniformly spaced collocation points and eventually have solved the system of 1050 algebraic equations.

In connection with some regularity of results observed in two previous examples, this time the effectiveness and accuracy of the proposed method were compared only with BEM. For this reason, the considered geometry was modeled in BEASY (Fig.7b), where as in the previous example, for the comparative analysis we have chosen such mesh for which the minimum number of elements gives stable numerical solutions. Finally, 104 quadrilateral and triangular quadratic elements defined by 475 nodes were used. The number of algebraic equations that had to be solved is 1425.

Solutions from various cross-sections were analyzed, but we have decided to include only one of them  $x = 1.5, -4.5 \leq y \leq 1, z = 1$ . Values of displacements obtained by PIES and classical BEM implemented by BEASY are presented in Fig.8. In order to detailed analysis of the reliability of obtained by PIES solutions we have also examined stress values, which selected components are presented in Fig. 9.

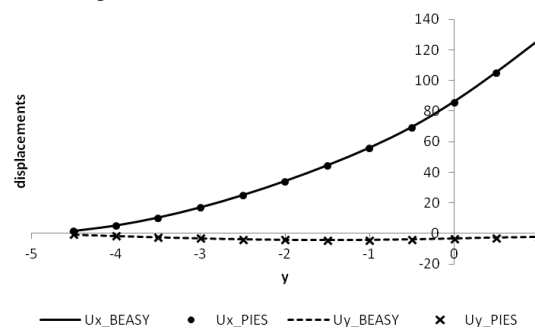


Fig. 8. Displacements in the considered cross-section

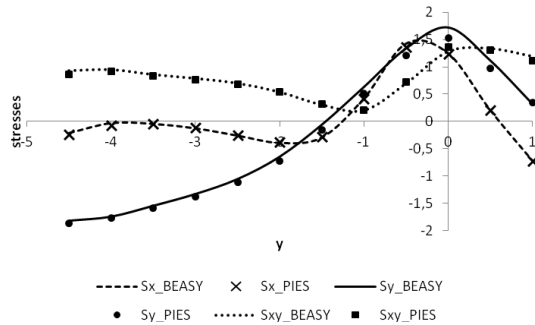


Fig. 9. Stresses in the analyzed cross-section

As can be seen from above figures, both displacements and stresses obtained using the proposed and developed by the authors method are characterized by high accuracy in comparison to BEM. Lines representing the various functions of solutions have the same shape, which shows a similar accuracy of considered techniques. It should be noted that in BEM the way of modeling of the boundary geometry is much less efficient (the discretization), which eventually leads to the solution of the equation system containing approximately 1.5-times more of algebraic equations. The use of PIES limits time to data preparation, their analysis, and finally minimize the set of solutions to those that are necessary for analysis.

## 4 Conclusions

The paper presents the effectiveness of solving 3D boundary value problems of elasticity using the proposed PIES method. A number of examples were solved, for which we have compared with classical element methods (FEM, BEM): the data necessary to model the geometry (the type and number), the way of the modeling and posing of boundary conditions, as well as the reliability of results (displacements and stresses). It can be stated that obtained using PIES solutions are comparable to those obtained with other numerical methods, however, they are obtained with a much smaller number of input data. Moreover, the system of equations solved in PIES is much less numerous, and solutions can be obtained at any points of the boundary and area, not previously selected nodes.

## 5 Acknowledgment

This work is funded by resources for science in the years 2010-2013 as a research project.

## 6 References

[1] O. C. Zienkiewicz, R. L. Taylor. "The Finite Element Method". McGraw- Hill, 1977.  
 [2] M. Ameen. "Computational elasticity". Alpha Science International Ltd., 2005.  
 [3] R. D. Cook. "Finite element modeling for stress analysis". John Wiley&Sons, 1995.

[4] M. H. Aliabadi. "The boundary element method vol. 2. Applications in Solid and Structures". John Wiley&Sons, Ltd, 2002.  
 [5] C. A. Brebbia, J. Dominguez. "Boundary element methods for potential problems"; Applied Mathematical Modelling, 1, 372-378, 1977.  
 [6] E. Zieniuk. "Potential problems with polygonal boundaries by a BEM with parametric linear functions"; Engineering Analysis with Boundary Elements, 25, 185-190, 2001.  
 [7] G. Farin. "Curves and Surfaces for computer Aided Geometric Design". Academic Press Inc., 1990.  
 [8] E. Zieniuk, A. Bołtuć. "Non-element method of solving 2D boundary problems defined on polygonal domains modeled by Navier equation"; International Journal of Solid and Structures, 43, 7939-7958, 2006.  
 [9] E. Zieniuk, A. Bołtuć. "Bézier curves in the modeling of boundary geometry for 2D boundary problems defined by Helmholtz equation"; Journal of Computational Acoustics, 14/3, 353-367, 2006.  
 [10] E. Zieniuk, K. Szerszeń. „Linear Coons surfaces in the modeling of polygonal geometries in 3D boundary problems modeled by Laplace equation”; Archiwum Informatyki Teoretycznej i Stosowanej, 17/2, 127-142, 2005 (in polish).  
 [11] E. Zieniuk, K. Szerszeń. "Triangular Bézier patches in modelling smooth boundary surface in exterior Helmholtz problems solved by PIES"; Archives of Acoustics, 1/34, 1-11, 2009.  
 [12] E. Zieniuk, K. Szerszeń, A. Bołtuć. „PIES in solving 3D boundary problems modeled by Navier-Lame equation in polyhedral areas”; Modelowanie Inżynierskie, 11/42, 487-494, 2011 (in polish).  
 [13] E. Zieniuk, A. Bołtuć, K. Szerszeń. "The effectiveness of PIES compared to BEM in the modelling of 3D polygonal problems defined by Navier-Lame equations"; Proceedings of 2nd International Conference on Information and Communication, 77-82, Amsterdam, Holland, 2011.  
 [14] P. Ladevèze, J.T. Oden (Editors). "Advances in Adaptive Computational Methods in Mechanics"; Studies in Applied Mechanics, 47, 1998.  
 [15] M. Mortenson. "Mathematics for Computer Graphics Applications: An Introduction to the Mathematics and Geometry of Cad/Cam". Geometric Modeling, Scientific Visualization, and Other CG Applications, Industrial Press, 1999.  
 [16] P. Kiciak. „Fundamentals of modeling curves and surfaces”. WNT, 2000 (in polish).

# Trading Space for Time: Constant-Speed Algorithms for Grouping Objects in Scientific Simulations

Adrienne Keen<sup>1</sup> and Clarence Lehman<sup>2</sup>

<sup>1</sup>London School of Hygiene and Tropical Medicine, Keppel Street, London WC1E 7HT, UK

<sup>2</sup>University of Minnesota, 123 Snyder Hall, 1474 Gortner Avenue, Saint Paul, MN 55108 USA

**Abstract**—*Microscale models that simulate discrete individuals commonly organize those individuals into groups of similar characteristics. During the simulation, individuals are frequently added to groups, removed from groups, and moved among groups. Moreover, individuals must be selected randomly based on probabilities that vary among the groups, and even that may vary within a group. Space and time limit the number of individuals, as the number desired can be large—up to the population of entire nations or more. Therefore algorithms for processing them should be efficient. Here we explain a set of Order-1 algorithms for managing groups. These algorithms run at a constant speed regardless of whether 100 individuals are included or 100 million. They use space–time tradeoffs recently made possible by large computer memories to bring the number of iterations per operation close to one. This increases the scale of problems that can be addressed by individual-based, agent-based, discrete-event, and other microscale simulations.*

**Keywords:** individual-based simulation, discrete event simulation, equation-free models, order-1 algorithms, memory–speed tradeoff

## 1. Introduction

One goal of the algorithms described here is to select individuals at random, from given groups, in the least possible time. Such operations can be needed billions of times during large-scale individual-based and other microscale simulations in science and industry [1] [2] [3]. For example, epidemiological models may need to select simulated individuals from given age groups, birthplaces, and susceptibility categories as targets of infection transmitted during the simulation.

Selection is easiest and fastest if the data structures for all members of a group occupy a contiguous block of memory, with no intervening gaps. Then selecting a random member is merely generating a random number between 1 and the number of members in the group, then indexing the corresponding member. If all individuals in the group have the same probability of being selected, that operation is clearly independent of the number of individuals in the group. In other words, it is of “Order-1,” running at the same speed regardless of how many individuals are in a group. If

the probabilities of being selected vary within the group, the algorithm remains Order-1, but with slightly more time required per selection.

During the course of the simulation, individuals will be added to groups, deleted from them, and moved between them. When an individual is deleted, for example, the gap in the group formed by that individual must be closed up. These addition–deletion operations occur frequently during the simulation, so ideally they should also be Order-1, as are the algorithms explained in this paper.

With high-speed algorithms for adding and deleting, individuals can be organized into groups even if random selection is not needed. For example, such organization can help tallying—keeping counts of those in different groups through time without scanning the array of individuals. Under the right circumstances, the algorithms can also be used within in many applications that require efficient priority queues [4].

Order-1 algorithms are rare, but they have been known since the early days of computer science [5]. They often rely on an abundance of computer memory to keep data structures sparse, and therefore were costly to apply in earlier days. Now, because allocating a gigabyte array is readily within the reach even of portable computers, new rules for memory use apply. Algorithms that trade additional memory for additional speed are possible and desirable.

Some groups may have myriad individuals and others only a few. The size of the groups may not be known in advance and may vary widely during the simulation. Therefore, it is not practical to allocate separate arrays large enough for each group, and dynamically allocating and reallocating memory would be unnecessarily inefficient. The algorithms described here eliminate the need for such reallocations, using space not needed by smaller groups to accommodate the needs of larger groups. The algorithms use buffer areas of allocated but unused memory to speed operations. Their running times become independent of the number of individuals and nearly independent of the number of groups.

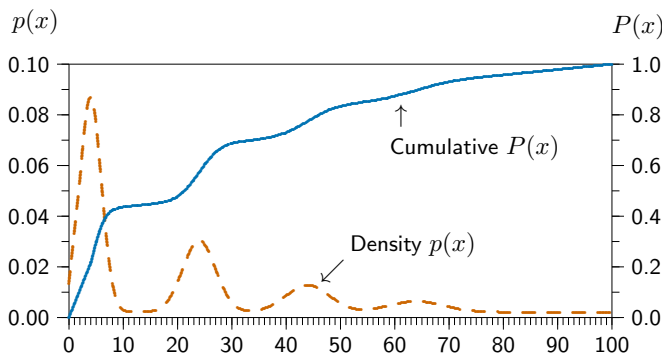
## 2. Sample application

For an example of use of these algorithms, consider an epidemiological model having what is called “age dependent



mixing.” Transmission of influenza, for example, is more likely between those in similar age groups, since individuals of similar ages spend much of their time in similar locales—such as day-cares, schools, business places, or assisted living facilities. These may have approximately uniform mixing within age groups, but reduced mixing between age groups [6]. An age-dependent mixing strategy can substitute for a more accurate but unknown contact network.

In an individual-based, discrete-event simulation of this type, when an infectious individual is about to infect another, the new individual must be selected efficiently from a list of perhaps tens of millions of individuals. Which group will receive the infection is determined in the program at large, outside the algorithms described here, via empirical or hypothetical probability distributions. Figure 1 is a hypothetical example of such a distribution. It represents the probability of transmission by age class from an infected four-year-old. The horizontal axis is the age of a susceptible individual, the vertical axis represents relative probability of receiving an infection from an infected four-year-old.



**Figure 1.** Sample probability function for selecting groups, illustrated as the probability of transmission of infection from an infected four-year-old to each of 100 groups, here representing one-year human age classes. The probability of being selected depends on the group, according to an empirical probability distribution, determined or surmised. Left axis, probability density function, dashed curve. Right axis, corresponding cumulative probability distribution, solid curve.

In this illustration, four-year olds can transmit infections to anyone but have the highest probability of transmitting to other four-year olds and to nearby ages, an increased probability of transmitting to those the age of their parents, peaking around 24 years old, and their grandparents, around 44 years old, and great-grandparents around 64 years old. Such distributions would be empirically estimated and groups to receive an infection would be selected many times during the simulation. This would be accomplished, for example, with a non-uniform random number generator that selects from arbitrary probability distributions [7] [8] [9], like the distribution in Figure 1. That is accomplished through the cumulative probability distribution by representing its inverse in an efficient way, as with piecewise-polynomial curves,

then sampling from that inverse distribution with uniform random numbers (e.g., [10]). That process, accomplished outside these algorithms, determines which group of individuals will receive the infection. The algorithms described here are then applied to select an individual from the group. They are also applied to add individuals to groups, delete them, or move them between groups.

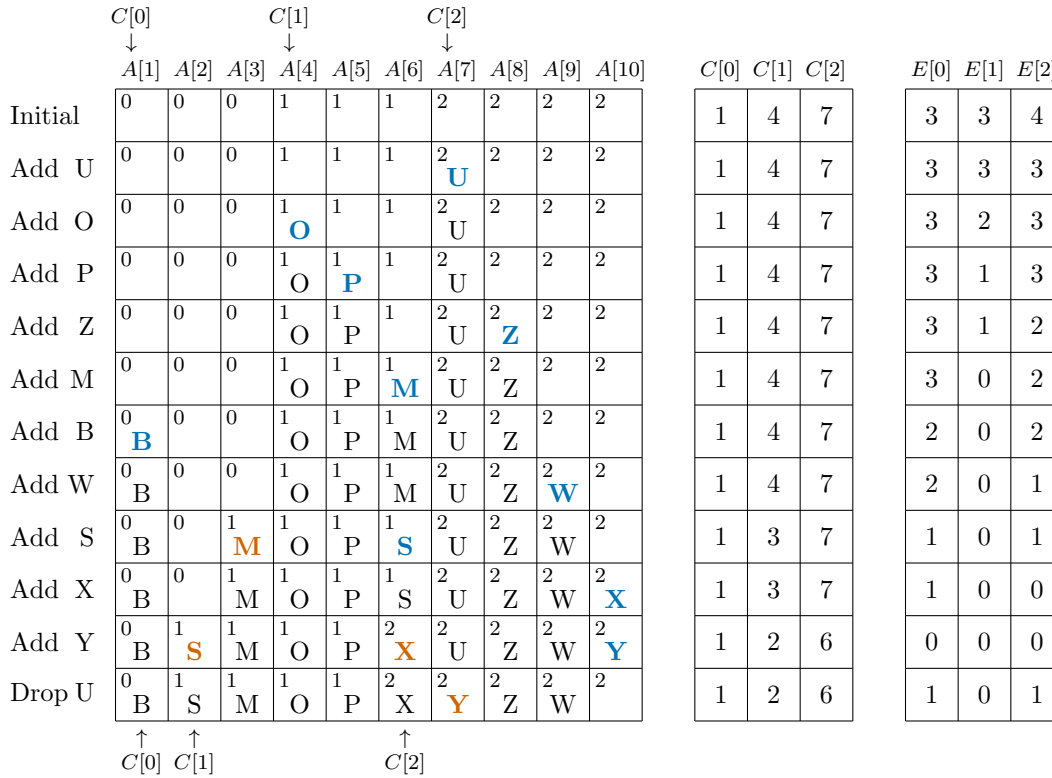
### 3. Algorithms and data structures

One array exists for individuals and two for groups (see appendix).  $A[n]$  is a one-dimensional array of structures representing individuals, in order by group and contiguous within each group, but in no particular order within groups, and with possible gaps between groups. The relevant simulation data for each individual is carried in this array, which varies by application, but in our example would include such items as sex, birth date, birthplace, and geographic coordinates. Each individual may also carry information on its own probability of being selected within a group, in data element  $A[n].v$ . By convention,  $A[1]$  is the first entry used, so that index 0 may be used as a null list index. This is the largest data structure, potentially containing tens or hundreds of millions of individuals and occupying gigabytes of memory.

$C[i]$  is a one-dimensional array of groups, identifying the lowest-numbered individual in  $A[n]$  for each group, structured so that  $C[0]$  is the index of the lowest numbered individual in the first group and that  $C[j+1] - C[j] - E[j]$  is the number of individuals in group  $j$ .  $E[i]$  is simply a one-dimensional array identifying the number of empty cells at the end of each group. Deletions increase the number of empty cells and additions draw from those empty cells. These are relatively small arrays, typically occupying only kilobytes each. They can be initialized by *GroupInit* (appendix), or by a custom routine. Global scalar variables are  $ma$ , the maximum number of individuals in  $A$ , and  $nc$ , the number of groups.

#### 3.1 Selecting from a group

Because all individuals in a group occupy contiguous elements of array  $A[n]$ , selecting one at random simply involves generating a uniformly distributed random number between one and the number of individuals in the group, then selecting that individual after biasing the number to align with index numbers in  $A[n]$  for the group. If all members of the group are equally likely, the process is complete. If probabilities of selection vary among members of the group, then the “sieve method” [11] applies within the group. That is implemented in Algorithm 1 (appendix). In effect, the sieve method tentatively selects a random member of the group and examines its probability of being selected, relative to others in the group. One additional uniform random number determines if the tentatively selected individual should remain unselected, based on the probability recorded



**Figure 2.** Illustration of addition and deletion. Each row shows the array of individuals  $A[n]$ ,  $1 \leq n \leq 10$ , of groups  $C[i]$ ,  $0 \leq i \leq 2$ , and of empty cells  $E[i]$ , same range on  $i$ . The top row is the initial empty state. The next 10 rows are individuals being added in random order by Algorithm 2. The bottom row is an individual being deleted by Algorithm 3. Numbers in the upper left of each cell of  $A[n]$  indicate the group to which that cell is presently allocated, as defined by array  $C[i]$ . The letters in the centers of the cells of  $A[n]$  represent distinct individuals assigned to the cells.

for it in  $A[n]$ . If so, the individual is ignored and the selection process is repeated. The entire process is still Order-1 on the number of individuals, though with a higher coefficient.

### 3.2 Adding to a group

Adding a member to a group is a relatively simple but exacting process. If space is available at the end of the group, the new individual is simply placed there and the number of unused entries in the group,  $E[i]$ , is reduced by 1. If, on the other hand, the area allocated to the group is full, then one member from each of one or more neighboring groups must be shifted to make room. The external function *Transfer* is called to actually move the entries, since the larger program may have other lists that must be updated when an individual is moved.

This process is defined precisely in Algorithm 2 (appendix) and illustrated in Figure 2. The latter is a step-by-step example starting with an empty list of 10 individuals and filling it in random order. In the example there are 26 possible individuals, each with a fixed “name”, ‘A’ through ‘Z’. Each is assigned an initial group, which organizes the list, such that ‘A’ through ‘J’ initially belong to group 0, ‘K’ through ‘T’ initially belong to group 1, and ‘U’ through ‘Z’

initially belong to group 2. Individuals can be moved from group to group as the simulation proceeds. In practice, large numbers of individuals would be processed, not just ten.

Array  $A[n]$  starts with 10 empty slots. The three groups, 0, 1, and 2, have an initial allocation of 3, 3, and 4 slots, respectively. Entries are added at the first available slot for their group, until that group is filled. Then entries cascade to the right or left, resting in the first available slot.

Individual ‘U’ is added first. It is initially in group 2, which begins at position  $A[7]$ . That is followed by ‘O’, ‘P’, ‘Z’, ‘M’, ‘B’, and ‘W’, all of which fall into empty slots pre-allocated in their initial groups. That situation is typical when the array is sparse, and in part leads to the algorithm’s speed. However, when ‘S’ is to be added, the space allocated to its group (cells  $A[4]$ ,  $A[5]$ , and  $A[6]$ ) is full. To make room, group 2 could move down or group 1 could move up. This example shows the latter, and accomplishes that by shifting ‘M’ into the open cell at  $A[3]$ , changing  $C[1]$  accordingly, and placing ‘S’ at the newly opened cell at  $A[6]$ . Note that ‘S’ could simply have been placed in position  $A[3]$ , rather than moving ‘M’ there. Various optimizations could be applied at the cost of a little additional complexity in the code, though the effects on overall timing would be minor.

Individual 'X', initially designated for group 2, falls immediately in the open cell at  $A[10]$ . Individual 'Y' is also of group 2, and the cells for that group are full. The algorithm moves 'S' to empty cell  $A[2]$ , then moves 'X' to  $A[6]$ , just vacated by 'S', and finally stores 'Y' in the vacated cell  $A[10]$ . Arrays  $C$  and  $E$  are updated in the process.

With this method, the maximum number of array elements moved is bounded by the number of groups, independent of the number of individuals. When sufficient memory is allocated to leave a fraction of the space free in  $A[n]$ , then typically no array elements must be moved. That makes the Order-1 coefficient as small as possible. It also makes it, for practical purposes, independent of the number of groups.

### 3.3 Deleting from a group

Deleting is simpler. The individual is removed and the member from the end of that group is moved into its place, which keeps the individuals in the group contiguous. For example, when individual 'U' is removed, that would leave a gap in the middle of group 2. Individual 'Y' at the end of the group is moved into its place, increasing by one the number  $E[2]$  of empty slots at the end of the group. Deleting is always independent of the number of individuals and the number of groups.

Moving an individual from one group to another is merely deleting from one group and subsequently adding to another. As before, the external function *Transfer* is called to actually move the entries.

## 4. Timing

Timing tests of the above algorithms, starting with an empty array and building to one-hundred million individuals ( $10^8$ ), averaged 1.04 seconds total on a 2.8 GHz processor, or 10.4 nanoseconds per addition. Individuals were added in random order, with all groups equally likely, into an array that had 15% more room than required. Selecting from 100 groups averaged 2.1 nanoseconds per selection. Deleting them all at the end averaged 6.8 nanoseconds each.

With sufficient space in array  $A[n]$ , as provided above, the algorithms become independent not only of the number of individuals but also of the number of groups. Keeping all else equal while increasing groups a thousand-fold, to 100,000 distinct groups, required no additional running time in the algorithm itself.

Nonetheless, large multi-gigabyte arrays such as these may exercise a processor's internal memory caches in various ways, and that can affect finer details of timing. In any case, the algorithms remain efficient for a very large number of groups.

## 5. Conclusions

The algorithms presented here can be incorporated into any individual-based or other microscale model, where they can speed simulations many orders of magnitude over alternative

methods that are not Order-1. The methods applied in these algorithms are part of a large-scale simulation model developed by one of us (A.K.) for tuberculosis in the UK. Compilable copies of the code described here and related simulation algorithms are available free from the authors upon request.

## 6. Acknowledgements

We are grateful to Todd Lehman and Shelby Williams for reading the manuscript and helping with the presentation of the material, and to Richard Barnes for enlightening discussions on the application of these algorithms to priority queues. The project was supported in part by a resident fellowship grant to C.L. from the University of Minnesota's Institute on the Environment, by grants of computer time from the Minnesota Supercomputer Institute, Minneapolis, Minnesota, and by doctoral research funding to A.K. from the Modelling and Economics Unit at the Health Protection Agency, London.

## 7. Contributions

Both authors contributed equally. The algorithms began with an Order-1 method by A.K. for managing two groups. C.L. and A.K. worked jointly to incorporate any number of groups. A.K. proposed how to handle variable probabilities within groups. C.L. initially coded the algorithms and both authors contributed to the code and to the manuscript.

## References

- [1] L. Gustafsson and M. Sternad, "Consistent micro, macro and state-based population modelling," *Mathematical Bioscience*, vol. 225, pp. 94–107, 2010.
- [2] I. G. Kevrekidis and G. Samaey, "Equation-free multiscale computation: Algorithms and applications," *Annual Review of Physical Chemistry*, vol. 60, pp. 321–344, 2009.
- [3] V. Grimm and S. F. Railsback, "Individual-based modeling and ecology," *Princeton University Press, New Jersey*, 2005.
- [4] S. Edelkamp and S. Schrödl, "Heuristic search: Theory and applications," *Morgan Kaufmann*, p. 152, 2011.
- [5] A. I. Dumey, "Indexing for rapid random access memory systems," *Computers and Automation*, vol. 6, pp. 6–9, 1956.
- [6] J. Mossong, N. Hens, M. Jit, P. Beutels, K. Auranen, R. Mikolajczyk, M. Massari, S. Salmaso, G. S. Tomba, J. Wallinga, J. Heijne, M. Sadkowska-Todys, M. Rosinska, and W. J. Edmunds, "Social contacts and mixing patterns relevant to the spread of infectious diseases," *PLoS Medicine*, vol. 5, p. e74, 2008.
- [7] C. Lehman and A. Keen, "Efficient pseudo-random numbers generated from any probability distribution," *Proceedings, International Conference on Modeling, Simulation, and Visualization Methods*, 2012.
- [8] W. Hörmann and J. Leydold, "Continuous random variate generation by fast numerical inversion," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, pp. 347–362, 2003.
- [9] L. Devroye, "Non-uniform random variate generation," *Springer-Verlag, New York*, 1986.
- [10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes: The art of scientific computing, third edition," *Cambridge University Press, New York*, 2007.
- [11] M.-T. Tsai, T.-C. Chern, J.-H. Chuang, C.-W. Hsueh, H.-S. Kuo, C.-J. Liao, S. Riley, B.-J. Shen, C.-H. Shen, D.-W. Wang, and T.-S. Hsu, "Efficient simulation of the spatial transmission dynamics of influenza," *PLoS ONE*, vol. 5, p. e13292, 2010.

## 8. Appendix: Grouping Algorithms

To use the algorithms described in this paper, it is only necessary to understand the entry and exit conditions that appear at the beginning of each, not the code itself. Nonetheless, to allow complete evaluation of the algorithms, and to encourage further development of them, we present them as pseudo-code inspired by and simplified from the programming languages C, R, and Python. The algorithms are defined

with sufficient precision that they can be tested, timed, or translated to other languages. Familiarity with a relatively few operators\* and with the syntax of flow control (if, for, while, etc.), is sufficient to follow the algorithms. Text copies of this pseudo-code translated into operational C are available from the authors upon request, or from the associated website, 'www.cbs.umn.edu/modeling'.

---

### DATA STRUCTURES

$ma \equiv 100000000$   
 $nc \equiv 10000$

**structure** *Individual*

**integer**  $g$ ;

**real**  $v$ ;

**real**  $t_{birth}$ ;

**integer**  $rob$ ;

**structure** *Individual*  $A[ma + 1]$ ;

**integer**  $C[nc + 2]$ ;

**integer**  $E[nc + 2]$ ;

Sample, maximum number of individuals in  $A$ .

Sample, maximum number of groups in  $C$ .

Sample, data structure for individuals

(Optional, group for this individual)

(Optional, relative probability of remaining unselected)

(Sample, time of birth)

(Sample, region of birth).

Array of individuals.

Array of beginning index for each group in  $A$ .

Number of empty cells trailing each group.

---

### Algorithm 1. SELECT RANDOM ELEMENT FROM GROUP

**Upon entry to the algorithm,** (1)  $k$  defines the group to be sampled. (2)  $nc$  specifies the number of groups. (3)  $C$  indexes the first individual in each group. (4)  $E[k]$  contains the number of empty cells at the end of the group. (5)  $A[i].v$  contains the probability of each individual in the group remaining unselected, relative to the individual least likely to remain unselected. All  $A[i].v$  are zero if all individuals are equally likely. (6)  $Rand$  returns a uniformly distributed random number between 0 and 1, including 0 but not including 1. **At exit,** *GroupSelect* indexes the individual selected. If zero, the group is empty.

**integer** *GroupSelect*( $k$ ) **integer**  $k$ ; **integer**  $h, n$ ;

**if**  $C[0] = 0$  **or**  $k \geq nc$  : **return** 0;

$C[k + 1] - C[k] - E[k] \rightarrow h$ ;

**if**  $h \leq 0$  : **return** 0;

**loop** until return :

$C[k] + h * Rand() \rightarrow n$ ;

**if**  $A[n].v = 0$  : **return**  $n$ ;

**if**  $Rand() \geq A[n].v$  :

**return**  $n$ ;

1. Guard against null cases.

2. Determine how many occupy the group.

3. Select an individual randomly and use it unless it has a probability of remaining unselected.

4. Otherwise use it only in proportion to its relative probability.

---

\* The pseudo-code given here is two-dimensional, as in the language Python, so that indentation completely defines the nested structure, with no need for bracketing characters such as '{' and '}'. Variables and function names are italicized and flow control and reserved words are bolded.

The assignment operator is represented either as ' $\leftarrow$ ' or ' $\rightarrow$ ', similar to assignments in R. The compound assignments ' $a + 1 \rightarrow a \rightarrow b \rightarrow W[i][j]$ ' and ' $W[i][j] \leftarrow b \leftarrow a \leftarrow a + 1$ ' are equivalent, first incrementing  $a$  and placing the results back in  $a$ , then in  $b$ , and then in the  $i, j$ th element of the array  $W$ .

Using up-tick and down-tick operators to write ' $\uparrow a$ ', ' $\downarrow a$ ', ' $a \uparrow$ ', and ' $a \downarrow$ ' form pre- and post-increments by one, as in ' $++a$ ', ' $--a$ ', ' $a++$ ', and ' $a--$ ' of C.

Arrays are indexed as in the language C, starting with 0. Data types are '**integer**' and '**real**', with the latter specifying floating point. Operator precedence is that of C, with assignments having lowest precedence. Logical operators such as '**and**' and '**or**' are preemptive, terminating a chain of logical operations as soon as the result is known. Permanent global assignments, as would be represented '#define  $\alpha \beta$ ' in C, are rendered as ' $\alpha \equiv \beta$ '.

**Algorithm 2. ADD ELEMENT TO GROUP**

**Upon entry to the algorithm,** (1)  $A$  contains the list of individuals, ordered by group, with sufficient room for at least one more individual. (2)  $C$  indexes the first individual in each group. (3)  $E$  contains the number of empty cells in each group. (4)  $k$  is the group for the individual to be added. (5)  $ma$  contains the maximum number of individuals that may reside in  $A$ . (6)  $nc$  contains the number of groups. (7)  $Transfer(m, n)$  is an external function to move individuals from entry  $m$  to entry  $n$ , including updating of any external information. **At exit,** (1)  $GroupAdd$  returns the index of an available entry in  $A$  where the individual is to be added. If zero, none can be added. (2)  $A$ ,  $C$ , and  $E$  are updated to include space for the new individual.

```

integer GroupAdd(k) integer k; integer i, m, n, d;
  if C[0] = 0 : return 0;
  0 → d;
  while k - d ≥ 0 or k + d + 1 ≤ nc :
    k - d → i; if i ≥ 0 and E[i] > 0 : exit loop;
    k + d + 1 → i; if i ≤ nc and E[i] > 0 : exit loop;
    -1 → i; ↑d;
  if i < 0 : return 0;
  while i ≥ k :
    ↓E[i]; C[i + 1] - E[i] - 1 → m;
    if i = k : return m;
    C[i] → n; if m ≠ n : Transfer(n, m);
    ↑E[i - 1]; ↑C[i]; ↓i;
  while i ≤ k :
    ↓E[i]; C[i + 1] - 1 → m;
    if i = k : return m;
    C[i + 2] - 1 → n; if m ≠ n : Transfer(n, m);
    ↑E[i + 1]; ↓C[i + 1]; ↑i;

```

1. Guard against null cases.
2. Search forward and backward simultaneously for the nearest group with an empty slot, returning with failure if the array is full.
3. If there is a slot at the present location, use it, or if forward, cascade it back to the current location.
4. Otherwise, if there is a slot earlier in the list of groups, cascade it forward to the current location.

**Algorithm 3. DELETE ELEMENT FROM GROUP**

**Upon entry to the algorithm,** (1)  $k$  is the group for the individual to be deleted. (2)  $n$  indexes the individual being deleted, whose entry is ready for reuse. (3)  $A$  contains the list of individuals, ordered by group. (4)  $C$  indexes the first individual in each group. (5)  $E$  contains the number of empty cells in each group. (6)  $nc$  contains the number of classes. (7)  $Transfer(m, n)$  is an external function to move individuals from entry  $m$  to entry  $n$ , including updating of any external information. **At exit,** (1)  $GroupDelete$  returns zero if the operation failed. (2)  $A$ ,  $C$ , and  $E$  are updated to exclude the deleted individual.

```

integer GroupDelete(k, n) integer k, n; integer m;
  if C[0] = 0 or k > nc or C[k + 1] - C[k] ≤ 0 : return 0;
  ↑E[k]; C[k + 1] - E[k] → m;
  if n ≠ m : Transfer(m, n);
  return 1;

```

1. Guard against null cases.
2. Transfer the last occupied entry to the deleted slot.
3. Return with success.

**Algorithm 4. INITIALIZE ALL GROUPS**

**Upon entry to the algorithm,** (1)  $nc$  is the number of groups. (2)  $ma$  is the maximum number of individuals. **At exit,** (1)  $C[i]$  indexes the location for the first individual of group  $i$ . (2)  $E[i]$  contains the number of entries initially in group  $i$ . Groups are of equal size to within the limits of integer arithmetic.

```

GroupInit() integer i, k, n, r;
  ma/nc → n; ma - nc*n → r;
  1 → k;
  for i from 0 to nc :
    n → E[i]; if i ≥ nc - r : ↑E[i];
    k → C[i]; k + E[i] → k;
  0 → E[nc];

```

1. Compute the group size and remainder.
2. Initialize the location and size of each group, distributing the remainder across higher-numbered groups.
3. Close the list of groups.

# A strategy for modeling 3D piecewise homogeneous regions and solving boundary value problems by generalized PIES

E. Zieniuk, K. Szerszen, A. Boltuc

Faculty of Mathematics and Computer Science, University of Bialystok  
Sosnowa 64, 15-887 Bialystok, Poland

**Abstract** - *The purpose of this paper is to generalize the parametric integral equation system (PIES) to three-dimensional piecewise homogeneous regions. Generalization is at the level of numerical solving of PIES as a results of combining properly modeled subregions with different parameters. The effectiveness of solving problems with homogeneous areas modeled using various differential equations was studied in previous works. To model the geometry of subregions we have used flat and curved surfaces. In order to check the reliability and effectiveness of the proposed method several examples were solved. These examples refer to 3D boundary problems modeled by Laplace's and Navier-Lame differential equations.*

**Keywords** - computer modeling and simulation, boundary problems, parametric integral equation system (PIES), subregions, parametric surfaces, Laplace's and Navier-Lame equations

## 1 Introduction

Computer analysis of a wide range of technical problems often reduces to application of the popular finite element method (FEM) [1] and boundary element method (BEM) [2]. The finite element method based on discretization of the whole area is naturally suited to direct analysis of problems defined over regions which are piecewise homogeneous, where each subregion has different properties. On the other hand, the use of FEM generates a very large number of algebraic equations, and the obtained solutions are closely related to nodes which define finite elements.

Because of these disadvantages occurring in FEM, it is reasonable to apply to mentioned problems alternative BEM, which reduces the need for the discretization only to the boundary. This limits the number of declared nodes, which eventually leads to obtain a smaller system of algebraic equations to solve. In contrast to FEM, the practical application of BEM to problems with subregions, however, is more complex. The complexity lies in the fact that different subregions should be treated as separate areas and the boundary of all of them should be discretized, and then they

have to be connected at interfaces between them into one global region.

Our previous research resulted in the creation and development of the alternative method compared to the classical BEM. The aim was to eliminate the discretization of both the area and the boundary. It was possible by analytical modifying the conventional boundary integral equations (BIE) and obtaining the parametric integral equation systems (PIES). The modification consisted of the direct definition of the geometry in BIE by proper functions defined in a parametric way. Therefore, the resulting PIES has eliminated the discretization of the boundary during its numerical solution. This became possible through including the geometry at the level of the mathematical formalism of the proposed PIES, and not at the level of the calculation of boundary integrals, as is in the traditional BEM.

Parametric integral equation systems (called the PIES method) have been obtained and tested for the entire spectrum of two-dimensional boundary problems modeled by differential equations: the Laplace [3], Helmholtz [4] and Navier-Lame [5]. Obtained results were much more accurate in relation to BEM. Therefore, it was encouraging to generalize PIES to three-dimensional problems modeled by these equations [6,7].

The purpose of this paper is to generalize PIES to three-dimensional problems modeled by the Laplace and Navier-Lame equations defined over regions which are piecewise homogeneous. To model the boundary geometry of subregions parametric surfaces were used [10,11]. Practically, they are declared in PIES using a small set of points, which in no way be equated to nodes of boundary element (as in BEM). In order to check the reliability and effectiveness of the proposed way of modeling subregions in PIES we have solved several examples. Verification was carried out in several stages. At the beginning, the Navier-Lame equation defined over elementary homogenous region was solved by PIES and the results were compared with known analytical solutions. Then the area was divided into subregions with the same properties. The reliability of the proposed way of modeling was verified by numerical tests. Obtained results (for subregions) were compared with exact solutions and obtained by means of PIES for the homogeneous region. We

also solved a problem for which there is no analytical solution and we compared the results with these obtained by known numerical method BEM realized by the commercial software BEASY.

## 2 A strategy for application of PIES to piecewise homogeneous regions

PIES are obtained by analytical modifying the classical boundary integral equations (BIE). This modification consisted of the inclusion of homogeneous areas defined by surface patches in the mathematical formalism of BIE. PIES for Laplace's equation is presented in [6], whilst for the Navier-Lame equations in [7] and [8] and takes the following form:

$$0.5u_i(v_1, w_1) = \sum_{j=1}^n \int_{v_{j-1}}^{v_j} \int_{w_{j-1}}^{w_j} \{ \bar{U}_{ij}^*(v_1, w_1, v, w) p_j(v, w) - \bar{P}_{ij}^*(v_1, w_1, v, w) u_j(v, w) \} J_j(v, w) dv dw, \quad (1)$$

where  $v_{l-1} < v_l < v_{l+1}$ ,  $w_{l-1} < w_l < w_{l+1}$ ,  $v_{j-1} < v < v_j$ ,  $w_{j-1} < w < w_j$  are the parameters of surfaces  $j = 1, 2, 3, \dots, n$ .

Integrands  $\bar{U}_{ij}^*$  and  $\bar{P}_{ij}^*$  for (1) in an explicit form are presented in [7] and [8].

In a very similar form as equation (1) is presented PIES for Laplace's equation. The detailed form of the function for this equation is presented in [6]. Integrands for this equation are presented in the following way:

$$\begin{aligned} \bar{U}_{ij}^*(v_1, w_1, v, w) &= \frac{1}{4\pi} \frac{1}{[\eta_1^2 + \eta_2^2 + \eta_3^2]^{0.5}}, \\ \bar{P}_{ij}^*(v_1, w_1, v, w) &= \frac{1}{4\pi} \frac{\eta_1 n_1 + \eta_2 n_2 + \eta_3 n_3}{[\eta_1^2 + \eta_2^2 + \eta_3^2]^{1.5}}, \end{aligned} \quad (1a)$$

where  $\eta_1 = P_i^{(1)}(v_1, w_1) - P_j^{(1)}(v, w)$ ,  $\eta_2 = P_i^{(2)}(v_1, w_1) - P_j^{(2)}(v, w)$ ,

$\eta_3 = P_i^{(3)}(v_1, w_1) - P_j^{(3)}(v, w)$ .

Kernels (1a) include in its mathematical formalism the shape of a closed boundary, created by means of functions  $P^{(i)}(v, w)$ ,  $i = 1, 2, 3$  that in the paper are represented by triangular or rectangular patches.

A characteristic feature of PIES, regardless of the differential equation, is that the corresponding integrands  $\bar{U}_{ij}^*$  and  $\bar{P}_{ij}^*$  allow to analytical include in their mathematical formalism the geometry modeled using parametric surface patches. The theoretical considerations assumed, that created by surfaces area is homogeneous and is characterized by constant parameters throughout the whole considered area. This gives possibility for solving boundary value problems only for unchanging properties (material constants) throughout the area. In many practical problems, it

appears that areas can be characterized by different parameters (properties). Resolving such problems by direct application of PIES is impossible. It would be possible only if, for such areas (piecewise homogeneous) we were able to analytically obtain solutions, which are integrands in PIES. Receipt of such functions in explicit form is very difficult and even impossible.

Therefore it became necessary to use a different strategy for solving problems defined over regions which are piecewise homogeneous. This strategy bases on the preliminary treatment of the area as the global area, composed of subregions with different properties. Then PIES is defined for each subregion separately, but with compatibility and equilibrium conditions between their interfaces. As a results, PIES for the whole region is obtained. Imposing the different characteristics for each subregion is reduced to consider relevant parameters in mentioned above integrands  $\bar{U}_{ij}^*$  and  $\bar{P}_{ij}^*$ , which correspond to individual subregions. Till now, three-dimensional analysis using PIES was restricted to homogenous areas.

## 3 Modeling piecewise homogenous regions in PIES

Modeling piecewise homogeneous regions by surfaces is analogous to the modeling of homogeneous areas [6,7]. The only problem is that in order to take into account the heterogeneity we have to define more patches. Properly modeled subregions are characterized by the fact that for each of them can be easily given different material properties. Figure 1 shows an elementary global area  $\Omega$  which is the sum of homogeneous subregions  $\Omega_1, \Omega_2, \Omega_3$ . Interfaces marked in gray are common for neighboring subregions. For each of these subregions in PIES can be easily prescribed different material constants. Defining the same material constants for all subregions reduces the problem to model the homogeneous area.

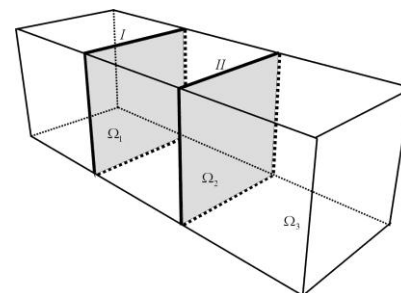


Fig. 1. The region composed of homogeneous subregions  $\Omega_1, \Omega_2, \Omega_3$ .

PIES represented by (1) can be applied separately for each subregion. In order to generalize PIES into the area which is piecewise homogeneous we have to add separately obtained PIESes for each subregion, together with compatibility conditions (3). Numerical solution of PIES for each subregion reduceses to solving the system of algebraic

equations. Therefore, the inclusion of compatibility conditions is most convenient at the level of systems of equations, which approximate PIES for individual subregions.

### 4 Numerical implementation of PIES for homogeneous subregions

Based on PIES represented by formula (1), the following systems of algebraic equations for each of the three subregions shown in Fig.1 can be written:

$$\begin{aligned} & \begin{bmatrix} H_1 & H_1^I \end{bmatrix} \begin{bmatrix} u_1 \\ u_1^I \end{bmatrix} = \begin{bmatrix} G_1 & G_1^I \end{bmatrix} \begin{bmatrix} p_1 \\ p_1^I \end{bmatrix}, \\ & \begin{bmatrix} H_2^I & H_2 & H_2^{II} \end{bmatrix} \begin{bmatrix} u_2^I \\ u_2 \\ u_2^{II} \end{bmatrix} = \begin{bmatrix} G_2^I & G_2 & G_2^{II} \end{bmatrix} \begin{bmatrix} p_2^I \\ p_2 \\ p_2^{II} \end{bmatrix}, \quad (2) \\ & \begin{bmatrix} H_3^{II} & H_3 \end{bmatrix} \begin{bmatrix} u_3^{II} \\ u_3 \end{bmatrix} = \begin{bmatrix} G_3^{II} & G_3 \end{bmatrix} \begin{bmatrix} p_3^{II} \\ p_3 \end{bmatrix}. \end{aligned}$$

In order to combine subregions into one region the compatibility conditions should be applied at interfaces (shown in Fig.1) *I* and *II* :

$$\begin{aligned} u_1^I &= u_2^I = u^I, u_2^{II} = u_3^{II} = u^{II}, \\ p_1^I &= -p_2^I = p^I, p_2^{II} = -p_3^{II} = p^{II}. \end{aligned} \quad (3)$$

Equations (2) with compatibility conditions (3) can be combined to form of the global system for three subregions:

$$\begin{aligned} & \begin{bmatrix} H_1 & H_1^I & -G_1^I & 0 & 0 & 0 & 0 \\ 0 & H_2^I & -G_2^I & H_2 & H_2^{II} & -G_2^{II} & 0 \\ 0 & 0 & 0 & 0 & H_3^{II} & -G_3^{II} & H_3 \end{bmatrix} \begin{bmatrix} u_1 \\ u^I \\ p^I \\ u_2 \\ u^{II} \\ p^{II} \\ u_3 \end{bmatrix} = \begin{bmatrix} G_1 & 0 & 0 \\ 0 & G_2 & 0 \\ 0 & 0 & G_3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}. \quad (4) \end{aligned}$$

By imposing the boundary conditions of the problem, the system (4) can be reordered as:

$$AX = B, \quad (5)$$

where *X* is a vector of unknown coefficients of series approximating boundary functions [7]. Then, having the

solution at the boundary we can obtain a solution at any point of subregions  $\Omega_1, \Omega_2, \Omega_3$  using the integral identity [7].

### 5 Verification of the reliability of modeling piecewise homogeneous regions

The reliability of presented modeling concept was verified and tested on examples with exact solutions. For this purpose, a linear elastic problem in the area shown in Fig. 2a and in the area modeled as the sum of three subregions  $\Omega_1, \Omega_2, \Omega_3$ , as shown in Fig. 2b, was considered. In order to verify the algorithm it was assumed that all three subregions have the same properties *E, ν* and proposed in the paper technique for piecewise homogeneous regions was applied to solve the problem. The results can be compared with the results obtained using widely tested PIES for homogeneous areas (Fig. 2a).

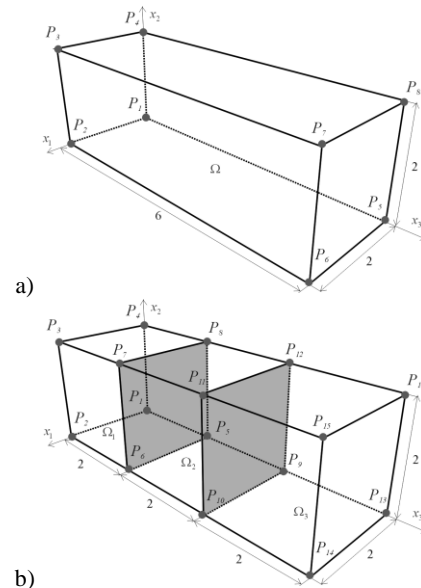


Fig. 2. Modeling in PIES by rectangular Coons surfaces: a) homogeneous area  $\Omega$ , b) piecewise homogeneous area  $\Omega_1, \Omega_2, \Omega_3$

Modeling areas in PIES is limited to corner points considering the area shown in Fig.2a and additionally corner points of interfaces between subregions (Fig.2b). The practical definition of these surfaces is reduced to posing 8 and 16 corner points. As a result, the geometry presented in Fig. 2a is created by 6, while the geometry from Fig. 2b, 18 connected Coons patches.

In order to numerical verification, for presented in Fig.2 area the Navier-Lame equations with displacements boundary conditions were solved. These conditions were determined based on the known analytical solutions:



$$\begin{aligned} u_1 &= (2x_1 + x_2 + x_3)/2, \\ u_2 &= (x_1 + 2x_2 + x_3)/2, \\ u_3 &= (x_1 + x_2 + 2x_3)/2. \end{aligned} \tag{6}$$

The values of relative error for displacements at selected points of the area for two variants of modeling are summarized in Table 1.

Table 1. Values of relative errors at selected area points

$(x_1, x_2, x_3)$	Relative error [%]	
	$\Omega$	$\Omega_1, \Omega_2, \Omega_3$
(1.0, 1.0, 0.5)	0.0014456	0.00251776
	0.0014456	0.00251776
	0.0089845	0.01575
(1.0, 1.0, 1.5)	0.000546507	0.000735146
	0.000546507	0.000735146
	0.00100551	0.00398909
(1.0, 1.0, 2.5)	0.00027155	0.00312324
	0.00027155	0.00312324
	0.00042753	0.011762
(1.0, 1.0, 3.5)	5.7268e-005	0.000740912
	5.7268e-005	0.000740912
	0.00016390	0.00842831
(1.0, 1.5, 4.5)	9.9359e-005	0.00238683
	9.9359e-005	0.00238683
	0.00011531	0.00797907
(1.0, 1.0, 5.5)	0.0010604	0.000366541
	0.0010604	0.000366541
	0.0028109	0.00418387

It is worth to emphasizing large similarity of the results for both variant of area modeling. Small relative errors for subregions confirm the reliability of PIES and the proposed way of modeling. These solutions have been obtained by solving respectively 24 and 72 algebraic equations.

In order to more accurately verify the modeling concept we considered more complex area than shown in Fig. 2. The hollow ball shown in Fig. 3a was modeled by Bezier patches, and the results were compared with analytical solution and these obtained by FEM known from the literature [9]. Therefore, the problem under consideration was modeled by Laplace's equation. Figures 3b,c show the cross section  $1.0 \leq r \leq 2.0$  of the hollow ball.

In order to test the software which realize PIES for piecewise homogeneous regions, the area from Fig. 3a,b as previously was divided into two closed subregions  $\Omega_1, \Omega_2$  with the common interface at  $r = 1.5$  (Fig. 3c). After imposing the potential boundary conditions ( $T_1 = 100^\circ C$  and  $T_2 = 200^\circ C$ ), assuming the same parameters and geometric dimensions the analytical solution for two variants of the geometry depends on the radius  $r$  and is presented by [9]:

$$T(r) = 300 - \frac{200}{r}. \tag{7}$$

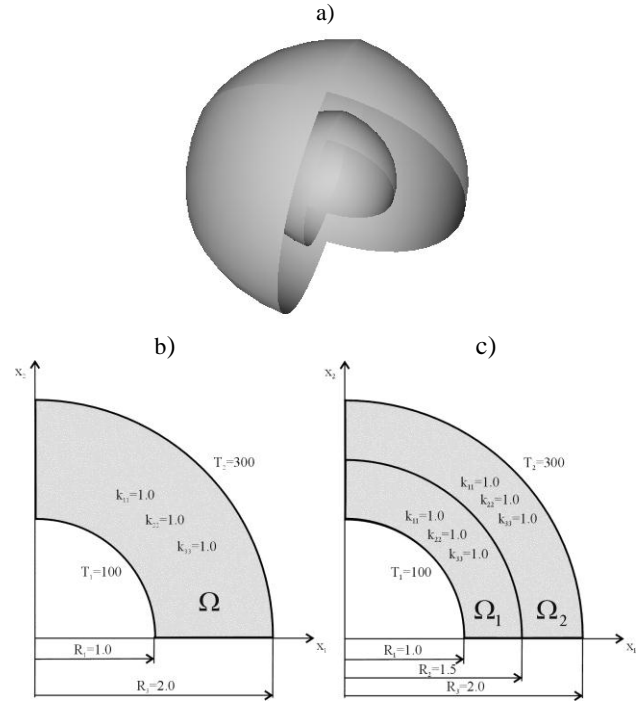


Fig. 3. The hollow ball: one region (a,b), two subregions (c)

In order to compare the effectiveness of modeling, Fig. 4 shows the modeling used in FEM and PIES. In the case of the classical FEM (Fig. 4b) a traditional element mesh for 1/8 part of the whole area was used. The mesh was made up of 64 finite elements. If we want to model the whole area number of elements would increase to 512 elements, declared using 800 nodes [9].

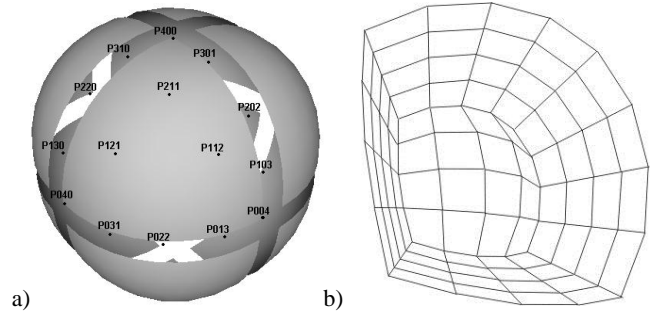


Fig. 4. Modeling of 3D area by Bezier triangular patches (a), FEM mesh for 1/8 part of the hollow ball (b).

To model the boundary in PIES in place of the finite elements are used shown in Fig. 4a triangular Bezier surface patches of the fourth degree, declared by 15 control points. The sphere of the ball is created in this case by combination of 8 mentioned patches. As a result, to declare the final geometry entering 16 surfaces declared by 112 control points is required. After the division of the input area into two subareas (Fig. 3c) the number of surfaces is doubled. The way of geometry modeling, in comparison with FEM, is simplified and is done by smaller set of points. Created in such way geometry is also directly used by the mathematical formalism

of PIES during the boundary problem solving, without any further division into finite or boundary elements.

Obtained by PIES numerical solutions for two variants of geometry show good agreement with analytical and FEM values, as shown in Table 2.

Table 2. Values of relative errors at selected area points for FEM and PIES

$r$	Relative error [%]		
	FEM [9]	PIES $\Omega$	PIES $\Omega_1, \Omega_2$
1.25	0.18	0.01	0.21
1.75	0.06	0.11	0.10

Based on the geometry shown in Fig. 5a was considered the problem of the region piecewise homogeneous, where subregions have different parameters. These parameters are as follows:  $k_{11} = k_{22} = k_{33} = 1$  for the internal region and  $k_{11} = k_{22} = k_{33} = 2$  for the external region.

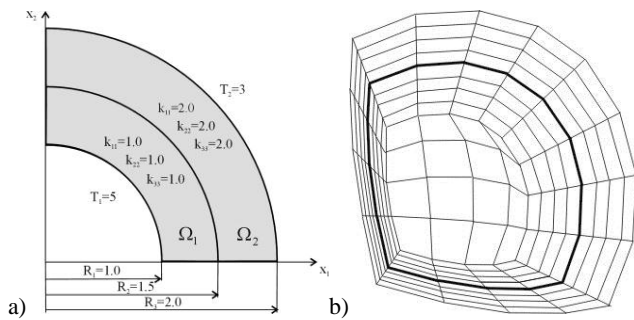


Fig. 5. The cross-section of the considered in PIES geometry (a), discretization of two subregions in FEM (b)

The exact solution in both subregions, with the prescribed as shown in Fig. 5a boundary conditions and geometrical dimensions is presented by the following function [9]:

$$T(r) = \frac{A}{r} + B. \tag{8}$$

The coefficients  $A$  and  $B$  are respectively 4.8 and 0.2 for the first and 2.4 and 1.8 for the second subregion.

The boundary geometry in PIES was modeled using 32 triangular Bezier patches, after defining 224 control points. This number of data is comparable with the number available in the literature for FEM [8], but only for 1/8 of the area. In this case 128 finite elements were used, and they were declared by 225 nodes. Thus the modeling of the entire area in case of PIES requires almost as much input as for FEM, but only to model 1/8 of the area. Table 3 summarizes the values obtained at certain points of the area modeled in PIES and FEM as mentioned above.

The advantage of the proposed PIES algorithm in comparison to the classical analysis using element methods is particularly the way of the modeling of the boundary geometry.

Table 3. Values of relative errors at selected area points for FEM and PIES

$r$	Relative error [%]	
	FEM	PIES
1.125	0.040	0.113
1.375	0.054	0.106
1.75	0.025	0.103
1.875	0.013	0.113

## 6 Numerical tests on examples

To verify the reliability of the presented way of subregions modeling by parametric patches two extreme examples are considered. The first example demonstrates the modeling of an elementary polyhedral domain described by only bilinear Coons surfaces, whereas the second shows the cylindrical domain generated by Bezier surfaces of the fourth degree. In the following part of the paper we considered more complex polyhedral shape without analytical solution and also the problem in which we should combine flat (first degree) and curved (higher degree) patches.

### 6.1 Example 1

Figure 6a shows a part of cylindrical domain containing a hole, which can be modeled by 2 bilinear Coons surfaces and 4 Bezier rectangular surfaces of the third degree. The domain is described unequivocally and accurately only by 48 control and corner points.

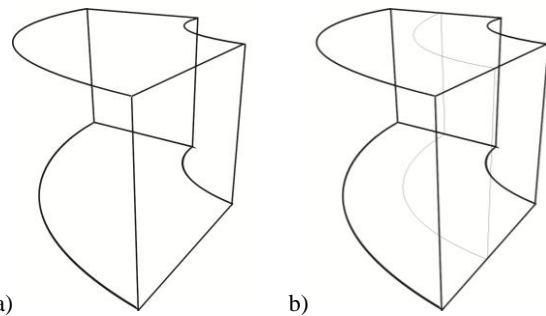


Fig. 6. The cylindrical domain: one region (a), two subregions (b)

The validation of the accuracy of solutions in PIES for generated by surface patches domain has been carried out for the classical Lamé problem of a hollow tube  $3.0 \leq r \leq 6.0$  subjected to an internal pressure  $p = 1Pa$ . Considering geometrical symmetry, for analysis is taken only 1/4 of the initial domain, as shown in Fig. 6a.

Simultaneously, in order to test the proposed concept, we also split the domain from Fig. 6a into two closed subregions  $\Omega_1, \Omega_2$  with the common interface for  $r = 4.5$  (Fig. 6b). After imposing the boundary conditions, identical material properties  $(E, \nu)$  and geometrical dimensions, the exact solution for both regions depends on the radius  $r$  and is given by the following formula [9]:

$$u_r = C_1 r + \frac{C_2}{r}, \tag{9}$$

where  $C_1 = 0.1733$ ,  $C_2 = 15.6$ .

Table 4. Values of relative errors at selected area points

$r$	Relative error [%]	
	PIES $\Omega$	PIES $\Omega_1, \Omega_2$
3.3234	0.90118	0.962047
3.46482	0.822381	0.873247
3.9598	0.594504	0.621568
4.24264	0.695472	1.54433
4.87904	0.853623	0.800289
5.16188	0.792898	0.788377
5.58614	0.863936	0.872808

The results summarized in Table 4 are in good agreement with the analytical values for both domains from Fig. 6.

### 6.2 Example 2

In the previous examples, the results obtained by PIES were compared with exact or numerical solutions taken from the literature. In this example, we solved problem using professional software BEASY and using authors software implementing PIES. A polyhedral region (shown in Fig. 7), composed of two symmetric subregions  $\Omega_1, \Omega_2$  and subjected to a uniform normal load  $p = 1Pa$ , is considered. Subregions are practically defined in PIES by 24 corner points, which model 20 rectangular Coons surface patches. Figure 6b illustrates the representation of the same geometry in BEM. The mesh has been generated using BEASY preprocessor and contains 784 quadrilateral boundary elements with 7929 nodes. As can be seen the number of input data to model the geometry in PIES (control points) is much less than the total number of nodes in presented BEM mesh.

We have considered subregions with different material properties, respectively  $E_1 = 1, \nu_1 = 0.3$  and  $E_2 = 2, \nu_2 = 0.3$ . Table 5 contains displacement values at selected area points obtained using PIES and BEM implemented by BEASY.

Comparing the results (presented in Table 5) it is possible to notice that we have obtained a comparative values of displacements for both methods with smaller input data needed to model the computational geometry in PIES.

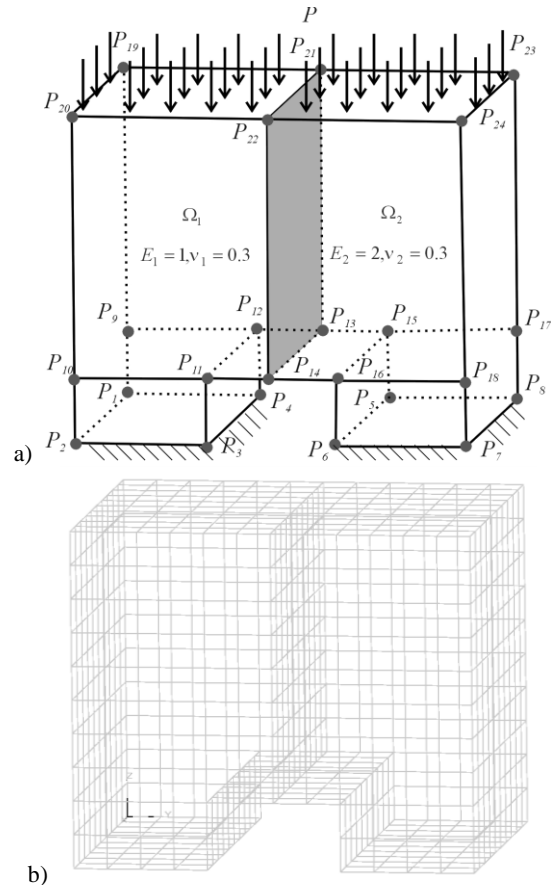


Fig. 7. The considered piecewise homogeneous region: a) modeled by Coons surfaces with posed boundary conditions, b) its discretization in BEM

## 7 Conclusions

The paper proposes modeling 3D piecewise homogeneous regions in boundary value problems solved by PIES. Modeling is characterized by defining boundaries of each subregion using the appropriate surface patches. Then, subregions were combined into one so-called global region with piecewise constant parameters. Such defined global region, after imposing the boundary conditions was used for solving boundary value problems by PIES.

The reliability of such modeling has been verified at several stages. At the beginning, the results obtained by PIES were compared with known analytical solutions for the same parameters in the whole area (without subregions). Then these results were obtained for the region which is piecewise homogeneous, but for all subregions were given the same material constants. The results were also compared with those known from the literature obtained by other numerical methods. The results confirmed the reliability of the applied modeling of such areas. This reliability was also confirmed by example where we considered the region with subregions characterized by different material constants. The results were compared with results obtained on the basis of professional software BEASY.

Table. 5. Solutions at selected area points

$x_1, x_2, x_3$	$BEASY_{\Omega_1, \Omega_2}$	$PIES_{\Omega_1, \Omega_2}$
5,1,6	-0.1596461E-07 -3.095816 -7.473452	0.00676691 -3.08303 -7.48228
5,2,6	-0.1628971E-07 -2.748688 -7.186318	0.00681254 -2.73575 -7.19854
5,3,6	-0.1660411E-07 -2.428255 -6.978831E	0.00898227 -2.41299 -6.99799
5,4,6	-0.1702067E-07 -2.174384 -6.799047	0.0132696 -2.15681 -6.82872
5,5,6	-0.1760653E-07 -2.020893 -6.556963	0.0173771 -2.00378 -6.59746
5,7,6	-0.1935037E-07 -1.882589 -5.633672	0.0121885 -1.86956 -5.67795
5,8,6	-0.2018353 -1.780836 -4.997009	0.00436346 -1.76742 -5.0351
5,9,6	-0.2106911E-07 -1.637185 -4.352804	-0.00282042 -1.62172 -4.38513
5,10,6	-0.2208607E-07 -1.468327 -3.755862	-0.00807627 -1.45112 -3.78555
5,11,6	-0.2319604E-07 -1.299597 -3.208792	-0.0121223 -1.28249 -3.23929

## 8 Acknowledgements

This work is funded by resources for science in the years 2010-2013 as a research project.

## 9 References

- [1] O. Zienkiewicz. "The Finite Element Methods", McGraw-Hill, London 1977.
- [2] C. A. Brebbia, J. C. F. Telles., L. C. Wrobel. "Boundary element techniques, theory and applications in engineering". Springer-Verlag, New York 1984.
- [3] E. Zieniuk. "Modelling and effective modification of smooth boundary geometry in boundary problems using B-spline curves". Engineering with Computers 23, pp. 39-48, 2007.
- [4] E. Zieniuk, A. Boltuc. "Bézier curves in the modeling of boundary geometries for 2D boundary problems defined by Helmholtz equation". Journal of Computational Acoustics 14/3, pp. 1-15, 2006.
- [5] E. Zieniuk, A. Boltuc. "Non-element method of solving 2D boundary problems defined on polygonal domains modeled by Navier equation". International Journal of Solids and Structures 43, pp. 7939-7958, 2006.
- [6] E. Zieniuk, K. Szerszen. „Linear Coons surfaces in the modeling of polygonal geometries in 3D boundary problems modeled by Laplace equation”, Archiwum Informatyki Teoretycznej i Stosowanej, 17/2, pp. 127-142, 2005 (in polish).
- [7] E. Zieniuk, A. Boltuc, K. Szerszen. "The effectiveness of PIES compared to BEM in the modelling of 3D polygonal problems defined by Navier-Lame equations", Proceedings of 2nd International Conference on Information and Communication, pp. 77-82, Amsterdam, Holland, 2011
- [8] E. Zieniuk, A. Boltuc, K. Szerszen. "Modeling complex homogeneous regions using surface patches and reliability verification for the Navier-Lame boundary problems", Proceedings of the 2012 International Conference on Scientific Computing, CSC 2012, unpublished.
- [9] E. Zieniuk, K. Szerszeń, A Boltuc. „PIES in solving 3D boundary problems modeled by Navier-Lame equations in polyhedral areas”, Modelowanie Inżynierskie 42, pp. 487-494, 2011 (in polish).
- [10] P. Kiciak. „Fundamentals of modeling curves and surfaces”, WNT, Warszawa, 2000 (in polish).
- [11] M. Mortenson. "Mathematics for Computer Graphics Applications: An Introduction to the Mathematics and Geometry of Cad/Cam", Geometric Modeling, Scientific Visualization, and Other CG Applications, Industrial Press, 1999.

# An S-System Analysis of the Sensitivity of Uric Acid Concentration to System Parameters in HGPRT Deficiency

Jack K. Horner  
PO Box 266  
Los Alamos NM 87544 USA  
email: jhorner@cybermesa.com

## Abstract

*Purine metabolism is a fundamental component of the production of nucleotides, the building blocks of DNA and RNA. In this pathway, hypoxanthine-guanine phosphoribosyltransferase (HGPRT) catalyzes the conversion of phosphoribosylpyrophosphate to inosine monophosphate and to phosphorylated guanosine. Mild HGPRT deficiency can result in gout and mild hyperuricemia. More severe HGPRT deficiency can result in Lesch-Nyhan syndrome, characterized by spasticity, choreoathetosis, mental retardation, self-mutilation, renal failure, and in the most severe cases, death. Here I present an S-system analysis of the (logarithmic gain) sensitivity of uric acid concentration in HGPRT deficiency to the parameters of the S-system model. The results suggests that there are no well leveraged targets for moderating uric acid concentration in the purine metabolism pathway that do not at the same time perturb primary DNA/RNA metabolism.*

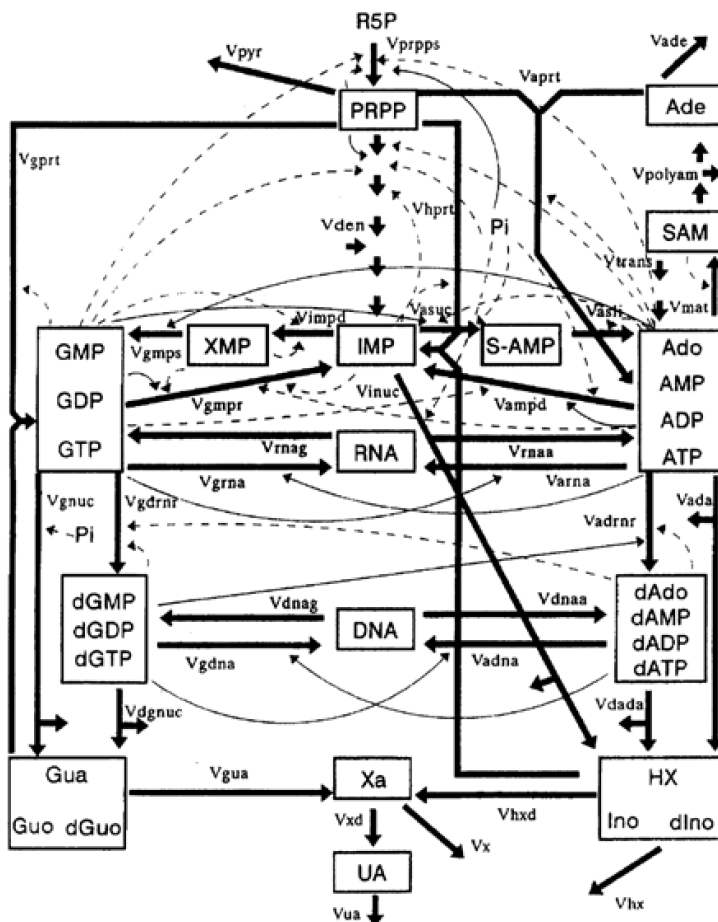
**Keywords:** purine metabolism, Lesch-Nyhan syndrome, HGPRT deficiency, S-system

## 1.0 Introduction

Purine metabolism (see Figure 1) is a fundamental component of the production of nucleotides, the building blocks of DNA and RNA. In this pathway, hypoxanthine-guanine phosphoribosyltransferase (HGPRT) catalyzes the conversion of phosphoribosylpyrophosphate (PRPP) to inosine monophosphate (IMP) and to phosphorylated guanosine. Mild HGPRT

deficiency can result in gout and mild hyperuricemia. More severe HGPRT deficiency can result in Lesch-Nyhan syndrome (LNS), characterized by spasticity, choreoathetosis, mental retardation, renal failure, self-mutilation, and in the most severe cases, death ([11]).

Can we systematically search for well leveraged targets for moderating uric acid concentrations in the purine metabolism pathway?



**Figure 1. Biochemical map of purine metabolism (adapted from [1]). Rectangles represent metabolites (some boxes abstract a collection of closely related metabolites (e.g., dGMP, dGDP, dGTP) whose differences are not essential to the model). Light solid arrows represent activation. Light dashed arrows represent inhibition. Curved heavy arrows entering or leaving the pathway indicate purine ring and ribose moieties that balance the stoichiometry of the system. Arrow labels represent flux in the direction of the arrow. The nomenclature in this diagram is defined in Tables 1 and 2.**

Curto et al. ([1], [2]) have extensively modeled purine metabolism and HGPRT deficiency using systems of ordinary differential equations (SODEs). For all but the simplest

SODEs, empirical surveys of system behavior are intractable due to the large size of the state space, and simulation is the only general system-characterization method available.

**Table 1. Purine metabolites and associated simulator variable names for Figure 1 (adapted from [1]).**

Simulator variable name	Biochemical name	Abbreviated name (in Figure 1)	Nominal initial concentration, microMolar
X1	Phosphoribosylpyrophosphate	PRPP	5
X2	Inosine_monophosphate	IMP	100
X3	Adenylosuccinate	S-AMP	0.2
X4	Adenosine	Ado	2500
X4	Adenosine_monophosphate	AMP	2500
X4	Adenosine_diphosphate	ADP	2500
X4	Adenosine_triphosphate	ATP	2500
X5	S-adenosyl-L-methionine	SAM	4
X6	Adenine	Ade	1
X7	Xanthosine_monophosphate	XMP	25
X8	Guanosine_monophosphate	GMP	400
X8	Guanosine_diphosphate	GDP	400
X8	Guanosine_triphosphate	GTP	400
X9	Deoxyadenosine	dAdo	6
X9	Deoxyadenosine_monophosphate	dAMP	6
X9	Deoxyadenosine_diphosphate	dADP	6
X9	Deoxyadenosine_triphosphate	dATP	6
X10	Deoxyguanosine_monophosphate	dGMP	3
X10	Deoxyguanosine_diphosphate	dGDP	3
X10	Deoxyguanosine_triphosphate	dGTP	3
X11	Ribonucleic_acid	RNA	28600
X12	Deoxyribonucleic_acid	DNA	5160
X13	Hypoxanthine	HX	10
X13	Inosine	Ino	10
X13	Deoxyinosine	dIno	10
X14	Xanthine	Xa	5
X15	Guanine	Gua	5
X15	Guanosine	Guo	5
X15	Deoxyguanosine	dGuo	5
X16	Uric_acid	UA	100
X17	Ribose-5-phosphate	R5P	18
X18	Phosphate	P_i	1400

**Table 2. Fluxes and enzymes for Figure 1 (adapted from [1]).**

Abbreviated flux name	Abbreviated enzyme name	Full name of enzyme that catalyzes reaction	E.C. enzyme identifier
vprpps	PRPPS	Phosphoribosylpyrophosphate synthetase	2.7.6.1.
vgprt	HGPRT	Hypoxanthine-guanine phosphoribosyltransferase	2.4.2.8.
vhprt	HGPRT	Hypoxanthine-guanine phosphoribosyltransferase	2.4.2.8.
vaprt	APRT	Adenine phosphoribosyltransferase	2.4.2.7.
vden	ATASE	`De novo synthesis' (Amidophosphoribosyltransferase)	2.4.2.14.
vpyr		`pyrimidine synthesis'	several enzymes
vasuc	ASUC	Adenylosuccinate synthetase	6.3.4.4.
vasli	ASLI	Adenylosuccinate lyase	4.3.2.2.
vimpd	IMPD	IMP dehydrogenase	1.1.1.205.
vgmps	GMPS	GMP synthetase	6.3.4.1.
vampd	AMPD	AMP deaminase	3.5.4.6.
vgmpr	GMPR	GMP reductase	1.6.6.8.
vtrans	MT	`transmethylation pathway' (Protein O-methyltransferase)	2.1.1.24.
vmat	MAT	Methionine adenosyltransferase	2.5.1.6.
vpolyam	SAMD	`Polyamine pathway' (S-adenosylmethionine decarboxylase)	4.1.1.50.
vade		`Adenine oxidation (xanthine oxidase)	1.2.1.37.

Abbreviated flux name	Abbreviated enzyme name	Full name of enzyme that catalyzes reaction	E.C. enzyme identifier
vinuc	5NUC	50-Nucleotidase	3.1.3.5.
vgnuc	5NUC	50-Nucleotidase	3.1.3.5.
varna	RNAP	RNA polymerase (from ATP)	2.7.7.6.
vgrna	RNAP	RNA polymerase (from GTP)	2.7.7.6.
vrnaa	RNAN	RNases (to AMP)	several enzymes
vrnag	RNAN	RNases (to GMP)	several enzymes
vdgnuc	3NUC	50(30) Nucleotidase	3.1.3.31.
vada	ADA	Adenosine deaminase	3.5.4.4.
vdada	ADA	Adenosine deaminase	3.5.4.4.
vadrnr	DRNR	Diribonucleotide reductase	1.17.4.1.
vgdrnr	DRNR	Diribonucleotide reductase	1.17.4.1.
vgua	GUA	Guanine hydrolase	3.5.4.3.
vadna	DNAP	DNA polymerase (from dATP)	2.7.7.7.
vgdna	DNAP	DNA polymerase (from dGTP)	2.7.7.7.
vdnaa	DNAN	DNases (to dAMP)	several enzymes
vdnag	DNAN	DNases (to dGMP)	several enzymes
vhx		`Hypoxanthine excretion'	Non-enzymatic step
vhxd	XD	Xanthine oxidase or xanthine dehydrogenase	1.2.1.37.
vxd	XD	Xanthine oxidase or xanthine dehydrogenase	1.2.1.37.
vx		`Xanthine excretion'	Non-enzymatic step
vua		`Uric acid excretion'	Non-enzymatic step

## 2.0 Method

The HGPRT deficiency model in [4] is an *S-system* ([3]). An *S-system* is a power-law-oriented, differential difference-equation SODE

each of whose dependent variables  $X_i$  is described by a kinetic equation of the form

$$dX_i / dt = \alpha_i \prod_j X_j^{g_{i,j}} - \beta_i \prod_j X_j^{h_{i,j}} \tag{Eq. 2.1}$$

where

- the left-hand side of Eq. 2.1 is the first derivative with respect to time of  $X_i$
- $i, j = 1, 2, 3, \dots, N$
- $\{X_i\}$  is the set of real-valued dependent variables of the system
- for any given  $X_i$ , only those independent and dependent variables  $X_j$  that have an action on  $X_i$  are included as factors in the products on the right-hand-side (RHS) of Eq. 2.1. The factors in the first term on the RHS of Eq. 2.1 correspond to just those entities that increase or inhibit the production of  $X_i$ ; the factors in the second term of the RHS of Eq. 2.1 correspond to just those entities that contribute to the consumption of  $X_i$ .
- $\alpha_i, \beta_i > 0$
- $g_{i,j}, h_{i,j}$  are real-valued



There is a natural mapping from a biochemical map,  $K$ , to equations that have the form of Eq 2.1. In particular, let  $K = \langle \{X_k\}, E \rangle$ ,  $E \in \{X_k\} \otimes \{X_k\}$ ,  $k = 1, 2, \dots, N$ , be a directed graph in which each distinct  $X_i \in \{X_k\}$  corresponds to a distinct dependent variable (e.g., the concentration of a distinct chemical species in the map), and  $w \in E$  if and only if  $w = (X_m, X_n)$  is a directed edge in  $K$ ,  $m \neq n = 1, 2, \dots, N$ .

$\alpha_i$  and  $\beta_i$  are called *generalized rate constants* (or just rate constants) for  $X_i$ , and  $g_{i_j}$  and  $h_{i_j}$  are called the *generalized kinetic orders* (or just kinetic orders) for  $X_i$ , on analogy with standard chemical kinetic theory. The subexpression  $i_j$  indicates the action of  $X_j$  on  $X_i$ .

An S-system has several desirable features, including the fact that it is fully characterized by its rate constants and kinetic orders, allowing us to comprehensively survey the system's (logarithmic gain) sensitivity to those parameters. Any SODE can be *recast*

([9],[10]) as an S-system without loss of accuracy or precision; the recasting, however, is not in general unique. In addition to biochemical systems, S-systems have been successfully used to model epidemics ([16]), forest diversification, and world dynamics ([15]).

The metabolic map shown in Figure 1 was translated to an S-system in *Mathematica* ([6]), then translated to the *Power Law Analysis and Simulation* (PLAS, [5]) language. The resulting model contains 16 dependent variables, two independent variables, and ~145 parameters (see Figure 2). In the model, HGPRT deficiency is represented as a factor,  $d$ , whose values lie in the interval [0.01, 0.99], that modulates the kinetic orders of several terms in the system. The larger the value of  $d$ , the more severe the deficiency.

---

```
// Phosphoribosylpyrophosphate
X1' = a1 X1^g1b1 X4^g1b4 X8^g1b8 X17^g1b17 X18^g1b18 >>
      - b1 X1^h1b1 X2^h1b2 X4^h1b4 X6^h1b6 X8^h1b8 X13^h1b13 X15^h1b15 X18^h1b18

// Inosine monophosphate
X2' = a2 X1^g2b1 X2^g2b2 X4^g2b4 X7^g2b7 X8^g2b8 X13^g2b13 X18^g2b18 >>
      - b2 X2^h2b2 X4^h2b4 X7^h2b7 X8^h2b8 X18^h2b18

// Adenylsuccinate
X3' = a3 X2^g3b2 X4^g3b4 X8^g3b8 X18^g3b18-b3 X3^h3b3 X4^h3b4

// Adenosine mono/di/tri phosphate
X4' = a4 X1^g4b1 X3^g4b3 X4^g4b4 X5^g4b5 X6^g4b6 X11^g4b11 >>
      - b4 X4^h4b4 X5^h4b5 X8^h4b8 X9^h4b9 X10^h4b10 X18^h4b18

// S-adenosyl-L-methionine
X5' = a5 X4^g5b4 X5^g5b5 - b5 X5^h5b5

// Adenine
X6' = a6 X5^g6b5 - b6 X1^h6b1 X4^h6b4 X6^h6b6

// Xanthine monophosphate
X7' = a7 X2^g7b2 X7^g7b7 X8^g7b8 - b7 X4^h7b4 X7^h7b7

// Guanosine mono/di/tri phosphate
X8' = a8 X1^g8b1 X4^g8b4 X7^g8b7 X8^g8b8 X11^g8b11 X15^g8b15 >>
      - b8 X2^h8b2 X4^h8b4 X7^h8b7 X8^0.133 X9^h8b9 X10^h8b10 X18^h8b18

// Deoxyadenosine mono/di/tri phosphate
X9' = a9 X4^g9b4 X9^g9b9 X10^g9b10 X12^g9b12 >>
      - b9 X9^h9b9 X10^h9b10

// Deoxyguanosine mono/di/tri phosphate
```

```

X10' = a10 X8^g10b8 X9^g10b9 X10^g10b10 X12^g10b12 >>
      - b10 X9^h10b9 X10^h10b10

// RNA
X11' = a11 X4^g11b4 X8^g11b8 - b11 X11

// Deoxyribonucleic acid
X12' = a12 X9^g12b9 X10^g12b10 >>
      - b12 X12

// Hypoxanthine/Inosine/Deoxyinosine
X13' = a13 X2^g13b2 X4^g13b4 X9^g13b9 X18^g13b18 >>
      - b13 X1^h13b1 X2^h13b2 X13^h13b13

// Xanthine
X14' = a14 X13^g14b13 X15^g14b15 - b14 X14^h14b14

// Guanine/Guanosine/Deoxygaunosine
X15' = a15 X8^g15b8 X10^g15b10 X18^g15b18 >>
      - b15 X1^h15b1 X8^h15b8 X15^h15b15

// Uric acid
X16' = a16 X14^g16b14 - b16 X16^h16b16

// INITIAL VALUES OF INDEPENDENT VARIABLES, microMolar

// Ribose-5-phosphate
X17 = 18.0

// Phosphate
X18 = 1400.0

```

**Figure 2. Parameterized equations of the HGPRT deficiency model. (Parameter values are not shown, but are available as noted in [17].) "'''' means first time derivative. "^" means exponentiation.**

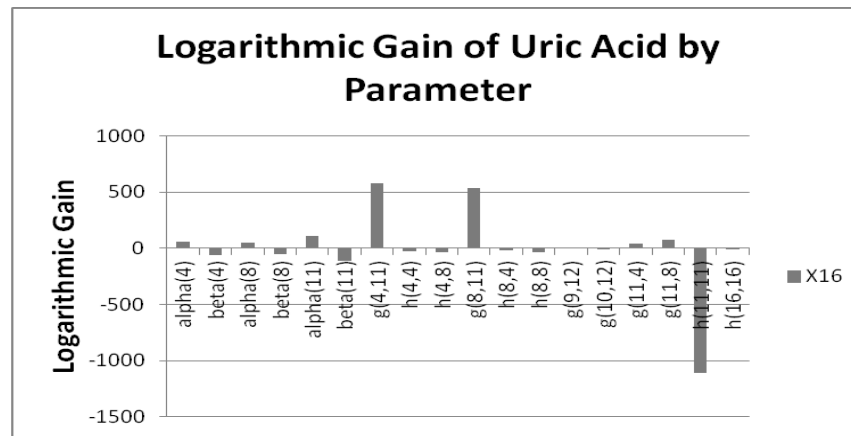
The model sketched in Figure 2 was executed to determine nominal system behavior. PLAS sensitivity analysis functions of that execution were used to determine the sensitivity of uric acid concentration to model parameters (gains less than 2 were excluded). All software was executed on a Dell Inspiron 545 with an Intel Core2 Quad CPU Q8200 (clocked @ 2.33 GHz) and 8.00 GB RAM, running under the *Windows Vista Home Premium* operating environment.

### 3.0 Results and discussion

Figure 3 shows the logarithmic gain (sensitivity) of uric acid to the model's parameters. Rate constants the absolute value of whose logarithmic gain is  $> \sim 10$  are potential targets for modulation of metabolic pathways ([18]). Taken together, Figure 1 and Figure 3 suggest that *there are no well leveraged targets in the purine metabolism pathway for moderating uric acid concentration that do not at the same time perturb primary DNA/RNA metabolism.*

These results illustrate the power of S-system modeling to help identify therapeutic targets.

The time-to-solution for all results on the platform described in Section 2.0 was less than 1 second.



**Figure 3. Logarithmic gain (sensitivity) of uric acid by S-system parameter in severe HGPRT deficiency ( $d = 0.99$ ). Gains with an absolute value less than 2 were excluded from the analysis.**

## 4.0 References

- [1] Curto R, Voit EO, Sorribas A, and Cascante M. Mathematical models of purine metabolism in man. *Mathematical Biosciences* 151 (1998), 1 - 49.
- [2] Curto R, Voit EO, Sorribas A, and Cascante M. Analysis of abnormalities in purine metabolism leading to gout and to neurological dysfunctions in man. *Biochemistry Journal* 329 (1998), 477 - 487.
- [3] Voit EO. *Computational Analysis of Biochemical Systems*. Cambridge. 2000. See especially Chapter 10.
- [4] File PurineSHGPRT.plc. A PLAS ([5]) script bundled with [3].
- [5] Ferreira AEN. *Power Law Analysis and Simulation (PLAS)* v1.2.120. <http://enzymology.fc.ul.pt/software.htm>. March 2011. Note: as of 16 January 2012, this link is broken; a copy of the tool is available on request from me.
- [6] Wolfram Research. *Mathematica* v8.0.1. 2011.
- [7] Liu JS. *Monte Carlo Strategies in Scientific Computing*. Springer. 2001.
- [8] Chung KL. *A Course in Probability Theory*. Third Edition. Academic Press. 2001.
- [9] Voit EO and Savageau MA. Equivalence between S-systems and volterra systems. *Mathematical Biosciences* 78 (1986), 47-55.
- [10] Voit EO. Recasting nonlinear models as S-systems. *Mathematical and Computer Modelling* 11 (1988), 140-145.
- [11] Harris JC. Disorders of purine and pyrimidine metabolism. In Kliegman RM, Behrman RE, Jenson HB, Stanton BF, eds. *Nelson Textbook of Pediatrics*. 18th ed. Saunders Elsevier. 2007.
- [12] von Neumann J. Various techniques used in connection with random digits. *National Bureau of Standards Applied Mathematics Series* 12 (1951), 36-38.
- [13] Rubinstein RY. *Simulation and the Monte Carlo Method*. Wiley. 1981.
- [14] Robert CP and Casella G. *Monte Carlo Statistical Methods*. Second Edition. Springer. 2004.
- [15] Horner JK. A dynamical implementation of the Stockholm Resilience Centre safe operating space (SOS) model. *Proceedings of the 2010 International Conference on Scientific Computing*. CSREA Press. 2010. 236-242.
- [16] Horner JK. A power-law model of dengue haemorrhagic fever epidemics in Thailand. *Proceedings of the 2004 Conference on Mathematical and Engineering Techniques in Medicine and Biological Sciences*. CSREA Press. 2004. 410-415.
- [17] The source code for the application is available on request from the author.
- [18] Torres NV, Voit EO, Glez-Alcón C, and Rodriguez F. An indirect optimization method for biochemical systems: description of method and application to the maximization of the rate of ethanol, glycerol, and carbohydrate production in *Saccharomyces cerevisiae*. *Biotechnology and Bioengineering* 55 (1997), 758-772.

# Modeling complex homogeneous regions using surface patches and reliability verification for Navier-Lame boundary problems

E. Zieniuk, A. Boltuc, K. Szerszen

Faculty of Mathematics and Computer Science, University of Bialystok,  
Sosnowa 64, 15-887 Bialystok, Poland

**Abstract** - *The paper presents a strategy for using different parametric surface patches for the modeling (approximation) of complex 3D boundary geometries in boundary value problems. Created by mentioned technique boundary is directly included in developed by the authors parametric integral equation systems (PIES) to solve boundary problems of linear elasticity. Included examples confirm the reliability of the approach and high accuracy of numerical solutions for boundary problems governed by the Navier-Lame equations.*

**Keywords** - computer modeling and simulation, parametric integral equation systems (PIES), Bézier surfaces, linear elasticity, boundary problems

## 1 Introduction

Modeling areas for boundary problems in element methods reduces to dividing the area into finite elements in the finite element method FEM [1,2] and into boundary elements in the boundary element method BEM [3,4]. These elements are used for the simultaneous modeling of areas and approximate solutions. To improve the accuracy of solutions we have to increase the number of elements. It makes sense from the standpoint of improving the accuracy of the solution, but it is pointless taking into account the accuracy of modeling the considered area. It comes from the fact that even a small number of these elements can be sufficient for very accurate modeling (approximating) the area.

For many years, we have used the parametric integral equation systems (PIES) for solving boundary value problems. In PIES, the approximation of the boundary is independent of the approximation of boundary functions, hence there is the possibility of effective modeling the boundary shape by selecting appropriate surface patches. Till now, PIES was mainly used to solve two-dimensional potential boundary problems modeled by a variety of differential equations such as Laplace, Helmholtz, Poisson and Navier-Lame.

To define the boundary in 2D problems we have used linear segments [5], Bézier [6], B-spline [7] and Hermite [8]

curves of the first and third degree. Using curves of the third degree we can very easily model the geometry in a continuous way, and to its practical definition a small number of control points is posed. By means of curves of the first degree we can model any polygonal area using only corner points. Finally, the number of input data in PIES is significantly lower than in FEM and BEM and the boundary is defined in a continuous way.

The resulting efficiency of PIES in the case of solving two-dimensional potential boundary problems was encouraging to generalize the method to 3D problems defined by various differential equations. Till now, we have considered only 3D problems modeled by the Laplace [9] and Helmholtz [10] equations. Current research focuses on the use of PIES for 3D linear elastic areas modeled by the Navier-Lame equations. We have done preliminary tests concerned with verification of PIES for very elementary shapes.

The aim of this paper is to analyze the reliability and effectiveness of modeling complex 3D geometries by surface patches in PIES for boundary value problems of linear elasticity. For this purpose, triangular and rectangular parametric surfaces were tested in terms of their direct application to the representation of the boundary in PIES. Both flat surfaces of the first degree, as well as curved patches of higher degrees were examined.

## 2 Bézier surface patches for boundary representation

Used in the paper parametric Bézier surface patches are an extension of well-known from computer graphics parametric Bézier curves. A common feature of both curves and surfaces is declaration of their shape using a small set of control points. Considering the Bézier surface the shape is defined by an array of control points. Presented and applied in the paper modeling uses triangular and rectangular Bézier patches.

Triangular Bézier surfaces of the  $n$  degree are declared by a set of  $0.5(n+1)(n+2)$  control points  $P_{ijk}$  and mathematically described by the following formula [11]:

$$P(v, w) = \sum_{\substack{i, j, k \geq 0 \\ i+j+k=n}} P_{ijk} B_{ijk}^n(v, w, 1-v-w), \quad (1)$$

$$0 \leq v, w \leq 1, v+w \leq 1,$$

with basis functions:

$$B_{ijk}^n(v, w, 1-v-w) = \frac{n!}{i!j!k!} v^i w^j (1-v-w)^k. \quad (2)$$

Figure 1a shows triangular patch of the third degree defined by 10 control points. By moving these points, we can effectively modify the shape of the surface.

A rectangular Bézier surface is described by an array of  $n \times m$  control points  $P_{ij}$  and is written mathematically by [11]:

$$P(v, w) = \sum_{i=0}^m \sum_{j=0}^n P_{ij} B_{im}(v) B_{jn}(w), \quad 0 \leq v, w \leq 1, \quad (3)$$

with basis functions represented by Bernstein polynomials as follows:

$$B_{im}(v) = \frac{m!}{i!(m-i)!} v^i (1-v)^{m-i}, \quad (4)$$

$$B_{jn}(w) = \frac{n!}{j!(n-j)!} w^j (1-w)^{n-j}.$$

In this paper, we deal with Bézier rectangular patches with  $n = m = 1$  and  $n = m = 3$ , called the first and third degree. A rectangular patch of the first degree takes the form of a flat surface defined by four corner points. A bicubic Bézier patch of the third degree is defined using 16 control points and is shown in Fig. 1b.

Defining the proper geometry in PIES reduces to connecting independent surface patches, so as to finally they model whole boundary of the problem under consideration. Examples of such created areas are shown in the next section of this paper. Declared in this way geometries are directly included into the mathematical formalism of PIES.

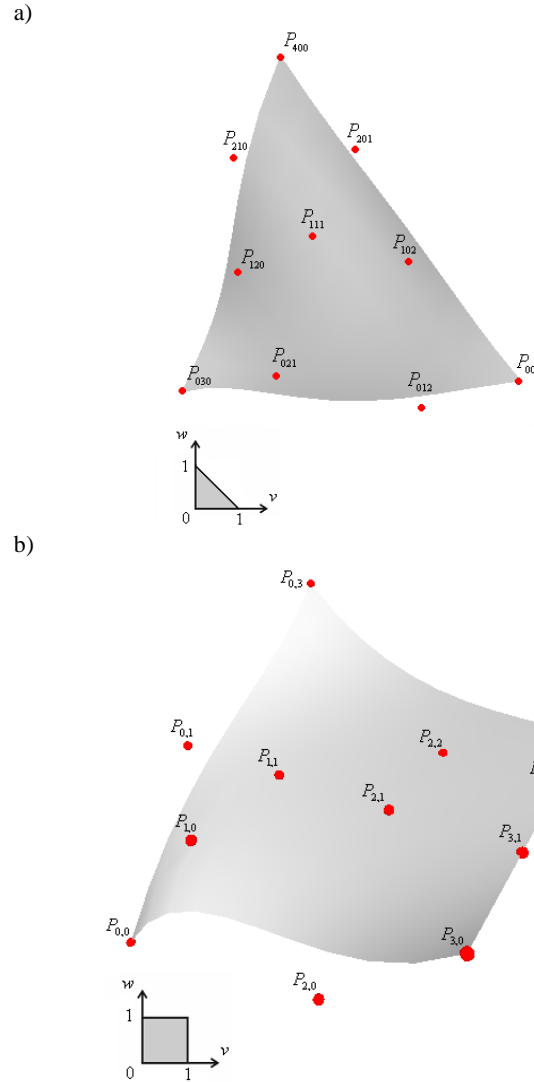


Fig. 1. Visualization of a) triangular and b) rectangular Bézier patches of the third degree with their control points

### 3 Modeling curvilinear geometries by control points

To model curvilinear geometries in 3D boundary value problems we have used triangular and rectangular Bézier patches discussed in the previous section. Practical modeling mentioned areas is reduced to its creation by a combination of surfaces of the lower degree. Such strategy was used in presented in the paper researches.

Figure 2 shows an axially symmetric torus generated in PIES by 16 rectangular Bézier patches of the third degree and a total number of 174 control points. The closed surface of a torus was formed after connecting the outer edges of surfaces. Such modeled torus directly represents the boundary geometry in PIES.

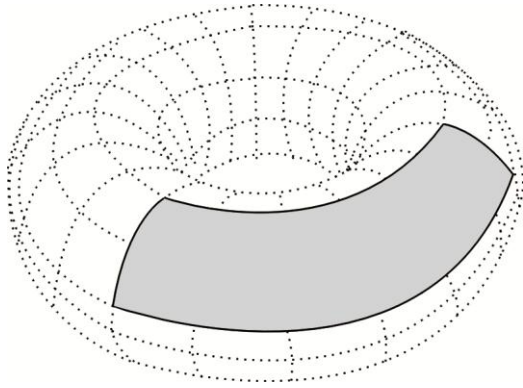


Fig. 2. The torus with marked the one of 16 rectangular patches used for its modeling

We can generalize the strategy used in the above example to modeling more complex shapes, for example the multi-connected area shown in Fig. 3h. In such cases it is necessary to connect flat (first degree) and curved (higher degree) surfaces to model the considered area. The use of different surfaces is presented in Figures 3a-g.

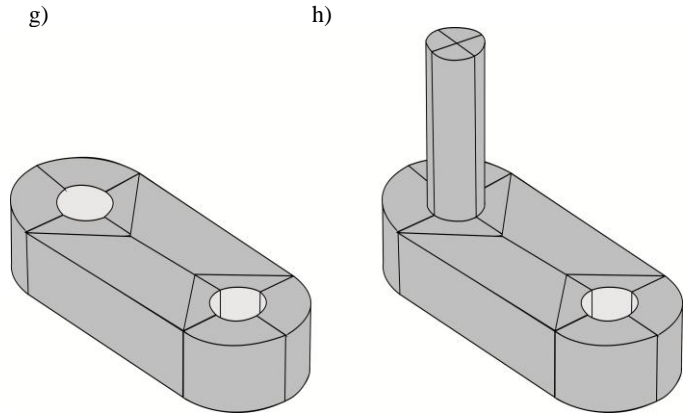
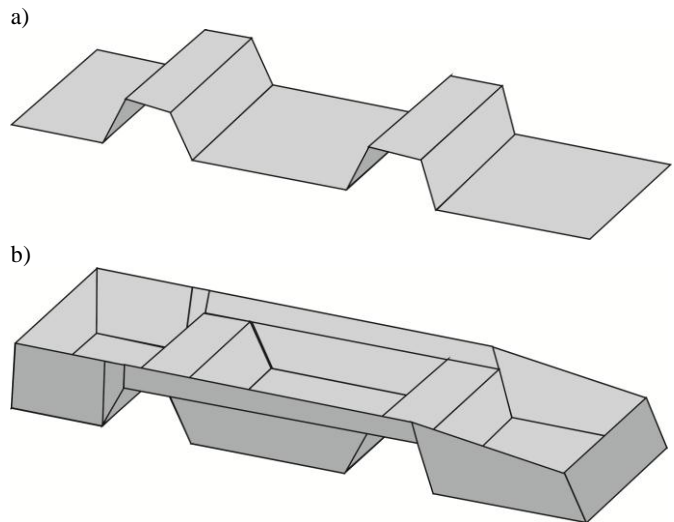
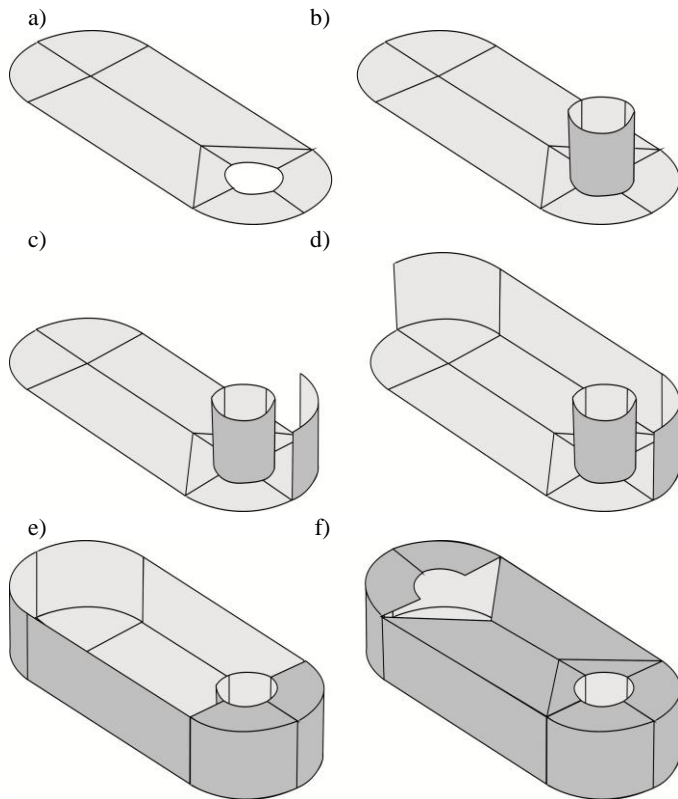


Fig.3. Further stages of defining the multi-connected domain: a) the base declared by rectangular and triangular surfaces, b-d) side faces modeled by rectangular surfaces of the third degree, e-g) defining next segments of the boundary, h) the complete geometry

Finally, the boundary is modeled by the following Bézier patches: 6 rectangular of the first degree, 24 rectangular and 6 triangular of the third degree. These patches are defined by 236 control points and they create the boundary geometry presented in Fig.3h.

Figures 4a,b,c show the following stages of modeling even more complex area shown in Fig. 4d. This area has been generated for the numerical analysis of PIES and verification of the reliability of proposed in the paper modeling concept. Its creation requires introduction of respectively: 26 rectangular patches of the first and third degree and 4 triangular of the third degree.



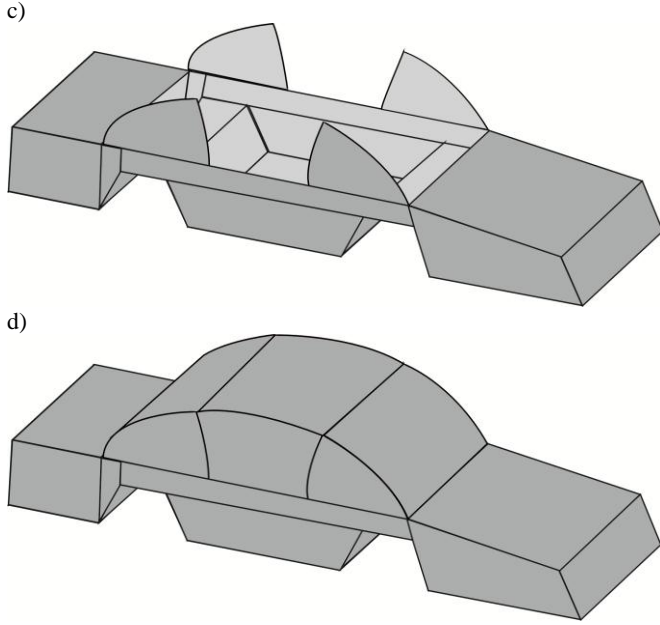


Fig. 4. The stages of defining sample automobile domain by Bézier patches

The created by patches area has been directly integrated with the mathematical formalism of PIES, which is used for numerical solution of boundary problems. As can be seen in Fig.4, this is the smallest number of patches (of the considered type) required to model the area under consideration. Taking into account the number of surfaces we can state that modeling is effective. It is also implemented without any further division of the area or boundary into elements as in FEM and BEM.

#### 4 PIES for the Navier-Lame equations in domains by Bézier surface patches

The presented way of modeling is directly used in parametric integral equation systems (PIES). PIES for the three-dimensional Navier-Lame equations were obtained as a result of analytical modification of the classical boundary integral equation (BIE). An applied modification strategy is the generalization of modification presented in [5], used for 2D problems. General form of PIES (for any shape of the boundary) is presented by the following formula [12]:

$$0.5\mathbf{u}_l(v_1, w_1) = \sum_{j=1}^n \int_{v_{j-1}}^{v_j} \int_{w_{j-1}}^{w_j} \{ \bar{U}_{ij}^*(v_1, w_1, v, w) \mathbf{p}_j(v, w) - \bar{P}_{ij}^*(v_1, w_1, v, w) \mathbf{u}_j(v, w) \} J_j(v, w) dv dw, \quad (5)$$

where

$v_{l-1} < v_1 < v_l, w_{l-1} < w_1 < w_l, v_{j-1} < v < v_j, w_{j-1} < w < w_j, l = 1, 2, 3, \dots, n,$  and  $n$  – is the number of parametric patches that create 3D boundary geometry.

Integrands  $\bar{U}_{ij}^*$  and  $\bar{P}_{ij}^*$  in (5) can be expressed in a following matrix form:

$$\bar{U}_{ij}^*(v_1, w_1, v, w) = \frac{(1+\nu)}{8\pi(1-\nu)E\eta} \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ U_{21} & U_{22} & U_{23} \\ U_{31} & U_{32} & U_{33} \end{bmatrix}, \quad (6)$$

where

$$U_{11} = (3-4\nu) + \frac{\eta_1^2}{\eta^2}, U_{12} = \frac{\eta_1\eta_2}{\eta^2}, U_{13} = \frac{\eta_1\eta_3}{\eta^2},$$

$$U_{21} = \frac{\eta_2\eta_1}{\eta^2}, U_{22} = (3-4\nu) + \frac{\eta_2^2}{\eta^2}, U_{23} = \frac{\eta_2\eta_3}{\eta^2},$$

$$U_{31} = \frac{\eta_3\eta_1}{\eta^2}, U_{32} = \frac{\eta_3\eta_2}{\eta^2}, U_{33} = (3-4\nu) + \frac{\eta_3^2}{\eta^2},$$

and

$$\bar{P}_{ij}^*(v_1, w_1, v, w) = \frac{-1}{8\pi(1-\nu)\eta^2} \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix}, \quad (7)$$

where

$$P_{11} = ((1-2\nu) + 3\frac{\eta_1^2}{\eta^2}) \frac{\partial \eta}{\partial n},$$

$$P_{12} = 3\frac{\eta_1\eta_2}{\eta^2} \frac{\partial \eta}{\partial n} - (1-2\nu) \frac{\eta_1 n_2 - \eta_2 n_1}{\eta},$$

$$P_{13} = 3\frac{\eta_1\eta_3}{\eta^2} \frac{\partial \eta}{\partial n} - (1-2\nu) \frac{\eta_1 n_3 - \eta_3 n_1}{\eta},$$

$$P_{21} = 3\frac{\eta_2\eta_1}{\eta^2} \frac{\partial \eta}{\partial n} - (1-2\nu) \frac{\eta_2 n_1 - \eta_1 n_2}{\eta},$$

$$P_{22} = ((1-2\nu) + 3\frac{\eta_2^2}{\eta^2}) \frac{\partial \eta}{\partial n},$$

$$P_{23} = 3\frac{\eta_2\eta_3}{\eta^2} \frac{\partial \eta}{\partial n} - (1-2\nu) \frac{\eta_2 n_3 - \eta_3 n_2}{\eta},$$

$$P_{31} = 3\frac{\eta_3\eta_1}{\eta^2} \frac{\partial \eta}{\partial n} - (1-2\nu) \frac{\eta_3 n_1 - \eta_1 n_3}{\eta},$$

$$P_{32} = 3\frac{\eta_3\eta_2}{\eta^2} \frac{\partial \eta}{\partial n} - (1-2\nu) \frac{\eta_3 n_2 - \eta_2 n_3}{\eta},$$

$$P_{33} = ((1-2\nu) + 3\frac{\eta_3^2}{\eta^2}) \frac{\partial \eta}{\partial n}.$$

Kernels (6,7) include in its mathematical formalism the shape of the closed boundary, created by means of functions  $P^{(i)}(v, w)$ ,  $i=1,2,3$  that in this paper are represented by triangular (1) or rectangular (3) patches through the following relations:

$$\eta_1 = P_j^{(1)}(v, w) - P_i^{(1)}(v_1, w_1), \quad \eta_2 = P_j^{(2)}(v, w) - P_i^{(2)}(v_1, w_1), \quad (8)$$

$$\eta_3 = P_j^{(3)}(v, w) - P_i^{(3)}(v_1, w_1) \quad \text{and} \quad \eta = [\eta_1^2 + \eta_2^2 + \eta_3^2]^{0.5}.$$

These relations give possibility for creation any boundary geometry by joining parametric patches directly in PIES. Three sample geometries are shown in Fig. 2-4.

Formula (5) requires the Jacobian  $J_j(v, w)$  and surface normal vectors  $n_1(v, w), n_2(v, w), n_3(v, w)$ , which are derived analytically from equations (1,3) in the following way:

$$J_j(v, w) = [A_1^2(v, w) + A_2^2(v, w) + A_3^2(v, w)]^{0.5}, \quad (9)$$

and

$$n_1(v, w) = \frac{A_1(v, w)}{J_j(v, w)}, \quad n_2(v, w) = \frac{A_2(v, w)}{J_j(v, w)},$$

$$n_3(v, w) = \frac{A_3(v, w)}{J_j(v, w)} \quad (10)$$

where

$$A_1(v, w) = \frac{\partial P_j^{(2)}(v, w)}{\partial w} \frac{\partial P_j^{(3)}(v, w)}{\partial v} - \frac{\partial P_j^{(2)}(v, w)}{\partial v} \frac{\partial P_j^{(3)}(v, w)}{\partial w},$$

$$A_2(v, w) = \frac{\partial P_j^{(3)}(v, w)}{\partial w} \frac{\partial P_j^{(1)}(v, w)}{\partial v} - \frac{\partial P_j^{(3)}(v, w)}{\partial v} \frac{\partial P_j^{(1)}(v, w)}{\partial w},$$

$$A_3(v, w) = \frac{\partial P_j^{(1)}(v, w)}{\partial w} \frac{\partial P_j^{(2)}(v, w)}{\partial v} - \frac{\partial P_j^{(1)}(v, w)}{\partial v} \frac{\partial P_j^{(2)}(v, w)}{\partial w}.$$

## 5 Numerical solution of PIES

Application of PIES for solving 2D and 3D boundary problems, taking into account presented modeling technique, allows eliminating the need for a discretization at the level of both the boundary modeling and the boundary function approximation. Boundary functions, which are vectors of displacements  $u_j(v, w)$  and stresses  $p_j(v, w)$  on each  $j$  surface can be approximated by the following series:

$$u_j(v, w) = \sum_{p=0}^N \sum_{r=0}^M u_j^{(pr)} T_j^{(p)}(v) T_j^{(r)}(w), \quad (11)$$

$$p_j(v, w) = \sum_{p=0}^N \sum_{r=0}^M p_j^{(pr)} T_j^{(p)}(v) T_j^{(r)}(w), \quad (12)$$

where  $u_j^{(pr)}, p_j^{(pr)}$  are unknown coefficients, while  $T_j^{(p)}(v), T_j^{(r)}(w)$  are Chebyshev polynomials. After substituting (11,12) to (5) and writing down at proper collocation points [13] we obtain a system of linear algebraic equations with respect to unknown coefficients. Coefficients  $u_j^{(pr)}$  or  $p_j^{(pr)}$  can be obtained by solving mentioned system.

## 6 Solutions in the domain

After solving PIES only the solution on the boundary is obtained, and it is represented by approximation series (11) or (12). The integral identity is used to obtain solutions in the area on the basis of solutions from the boundary and is formulated in the same way as for 2D problems [5]. In order to do it solutions obtained on the boundary are used. This identity is presented in the following form [12]:

$$u(x) = \sum_{j=1}^n \int_{v_{j-1}}^{v_j} \int_{w_{j-1}}^{w_j} \{ \hat{U}_j^*(x, v, w) p_j(v, w) - \hat{P}_j^*(x, v, w) u_j(v, w) u(x) \} J(v, w) dv dw \quad (13)$$

while integrands can be expressed by:

$$\hat{U}_j^*(x, v, w) = \frac{(1+v)}{8\pi(1-v)E\bar{r}} \begin{bmatrix} \hat{U}_{11} & \hat{U}_{12} & \hat{U}_{13} \\ \hat{U}_{21} & \hat{U}_{22} & \hat{U}_{23} \\ \hat{U}_{31} & \hat{U}_{32} & \hat{U}_{33} \end{bmatrix}, \quad (14)$$

where

$$\hat{U}_{11} = (3-4\nu) + \frac{\bar{r}_1^2}{\bar{r}^2}, \quad \hat{U}_{12} = \frac{\bar{r}_1 \bar{r}_2}{\bar{r}^2}, \quad \hat{U}_{13} = \frac{\bar{r}_1 \bar{r}_3}{\bar{r}^2},$$

$$\hat{U}_{21} = \frac{\bar{r}_2 \bar{r}_1}{\bar{r}^2}, \quad \hat{U}_{22} = (3-4\nu) + \frac{\bar{r}_2^2}{\bar{r}^2}, \quad \hat{U}_{23} = \frac{\bar{r}_2 \bar{r}_3}{\bar{r}^2},$$

$$\hat{U}_{31} = \frac{\bar{r}_3 \bar{r}_1}{\bar{r}^2}, \quad \hat{U}_{32} = \frac{\bar{r}_3 \bar{r}_2}{\bar{r}^2}, \quad \hat{U}_{33} = (3-4\nu) + \frac{\bar{r}_3^2}{\bar{r}^2},$$

and

$$\hat{P}_j^*(x, v, w) = \frac{-1}{8\pi(1-v)\bar{r}^2} \begin{bmatrix} \hat{P}_{11} & \hat{P}_{12} & \hat{P}_{13} \\ \hat{P}_{21} & \hat{P}_{22} & \hat{P}_{23} \\ \hat{P}_{31} & \hat{P}_{32} & \hat{P}_{33} \end{bmatrix}, \quad (15)$$

where



$$\begin{aligned} \hat{P}_{11} &= ((1-2\nu) + 3\frac{\bar{r}_1^2}{\bar{r}^2}) \frac{\partial \bar{r}}{\partial n}, \\ \hat{P}_{12} &= 3\frac{\bar{r}_1\bar{r}_2}{\bar{r}^2} \frac{\partial \bar{r}}{\partial n} - (1-2\nu) \frac{\bar{r}_1 n_2 - \bar{r}_2 n_1}{\bar{r}}, \\ \hat{P}_{13} &= 3\frac{\bar{r}_1\bar{r}_3}{\bar{r}^2} \frac{\partial \bar{r}}{\partial n} - (1-2\nu) \frac{\bar{r}_1 n_3 - \bar{r}_3 n_1}{\bar{r}}, \\ \hat{P}_{21} &= 3\frac{\bar{r}_2\bar{r}_1}{\bar{r}^2} \frac{\partial \bar{r}}{\partial n} - (1-2\nu) \frac{\bar{r}_2 n_1 - \bar{r}_1 n_2}{\bar{r}}, \\ \hat{P}_{22} &= ((1-2\nu) + 3\frac{\bar{r}_2^2}{\bar{r}^2}) \frac{\partial \bar{r}}{\partial n}, \\ \hat{P}_{23} &= 3\frac{\bar{r}_2\bar{r}_3}{\bar{r}^2} \frac{\partial \bar{r}}{\partial n} - (1-2\nu) \frac{\bar{r}_2 n_3 - \bar{r}_3 n_2}{\bar{r}}, \\ \hat{P}_{31} &= 3\frac{\bar{r}_3\bar{r}_1}{\bar{r}^2} \frac{\partial \bar{r}}{\partial n} - (1-2\nu) \frac{\bar{r}_3 n_1 - \bar{r}_1 n_3}{\bar{r}}, \\ \hat{P}_{32} &= 3\frac{\bar{r}_3\bar{r}_2}{\bar{r}^2} \frac{\partial \bar{r}}{\partial n} - (1-2\nu) \frac{\bar{r}_3 n_2 - \bar{r}_2 n_3}{\bar{r}}, \\ \hat{P}_{33} &= ((1-2\nu) + 3\frac{\bar{r}_3^2}{\bar{r}^2}) \frac{\partial \bar{r}}{\partial n}. \end{aligned}$$

Above formulas are analogous to those previously described in (6,7). The difference lies in the following expressions:

$$\begin{aligned} \bar{r}_1 &= P_j^{(1)}(v, w) - x_1, \quad \bar{r}_2 = P_j^{(2)}(v, w) - x_2, \\ \bar{r}_3 &= P_j^{(3)}(v, w) - x_3, \quad \bar{r} = [\bar{r}_1^2 + \bar{r}_2^2 + \bar{r}_3^2]^{0.5}, \end{aligned} \quad (16)$$

where beyond surface patches that define the geometry are the coordinates of points in the domain  $\mathbf{x} \equiv \{x_1, x_2, x_3\}$ , at which we are interested in the solution.

### 7 Numerical examples

In order to practical verification of the presented way of modeling numerical tests were carried out. We considered boundary problems defined over linear elastic bodies. Verification concerned the examination of the similarity of the numerical results with available in the literature analytical solutions. Imposed displacement boundary conditions for different areas were obtained by the following known analytical solutions [14,15]:

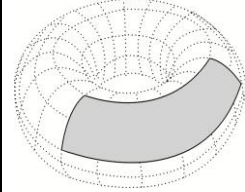
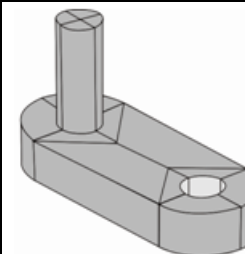
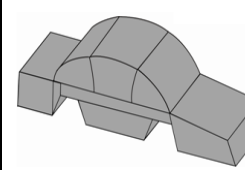
$$\begin{aligned} u_1 &= (2x_1 + x_2 + x_3) / 2, \\ u_2 &= (x_1 + 2x_2 + x_3) / 2, \\ u_3 &= (x_1 + x_2 + 2x_3) / 2, \end{aligned} \quad (17)$$

and

$$\begin{aligned} u_1 &= x_2^3 - 3x_2x_3^2, \\ u_2 &= x_3^3 - 3x_3x_1^2, \\ u_3 &= x_1^3 - 3x_1x_2^2. \end{aligned} \quad (18)$$

On the basis of solutions (17,18) we can easily calculate derivatives with respect to relevant variables. Then, having derivatives and proper expressions of elasticity we can calculate loads on the boundary corresponding to the given displacements. The resulting boundary conditions give the possibility to compare them with the result obtained by numerical solution of PIES. Numerical tests were performed for three areas shown in Figures 2-4. Obtained on the boundary solutions in comparison with the exact results with  $L_2$  error norms are shown in Table I.

Table. 1.  $L_2$  error norms for three geometries and functions (17,18)

Geometry	Error norm $L_2$	
	$\ e\ _p$ (17)	$\ e\ _p$ (18)
	$\ e\ _{p_1} = 0.1715\%$ $\ e\ _{p_2} = 0.1727\%$ $\ e\ _{p_3} = 0.2336\%$	$\ e\ _{p_1} = 0.1525\%$ $\ e\ _{p_2} = 0.1594\%$ $\ e\ _{p_3} = 0.3235\%$
	$\ e\ _{p_1} = 0.5553\%$ $\ e\ _{p_2} = 0.5531\%$ $\ e\ _{p_3} = 0.3318\%$	$\ e\ _{p_1} = 0.2649\%$ $\ e\ _{p_2} = 0.2883\%$ $\ e\ _{p_3} = 0.5358\%$
	$\ e\ _{p_1} = 1.0487\%$ $\ e\ _{p_2} = 0.8459\%$ $\ e\ _{p_3} = 0.7775\%$	$\ e\ _{p_1} = 0.1799\%$ $\ e\ _{p_2} = 0.4189\%$ $\ e\ _{p_3} = 0.5097\%$

An analysis of the results summarized in the table confirms the high accuracy of the solutions for all geometry variants modeled by surface patches.

## 8 Conclusions

In the paper triangular and rectangular surface patches for modeling various complex 3D regions are used. The proposed representation of the boundary seems to be a promising alternative to conventional mesh generation procedures known from FEM and BEM. These procedures, as we know, are based on dividing the physical domain into large number of finite or boundary elements.

Three different geometries modeled by various surfaces whose number is significantly less than the number of elements in FEM and BEM, have been considered. The reliability of the modeling technique in connection with PIES has been verified by analyzing boundary value problems defined over regions with different shapes. Obtained numerical results confirm its reliability. According to the authors, performed analysis shows advantages of PIES also in relation to linear elasticity problems. These advantages are related to the simplicity of modeling also complex shapes of linear elastic bodies.

Having validated the method on the basis of analytical solutions it is interesting to examine the effectiveness of PIES compared to classical FEM and BEM. This is the topic of another paper.

## 9 Acknowledgment

This work is funded by resources for science in the years 2010-2013 as a research project.

## 10 References

- [1] O.C. Zienkiewicz, R.L. Taylor. "The Finite Element Method. vol. 1-3", Butterworth, Oxford, 2000.
- [2] C. Johnson, "Numerical Solution of Partial Differential Equations by the Finite Element Method". Cambridge University Press, 1990.
- [3] C.A. Brebbia, J.C. Telles. and L.C. Wrobel. "Boundary element techniques, theory and applications in engineering". Springer, New York, 1984.
- [4] A.A. Becker. "The Boundary Element Method in Engineering: a complete course". McGraw-Hill Book Company Cambridge, 1992.
- [5] E. Zieniuk, A. Bołtuc. "Non-element method of solving 2D boundary problems defined on polygonal domains modeled by Navier equation", International Journal of Solids and Structures, 43, pp. 7939-7958, 2006.
- [6] E. Zieniuk. "Bézier curves in the modification of boundary integral equations (BIE) for potential boundary-values problems", International Journal of Solids and Structures, 9(40), pp. 2301-2320, 2003.
- [7] E. Zieniuk. "Modelling and effective modification of smooth boundary geometry in boundary problems using B-spline curves". Engineering with Computers, 23(1), pp. 39-48, 2007.
- [8] E. Zieniuk. "Hermite curves in the modification of integral equations for potential boundary-value problems", Engineering Computations, 20(2), pp. 112-128, 2003.
- [9] E. Zieniuk, K. Szerszeń. „Linear Coons surfaces in the modeling of polygonal geometries in 3D boundary problems modeled by Laplace equation”, *Archiwum Informatyki Teoretycznej i Stosowanej*, 17/2, pp. 127-142, 2005 (in polish).
- [10] E. Zieniuk, K. Szerszen. "Triangular Bézier patches in modelling smooth boundary surface in exterior Helmholtz problems solved by PIES", *Archives of Acoustics*, 34, pp. 1-11, 2009.
- [11] G. Farin, J. Hoschek, M.S. Kim. "Handbook of computer aided geometric design". Elsevier, Amsterdam, 2002.
- [12] E. Zieniuk, K. Szerszeń, A. Bołtuc. „PIES in solving 3D boundary problems modeled by Navier-Lame equations in polyhedral areas”, *Modelowanie Inżynierskie* 42, pp. 487-494, 2011 (in polish).
- [13] D. Gottlieb and S.A. Orszag. "Numerical Analysis of Spectral Methods: Theory and Applications", SIAM, Philadelphia, 1977.
- [14] J. Zhang, Z. Yao. "The regular hybrid boundary node method for three-dimensional linear elasticity" *Engineering Analysis with Boundary Elements*, 28, pp. 525-534, 2004.
- [15] Y.X. Mukherjee, S. Mukherjee, X. Shi and A. Nagarajan. "The boundary contour method for three-dimensional linear elasticity with a new quadratic boundary element" *Engineering Analysis with Boundary Elements*, 20, pp. 35-44, 1997.

# Application of SolidWorks<sup>®</sup> & AMESim<sup>®</sup> - based Simulation Technique to Modeling, Cavitation, and Back-flow Analyses of Trochoid Hydraulic Pump for Multi-step Transmission

Myung Sik Kim<sup>1</sup>, Won Jee Chung<sup>1</sup>, Jun Ho Jang<sup>1</sup>, Chang Doo Jung<sup>1</sup>

<sup>1</sup> School of Mechatronics, Changwon National University, South Korea

**Abstract** - Flow rate control is the uppermost concern for trochoid hydraulic pump. Cavitation within the flow field of pump has the most influence on flow rate control reason at approximately 3500 ~ 4000 RPM high speed rotation of pump. In this paper, based on AMESim<sup>®</sup> and SolidWorks<sup>®</sup>, we will present how to simulate cavitation by analyzing the control factors of trochoid pump; hydraulic pressure change of outlet, flow rate according to rotation speed of inner rotor, leakage through gap between outer rotor and inner rotor, and discharging angle of outlet. The proposed methodology of cavitation simulation will enables field engineers to have access to the design of trochoid pump more easily and thereby to have more concrete control over the flow rate of pump by realizing its analysis model similar to its actual product model.

**Keywords:** Trochoid Hydraulic Pump, Modeling, Cavitation, Back-flow, AMESim<sup>®</sup>, SolidWorks<sup>®</sup>.

## 1 Introduction

Recently the role of transmission has been focused on according to the emphasis of driving performance, and the competitiveness of the transmission takes the largest part in the competition of automobiles together with engine. It is essential to improve the lubrication system and its performance due to the increase in the role and performance level of transmission, and the interests on these issues have also been increased. A trochoid (hydraulic) pump has been largely used in the lubrication system. Because the trochoid pump shows a simple structure, easy control in the flow rate per one rotation and the flow rate, and an advantage in its miniaturizing, it is very adaptable for the engine and transmission of automobiles due to the low variance in its efficiency because of the relatively small movement between outer and inner rotors.

It is essential to improve the lubrication system and its performance due to the increase in the role and performance level of transmission, and the interests on these issues have also been increased. A trochoid hydraulic pump has been largely used in the lubrication system. Because the trochoid pump shows a simple structure, easy controls in the flow rate per one rotation and the flow rate, and an advantage in its miniaturizing, it is very adaptable for the engine and

transmission of automobiles due to the low variance in its efficiency because of the relatively small movement between outer and inner rotors.

As shown in Fig. 1, the trochoid pump investigated in this paper represents an eccentricity in its rotation axis due to the structure of outer and inner rotors and that shows sliding contact (see Fig. 2 in detail). Also, it shows a difference in rotation speed as much as the difference in the number of teeth. While the rotors are rotated, the space generated by the contact between rotors is also rotated and generates an increase in volume. Then, it discharges a fluid to the outlet by absorbing a fluid from the inlet according to the increase or decrease in the volume. In addition, it prevents reverse flow (or back-flow) because the chamber between the inlet and the outlet is not connected.



Fig. 1 Trochoid Hydraulic Pump

Machining error and operating condition can be enumerated as the reason which can degrade the efficiency of trochoid pump. But cavitation within the flow field of pump can be placed as the most dominant reason during high speed rotation. Such cavitation can cause noise and vibration by increasing pulsation as well as the falloff of flow rate efficiency. Thus it is important to design the trochoid pump which can avoid the occurrence of cavitation in terms of the performance, endurance, noise and vibration of the pump. However it is well known that cavitation can inevitably occur at approximately 3500 ~ 4000 RPM (Revolution Per Minute) high-speed rotation speed of pump [1]. Therefore it is required to examine the phenomena of the cavitation factors

which have influence on the degradation of flow rate efficiency through cavitation simulation.

In previous papers including Yang et al.[1] and Nam et al. [2], the simulation of trochoid pump has utilized a professional analysis program or language such as CFD<sup>®</sup> (specifically CFX<sup>®</sup>) or C-code (e.g. C++) which is not an easy tool for a field engineer. This software tool can realize mesh and then make a cavitation model. But the field engineer based on his experience has a hurdle in modifying the cavitation model according to the need of a customer. In this paper, we will present how to simulate cavitation by using the most popular 3-dimensional (or 3D) modeling tool SolidWorks<sup>®</sup> and a hydraulic analysis program AMESim<sup>®</sup>. This proposed methodology of cavitation simulation will be useful for fast modification of trochoid pump design.

## 2 Trochoid Modeling using SolidWorks<sup>®</sup>

Figure 2 shows the rotor shape of pump based on trochoid profile. The trochoid pump is a type of gear pump, which is considered as a positive displacement pump in the upper level. The positive displacement pump has constant dispensing flow rate according to the single rotation of shaft regardless of loading pressure.

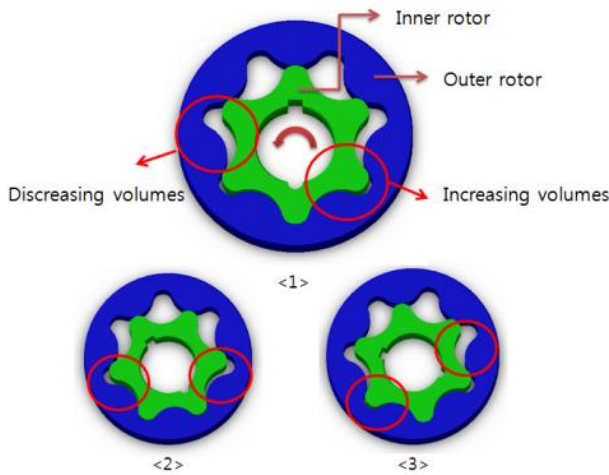


Fig. 2 Rotor shape of trochoid pump

Drawing of trochoid curve was suggested as various methods in many theses. The mathematical model composed of Eqs. (1) and (2), which has been mentioned representatively in previous papers including Jang et al.[3], forms basically-known trochoid curve condition:

$$K = \frac{-R_r \sin \theta + e \sin(N+1)\theta}{R_r \cos \theta - e \cos(N+1)\theta} \quad (1)$$

$$X = x + \frac{R_c}{\sqrt{1+K^2}}$$

$$Y = y - \frac{R_c K}{\sqrt{1+K^2}}$$

$$X = x - \frac{R_c}{\sqrt{1+K^2}}$$

$$Y = y + \frac{R_c K}{\sqrt{1+K^2}} \quad (2)$$

where  $R_r$  is the radius of rolling circle,  $N$  is the number of teeth of outer rotor, and  $R_c$  is the radius of circle of trace of wheel,  $\theta$  is rotation angle of basic circle,  $e$  is eccentricity as shown in Fig. 3 (the drawing of trochoid curve SolidWorks<sup>®</sup>).

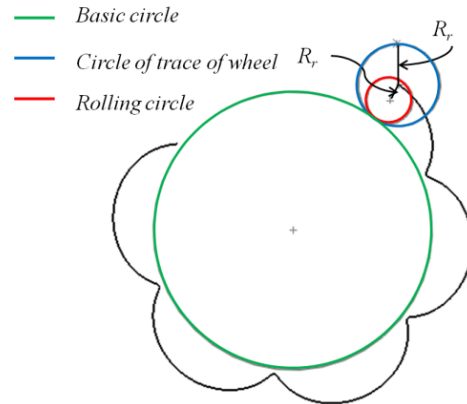


Fig. 3 Trochoid Curve Profile

Figure 3 means that filed engineers who are familiar with SolidWorks<sup>®</sup> can draw easily the trochoid model by using SolidWorks<sup>®</sup> based on Eqs. (1) and (2) not by virtue of C-code which is difficult for the engineers. Specifically, using the motion analysis module of SolidWorks<sup>®</sup>, the rolling circle can be rotated based on the basic circle in order to generate trochoid curve as shown in Fig. 3. Only the basic condition,  $R_r > e$ , should be satisfied. Next, when the circle of trace of wheel is made to be rotated according to the trochoid curve through the motion analysis module of SolidWorks<sup>®</sup>, the shape of inner rotor can be designed as shown in Fig. 4.

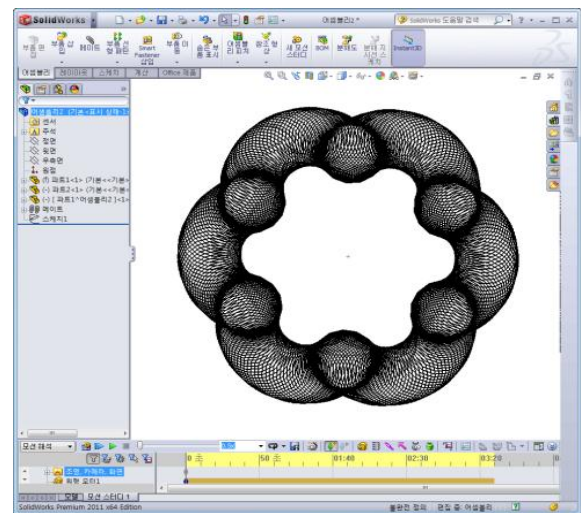


Fig. 4 Generation of Inner Rotor Shape

### 3 Flow Field Area Modeling Using SolidWorks®

As shown in Eq. (3), the dispensing flow rate  $Q_{pump}$  of the positive displacement pump can be obtained by using the change of flow volume  $displ$  per one rotation and the rotational angular speed  $\omega_{pump}$ . Especially  $displ$  can be expressed by Eq. (4) because the thickness  $H$  of trochoid pump is constant. In Eq. (4),  $Area$  denotes the cross-sectional area of flow rate while  $\theta$  indicates rotation angle. Now we need the data of area change according to rotation angle, i.e.  $\frac{dArea}{d\theta}$  by using SolidWorks® as a CAD (Computer Aided Design) tool.

$$Q_{pump} = displ \cdot \omega_{pump} \quad (3)$$

$$displ = \frac{dV}{d\theta} = H \cdot \frac{dArea}{d\theta} \left[ \frac{m^3}{rad} \right] \quad (4)$$

The data of area change ( $dArea$ ) can be obtained by using angular velocity ratio of trochoid pump. It can be performed by changing the area through the rotation of each rotor according to the rotation ratio of inner and outer rotors. The speed ratio can be determined by Eq. (5) according to the number of outer rotor teeth ( $N$ ):

$$\frac{\omega_{out}}{\omega_{in}} = \frac{N-1}{N} \quad (5)$$

where  $\omega_{out}$  ( $\omega_{in}$ ) denotes the angular velocity of outer (inner) rotor.

When the drawings of the inner and outer rotors are completed, flow field area modeling can be done through the element conversion technique of SolidWorks® as shown in Fig. 5 where inlet and outlet are simplified. In specific, the sketch of 3 parts, i.e., inner rotor, outer rotor and inlet/outlet, are first designated as 'BLOCKs' by using BLOCK technique of SolidWorks®. Then the flow field area (to be explained later) can be modeled by performing the PROTRUSION BASE (similar to PAD technique of Solidworks®) technique of SolidWorks®. Consequently 7 models of flow field area are designated as shown in Fig. 5.

Now, the area change of one flow field should be investigated according to Eq. (5) (in other words, according to the rotation of inner rotor BLOCK). Specifically, while making inlet and outlet BLOCK fixed, the outer rotor BLOCK is rotated based on Eq. (5) as the inner rotor BLOCK is rotated, by using FORMULA EDIT technique of SolidWorks®. In every rotation, the area of one flow field can be changed in shape according to rotation angle of inner rotor as shown in Fig. 6. Thus the area change of one flow field can be depicted as a graph of Fig. 7 which will be used later for the simulation of trochoid pump using AMESim®.

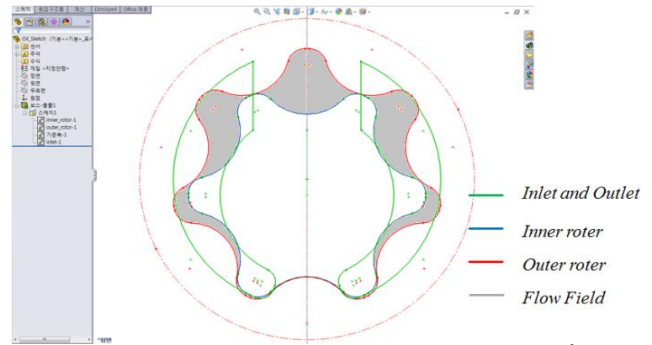


Fig. 5 Flow field modeling using SolidWorks®

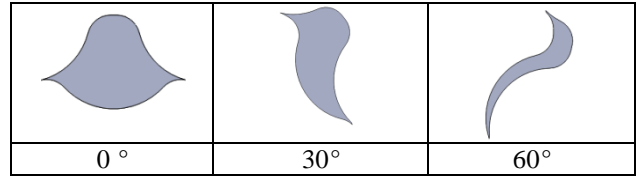


Fig. 6 Area change of one flow field according to rotation angle of inner rotor (shape)

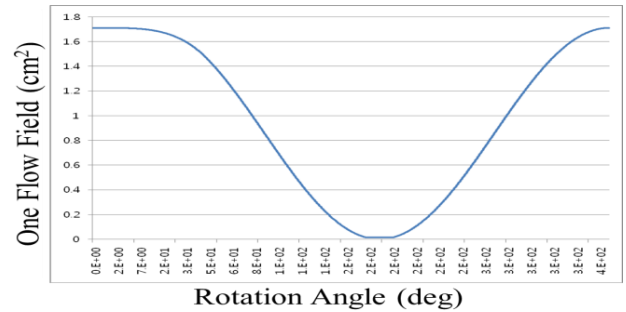


Fig. 7 Area change of one flow field according to rotation angle of inner rotor (graph)

In a similar manner to the method mentioned above, the area change of one flow field in inlet and outlet of trochoid pump (see Fig. 8) can be easily obtained in the graph of Fig. 9, which will be also used for cavitation simulation in AMESim®.

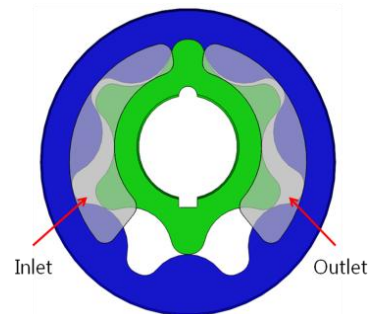


Fig. 8 Inlet and outlet flow field modeling using SolidWorks®



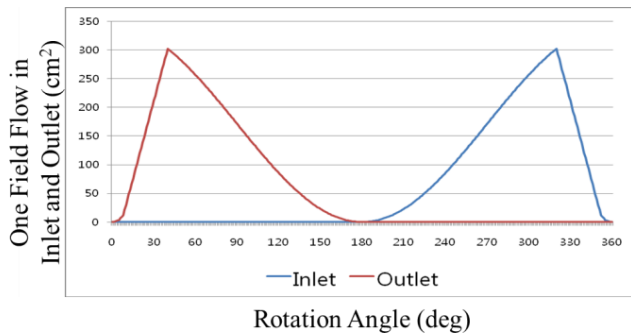


Fig. 9 Area change for inlet and outlet flow fields

### 4 Hydraulic Circuit For Trochoid Pump And Cavitation Simulation Using AMESim®

The objective of hydraulic circuit modeling for trochoid pump using AMESim® is to realize flow and simulate cavitation so as to control the flow rate control of trochoid pump. In case of hydraulic circuit modeling of only one flow field for trochoid pump, the area change data of Fig. 7 and the thickness of rotor can be made equivalent with the piston model as shown in Fig. 10.

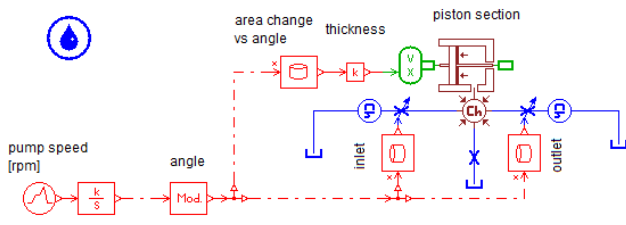


Fig. 10 Hydraulic circuit modeling of only one flow field using AMESim®

Hydraulic circuit modeling of trochoid pump is required to show the volume generated at the location of each particular angle when the inner rotor comes in contact with the outer rotor. This can be carried out at the location of each phase change, *i.e.*,  $360^\circ/N$ . Finally  $N$  models are generated similarly as Fig.10 and connected each other as shown in Fig. 11.

The factors that can be simply controlled in real time through the formation of the AMESim® hydraulic circuit of a trochoid pump include: the rotation speed of pump, the shape angle of inlet and outlet, gap between outer rotor and pump casing, and gap between inner rotor and outer rotor. In cavitation simulation, these control factors are important for the flow rate control of trochoid pump. Especially, from the viewpoint of a field engineer, AMESim® is more useful for this cavitation simulation, compared with a traditional analysis software of fluid mechanics, *i.e.*, CFD®, because CFD® needs the renewal of mesh modeling every time each control factor has a different value while AMESim® needs only the input values of control factors without any change of hydraulic circuit modeling.

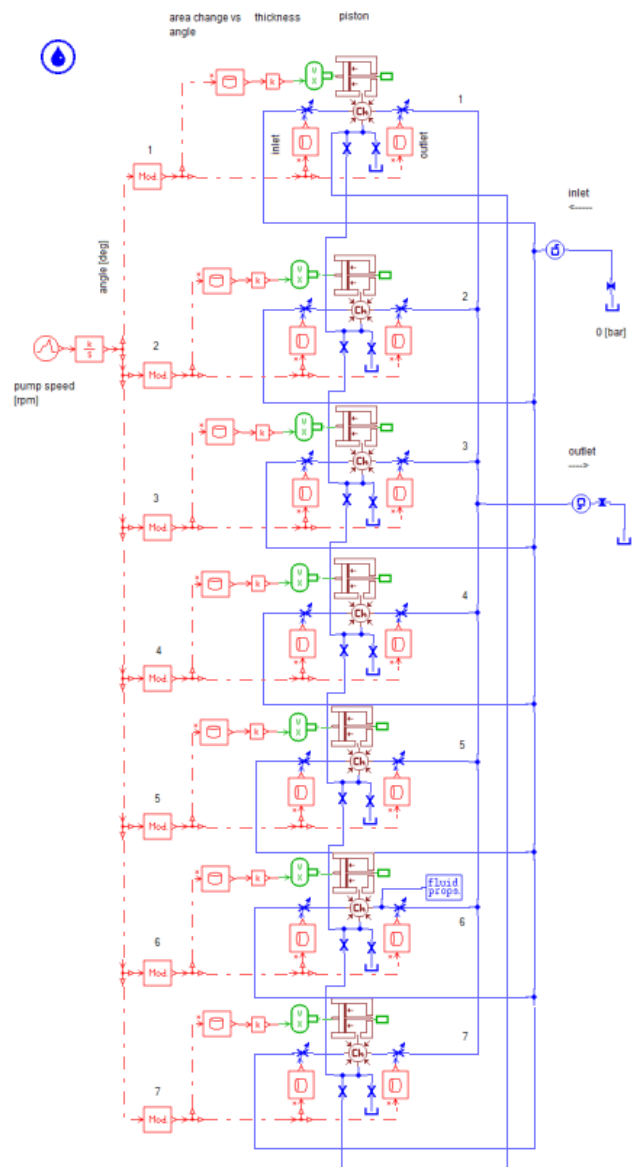


Fig.11 Connected  $N$  hydraulic circuit modeling of trochoid pump using AMESim®

As mentioned before, cavitation can inevitably occur at approximately 3500 ~ 4000 RPM high-speed rotation speed of pump. The cavitation usually results in increasing pulsation and thereby degrades flow rate efficiency. This is why cavitation simulation is required for the control of flow rate. In this paper, the control factors of trochoid pump including the rotation speed of pump, the pressure difference at inlet and outlet, the shape angle of outlet, gap between outer rotor and pump casing, and gap between inner rotor and outer rotor are analyzed in the cavitation simulation through the connected  $N$  hydraulic circuit modeling of trochoid pump using AMESim® (see Fig. 11).

### 4.1 Hydraulic Pressure Change of Outlet

As the first result of cavitation simulation, Fig. 12 shows the hydraulic pressure change of outlet at the 3000RPM rotation speed of inner rotor in a stabilized zone except a transient status of outlet pressure. The pressure change shows a cyclic (or periodic) characteristic at every  $360^\circ/N$  depending on the number of teeth for the outer rotor. Since  $N$  is 7 in this paper, in a stabilized zone, the pressure change cycle is repeated seven times for one rotation of inner rotor as shown in Fig. 12. This confirms the periodicity shown in the analyses using CFD® (Won *et al.* [3] and Yang *et al.*[1].)

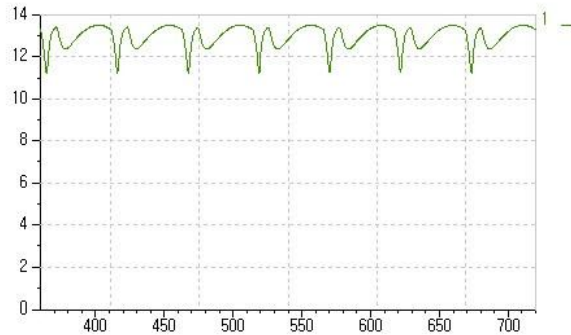


Fig. 12 Hydraulic pressure change of outlet

### 4.2 Flow rate according to Rotation Speed of Inner Rotor

Figure.13 shows the result of flow rate according to the rotation speed of inner rotor. As shown in this figure, a theoretical flow rate (2) delineates linear profile according to the rotation speed of the rotor. This means that the fluid of pump is filled 100% in the flow field without cavitation because the gap between the inner rotor and the outer rotor is not taken into consideration for theoretical study. However, since cavitation has been considered in the simulation, it can be noticed that the flow rate decreases at more than 4000RPM. As stated in refs. [1] and [2], it is confirmed that as the rotation speed of pump increases, cavitation is generated in the flow field, which makes the flow rate decreased due to the leakage of fluid into the outlet.

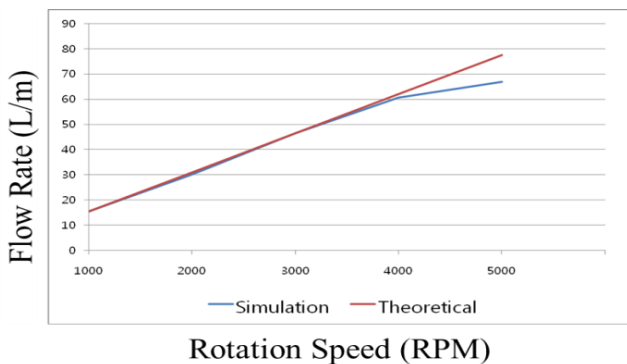


Fig. 13 Flow rate according to rotation speed of inner rotor

### 4.3 Discharging Angle of Outlet

In cavitation, rise of pulsation is inevitable. Moreover, in the high speed operation of trochoid pump, inlet flow resistance is enlarged so that cavitation phenomena itself can be increased. This means that the quantity of fluid to be transported in an isolated flow field can be decreased due to the increase of cavitation which has been induced by the decrease of the hydraulic pressure for the flow field. Thus back-flow into the isolated flow field can be resulted in at the position shown in Fig. 14. When back-flow to the isolated flow field is discharged to the outlet again by the rotation of rotor, it can result in higher pulsation by increasing the discharge pressure of outlet.

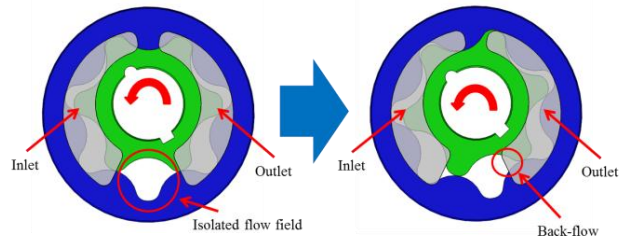
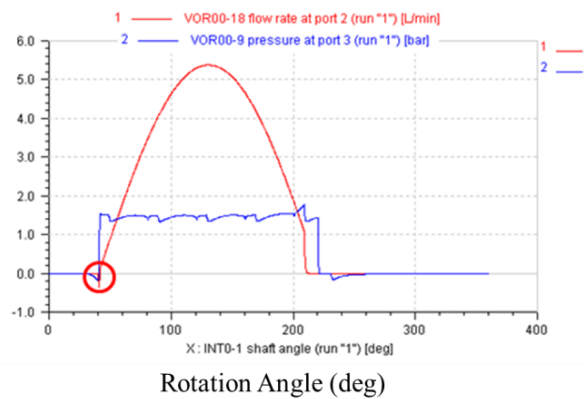


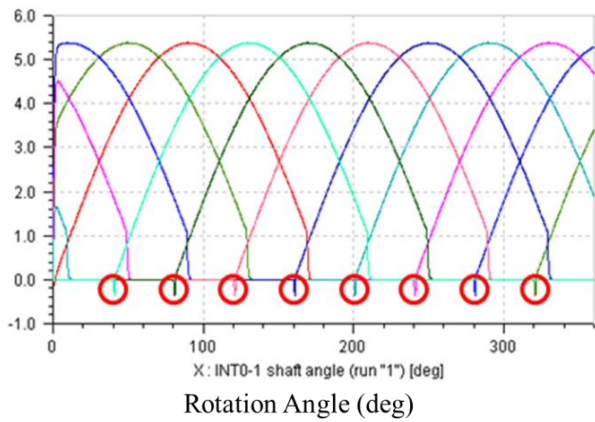
Fig. 14 Back-flow due to cavitation

The flow rate of back-flow is observed through Fig. 15 (a). In one flow field, the out-flow shows under 0 due to momentarily deterioration of pressure when the isolated flow field (see Fig. 14) meet the outlet position. Thus this showed the back-flow phenomenon. As shown in Fig.15 (b), when it is applied to all flow fields (*i.e.*, chambers), it can be seen that this back-flow has occurred so as to have a significant effects on flow rate.

To cope with this back-flow problem, the method to delay the discharging instant of outlet (in other words, the method to decrease the discharging angle of outlet) has been utilized in ref. [6,7]. In this paper, this method has been also adopted in order to prevent the back-flow phenomenon induced by the isolated flow field.



(a) Back-flow of one flow field



(b) Back of all flow fields

Fig. 15 Flow rate of back-flow

As shown in Fig. 16, the discharge instant of the outlet position has been delayed so that the compression time of isolated flow field has been increased. As a result, the pressure difference between the isolated flow field and the outlet position has been decreased so that the back-flow problem has been suppressed. This can be verified in Fig. 17 as the discharging angle of outlet is delayed at intervals of 0°, 1°, 2°, 3°, 4°, and 5°. As shown in this figure, the effect of back-flow has not occurred for 4° above even though it has occurred for 4° below. Moreover the flow rate has been increased to small extent for 4° above. Unfortunately, Fig. 18 shows that the pressure peak at the discharging instant has been observed so that the leakage through gap between outer rotor and inner rotor should be inevitable.

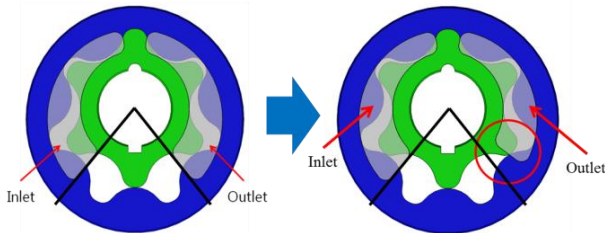


Fig. 16 Outlet configuration when discharging angle is decreased by 5°

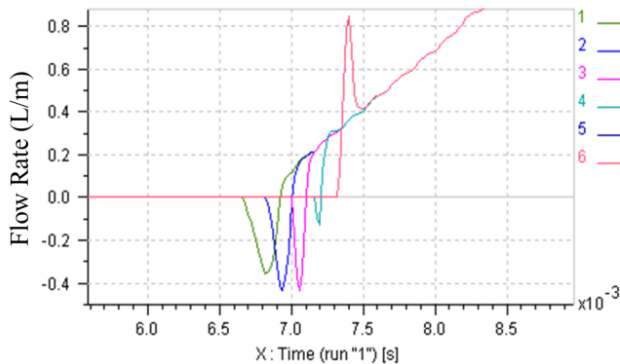


Fig. 17 Flow rate according to discharging angle of outlet

As mentioned in Nam *et al.* [2], it can be stated that the reduction of back-flow can decrease the loss of flow rate more effectively than the leakage between the gap. Consequently to decrease the discharging angle to some extent is to prevent the degradation of flow rate. Further study on the decreasing extent of discharging angle will be left for the flow rate control in detail.

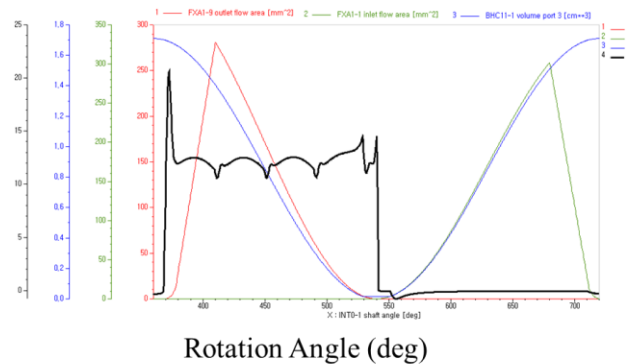


Fig. 18 Pressure distribution of outlet when the discharging time is delayed

## 5 Conclusions

The purpose of this paper aims at enabling field engineers to have access to the design of trochoid pump more easily and thereby to have more concrete control over the flow rate of pump by realizing its analysis model similar to its actual product model. For this purpose, first, we have used AMESim® which gives field engineers easy tool for analyzing the cavitation factors of trochoid pump including hydraulic pressure change of outlet, flow rate according to rotation speed of inner rotor, leakage through gap between outer rotor and inner rotor, and discharging angle of outlet, rather than a professional hydraulic analysis program or language such as CFD®.

In this paper, based on AMESim® with SolidWorks®, we have presented how to simulate cavitation by analyzing the control factors of trochoid pump which have influence on the degradation of flow rate efficiency. This proposed methodology of cavitation simulation will be useful for flow rate control through the fast modification of trochoid pump design. In further research, we expect that the flow rate over which a designer wants to have control would be optimized rapidly according to the change of environmental condition (e.g. the kind of hydraulic fluid, the application area of trochoid pump such as transmission or engine, *etc.*) by applying the proposed cavitation simulation methodology to the flow rate control using with MATLAB® or LabVIEW® at real time.

## 6 Acknowledgement

The authors of this paper were partly supported by the Second Stage of Brain Korea21 Projects



## 7 References

- [1] S. Y. Yang, and S.J. Cha, "Simulation of Cavitating Flow in a Gerotor Oil Pump," The Korean Society of Automotive Engineers, Autumn Conference, 2006, pp. 152~158.
- [2] K. W. Nam, S. H. Jo, and J. I. Park, "Numerical Simulation in the IC Engine Lubricating Gerotor Oil Pump," The Korea Society of Mechanical Engineers, vol. 30, no.10, pp.1019~1025, 2006
- [3] C. S. W, N. G. Kwon, and S. H. Kwon, "Flow Analysis of Automotive Oil Pump of Gerotor Type"
- [4] J. S. Seo, H. S. Chung, and H. M. Jeong, "A Study on Gerotor Design with Optimum Tip Clearance for Low Speed High Torque Gerotor Hydraulic Motor"
- [5] S. Neyrat, N. Orand, and D. Jonquet, "Modeling and Analysis of an Automatic Transmission Internal Gear Oil Pump with Cavitation"
- [6] B. J. Kim, S. H. Seong, and S. H. Yoon, "Flow Analysis and Port Optimization of GeRotor Pump Using Commercial CFD Code"
- [7] H. K. Moon, S, J. Jang, C. Kim, and H. Y. Cho, "Port Configurations on CFD Analysis in Gerotor Pump"

# An S-system Parameter Sensitivity Analysis of Biohydrogen Production by the Microalga *Chlamydomonas reinhardtii*

Jack K. Horner  
 PO Box 266  
 Los Alamos NM 87544 USA  
 email: jhorner@cybermesa.com

## Abstract

*Producing biohydrogen on a commercial scale will likely require the genetic re-engineering of natural hydrogen-producing organisms. Kinetic modeling of hydrogen-producing metabolic pathways can cost-effectively help to characterize systemic (e.g., mass/energy/charge conservation) constraints in these organisms. In vitro kinetic studies suggest that the activity of the hydrogenases in several photolytic biohydrogen producers (PBPs) could be increased to as much as four times their nominal in vivo rate. It is much less clear, however, whether the in vitro activity maximum could be realized in vivo. Here I use an S-system photosynthesis-based PBP (PS-PBP) simulator to survey the sensitivity of C. reinhardtii to variation in system parameters. The analysis strongly suggests that the H<sub>2</sub> production efficiency of the alga cannot be increased by more than a factor of two through single-enzyme genetic modifications.*

Keywords: biohydrogen, S-system, metabolic modeling

## 1.0 Introduction

Kinetic modeling of hydrogen-producing metabolic pathways can cost-effectively help to characterize systemic (e.g., mass/energy conservation) sensitivities in photolytic biohydrogen producers, even if all the details of hydrogen-gas producing metabolic pathways are not known. Among the more promising candidates for hydrogen-production optimization are photolytic biohydrogen producers (PBPs) such as the microalga *Chlamydomonas reinhardtii* ([7], [8]). It is generally held that the hydrogen-producing pathways in many PBPs incorporate segments of the PS-I and

PS-II photosynthetic pathways ([6],[13]), and electrons from the anaerobic degradation of starch, to help accumulate the electron free energy required to allow a hydrogenase to convert protons to H<sub>2</sub> ([14]). *In vitro* kinetic studies suggest that the activity of hydrogenases isolated from several PBPs could be increased to as much as four times their nominal *in vivo* rate ([1]). Here I use *bioh2gen* ([15]), an S-system ([2], [11]) PS-PBP kinetics simulator, to argue that within the context of the model, the H<sub>2</sub> production efficiency of *C. reinhardtii* cannot be increased by more than a factor of two through single-enzyme genetic modifications

## 2.0 S-systems

An S-system ([11],[12]) is a power-law-oriented, finite-difference system of

ordinary differential equations (SODE) each of whose dependent variables  $X_i$  is described by a kinetic equation of the form

$$dX_i/dt = \alpha_i \prod_j X_j^{g_{i,j}} - \beta_i \prod_j X_j^{h_{i,j}}$$

Eq. 2.1

where

- the left-hand side of Eq. 2.1 is the first derivative of  $X_i$  with respect to time
- $i, j = 1, 2, 3, \dots, N$
- $\{X_i\}$  is the set of real-valued dependent variables of the system
- for any given  $X_i$ , only those independent and dependent variables  $X_j$  that have an action on  $X_i$  are included as factors in the products on the right-hand-side (RHS) of Eq. 2.1. The factors in the first term on the RHS of Eq. 2.1 correspond to just those entities that increase or inhibit the production of  $X_i$ ; the factors in the second term of the RHS of Eq. 2.1 correspond to just those entities that contribute to, or inhibit, the consumption of  $X_i$ .
- $\alpha_i, \beta_i > 0$
- $g_{i,j}, h_{i,j}$  are real-valued

There is a natural mapping from a biochemical map,  $K$ , to equations that have the form of Eq. 2.1. In particular, let  $K = \langle \{X_k\}, E \rangle$ ,  $E \in \{X_k\} \otimes \{X_k\}$ ,  $k = 1, 2, \dots, N$ , be a directed graph in which each distinct  $X_i \in \{X_k\}$  corresponds to a distinct variable (e.g., the concentration of a distinct chemical species in the map), and  $w \in E$  if and only if  $w = (X_m, X_n)$  is a directed edge in  $K$ ,  $m \neq n = 1, 2, \dots, N$ .

$\alpha_i$  and  $\beta_i$  are called *generalized rate constants* (or just rate constants) for  $X_i$ , and  $g_{i,j}$  and  $h_{i,j}$  are called the *generalized kinetic orders* (or just kinetic orders) for  $X_i$ , on analogy with standard chemical kinetic

theory. The subexpression  $i_j$  indicates the action of  $X_j$  on  $X_i$ .

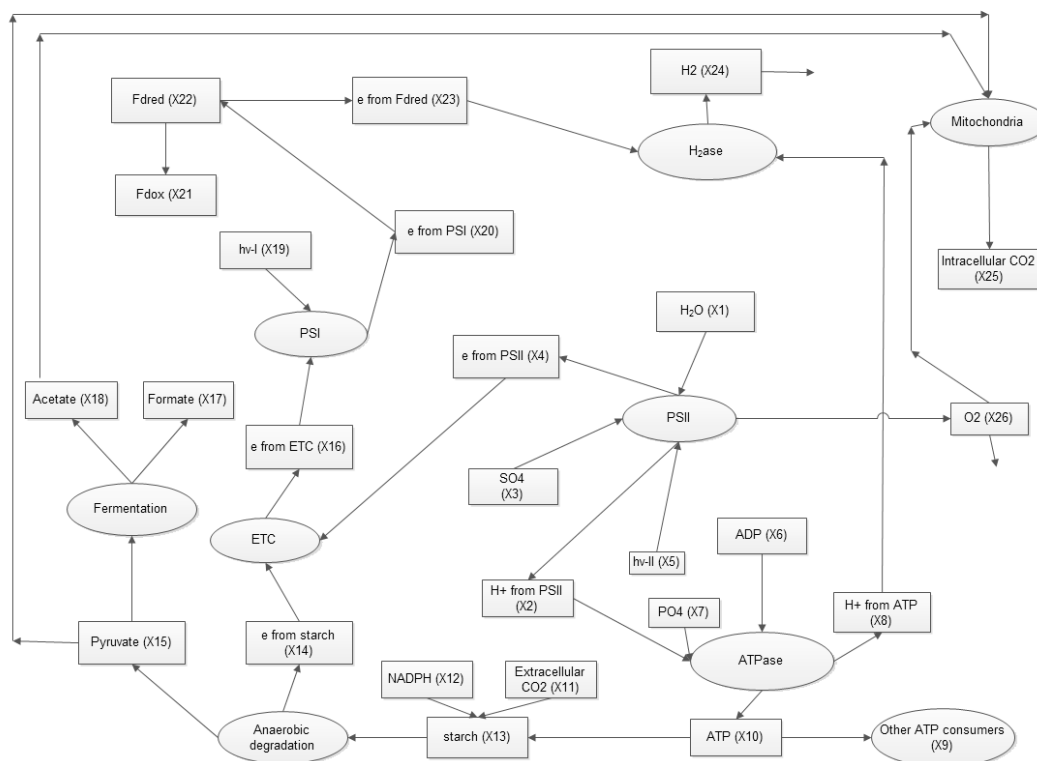
An S-system has several desirable features, including the fact that it is fully characterized by its rate constants and kinetic orders, allowing us to comprehensively survey the system's (logarithmic gain) sensitivity to its parameters. Any SODE can be *recast* ([10],[11]) as an S-system without loss of accuracy or precision; the recasting, however, is not in general unique. In addition to biochemical systems, S-systems have been successfully used to model

epidemics, forest diversification, and world dynamics.

Performing (rate-constant, and kinetic-order) parameter logarithmic gain (sensitivity) surveys on S-systems is straightforward ([11], Chapter 7); performing parameter gain surveys on a SODE which is not an S-system, in general, is ill-defined.

### 3.0 A network model of hydrogen production in PS-PBPs

I will call bioH<sub>2</sub> producers that exploit portions of the PSII or PSI pathways “photosynthetic” PBPs (PS-PBPs). The schematized PS-PBP model used in the present study is shown in Figure 1 and is similar to [3], [4], [5], [9] and [14]. It represents a consensus working hypothesis held by the biohydrogen research community about the high-level metabolics of hydrogen production in PS-PBPs ([7]).



**Figure 1. Schematized hydrogen producing metabolic network for PS-PBPs. Rectangles represent sources or sinks of physical quantities of interest (such as mass, concentration, or photon count) named in those rectangles, ellipses represent transforms (which may be complexes of reactions not individually modeled here), and an arrow from an ellipse to a rectangle means that the transform named in the ellipse affects the quantity/concentration of the chemical species named in the rectangle. Legend: PSI = photosynthesis stage I; PSII = photosynthesis stage II; SO4 = sulfate; hv-I = photons incident to PSI; hv-II = photons incident to photosynthesis PSII; ADP = adenosine diphosphate; ATP = adenosine triphosphate; PO4 = inorganic phosphate; O2 = oxygen gas; ATPase = adenosine**

triphosphatase; e from starch = electrons from anaerobic starch degradation; H<sub>2</sub>ase = hydrogenase; ETC = electron transport chain; e from PSII = electrons from PSII; e from PSI = electrons from PSI; F<sub>dred</sub> = ferredoxin, reduced; F<sub>dox</sub> = ferredoxin, oxidized; H<sub>2</sub> = hydrogen gas; H<sub>+</sub> from PSII = protons from PSII; H<sub>+</sub> from ATP = protons from ATPase.  
Not all interactions exist in all PS-PBP species.

In sulfur-deprived *C. reinhardtii*, oxygen gas production under the experimental conditions of [7] (1-L, 6 x 10<sup>6</sup> cell/mL preparation) is about 1 mmol/h after beginning of sulfur deprivation, and spontaneously ceases ~10 h thereafter. 30 - 50 h after beginning of sulfur deprivation, the algae begins releasing hydrogen at a rate of ~0.17 millimole H<sub>2</sub>/h (1-L, 6 x 10<sup>6</sup> cell/mL preparation) after beginning of

sulfur deprivation. ~100 h after beginning of sulfur deprivation, hydrogen production ceases. These trajectories provide strong constraints on any model of bioH<sub>2</sub> production by *C. reinhardtii*.

The S-system equations used in this study are shown in Figure 2.

---

```
// protons from PSII
X2' = a2 X1^g2_1 X3^g2_3 X5^g2_5 - b2 X10^h2_8 X2^h2_2 X5^h2_5

// e from PSII
X4' = a4 X1^g4_1 X3^g4_3 X5^g4_5 - b4 X16^h4_16 X4^h4_4

// protons from ATPase
X8' = a8 X6^g8_6 X7^g8_7 X2^g8_2 - b8 X8^h8_8 X24^h8_24

// other ATP consumers
X9' = a9 X10^g9_10 - b9 X9^h9_9

// ATP
X10' = a10 X2^g10_2 X7^g10_7 X6^g10_6 - b10 X13^h10_13 X9^h10_9 X10^h10_10

// starch
X13' = a13 X12^g13_12 X11^g13_11 X10^g13_10 - b13 X14^h13_14 X15^h13_15 X13^h13_13

// e from starch
X14' = a14 X13^g14_13 - b14 X16^h14_16 X14^h14_14

// pyruvate
X15' = a15 X13^g15_13 - b15 X25^h15_25 X18^h15_18 X17^h15_17 X15^h15_15

// e from ETC
X16' = a16 X14^g16_14 X4^g16_4 - b16 X20^h16_20 X16^h16_16

// formate
X17' = a17 X15^g17_15 - b17 X17^h17_17

// acetate
X18' = a18 X15^g18_15 - b18 X15^h18_25 X18^h18_18

// e from PSI
X20' = a20 X16^g20_16 - b20 X22^h20_22 X20^h20_20

// Fdox
X21' = a21 X22^g21_22 - b21 X21^h21_21

// Fdred
X22' = a22 X20^g22_20 - b22 X21^g22_21 X23^g22_23 X22^h22_22
```

```

// e from Fdred
X23' = a23 X22^g23_22 - b23 X24^h23_24 X23^h23_23

// H2 gas
X24' = a24 X23^g24_23 X8^g24_8 - b24 X24^h24_24

// Intracellular CO2
X25' = a25 X15^g25_15 X18^g25_18 X26^g25_26 - b25 X25^h25_25

// oxygen
X26' = a26 X1^g26_1 X3^g26_3 X5^g26_5 - b26 X26^h26_26 X25^h26_25 X5^h26_5

```

**Figure 2. S-system equations for the dependent variables used in this study. “^” is exponentiation. “>” means “expression continuation”. “'” means “first derivative with respect to time”. Note that the equation for X2' has light as a *consumption* factor because activity *decreases* as light intensity increases above an optimal value.**

Table 1 shows the values of the independent variables of the system.

**Table 1. Values of the independent variables of the system.**

Independent variable	Value (relative units)
X1 (water)	1
X3 (SO4)	0.3
X5 (hv-II)	2.363
X6 (ADP)	100
X7 (PO4)	100
X11 (Extracellular CO2)	3e-3
X12 (NADPH)	1e-6
X19 (hv-I)	2.363

Much of the system in Figure 1 is based on PSII and PSI kinetics. Based on PSII/PSI kinetic data in [16], all generalized rate constants were set to 0.1, except a2 (= 3e-4), b2 (= 1e-4), a4 (=0.01), a24 (=1e-4), b24 (= 0.001), a26 (=10), and b26 (=1000); these exceptions were based on *in vitro* experimental values obtained in [7]. All generalized kinetic orders were set to 1.

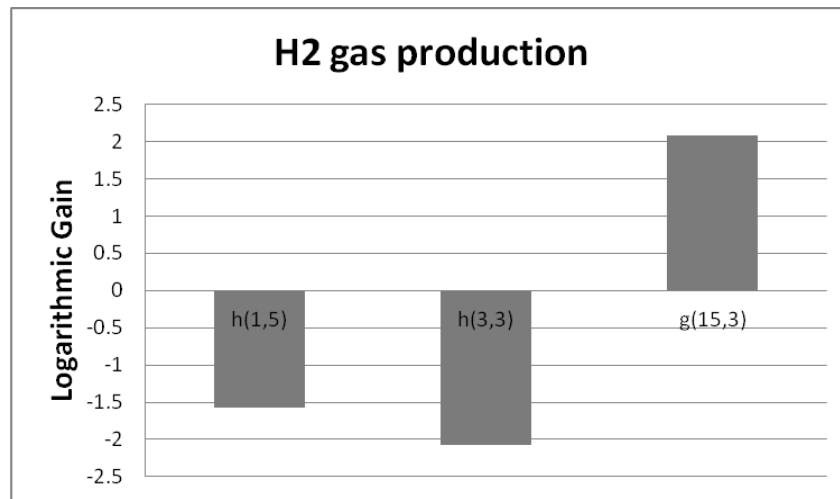
*bioh2gen* and the model used in [14] differ in a few ways. First, following the conventions in [11] for modeling metabolic systems in the absence of gene-circuit dynamics, no enzyme is an explicit variable of *bioh2gen*; several enzymes are variables in [14]. Second, *bioh2gen* employs more rate constants derived from experiment than does the model used in [14]. Third, all the kinetic orders in *bioh2gen* were set to 1; two kinetic orders were set to 2 in [14]. Fourth, *bioh2gen* study models the photon inputs to each of PSII and PSI individually; the model in [14] represents only the photon inputs to PSII.

The H<sub>2</sub> and O<sub>2</sub> production rates of *bioh2gen* were compared to [7], and the logarithmic gains ([11], Chapter 7) of the H<sub>2</sub> production rates were computed as a function of the generalized rate constants and kinetic orders in the model. Gains whose absolute values were less than 1 were excluded from consideration.

## 4.0 Results and discussion

The generalized rate constants in the model described in Section 3.0 exhibited no logarithmic gain  $\geq 1$ . Figure 4 shows the logarithmic gain of H<sub>2</sub> gas production in the model to kinetic orders, if the sensitivity is  $\geq 1$ . Changing generalized kinetic orders

typically requires changing the genetics of enzymes associated with those kinetic orders. Logarithmic gains  $> \sim 10$  can be opportunities for single-enzyme genetic modification; gains less than  $\sim 10$  typically are not ([17]).



**Figure 4. Logarithmic gain of H<sub>2</sub> production as a function of kinetic order. (The kinetic orders are shown with PLAS internal names. In each of the following, the PLAS internal name appears on the left-hand side of the equality; the name shown in Figure 1, on the right-hand-side:  $h(1,5) = h2\_10$ ;  $h(3,3) = h8\_8$ ;  $g(15,3) = g24\_8$ . The gains suggest that ([11], p. 226) H<sub>2</sub> production efficiency cannot be increased by more than a factor of two through single-enzyme genetic modification.**

Figure 4 strongly suggests that, within the model described in Section 3.0, the H<sub>2</sub> production efficiency of *C. reinhardtii* cannot be increased by more than a factor of two ([11], p. 226) through single-enzyme genetic modification. This is consistent with Figure 1: restrictions on the photolytically generated protons from PSII, and the proton turnover rate of ATPase, constrain the

throughput of protons available to produce H<sub>2</sub>. These results are consistent with the implications of [14].

## 5.0 Acknowledgements

This work benefited from discussions with Maria Ghirardi and Michael Seibert of the

National Renewable Energy Laboratory, Anastasios Melis of the University of California/Berkeley, Anatoly Tsygankov of the Institute of Basic Biological Problems (Pushchino, Russia), Orlando Jorquera of the Federal University of Bahia, Murray Wolinsky of Los Alamos National Laboratory, and Jorge Soberón of the University of Kansas Biodiversity Institute. For any errors that remain, I am solely responsible.

## 6.0 References

- [1] Cammack R. Hydrogenases and their activities. In Cammack R, Frey M, and Robson R, eds. *Hydrogen as a Fuel: Learning from Nature*. Taylor and Francis. 2001.
- [2] Ferreira AEN. *Power Law Analysis and Simulation (PLAS)*. Version 1.2 beta, Build 0.120. URL <http://correio.cc.fc.ul.pt/~aenf/plas.html>. March 2011. Note: the link to the PLAS software appears is broken as of 1 January 2012. A copy of the software is available on request from the author of the present paper.
- [3] Horner JK. An S-system model of hydrogen production in microalgae. *International Society for Computational Biology 2002, Special Interest Group for Biological Simulation Satellite Meeting (SIGSIM2002), Computer Modeling of Cellular Processes*. Edmonton, Alberta, Canada.
- [4] Horner JK. Leveraging biohydrogen research: a kinetic modeling approach. *Hydrogen and Fuel Cells Conference 2003*. Vancouver, British Columbia, Canada.
- [5] Horner JK and Wolinsky MA. A power-law sensitivity analysis of the hydrogen-producing metabolic pathway in *Chlamydomonas reinhardtii*. *International Journal of Hydrogen Energy* 27 (2002), 1251-1255.
- [6] Lawlor DW. *Photosynthesis*. Third Edition. Springer. 2001.
- [7] Melis A et al. Sustained photobiological hydrogen gas production upon reversible inactivation of oxygen evolution in the green algae *Chlamydomonas reinhardtii*. *Plant Physiology* 122 (2000), 127-135.
- [8] Melis A. Green alga hydrogen production: progress, problems, and prospects. *International Journal of Hydrogen Energy* 27 (2002), 1217-1228.
- [9] Horner JK. *bioh2gen, Version 1*. Available on request from the author. 2004.
- [10] Savageau MA. Growth of complex systems can be related to the properties of their underlying determinants. *Proceedings of the National Academy of Sciences* 76 (1979), 5413-5417.
- [11] Voit EO. *Computational Analysis of Biochemical Systems*. Cambridge. 2000.
- [12] Drazin PG. *Nonlinear Dynamics*. Cambridge. 1992.
- [13] Markqvart T and Landsberg PT. Solar cell model for electron transport in photosynthesis. *Proceedings of the 29th IEEE Photovoltaic Specialists Conference (2002)*, 1348-1351.
- [14] Jorquera O, Kiperstok A, Sales EA, Embiruçu M, and Ghiardi ML. S-systems sensitivity analysis of the factors that may influence hydrogen production by sulfur-deprived *Chlamydomonas reinhardtii*. *International Journal of Hydrogen Energy* 33 (2008), 2167-2177.
- [15] Horner JK. *bioh2gen, Version 5*, a PLAS simulator for biohydrogen production by photosynthetic biohydrogen producers. Source code is available on request from the author.
- [16] NPO Bioinformatics Japan. *KEGG: Kyoto Encyclopedia of Genes and Genomes*. <http://www.genome.jp/kegg/>. 2012.
- [17] Torres NV, Voit EO, Glez-Alcón C, and Rodriguez F. An indirect optimization method for biochemical systems: description of method and application to the maximization of the rate of ethanol, glycerol, and carbohydrate production in *Saccharomyces cerevisiae*. *Biotechnology and Bioengineering* 55 (1997), 758-772.



# Stochastic Dynamic Analysis of Nonlinear Vibration of Fluid-conveying Double-walled Carbon Nanotubes Based on Nonlocal Elasticity Theory

Tai-Ping Chang<sup>1</sup>

<sup>1</sup>Department of Construction Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan

**Abstract** - This study deals with the stochastic dynamic behaviors of nonlinear vibration of the fluid-conveying double-walled carbon nanotubes (DWCNTs) by considering the effects of the geometric nonlinearity and the nonlinearity of van der Waals (vdW) force. Besides, the small scale effects of the nonlinear vibration of the DWCNTs are investigated by using the theory of nonlocal elasticity. Some statistical dynamic response of the DWCNTs such as the mean values and standard deviations of the amplitude of the displacement are computed, meanwhile the effects of the flow velocity and small scale coefficients on the statistical dynamic response of the DWCNTs are investigated. It is concluded that the mean value and standard deviation of the amplitude of the displacement increase nonlinearly with the increase of the frequencies and change slightly as the flow velocity increases. Furthermore, small scale coefficients have significant influence on the mean value and standard deviation of the amplitude of the DWCNTs.

**Keywords:** Nonlinear vibration; Double-walled carbon nanotubes; Stochastic dynamic response; Galerkin's method; Small scale effect; Nonlocal elasticity theory.

## 1 Introduction

Since the landmark paper published by Iijima [1], carbon nanotubes (CNTs) have attracted worldwide attention due to their potential use in the fields of chemistry, physics, nano-engineering, electrical engineering, materials science, reinforced composite structures and construction engineering. Carbon nanotubes (CNTs) are used for a variety of technological and biomedical applications including nanocontainers for gas storage and nanopipes conveying fluids [2-8]. Some important applications of carbon nanotubes (CNTs) are such as nanotubes conveying fluids [3,7-8], different types of fluid flows like water [9], dynamic flow of methane, ethane and ethylene molecules [10] and the diffusive transport of light gases [11] had been reported, and the effects of these fluids on the mechanical properties of CNTs had been investigated. Generally there are two methods widely adopted to study the CNTs conveying fluids. One is the molecular dynamics simulations (MDS) [10-11], however, MDS needs a tremendous amount of computational time and effort so that only a very small system can be tackled. The other is the continuum mechanics model. Natsuki et al. [12] adopted a simplified Flügge shell model to investigate the wave

propagation of single- and double-walled CNTs conveying fluid. The single-elastic beam model [13-14] and the multiple-elastic beam model [15-19] were also broadly adopted to study the dynamic behaviors of fluid-conveying single-walled carbon nanotubes (SWCNTs) and multi-walled carbon nanotubes (MWCNTs). The vibration frequencies of the linear system and the system's stability related to the internal moving fluid were investigated. Moreover, the nonlocal elasticity theory was incorporated into the elastic beam model to study the small scale effect on the dynamics of SWCNT conveying fluid [20]. Chang and Liu [21-22] studied small scale effects on the flow-induced instability of double-walled carbon nanotubes (DWCNTs) by using the nonlocal elasticity theory. More recently, Chang [23-24] investigated the thermal-mechanical vibration and instability of fluid-conveying single-walled carbon nanotubes (SWCNTs) based on nonlocal elasticity theory. Generally speaking, the beam models mentioned above are linear; however, the vdW forces in the interlay space of MWCNTs are essentially nonlinear. Furthermore, the slender ratios are normally large if the beam models are adopted, that is, the large deformation will occur. Therefore, it is quite essential to consider two types of nonlinear factors, namely, the geometric nonlinearity and the nonlinearity of vdW force in investigating the dynamic behaviors of fluid-conveying MWCNTs. Kuang et al. [25] investigated the dynamic behaviors of double-walled carbon nanotubes (DWCNTs) conveying fluid by considering two types of nonlinearities mentioned above. Salvetat et al. [26] measured the flexural Young's modulus and shear modulus using AFM test on clamped-clamped nanoropes, getting values with 50% of error. Information related to statistical distributions of experimental data is also rare, and the important study from Krishnan et al. [27] provides one of the few examples available of histogram distribution of the flexural Young's modulus derived from 27 CNTs. The Young's modulus was estimated observing free-standing vibrations at room temperature using transmission electron microscope (TEM), with a mean value of 1.3 TPa - 0.4 TPa/+0.6 TPa. Pronouncedly, in [28], stochastically averaged probability amplitude for the vibration modes is computed to obtain the rms vibration profile along the length of the tubes. Uncertainty is also associated to the equivalent atomistic-continuum models adopted extensively in particular by the engineering and materials science communities. Hence, to be realistic, the Young's modulus of elasticity of carbon

nanotube (CNTs) should be considered as stochastic with respect to the position to actually describe the random property of the CNTs under certain conditions. In the present study, we investigate the stochastic dynamic behaviors of nonlinear vibration of the double-walled carbon nanotubes (DWCNTs) conveying fluid by considering the effects of the geometric nonlinearity and the nonlinearity of van der Waals (vdW) force. In addition, the small scale effects on the nonlinear vibration of the DWCNTs are studied by using the theory of nonlocal elasticity. Based on the Hamilton's principle, the nonlinear governing equations of the fluid-conveying double-walled carbon nanotubes are formulated. The Young's modulus of elasticity of the DWCNTs is considered as stochastic with respect to the position to actually characterize the random material properties of the DWCNTs. The effects of the flow velocity and small scale coefficients on the statistical dynamic response of the DWCNTs are investigated.

## 2 Nonlinear beam model for fluid-conveying DWCNTs

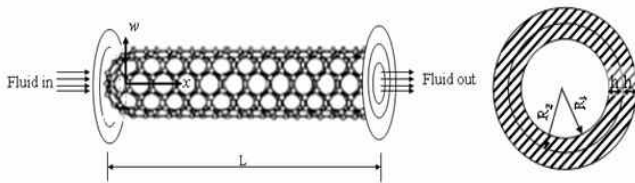


Fig. 1. Double-walled carbon nanotubes conveying fluid.

In Fig. 1, the double-walled carbon nanotubes (DWCNTs) is modeled as a double-tube pipe which is composed of the inner tube of radius  $R_1$  and the outer tube of radius  $R_2$ . The thickness of each tube is  $h$ , the length is  $L$ , and Young's modulus of elasticity is  $E$ . It is noted that the Young's modulus of elasticity  $E$  is assumed as stochastic with respect to the position to actually describe the random material property of the DWCNTs. The internal fluid is assumed to flow steadily through the inner tube with a constant velocity  $U$ . Besides, the boundary conditions of the DWCNTs are assumed as simply-supported at both ends. Based on the theory of Euler-Bernoulli beam and a nonlinear strain-displacement relationship of Von Karman type, the displacement field and strain-displacement relation can be written as follows:

$$\begin{aligned}\bar{u}_i(x, z, t) &= u_i(x, t) - z \frac{\partial w_i}{\partial x} \\ \bar{w}_i(x, z, t) &= w_i(x, t) \\ \varepsilon_i &= \frac{\partial \bar{u}_i}{\partial x} + \frac{1}{2} \left( \frac{\partial \bar{w}_i}{\partial x} \right)^2\end{aligned}\quad (1)$$

where  $x$  is the axial coordinate,  $t$  is time,  $\bar{u}_i$  and  $\bar{w}_i$  denote the total displacements of the  $i$ th tube along the  $x$  coordinate directions,  $u_i$  and  $w_i$  define the axial and transverse displacements of the  $i$ th tube on the neutral axis,  $\varepsilon_i$  the corresponding total strain, and the subscript  $i = 1$  and  $i = 2$ . Notice that tube 1 is the inner tube while tube 2 is the outer tube.

Based on Eq. (1), the potential energy  $V$  stored in a DWCNTs and the virtual kinetic energy  $T$  in the DWCNTs as well as the fluid inside the DWCNTs can be individually determined.

Based on Hamilton's principle, the variational form of the equations of motion for the DWCNTs can be given by

$$\int_{t_0}^{t_1} (\delta V - \delta T - \delta \Psi) dt = 0 \quad (2)$$

where  $\delta \Psi$  is the virtual work due to the vdW interaction and the interaction between tube 1 and the flowing fluid.

Based on Eq. (2) and the formulations derived by Chang [22, 24], the coupled nonlinear governing equations for the free vibration of DWCNTs conveying fluid based on nonlocal elasticity theory are given as follows:

$$\begin{aligned}& \frac{\partial^2}{\partial x^2} \left\{ [E(x)I_1 - (e_0 a)^2 MU^2] \frac{\partial^2 w_1}{\partial x^2} \right\} - 2(e_0 a)^2 MU \frac{\partial^4 w_1}{\partial x^2 \partial t} - (e_0 a)^2 (M + m_1) \frac{\partial^4 w_1}{\partial x^2 \partial t^2} \\ & + MU^2 \frac{\partial^2 w_1}{\partial x^2} - \int_0^L \left( \frac{\partial w_1}{\partial x} \right)^2 \left( \frac{E(x)A_1}{2L} + \frac{MU^2}{2L} \right) dx \frac{\partial^2 w_1}{\partial x^2} + \frac{3MU^2}{2} \left( \frac{\partial w_1}{\partial x} \right)^2 \frac{\partial^2 w_1}{\partial x^2} + \\ & (M + m_1) \frac{\partial^2 w_1}{\partial t^2} + 2MU \frac{\partial^2 w_1}{\partial x \partial t} - MU \frac{\partial w_1}{\partial t} \frac{\partial w_1}{\partial x} \frac{\partial^2 w_1}{\partial x^2} \\ & = \left( 1 - (e_0 a)^2 \frac{\partial^2}{\partial x^2} \right) \left( c_1 (w_2 - w_1) \right) + c_3 \left\{ \left[ \left( 1 - (e_0 a)^2 \frac{\partial^2}{\partial x^2} \right) w_2 \right] - \left[ \left( 1 - (e_0 a)^2 \frac{\partial^2}{\partial x^2} \right) w_1 \right] \right\}^3\end{aligned}\quad (3)$$

$$\begin{aligned}& \frac{\partial^2}{\partial x^2} \left\{ [E(x)I_2] \frac{\partial^2 w_2}{\partial x^2} \right\} + m_2 \frac{\partial^2 w_2}{\partial t^2} - (e_0 a)^2 m_2 \frac{\partial^4 w_2}{\partial x^2 \partial t^2} - \int_0^L \left( \frac{\partial w_2}{\partial x} \right)^2 \left( \frac{E(x)A_2}{2L} \right) dx \frac{\partial^2 w_2}{\partial x^2} \\ & = -(1 - (e_0 a)^2 \frac{\partial^2}{\partial x^2}) (c_1 (w_2 - w_1)) - c_3 \left\{ \left[ \left( 1 - (e_0 a)^2 \frac{\partial^2}{\partial x^2} \right) w_2 \right] - \left[ \left( 1 - (e_0 a)^2 \frac{\partial^2}{\partial x^2} \right) w_1 \right] \right\}^3\end{aligned}\quad (4)$$

It is noted that the scale  $e_0 a$  in the Eq. (3-4) will lead to small scale effect on the response of structures in nano-size. In Eqs. (3-4), it is assumed that the small scale effects on the

nonlinear terms due to geometrical nonlinearity are neglected since they are normally small compared with those on the linear terms.

### 3 Stochastic dynamic analysis of nonlinear vibration of DWCNTs

In the present study, the Young's modulus of elasticity  $E(x)$  is considered as stochastic with respect to the position to actually characterize the random properties of the DWCNTs and it is assumed as Gaussian distributed. Applying the perturbation technique on the Young's modulus of elasticity  $E(x)$ , the following equations can be written:

$$E(x) = E^0(x) + \varepsilon E^I(x) + \dots \quad (5)$$

where  $E^0(x)$  is the mean value of the Young's modulus of elasticity  $E(x)$ ,  $\varepsilon$  is a zero-mean small parameter, and  $\varepsilon E^I(x)$  is the first variation of the Young's modulus of elasticity  $E(x)$ . Similarly, the displacement  $w_1(x), w_2(x)$  of the DWCNTs can be written as follows:

$$w_1(x) = w_1^0(x) + \varepsilon w_1^I(x) + \dots \quad (6)$$

$$w_2(x) = w_2^0(x) + \varepsilon w_2^I(x) + \dots \quad (7)$$

where  $w_1^0(x), w_2^0(x)$  are the mean values of displacement of the inner and outer tubes separately. Substituting Eqs. (5-7) into Eqs. (3-4), we can obtain the following coupled equations based on the zero order of  $\varepsilon$ :

$$\begin{aligned} & E^0 I_1 \frac{\partial^4 (w_1^0)}{\partial x^4} + MU^2 \frac{\partial^2 (w_1^0)}{\partial x} - \left( \frac{MU^2}{2L} + \frac{E^0 A_1}{2L} \right) \int_0^L \left( \frac{\partial (w_1^0)}{\partial x^2} \right)^2 dx \frac{\partial^2 (w_1^0)}{\partial x^2} \\ & + \frac{3MU^2}{2} \left( \frac{\partial w_1^0}{\partial x} \right)^2 \left( \frac{\partial^2 w_1^0}{\partial x^2} \right) + (M + m_1) \frac{\partial^2 (w_1^0)}{\partial t^2} \\ & + 2MU \frac{\partial^2 w_1^0}{\partial x \partial t} - MU \left( \frac{\partial w_1^0}{\partial t} \right) \left( \frac{\partial w_1^0}{\partial x} \right) \left( \frac{\partial^2 w_1^0}{\partial x^2} \right) \\ & = c_1 (w_2^0 - w_1^0) + c_3 (w_2^0 - w_1^0)^3 \end{aligned} \quad (8)$$

$$\begin{aligned} & E^0 I_2 \frac{\partial^4 (w_2^0)}{\partial x^4} + m_2 \frac{\partial^2 (w_2^0)}{\partial t^2} - \frac{E^0 A_2}{2L} \int_0^L \left( \frac{\partial w_2^0}{\partial x} \right)^2 dx \frac{\partial^2 (w_2^0)}{\partial x^2} \\ & = -c_1 (w_2^0 - w_1^0) - c_3 (w_2^0 - w_1^0)^3 \end{aligned} \quad (9)$$

First of all, we have to solve  $w_1^0, w_2^0$  in Eqs. (8-9). By applying the harmonic balance method and Galerkin's method and substituting  $w_i^0 = A_i \phi_i(x) \sin(\omega t)$  ( $i=1, 2$ ) into Eqs. (8-9), after some tedious derivations the relationship between

the amplitude  $A_i$  and the resonant frequency  $\omega$  of the lowest-order mode  $\phi_1(x)$  can be achieved as follows

$$G_1 A_1 + G_2 A_1^3 + G_3 (A_2 - A_1) + G_4 (A_2 - A_1)^3 = 0 \quad (10)$$

$$G_5 A_2 + G_6 A_2^3 + G_3 (A_2 - A_1) + G_4 (A_2 - A_1)^3 = 0 \quad (11)$$

where  $G_i$  are constants which can be determined by performing the integration.

After solving coupled Eqs. (8-9) for the amplitudes  $A_1, A_2$ , we can obtain  $w_1^0, w_2^0$  readily. Substituting  $w_1^0, w_2^0$  into nonlinear coupled differential equations based on the first order of  $\varepsilon$ , and adopting the same technique for solving  $w_1^I, w_2^I$ , finally we can obtain  $w_1^I, w_2^I$  without any difficulties except the derivations are somewhat lengthy.

### 4 Numerical examples and discussion

In the numerical computations, the simply supported boundary condition is considered for the DWCNTs conveying fluid. The inner and the outer tubes are assumed to have the same Young's modulus, the same thickness and the same mass density. The numerical values of the parameters are adopted as follows:

Mean value of Young's modulus  $E=1$  Tpa, tube thickness  $h=0.34$  nm, mass density  $\rho = 2300 \text{ Kg/m}^3$ , the mass density of water flow is  $\rho_f = 1000 \text{ Kg/m}^3$ , the inner radius  $R_1 = 0.7 \text{ nm}$  and the outer radius  $R_2 = 1.04 \text{ nm}$  and

mean square values of  $\varepsilon$  is assumed as  $E[\varepsilon^2] = 0.01$ . The

velocity of the fluid is assumed as  $U=400$  m/sec unless it is specified otherwise. First of all, we examine the effect of the nonlinearity on the amplitude-frequency properties of the nonlinear vibration. The relations of the mean value of amplitude versus frequency are depicted in Fig. 2. It can be seen that the mean value of the amplitude increases with the increase of the frequencies. It is completely reasonable that the relation between the mean value of the amplitude and the frequency is nonlinear; in addition, the mean value of the amplitude of the outer tube is larger than that of the inner tube. Furthermore, it is noted that the mean value of the amplitude gets smaller as the small scale coefficient  $e_0 a$  increases for the fixed frequency. In Fig. 3, the standard deviation of the amplitude is plotted with respect to the frequency. As it can be found from the figure that the standard deviation of the amplitude increases nonlinearly with the increase of the frequencies, and it is noted that the standard deviation of the amplitude of the outer tube is larger than that of the inner tube. In Fig. 4, the coefficient of variation (COV) of the amplitude is depicted with respect to the frequency. It is noticed that the

coefficient of variation of the amplitude of the inner tube is around 0.10, however, the coefficient of variation of the amplitude of the outer tube is around 0.12. Finally, Fig. 5 presents the coefficient of variation of amplitude versus frequency with  $e_0a/L = 0.1$  for different values of flow velocity. It is found that the coefficient of variation of amplitude fluctuates between 0.10 and 0.12 and no specific relation between COV and flow velocity can be established despite they are correlated. Therefore, based on the results from Figs. 2-5, it can be concluded that the small scale coefficient has significant influence on the mean value, standard deviation and coefficient of variation of the amplitude of the DWCNTs, however, the flow velocity has only a little effect on the stastical dynamic response of the DWCNTs.

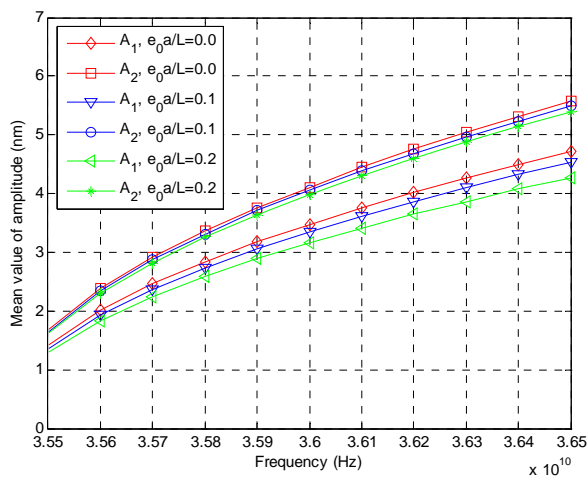


Fig. 2. Mean value of amplitude versus frequency for different values of  $e_0a/L$ .

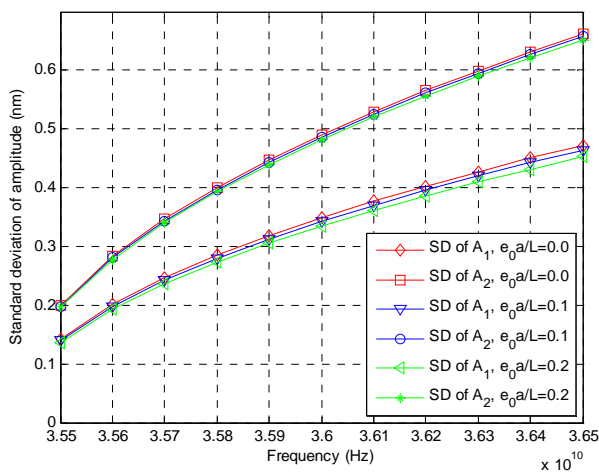


Fig. 3. Standard deviation of amplitude versus frequency for different values of  $e_0a/L$ .

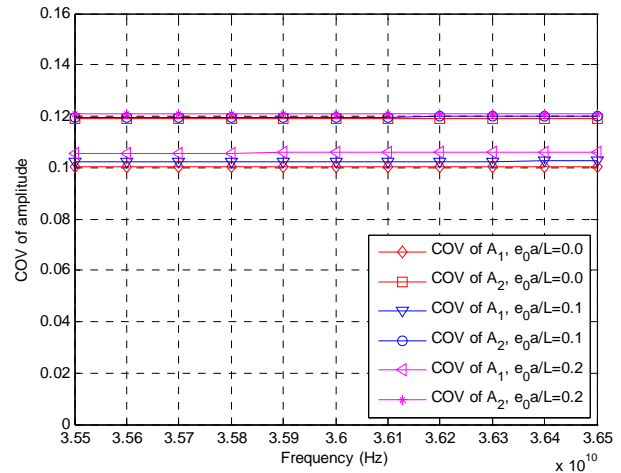


Fig. 4. Coefficient of variation of amplitude versus frequency for different values of  $e_0a/L$ .

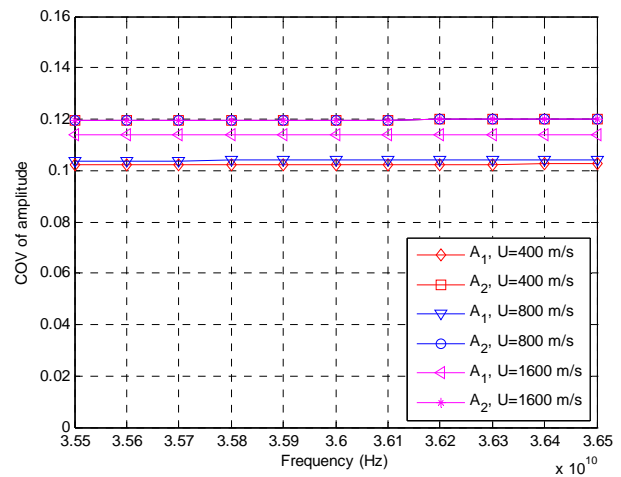


Fig. 5. Coefficient of variation of amplitude versus frequency with  $e_0a/L = 0.1$  for different values of flow velocity.

## 5 Conclusions

In the present study, we investigate the stochastic dynamic behaviors of nonlinear vibration of the double-walled carbon nanotubes (DWCNTs) conveying fluid by considering the effects of the geometric nonlinearity and the nonlinearity of van der Waals (vdW) force. In addition, the small scale effects of the nonlinear vibration of the DWCNTs are studied by using the theory of nonlocal elasticity. Based on the Hamilton's principle, the nonlinear governing equations of the fluid-conveying double-walled carbon nanotubes are formulated. The Young's modulus of elasticity of the DWCNTs is considered as stochastic with respect to the

position to actually characterize the random material properties of the DWCNTs. By using the perturbation technique, the nonlinear governing equations of the fluid-conveying double-walled carbon nanotubes can be decomposed into two sets of nonlinear differential equations involving the mean value of the displacement and the first variation of the displacement separately. Then the harmonic balance method and Galerkin's method are adopted to solve the nonlinear differential equations successively. Some statistical dynamic response of the DWCNTs such as the mean values and standard deviations of the amplitude of the displacement are calculated, meanwhile the effects of the flow velocity and nonlocal scale coefficients on the statistical dynamic response of the DWCNTs are investigated. It can be concluded that the mean value and standard deviation of the amplitude of the displacement increase nonlinearly with the increase of the frequencies. Besides, these stochastic dynamic responses change slightly as the flow velocity increases, furthermore, they are smaller as the small scale coefficients get larger. However, as the values of flow velocity or small scale coefficients increase, the coefficients of variation (COV) of the amplitude of the displacement remain almost constant and stay within certain range with respect to the frequency. It is noted that the computed stochastic dynamic response plays an important role in estimating the structural reliability of the DWCNTs.

**Acknowledgments** This research was partially supported by the National Science Council in Taiwan through Grant NSC-99-2221-E-327-020. The author is grateful for this support.

## 6 References

- [1] S. Iijima, Helical microtubules of graphitic carbon, *Nature* 354 (1991) 56-58.
- [2] E. Evans, H. Bowman, A. Leung, D. Needham, D. Tirrell, Biomembrane templates for nanoscale conduits and networks, *Science* 273 (1996) 933-935.
- [3] G.E. Gadd et. al., The World's Smallest Gas Cylinders?, *Science* 277 (1997) 933-936.
- [4] G. Che et. al., Carbon nanotubule membranes for electrochemical energy storage and production, *Nature* 393 (1998) 346-349.
- [5] J. Liu et. al., Fullerene Pipes, *Science* 280 (1998) 1253-1256.
- [6] A. Karlsson et. al., Networks of nanotubes and containers, *Nature* 409 (2001) 150-152.
- [7] Y. Gao, Y. Bando, Carbon nanothermometer containing gallium, *Nature* 415 (2002) 599.
- [8] G. Hummer, J.C. Rasaiah, J.P. Noworyta, Water conduction through the hydrophobic channel of a carbon nanotube, *Nature* 414 (2001) 188-190.
- [9] I. Hanasaki, A. Nakatani, Water flow through carbon nanotube junctions as molecular convergent nozzles, *Nanotechnology* 17 (2006) 2794-2804.
- [10] Z. Mao, S.B. Sinnott, A computational study of molecular diffusion and dynamic flow through carbon nanotubes, *J. Phys. Chem. B* 104 (2000) 4618-4624.
- [11] A. Skoulidas, D.M. Ackerman, K.J. Johnson, D.S. Sholl, Rapid transport of gases in carbon nanotubes, *Phys. Rev. Lett.* 89 (2002) 185901-185911.
- [12] T. Natsuki, Q.Q. Ni, M. Endo, Wave propagation in single- and double-walled carbon nanotubes filled with fluids, *J. Appl. Phys.* 101 (2007) 034319-034319.
- [13] J. Yoon, C.Q. Ru, A. Mioduchowski, Flow-induced flutter instability of cantilever CNTs, *Int. J. Solids Struct.* 43 (2006) 3337-3349.
- [14] L. Wang, Q. Ni, On vibration and instability of carbon nanotubes conveying fluid, *Comput. Mater. Sci.* 43 (2008) 399-402.
- [15] X.Q. He, C.M. Wang, Y. Yan, L.X. Zhang, G.H. Ni, Pressure dependence of the instability of multiwalled carbon nanotubes conveying fluids, *Arch. Appl. Mech.* 78 (2008) 637-648.
- [16] Y. Yan, X.Q. He, L.X. Zhang, C.M. Wang, Dynamic behavior of triple-walled carbon nanotubes conveying fluid, *J. Sound Vib.* 319 (2009) 1003-1018.
- [17] L. Wang, Q. Ni, M. Li, Q. Qian, The thermal effect on vibration and instability of carbon nanotubes conveying fluid, *Physica E* 40 (2008) 3179-3182.
- [18] Y. Yan, W.Q. Wang, L.X. Zhang, Dynamical behaviors of fluid-conveyed multi-walled carbon nanotubes, *Appl. Math. Modell.* 33 (2009) 1430-1440.
- [19] L. Wang, Q. Li, M. Li, Buckling instability of double-wall carbon nanotubes conveying fluid, *Comput. Mater. Sci.* 44 (2008) 821-825.
- [20] H. Lee, W. Chang, Comment on Free transverse vibration of the fluid-conveying single-walled carbon nanotube using nonlocal elastic theory, *J. Appl. Phys.* 103 (2008) 024302-024302.
- [21] T.P. Chang, M.F. Liu, Flow-induced instability of double-walled carbon nanotubes based on nonlocal elasticity theory, *Physica E* 43 (2011) 1419-1426.
- [22] T.P. Chang, M.F. Liu, Small scale effect on flow-induced instability of double-walled carbon nanotubes, *Eur. J. Mech. A. Solids* 30 (2011) 992-998.
- [23] T.P. Chang, Thermal-nonlocal vibration and instability of single-walled carbon nanotubes conveying fluid, *J. Mech.* 27 (2011) 567-573.
- [24] T.P. Chang, Thermal-mechanical vibration and instability of a fluid-conveying single-walled carbon nanotube embedded in an elastic medium based on nonlocal elasticity theory, *Appl. Math. Model.* 36 (2012) 1964-1973.
- [25] Y.D. Kuang et al, Analysis of nonlinear vibrations of double-walled carbon nanotubes conveying fluid, *Comput. Mater. Sci.* 45 (2009) 875-880.
- [26] J.P. Salvetat, J.A.D. Briggs, J.M. Bonard, R.R. Bacsa, A.J. Kulik, T. Stöckli, N.A. Burnham, L. Forró, Elastic

and shear moduli of single-walled carbon nanotube ropes, Phys. Rev. Lett. 82 (5) (1999) 944-947.

- [27] A. Krishnan, E. Dujardin, T.W. Ebbesen, P.N. Yianilos, M.M.J. Treacy, Young's modulus of single-walled nanotubes, Phys. Rev. B 58 (20) (1998) 14013-14019.
- [28] A. J. Mieszawska, R. Jalilian, G. U. Sumanasekera, F. P. Zamborini, The synthesis and fabrication of one-dimensional and nanoscale heterojunctions, Small 3 (2007) 722-756.

# Simulation of Fully-Developed Turbulent MHD Pipe Flow

Khalid N. Alammam<sup>1, a</sup>, Maher M. Shariff<sup>2</sup>, Regis D. Vilagines<sup>2</sup>, Zakariya M. Kaneesamkandi<sup>1</sup>, and Shaker S. Abdullah<sup>1</sup>

<sup>1</sup>Mechanical Engineering Department, King Saud University, Riyadh, Saudi Arabia

<sup>2</sup>R&D Center, Saudi Aramco, Dhahran, Saudi Arabia

**Abstract** - Using an eddy viscosity turbulence model, fully-developed turbulent MHD pipe flow was simulated. Uncertainty was approximated through grid-independence and model validation. Effect of Reynolds and Hartmann numbers on flow characteristics was investigated. Typical characteristics of flow structure in MHD flow were observed, including evolution of the velocity profiles with the magnetic field strength and formation of the M-shaped velocity distribution. The core region was shown to be increasingly retarded with increasing Lorentz force leading to the formation of side layers. Wall friction was shown to increase with increasing Hartmann number. The effect of Hartmann number was found to diminish with increasing Reynolds number.

**Keywords:** Turbulence, Magneto-Hydrodynamics, Hartmann number, Lorentz force

## Nomenclature

$\bar{B}$	magnetic field, T
$B_0$	magnetic field magnitude, T
DNS	direct numerical simulation
$E$	electric field, N/C
$J$	electric current density, A/m <sup>2</sup>
MHD	magnetohydrodynamic
$u$	average velocity, m/s
$p$	pressure, Pa
$r_0$	pipe radius, m
$r$	radial distance, m
RANS	Reynolds averaged Navier Stokes
$x, y$	coordinates, m

## Non-dimensional Parameters

Re	Reynolds number	$= \rho u D / \mu$
R	Modified Reynolds number	$= Re / Ha$
Ha	Hartmann Number	$= BD / \sqrt{\sigma / \mu}$
N	Stuart number	$= Ha^2 / Re$

## Greek Symbols

$\rho$	fluid density, Kg/m <sup>3</sup>
$\mu$	fluid dynamic viscosity Kg/(m s)
$\phi$	electric potential, volt
$\sigma$	fluid electric conductivity (ohm <sup>-1</sup> /m)
$\tau$	shear stress (N/m <sup>2</sup> )

## 1 Introduction

One of the techniques for controlling turbulent boundary layers consists of acting on the flow by means of electromagnetic force (Lorentz force). The technique is relatively old and traces its origin at least in the sixties, Gold [1], Shercliff [2], and Moffat [3]. Applications of MHD flow include heat transfer enhancement, drag reduction, power generation, and fusion reactors. A different class of applications is found in blood flow, micro pumps, porous media, chemical reactions, and material production. A relatively recent review on fusion reactors and MHD flow is found in Moreley [4].

Due to fundamental and practical importance, effect of MHD on turbulence has drawn special attention. Moffatt [3] was the first to show analytically that turbulent velocity fluctuations can be suppressed by application of a uniform magnetic field. Fraim and Heiser [5] investigated the effect of a strong longitudinal magnetic field on the turbulent flow of mercury in a circular tube. Depending on the intensity of the applied field, it was reported that the magnetic field had an important effect on the generation of turbulence, as well as damping of already existing turbulence. Early experiments include the work of Hensch and Stace [6] where it was shown that an MHD force, when applied to an electroconducting fluid and acting in a streamwise direction, can generate a near-wall jet, decreasing the boundary layer thickness and suppressing the intensity of the turbulent fluctuations across the boundary layer. At very high interactions, the force causes an increase in mean wall shear and turbulence.

Berger et al. [7] investigated the effect of MHD on turbulent saltwater channel flow. It was shown that skin-friction drag can be reduced by approximately 40% if a temporally oscillating spanwise Lorentz force is applied. However, the power to generate the required Lorentz force is an order of

<sup>a</sup> Corresponding Author: Phone: +96614676650, email: alammam@ksu.edu.sa



magnitude larger than the power saved due to the reduced drag.

More recently, A DNS and k- $\epsilon$  model simulation of MHD flow was carried out by Yamamoto et al [8]. In this study, flow and heat transfer characteristics of low-magnetic Reynolds number and Prandtl number fluids were investigated. It was found that the similarity-law between the velocity and the temperature profiles was not satisfied with increasing of Hartman number, noticeably near the critical Hartmann condition to maintain turbulent flow. At higher Reynolds number conditions, MHD models coupled with k- $\epsilon$  model was able to reproduce the MHD pressure loss trend with increasing Hartmann number.

DNS and LE are relatively accurate for modeling turbulent MHD flow; but due to limited computational resources, both remain limited to low-range Reynolds numbers. And because the turbulent boundary layer is not resolved all the way to the wall, the k- $\epsilon$  and similar turbulence models lack the means by which they can accurately predict MHD effect on turbulence. While the universal law of the wall is applicable in absence of MHD effects, such effects modify the turbulent boundary layer so that the universal law of the wall would no longer be applicable. In this study, a zero-equation model which resolves the entire boundary layer is used to model the average turbulent MHD flow. Main objective of the study, therefore, is to validate the turbulence model, and to characterize average turbulent MHD pipe flow in the presence of a transversal magnetic field  $\vec{B}$ , Fig. 1. Several Reynolds and Hartmann numbers are tested.

## 2 Theory

The mathematical model consisted of the following Reynolds-averaged Navier-Stokes (RANS) and electric potential equations.

### 2.1 Continuity equation

$$\frac{\partial u}{\partial z} = 0 \quad (1)$$

### 2.2 Axial momentum

$$\frac{\partial P}{\partial z} + \sigma B_o \frac{\partial \phi}{\partial x} + \sigma u B_o^2 = \frac{\partial}{\partial x} \left[ (\mu + \mu_t) \frac{\partial u}{\partial x} \right] + \frac{\partial}{\partial y} \left[ (\mu + \mu_t) \frac{\partial u}{\partial y} \right] \quad (2)$$

With the eddy viscosity, Alammari [9], given by

$$\mu_t = b \rho u d \quad (3)$$

Where  $b = 0.016$  and  $d$  is the distance from the wall.

### 2.3 Electric potential

$$-B_o \frac{\partial u}{\partial x} = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \quad (4)$$

### 2.4 Assumptions and boundary conditions

The flow is assumed steady, fully developed, and two-dimensional. The effect of magnetic field on turbulence is negligible (small Stuart number). Properties are assumed constant. The boundary conditions include the no-slip condition, given velocity inlet, and electrically insulated wall.

## 3 Numerical Procedure

Fluent 6.1 was used as the solver. The structured grid was built using Gambit 2.0. The mesh consisted of approximately 40,000 hybrid cells. The simulation was carried out using SIMPLE, Patankar and Spalding [10], and second-order schemes. The linearized equations were solved using Gauss-Seidel method, in conjunction with an algebraic Multigrid scheme, Sharov and Nakahashi [11].

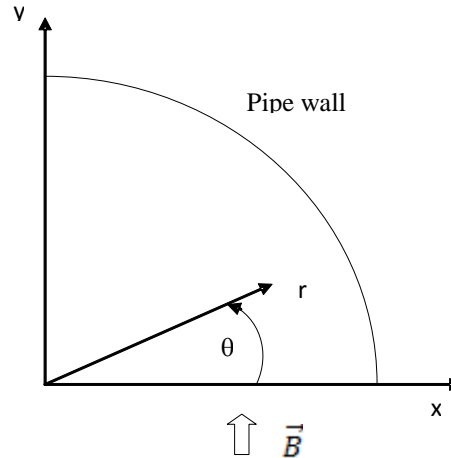


Fig. 1 Cross section of the pipe (first quadrant) with the constant-field vector

## 4 Uncertainty analysis

There are mainly two sources of uncertainty in CFD, namely modeling and numerical, Stern et al. [12]. Modeling uncertainty can be approximated through theoretical or experimental validation while numerical uncertainty can be approximated through grid independence. Numerical uncertainty has two main sources, namely truncation and round-off errors. Higher order schemes have less truncation



error, and as was outlined earlier, the discretization schemes invoked were second-order. In explicit schemes, round-off error increases with increasing iterations, and is reduced by increasing significant digits (machine precision). However, having used Gauss-Seidel iterative procedure in a steady-state simulation renders the calculation insensitive to round-off error.

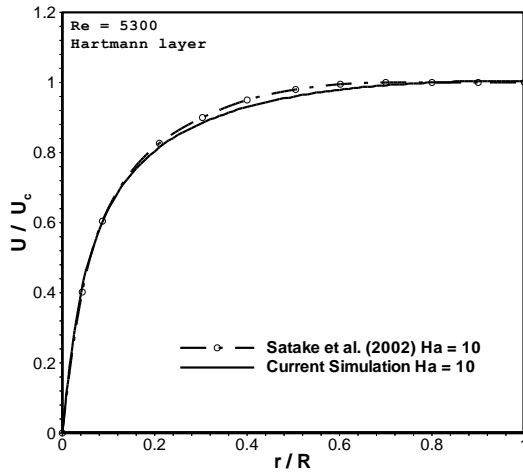


Fig. 2 Mean velocity distribution; Ha = 10

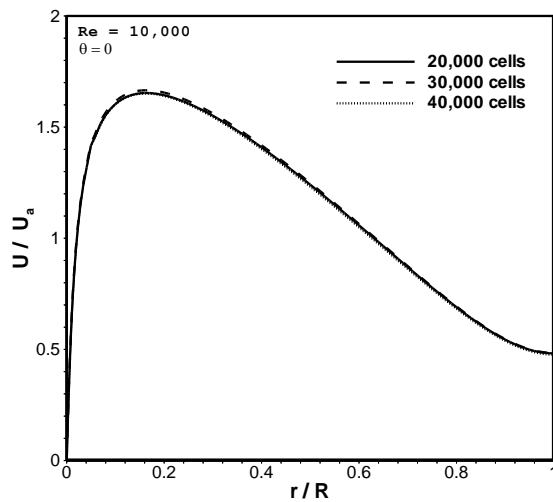


Fig. 3 Mean velocity distribution; Ha = 100

A comparison between the current simulation and DNS of Satake et al.[13] is depicted in Fig. 2. The numerical prediction with 40,000 cells is within  $\pm 5\%$ . Therefore, we assume the modeling uncertainty to be  $\pm 5\%$ . The small discrepancy is likely due to transitional effects which have not been incorporated into the current turbulence model. A grid-

independence test is shown in Fig. 3 where the mean velocity distribution is depicted for Hartmann number of 100. The numerical error is shown to be negligible. Hence, we conclude that the overall uncertainty is  $\pm 5\%$ .

## 5 Results and Discussion

The mean velocity distribution is depicted in Fig. 4 for Reynolds number of  $10^5$  for different Hartmann numbers and orientations. Typical characteristics of flow structure in MHD flow are observed, including evolution of the velocity profiles with the magnetic field strength and formation of the M-shaped and flat velocity distributions. The core region is observed to be increasingly retarded with increasing Lorentz force (Hartmann number) leading to the formation of different side layers.

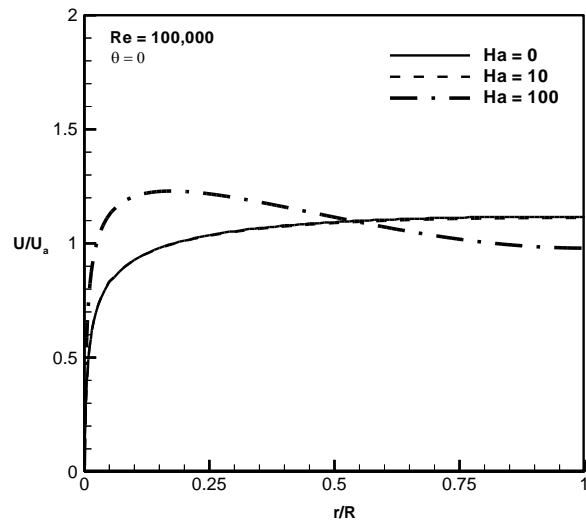


Fig. 4a Mean velocity distribution;  $\theta = 0$ ,  $Re = 10^5$

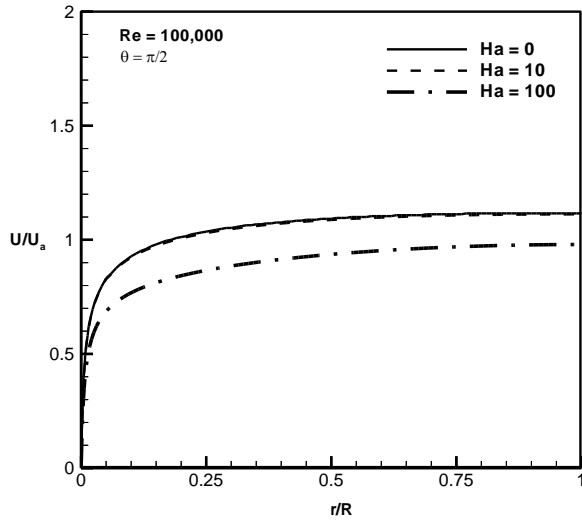


Fig. 4b Mean velocity distribution;  $\theta = \pi/2$ ,  $Re = 10^5$

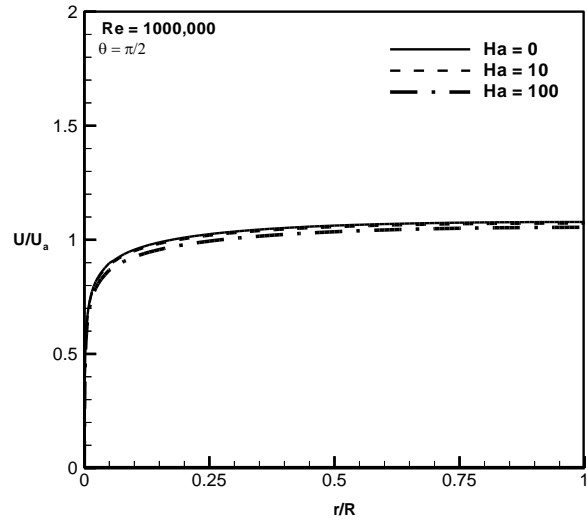


Fig. 5b Mean velocity distribution;  $\theta = \pi/2$ ,  $Re = 10^6$

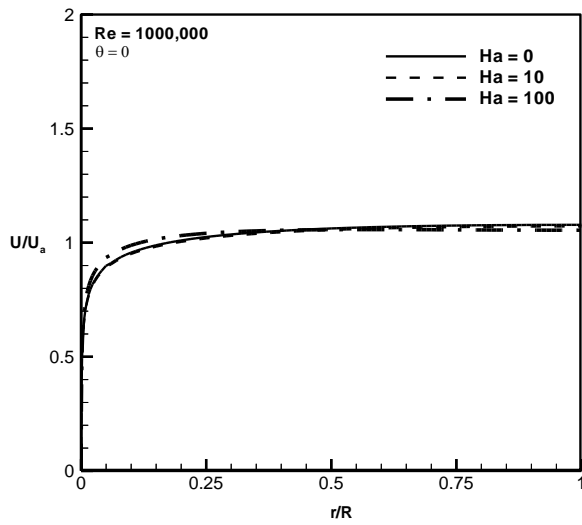


Fig. 5a Mean velocity distribution;  $\theta = 0$ ,  $Re = 10^6$

The mean velocity distribution is shown in Fig. 5 for Reynolds number of  $10^6$  for different Hartmann numbers and azimuth. The effect of Hartmann number is shown to diminish with increasing Reynolds number. This is due to increasing dominance of the inertial forces relative to the Lorentz force at higher Reynolds numbers.

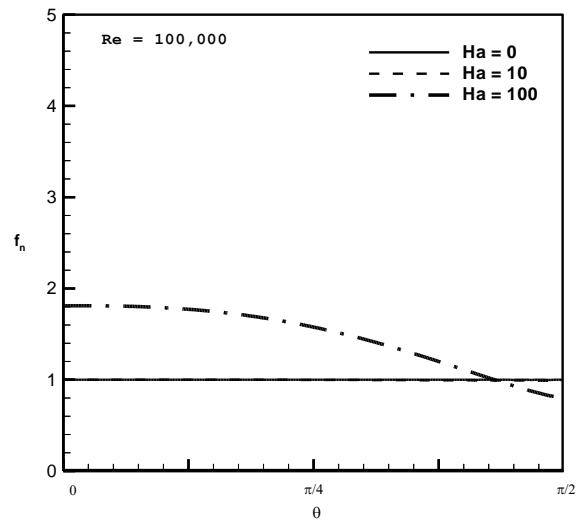


Fig. 6b Friction factor distribution;  $Re = 10^5$

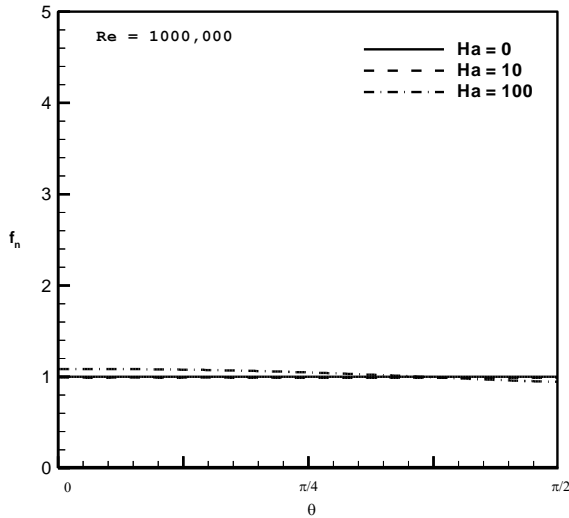


Fig. 6a Friction factor distribution;  $Re = 10^6$

The friction factor distribution is depicted in Fig. 6 for different Reynolds and Hartmann numbers. The profiles are normalized by the baseline case ( $Ha = 0$ ). The wall friction is shown to increase with increasing field strength. Due to decreasing boundary layer thickness, the friction factor is shown to increase towards the side layer. Again, the effect of increasing Reynolds number is to diminish the role of Lorentz force.

## 6 Conclusions

Using a zero-equation turbulence model, a fully-developed turbulent MHD pipe flow was simulated. Effect of Reynolds and Hartmann numbers on flow characteristics was investigated. Typical characteristics of flow structure in MHD flow were observed, including evolution of the velocity profiles with the magnetic field strength and formation of the Hartmann and side layers. The core region was observed to be increasingly retarded with increasing Lorentz force leading to the formation of different layers. The wall friction was shown to increase with increasing field strength. The effect of Hartmann number was shown to diminish with increasing Reynolds number.

## 7 Acknowledgments

This research was sponsored by Saudi ARAMCO under Contract NO. 6600023423. The authors are very grateful for Saudi ARAMCO's support of this research

## 8 References

[1] GOLD, R.R.: Magnetohydrodynamic Pipe Flow, Part 1, *J. Fluid Mech.* 13, 505-512 (1962)

[2] Shercliff, J.A.: Magnetohydrodynamic Pipe Flow Part 2, High Hartmann Number, Department of Engineering, University of Cambridge (1962)

[3] Moffat, H.K.: On the suppression of turbulence by a uniform magnetic field. *J. Fluid Mech* 28, 571-592 (1967)

[4] Morley, N.B., Smolentsev, S., Barleon, L., Kirillov, I.R., and Takahashi, M.: Liquid magnetohydrodynamics-recent progress and future directions for fusion, *Fusion Engineering and Design.* 51-52, 701-713 (2000)

[5] Fraim, F.W. and Heiser, W.H.: The effect of a strong longitudinal magnetic field on the flow of mercury in a circular tube. *Journal of Fluid Mechanics.* 33, 397-413 (1968)

[6] Henoeh, C and Stace, J.: Experimental investigation of a salt water turbulent boundary layer modified by an applied streamwise magnetohydrodynamic body force, *Physics of Fluids.* A7, 1371-1383 (1994)

[7] 8. Berger, T.W., Kim, J., Lee, C., and Lim, J.: Turbulent boundary layer control utilizing the Lorentz force, *Physics of Fluids.* 12, Issue 3, 631-649 (2000)

[8] Yamamoto, Y., Kunugi, T., Satake, S., and Smolentsev, S.: DNS and  $k-\epsilon$  model simulation of MHD turbulent channel flows with heat transfer, *Fusion Engineering and Design.* 83, Issues 7-9, 1309-1312 (2008)

[9] Alammr, K.: Turbulence: A new zero-equation model, 7th International Conference on Advancements in Fluid Mechanics, Wessex Institute of Technology, New Forest, UK, 21-23 May, (2008)

[10] Patankar, S.V. and Spalding, D.B.: A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows, *International Journal of Heat and Mass Transfer.* 15, 1787-1806 (1972)

[11] Sharov, D. and Nakahashi, K.: Hybrid Prismatic/Tetrahedral Grid Generation for Viscous Flow Applications, *AIAA Journal.* 36, 157-162, (1998)

[12] Stern, F., Wilson, R., Coleman, H., and Paterson, E: Verification and Validation of CFD Simulations, Iowa Institute of Hydraulic Research IIHR 407, College of Engineering, University of Iowa, Iowa City, IA, USA. 3, (1999)

[13] Satake, S., Kunugi, T., and Smolentsev, S.: Direct numerical simulations of turbulent pipe flow in a transverse magnetic field, *Journal of Turbulence.* 3, N20, (2002)

# An interface reconstruction method to deal with filaments in multi-material simulations

C. Fochesato<sup>1</sup>, R. Loubère<sup>2</sup>, R. Motte<sup>1</sup>, and J. Ovidia<sup>3</sup>

<sup>1</sup>CEA, DAM, DIF, F-91297 Arpajon, France

<sup>2</sup>Institut de Mathématiques de Toulouse, CNRS, Université de Toulouse, Toulouse, France

<sup>3</sup>Retired fellow from CEA, CESTA, F-33114 Le Barp, France

**Abstract**—*In this work, we propose a method for a two fluids interface reconstruction which is able to detect and deal with filaments and small fluid structures in 2D multi-material simulations. It is an extension of the usual VoF method, which remains the chosen method for most of the mixed cells of the mesh. Only for detected cells, the new algorithm is applied in order to improve the representation and advection of small filament structures.*

**Keywords:** interface reconstruction, multi-material simulations, filaments, VoF method, Youngs' method.

## 1. Introduction

Actually, most of interface reconstruction techniques assume that a mixed cell can be split into pure regions separated by a straight line. The straight line is usually computed as the perpendicular to a direction being the gradient of the volume fraction function. However, if the fluid is elongated enough, its characteristics size may drop below the characteristics size of the cell. The fluid element then becomes a filament with respect to cell size. Consequently, an accurate representation of fluid interface cannot be made with only one straight line per mixed cell. Typically, a classical Youngs' method [3] is known to have a behaviour similar to a surface tension effect but at mesh scale. We can say that the method has a too large numerical surface tension.

## 2. Subgradients Method

The idea is to compute subgradients in a cell from a local stencil associated with each corner ( $2 \times 2$  cells for instance) in order to reconstruct up to 4 interfaces, one per subzone, with normals given by these subgradients, and then to reconstruct a straight line by subzone (Figure 1). The method does not contain more physics. It is only another geometrical choice that reduces the numerical surface tension, so that small filaments are maintained longer through the mesh. A previous method, using such a subdivision strategy for improving accuracy of reconstruction results, has been described by B. Rebouret [1]. A similar idea is followed here for the subdivision and then used to deal with filaments by analyzing the volume fractions field and constructing subcell gradients.

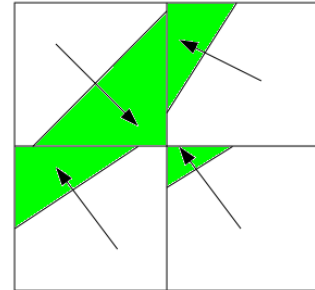


Figure 1: Subdivision of a cell containing a filament.

A summary of the algorithm is as follows:

- analyzing volume fraction gradients to detect such possible pathological situations,
- evaluating and selecting pertinent subcell gradients,
- defining 1, 2, 3, or 4 reconstruction subzones within the mixed cell,
- scattering the fluids onto the reconstruction subzones,
- constructing one straight line interface per subzones.

Such a technique is to be coupled with a classical interface reconstruction technique such as Youngs [3] as nonpathological mixed cells must be treated as usual.

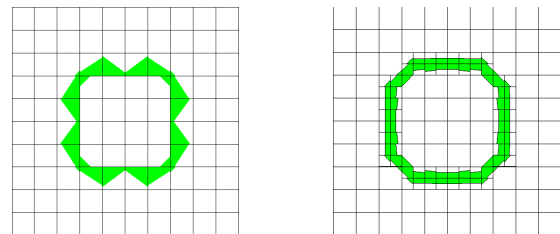


Figure 2: Illustration of a thin circle reconstructed. Left: with the usual VoF method. Right: with the proposed algorithm.

## 3. Numerical results

First numerical results on a quadrangular mesh have shown that the technique is able to deal with complex interfaces in some static scenarios (Figures 2 and 3). However, it appears that for dynamic cases, it is necessary to use more information in order to deal with the advection of a piece

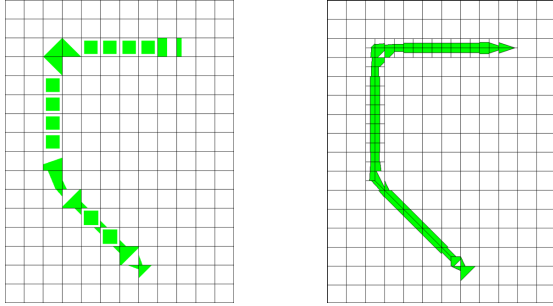


Figure 3: Illustration of a filament reconstructed. Left: with the usual VoF method. Right: with the proposed algorithm.

of fluid whose width is smaller than the cell: the volume fractions field does not allow to specify the relative location of the filament in the cell, and consequently to move it through the cell during a time step.

## 4. Conclusions

A Subgradients method to compute interfaces in subzones has been proposed. At given mesh, it allows better representation of thin structures of fluids.

The proposed improvements from the algorithm described above are:

- define the center of the subdivision in relation with the center of mass of the piece of fluid (Figure 4),
- store the volume fractions field and centroids of the previous time step, and use the velocity field, not only for the global advection of the mesh, but also in order to specify the relative motion of a filament versus the cell.

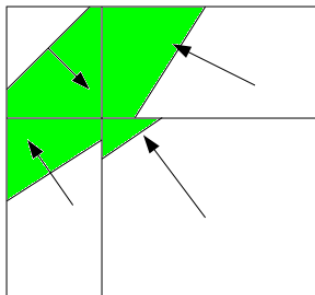


Figure 4: Subdivision of a cell containing a filament. Displaced center of subdivision.

This interface reconstruction method will be compared with a classical Youngs' method to show the improvement. Quantitative results will be measured on usual test cases for interface reconstruction methods such as the stretching of a circle in a vortex velocity field [2].

## References

- [1] B. Rebouret, "CUBIC: a scalable extension of PLIC technics for interface reconstruction", Los Alamos National Laboratory, unpublished, 1995.
- [2] W.J. Rider and D.B. Kothe, "Stretching and tearing interface tracking methods", *AIAA paper*, 95, 1-11, 1995.
- [3] D. Youngs, "Time dependent multi-material flow with large fluid distortion", *Numerical Methods for Fluid Dynamics*, K. W. Morton and M. J. Baines, editors, Academic Press, pp. 273-285, 1982.



**SESSION**  
**MATRIX OPERATIONS AND METHODS**

**Chair(s)**

**TBA**





# Combining Automated Multilevel Sub-structuring and Subspace Iteration for Huge Gyroscopic Eigenproblems

J. Yin<sup>1</sup>, H. Voss<sup>2</sup>, and P. Chen<sup>1</sup>

<sup>1</sup>Department of Mechanics and Aerospace Engineering, Peking University, Beijing, China

<sup>2</sup>Institute of Mathematics, Hamburg University of Technology, Hamburg, Germany

**Abstract**—*The Automated Multilevel Sub-structuring (AMLS) method is a powerful technique for computing a large number of eigenpairs with moderate accuracy for huge definite eigenproblems in structural analysis. It also turned out to be a useful tool to construct a suitable ansatz space for projection methods for gyroscopic problems. This paper improves the eigenpairs obtained with AMLS via a small number of subspace iteration steps. It takes advantage of a transformation of the stiffness matrix to block diagonal form from AMLS while using the original mass and gyroscopic matrices. A numerical example demonstrates the efficiency of this approach and its pronounced superiority upon the the subspace iteration for the original gyroscopic eigenproblem.*

**Keywords:** Eigenvalue, eigenvector, AMLS, subspace iteration, gyroscopic eigenproblem

## 1. Introduction

Simulation of acoustic properties has gained increasing importance in the engineering design process, in particular in automobile industries. Sound radiation from the rolling tires has been identified as a major source of noise generated by vehicles moving at speeds above 50 km/h [1], [2]. Therefore, much effort has been directed into development of methods allowing for a simulation of the effect of tire-road surface interaction.

According to Nackenhorst [1] the simulation of the tire noise is performed in three steps. First, the nonlinear tire deflections under steady state conditions are computed using an Arbitrary Lagrangian Eulerian (ALE) approach. Next, the transient vibrations governed by the eigenpairs of a gyroscopic eigenvalue problem

$$Q(\omega)x := Kx + \omega iGx - \omega^2 Mx = 0. \quad (1)$$

are assumed to be superimposed onto the nonlinear deflections. Finally, the acoustic analysis is carried out solving Helmholtz's equation on the exterior of the domain occupied by the tire where the normal velocities at the wheel surface, extracted from the vibration analysis, are taken as boundary conditions.

In this paper we consider only the second step, i.e. the numerical solution of the eigenproblem (1) where  $K$  is the stiffness matrix modified by the inertia forces due to

the stationary rolling,  $M$  is the mass matrix, and  $G$  is the gyroscopic matrix stemming from the Coriolis force. Clearly,  $K$  and  $M$  are symmetric and positive definite, and  $G$  is skew-symmetric. The eigenvalues  $\omega$  (which are influenced by the rotational speed of the tire) are real, whereas the eigenvectors are complex, and have to be interpreted as traveling waves on the surface of the tire rather than standing vibrations.

Due to the complicated interior structure of a belted tire the matrices  $K$ ,  $M$  and  $G$  of a sufficiently accurate FE model are very large and sparse. Moreover, for the acoustic analysis many eigenpairs (up to 2000 Hz) are needed, when determining the initial conditions for the Helmholtz equation for the third step, which are computed in a Fourier analysis of the tire excitations by the roughness of the road surface.

A common approach for solving the quadratic eigenvalue problem is linearization, i.e. to transform (1) into an equivalent linear eigenvalue problem

$$\begin{aligned} Aq &:= \begin{bmatrix} iG & K \\ K & O \end{bmatrix} \begin{bmatrix} \omega x \\ x \end{bmatrix} \\ &= \omega \begin{bmatrix} M & O \\ O & K \end{bmatrix} \begin{bmatrix} \omega x \\ x \end{bmatrix} =: \omega Bq \end{aligned} \quad (2)$$

and to apply the shift-and-invert Lanczos method as implemented in the software package ARPACK [3]. Then in every iteration step ARPACK interrupts and in reverse communication the user has to supply the solutions of a linear complex valued system  $(A - \sigma B)z = Bc$  for some shift  $\sigma$  and right hand side  $c$ . Although the special structure of the matrices in (2) allows for an efficient solution of these systems this approach requires an excessive amount of storage and computing time (cf. [2]). More efficient for solving the gyroscopic eigenvalue problem (1) than the implicitly restarted Lanczos method is the *Automated Multi-Level Sub-structuring* (AMLS) method [4].

The (AMLS) method, proposed by Bennighof an co-authors [5], [6], is an efficient condensation method for computing hundreds and thousands of eigenmodes and frequency responses for large and complex structures. The standard AMLS has been designed for linear symmetric eigenvalue problems and has been successfully applied to many engineering problems in recent years including vibro-acoustic analysis in automotive industry [7], ship vibrations [8], electromagnetic problems [9], [10], and has been generalized

to gyroscopic problems [4], and vibrations of fluid–solid structures [11]. Details of implementations are contained in [12], [13], [14], [15].

Compared to Krylov type approaches AMLS reduces computational resources in terms of computing time and hardware requirements to determine a large number of eigenpairs at the lower end of the spectrum. An evaluation and the comparison to the block Lanczos method for a broadband vibro-acoustic analysis of a passenger car body is contained in [7], demonstrating that AMLS enabled a reduction in runtime from several days on a supercomputer to a few hours using an off-the-shelf workstation.

It is important to note that AMLS usually provides approximate solutions which are less accurate than the ones obtained with Krylov type methods. However, in many applications, the underlying algebraic eigenvalue problem is a finite element model of the original continuous problem, and so the level of accuracy required for its numerical solution is no more than what is furnished by the FE model. Numerical examples demonstrate that the approximations to eigenvalues computed with AMLS are often of this limited but sufficient accuracy, whereas the modal errors of eigenvectors are usually still quite large.

A common way to enhance the approximation quality of a group of eigenpairs is subspace iteration [16], [17] which generalizes inverse iteration. However, applying subspace iteration to the linearization (2) of problem (1) directly is very expensive. AMLS offers a permutation of the unknowns such that the stiffness matrix becomes block diagonal with not too large block sizes. But transforming the mass and gyroscopic matrices correspondingly yields very crowded matrices with lots of dense blocks which require a huge amount of storage.

For the generalized eigenvalue problem we proposed in [15] a combination of AMLS with subspace iteration which takes advantage of the block structure of the transformed stiffness matrix, but performs multiplications with the mass matrix in the original ordering of variables. In this paper we transfer this approach to the gyroscopic eigenvalue problem taking advantage of knowledge from the variational characterizations of its spectrum to improve the interesting eigenpairs corresponding to the smallest positive eigenvalues although these are in the interior of the spectrum.

The paper is organized as follows. In Chapter 2 we briefly summarize the AMLS method for linear positive definite eigenvalue problems. Chapter 3 considers AMLS for gyroscopic problems, and Chapter 4 improves the eigenvalue and eigenvector approximations via subspace iteration. Chapter 5 demonstrates the efficiency of this approach for a finite element model of a rotating tire.

## 2. AMLS for linear eigenvalue problems

In this subsection we summarize the AMLS method for computing eigenvalues and corresponding eigenvectors of a

linear eigenvalue problem

$$Kx = \lambda Mx \quad (3)$$

in a frequency range of interest. Usually (3) is a finite element model of some problem, where the stiffness matrix  $K \in \mathbb{R}^{n \times n}$  and the mass matrix  $M \in \mathbb{R}^{n \times n}$  are symmetric and  $M$  is positive definite.

Similarly as in the component mode synthesis (CMS) [18] the structure is partitioned into a small number of substructures based on the sparsity pattern of the system matrices, but more generally than in CMS these substructures in turn are sub-structured on a number of levels yielding a tree topology for the substructures.

AMLS consists of two ingredients. First, the stiffness matrix  $K$  is transformed to block diagonal form, and secondly, the dimension is reduced substantially by modal condensation of the substructures. If  $K_{ss}$  is a sub-matrix of  $K$  corresponding to a particular substructure, then after reordering rows and columns in (3) the pencil obtains the form

$$\left( \begin{bmatrix} K_{ss} & K_{sr} \\ K_{rs} & K_{rr} \end{bmatrix}, \begin{bmatrix} M_{ss} & M_{sr} \\ M_{rs} & M_{rr} \end{bmatrix} \right),$$

and with block Gaussian elimination, i.e. post- and premultiplying this pencil with

$$U_s = \begin{bmatrix} I & -K_{ss}^{-1}K_{sr} \\ 0 & I \end{bmatrix}$$

and  $U_s^T$ , respectively,  $K_{ss}$  is decoupled, and the pencil obtains the following form

$$(U_s^T K U_s, U_s^T M U_s) = \left( \begin{bmatrix} K_{ss} & 0 \\ 0 & \tilde{K}_{rr} \end{bmatrix}, \begin{bmatrix} M_{ss} & \tilde{M}_{sr} \\ \tilde{M}_{sr}^T & \tilde{M}_{rr} \end{bmatrix} \right).$$

Repeating the block elimination for all substructures  $1, \dots, m$  we get

$$\tilde{K} = U^T K U, \quad \tilde{M} = U^T M U \quad \text{with } U = U_1 U_2 \dots U_m$$

where the transformed stiffness matrix  $\tilde{K}$  has block diagonal form. Notice that in an implementation of AMLS the reordering of the matrices is incorporated implicitly.

To reduce the dimension of the eigenproblem we determine for every substructure (after decoupling it from the remaining DoFs as above) all eigenvalues  $\lambda_{sj}$  not exceeding a cut off frequency  $\lambda_{\text{cutoff}}$  and corresponding eigenvectors  $z_{sj}$ ,  $j = 1, \dots, m_s$ . Then with  $Z_s = [z_{s1}, \dots, z_{sm_s}]$  and the global projection matrix  $Z = \text{diag}\{Z_1, \dots, Z_m\}$  we finally get the reduced eigenvalue problem

$$K_c x_c = \lambda M_c x_c \quad (4)$$

where  $K_c = Z^T \tilde{K} Z = Z^T U^T K U Z$  is a diagonal matrix and  $M_c = Z^T \tilde{M} Z = Z^T U^T M U Z$  has generalized block arrowhead form.

It is important to note that in an implementation the block Gaussian eliminations and the condensations are performed in an interleaving way to avoid the storage of large dense

sub-matrices of the transformed mass matrix which would occur in the course of the block elimination: as soon as a sub-matrix pencil  $(\tilde{K}_{ss}, \tilde{M}_{ss})$  has been formed the eigenproblem  $\tilde{K}_{ss}Z_s = \tilde{M}_{ss}Z_s\Lambda_s$  is solved and the corresponding projection is executed. Details of an implementation of AMLS are contained in [12], [15].

### 3. AMLS for gyroscopic eigenvalue problems

The gyroscopic eigenvalue problem (1) is equivalent to its Hermitian linearization

$$\begin{bmatrix} iG & K \\ K & O \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \omega \begin{bmatrix} M & O \\ O & K \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} \quad (5)$$

to which AMLS does not apply directly since the matrix on the left hand side is not positive definite, and for the transformed problem with  $\mu := \omega^{-1}$  AMLS yields approximations to eigenpairs corresponding to the smallest eigenvalues  $\mu_j$ , i.e. to the largest eigenvalues of (1) in modulus.

Since the influence of the gyroscopic matrix  $G$  on the eigenvectors of (1) is usually not very high compared to the mass and stiffness matrices, it is reasonable to neglect the linear term in (1) when defining the substructuring, the transformations of  $K$  to block diagonal form. and the modal reductions corresponding to the substructures. Hence the AMLS reduction is applied to the pencil  $(K, M)$ , and the same congruence transformations and modal reductions are employed for the skew-symmetric matrix  $G$ .

Thus, one obtains a reduced model

$$K_c y + i\omega G_c y - \omega^2 M_c y = 0, \quad (6)$$

where the reduced stiffness matrix  $K_c = Z^T U^T K U Z$  and mass matrix  $M_c$  are the ones from Section 2, and  $G_c = Z^T U^T G U Z$  is the projected gyroscopic matrix which can be evaluated along with  $K_c$  and  $M_c$  within the AMLS process for  $(K, M)$ . Hence, the reduced problem (6) has the same structure as (1), but it is of much smaller dimension and can therefore be solved by the Lanczos method using a linearization like (5) or by the nonlinear Arnoldi method. Notice, that all transformations in AMLS are real, and therefore the reduction can be performed in real arithmetic.

The paper [4] contains an example demonstrating the efficiency of the approach. A FE model of a deformable wheel rolling on a rigid plane surface of dimension approximately 125000 was reduced by AMLS to a gyroscopic problem of dimension 2635 on a personal computer (namely a Pentium 4 processor with 3.0 GHz and 1 GB storage) requiring a CPU time of 976 seconds. Solving its linearization (5) with the matlab function `eigs` (i.e. by ARPACK) needed another 124 seconds. Thus approximate eigenvalues for the smallest 180 eigenvalues (up to 2000 Hz) were obtained the relative errors of which were all less than 0.65%. We will come back to this example in Section 5.

### 4. AMLS with subspace iteration

AMLS is a one shot projection method, i.e. after having chosen a cut-off frequency the method produces a fixed subspace  $\mathcal{V} := \text{span}\{UZ\}$  and the corresponding projected eigenproblem. Differently from Krylov subspace methods there is no way to expand the subspace  $\mathcal{V}$  further reusing the projected problem if the computed approximate eigenpairs turn out to be not accurate enough. One has to repeat the reduction with a higher cut-off frequency, or one can improve the subspace  $\mathcal{V}$  using subspace iteration.

In [15] we discussed a method how to employ the transformed block diagonal stiffness matrix  $\tilde{K}$  in subspace iteration for (3) efficiently. It is important to note that we do not use the transformed mass matrix  $\tilde{M} = U^T M U$  which usually contains many dense sub-matrices requiring a huge amount of storage. In the following we modify this approach for improving the AMLS approximation of a gyroscopic problem (1).

The gyroscopic eigenvalue problem (1) where  $K \in \mathbb{C}^{n \times n}$  and  $M \in \mathbb{C}^{n \times n}$  are Hermitian and positive definite and  $G \in \mathbb{C}^{n \times n}$  is skew-Hermitian has  $2n$  real eigenvalues,  $n$  of which are negative and  $n$  are positive. For real matrices we even have that  $-\omega_j$  are the negative eigenvalues of (1) if  $\omega_j, j = 1, \dots, n$  denote its positive eigenvalues. For the general complex case positive and negative eigenvalues are of the same magnitude in modulus if  $G$  is small compared to  $K$  and  $M$ . The eigenvectors corresponding to the positive eigenvalues form a basis of  $\mathbb{C}^n$ , and the same holds true for the eigenvectors corresponding to the negative eigenvalues (cf. [19]).

If the subspace iteration is applied to the linearized eigenvalue problem (5) with shift  $\theta = 0$  (this is the only way to take advantage of the transformed block diagonal matrix  $\tilde{K}$ ) with an initial basis  $X \in \mathbb{R}^{n \times m}$ , then one obtains convergence to eigenvalues,  $m/2$  of which are negative and  $m/2$  are positive. This suggests to apply the subspace iteration in the following way:

Let  $V \in \mathbb{R}^{n \times m}$  be the matrix of eigenvector approximations obtained from AMLS and  $\Lambda \in \mathbb{R}^{m \times m}$  be the diagonal matrix containing the approximations of the  $m$  smallest positive eigenvalues then we apply the subspace iteration to (5) with the initial basis

$$\begin{bmatrix} V\Lambda^{1/2} & -V\Lambda^{1/2} \\ V & V \end{bmatrix}.$$

To take advantage of the particular form of (5) we set

$$\begin{bmatrix} P \\ Q \end{bmatrix} = \begin{bmatrix} V\Lambda^{1/2} & -V\Lambda^{1/2} \\ V & V \end{bmatrix}.$$

Then one step of subspace iteration

$$\begin{bmatrix} iG & K \\ K & O \end{bmatrix} \begin{bmatrix} P^{(k)} \\ Q^{(k)} \end{bmatrix} = \begin{bmatrix} M & O \\ O & K \end{bmatrix} \begin{bmatrix} P^{(k-1)} \\ Q^{(k-1)} \end{bmatrix}$$

reads as follows:

$$P^{(k)} = Q^{(k-1)}, \quad KQ^{(k)} = MP^{(k-1)} - iGQ^{(k-1)}. \quad (7)$$

After  $n_k$  steps of subspace iteration one gets  $\hat{P} := P^{(n_k)}$  and  $\hat{Q} := Q^{(n_k)}$  and the projected eigenvalue problem has the form

$$\begin{aligned} & [\hat{P}^H \quad \hat{Q}^H] \begin{bmatrix} iG & K \\ K & O \end{bmatrix} \begin{bmatrix} \hat{P} \\ \hat{Q} \end{bmatrix} z \\ &= \omega [\hat{P}^H \quad \hat{Q}^H] \begin{bmatrix} M & O \\ O & K \end{bmatrix} \begin{bmatrix} \hat{P} \\ \hat{Q} \end{bmatrix} z \end{aligned}$$

which is equivalent to

$$\begin{aligned} \hat{K}z &:= (i\hat{P}^H G \hat{P} + \hat{P}^H K \hat{Q} + \hat{Q}^H K \hat{P})z \\ &= \omega(\hat{P}^H M \hat{P} + \hat{Q}^H K \hat{Q})z =: \hat{M}z. \end{aligned}$$

To avoid the use of the transformed matrix  $\tilde{M} = U^T M U$  which is not computed in the AMLS method and which is much too memory-consuming we determine the right hand side of the linear system in (7) in original variables as  $R = MP^{(k-1)} - iGQ^{(k-1)}$ . But to take advantage of the block structure of the transformed stiffness matrix  $\tilde{K} = U^T K U$  we solve the transformed system  $\tilde{K}\tilde{Q}^{(k)} = \tilde{R}$ . Hence, in every iteration step two forward transformations  $\tilde{Q}^{(k)} := U^T Q^{(k)}$  and  $\tilde{R} = U^T R$  and one backward transformation  $Q^{(k)} = U\tilde{Q}^{(k)}$  are required. Details on the implementation of the forward and backward transformations are contained in [15].

Algorithm 1 contains a pseudocode of the resulting method. It is interesting to note that usually a small number of subspace iteration steps improves the approximations sufficiently, which also guarantees the stability of iterations (i.e. no orthogonalization between the individual iterations is necessary) with only one Rayleigh-Ritz analysis of  $\hat{K}z = \omega\hat{M}z$ .

## 5. Numerical results

We consider a FE model of a deformable wheel rolling on a rigid plane surface which is obtained by an Arbitrary Lagrangian Eulerian (ALE) formulation according to the derivation and presentation in [1]. Our model of a rotating tire consists of 39204 brick elements with 124992 degrees of freedom and accounts for 20 different material groups. The speed is assumed to be 60 km/h. Our aim is to determine approximations to the smallest 200 eigenvalues and corresponding eigenvectors.

The numerical tests were performed on a 64-bit HP workstation with an Intel Xeon CPU (3.20 GHz, 2 cores) and 24GB memory. AMLS and the two subspace iteration algorithms (AMLS-SIM which combines the transformation of  $K$  to block diagonal form in AMLS with the subspace iteration, and NormalSIM which applies subspace iteration directly to problem (5)) were implemented with Matlab R2009a.

---

### Algorithm 1 Subspace iteration cooperating with AMLS.

---

**Require:** Diagonal matrix  $\tilde{\Lambda}$  containing eigenvalue approximations from AMLS, transformed eigenvectors  $\tilde{V}$ , the transformed stiffness matrix  $\tilde{K}$  and the transformation matrix  $U$  from AMLS, and the maximum iteration number  $n_k$

- 1: initialize the iteration matrices  $\tilde{Q}^{(0)} = [\tilde{V}, \tilde{V}]$  and  $\tilde{P} = [\tilde{V}\tilde{\Lambda}^{1/2}, -\tilde{V}\tilde{\Lambda}^{1/2}]$
  - 2: transform backward  $P^{(0)} = U\tilde{P}^{(0)}$
  - 3: **for**  $k = 1, 2, \dots, n_k$  **do**
  - 4:   transform backward  $Q^{(k-1)} = U\tilde{Q}^{(k-1)}$
  - 5:   compute  $R = MP^{(k-1)} - iGQ^{(k-1)}$
  - 6:   transform forward  $\tilde{R} = U^T R$
  - 7:    $P^{(k)} = Q^{(k-1)}$
  - 8:   solve for  $\tilde{Q}^{(k)}$ :  $\tilde{K}\tilde{Q}^{(k)} = \tilde{R}$
  - 9: **end for**
  - 10:  $T = \tilde{R}^H \tilde{Q}^{(n_k)}$
  - 11: projected mass matrix  $\hat{M} = (P^{(n_k)})^H M P^{(n_k)} + T$
  - 12: reload  $R$
  - 13:  $S = R^H P^{(n_k)}$
  - 14: projected stiffness matrix  $\hat{K} = i(P^{(n_k)})^H G P^{(n_k)} + S + S^H$ .
  - 15: solve projected problem  $\hat{K}Z = \hat{M}Z\hat{\Lambda}$
  - 16: sort out positive eigenvalue  $\hat{\Lambda}_+$  and corresponding eigenvectors  $Z_+$
  - 17: compute improved eigenvectors  $V^{(n_k)} = \hat{Q}Z_+$ .
- 

Table 1: Computation time of NormalSIM and AMLS-SIM with 208 iteration vectors

Computational Steps	NormalSIM(s)	AMLS-SIM(s)
Compute initial eigenvectors from AMLS	9.2	2.4
1 Iteration	2401.0	148.9
2 Iterations	Not computed	304.9
3 Iterations	Not computed	462.1
4 Iterations	Not computed	621.8
Compute $\hat{K}$ and $\hat{M}$ if $n_k = 1$	10.0	9.7
Compute $\hat{K}$ and $\hat{M}$ if $n_k > 1$	Not computed	19.6
Solve $\hat{K}Z = \hat{M}V\hat{\Lambda}$ by eig	2.6	2.6
Compute final eigenvectors	0.9	25.6

The AMLS method addressing the linear eigenvalue problem  $Kx = \lambda Mx$  costs 881.2 seconds for the AMLS projection and 270.4 seconds for solving the projected linear eigenvalue problem of dimension 2263 by eig. Following Bathe's recommendation to use  $\min\{2p, p+8\}$  initial vectors in subspace iteration if  $p$  eigenpairs are wanted we initialized the subspace iteration methods with the lowest 208 positive eigenvalues and corresponding eigenvectors. The computation times are listed in Table 1. We can see that NormalSIM needs 2401.0 seconds to finish the first iteration step excluding computation of  $\hat{K}$  and  $\hat{M}$ . But AMLS-SIM only costs 148.9 seconds, much less than NormalSIM.

The relative errors of eigenvalues computed by AMLS-

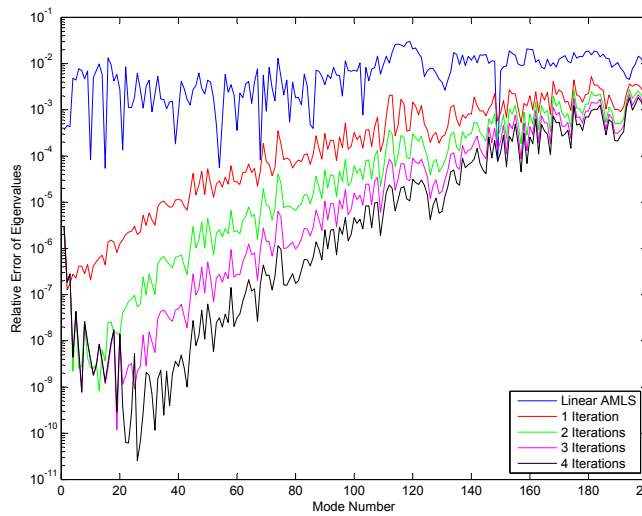


Fig. 1: Relative errors of eigenvalues computed with subspace iteration with AMLS utilizing 208 iteration vectors

SIM with four iteration steps are given in Fig.1. To evaluate the accuracy of eigenpairs we use modal errors

$$\epsilon_g = \frac{\|Kx + i\omega Gx - \omega^2 Mx\|}{\|\omega^2 Mx\|}. \quad (8)$$

Fig.2 shows the reduction of modal errors for four iterations with AMLS-SIM. It is interesting to note that essential improvements are obtained only every other iteration step.

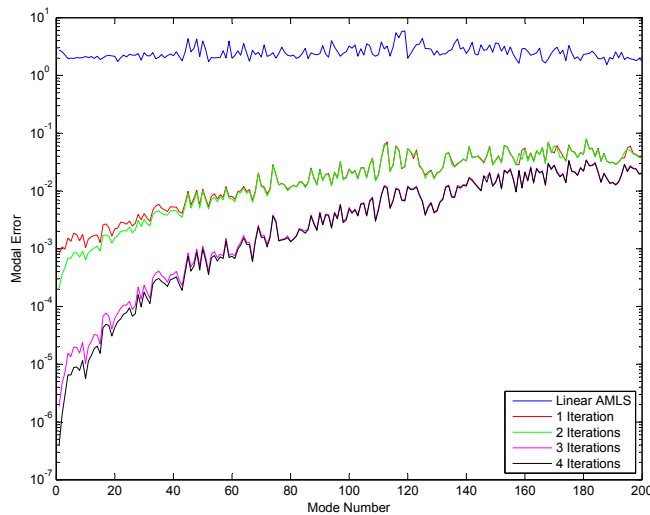


Fig. 2: Modal errors computed with subspace iteration with AMLS utilizing 208 iteration vectors

## 6. Conclusions

AMLS is an efficient condensation method for computing a huge number of eigenmodes and frequency responses for large complex structures. It usually provides approximate solutions which are less accurate than the ones obtained with standard Krylov type methods. However, in many applications the underlying algebraic eigenproblem is a FE model of a continuous structure, and so the required level of accuracy is no more than what is furnished by the FE model. Numerical examples demonstrate that the approximations to eigenvalues computed with AMLS are often of this limited but sufficient accuracy, whereas the modal errors of eigenvectors are usually still quite large. In a recent paper we proposed a combination of AMLS with subspace iteration taking advantage of the block structure of the transformed stiffness matrix, but avoiding the use of the highly populated transformed mass matrix. In this paper we generalized this approach to gyroscopic eigenvalue problems taking advantage of knowledge from variational characterizations of the spectrum to improve the interesting eigenpairs corresponding to the smallest positive eigenvalues although these are in the interior of the spectrum.

## Acknowledgement

This work was supported by the National Natural Science Foundation of China under the grant No.10972005 and the China Scholarship Council with the contract No.2010601199.

## References

- [1] U. Nackenhorst, "The ALE-formulation of bodies in rolling contact. Theoretical foundations and finite element approach," *Comput. Meth. Appl. Mech. Engrg.*, vol. 193, pp. 4299 – 4322, 2004.
- [2] M. Brinkmeier and U. Nackenhorst, "An approach for large-scale gyroscopic eigenvalue problems with application to high-frequency response of rolling tires," *Comput. Mech.*, vol. 41, pp. 503–515, 2008.
- [3] R. Lehoucq, D. Sorensen, and C. Yang, *ARPACK Users' Guide. Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Philadelphia: SIAM, 1998.
- [4] K. Elssel and H. Voss, "Reducing huge gyroscopic eigenproblem by Automated Multi-Level Substructuring," *Arch. Appl. Mech.*, vol. 76, pp. 171 – 179, 2006.
- [5] J. Bennighof and C. Kim, "An adaptive multi-level substructuring method for efficient modeling of complex structures," in *Proceedings of the AIAA 33rd SDM Conference*, Dallas, Texas, 1992, pp. 1631 – 1639.
- [6] J. Bennighof and R. Lehoucq, "An automated multilevel substructuring method for the eigenspace computation in linear elastodynamics," *SIAM J. Sci. Comput.*, vol. 25, pp. 2084 – 2106, 2004.
- [7] A. Kropp and D. Heiserer, "Efficient broadband vibro-acoustic analysis of passenger car bodies using an FE-based component mode synthesis approach," *J. Comput. Acoustics*, vol. 11, pp. 139 – 157, 2003.
- [8] D. N. Le, "Extending deterministic vibration analysis of ships into the medium frequency range," Ph.D. dissertation, Institute of Numerical Simulation, Hamburg University of Technology, 2009.
- [9] W. Rachowicz and A. Zdunek, "Automated multi-level substructuring (amls) for electromagnetics," *Comput. Meth. Appl. Mech. Engrg.*, vol. 198, pp. 1224 – 1234, 2009.

- [10] C. Yang, W. Gao, Z. Bai, X. Li, L. Lee, P. Husbands, and E. Ng, "An algebraic sub-structuring method for large-scale eigenvalue calculations," *SIAM J. Sci. Comput.*, vol. 27, pp. 873 – 892, 2005.
- [11] M. Stammberger and H. Voss, "Automated multi-level sub-structuring for fluid-solid interaction problems," *Numer. Lin. Alg. Appl.*, vol. 18, pp. 411 – 427, 2011.
- [12] K. Elssel, "Automated multilevel substructuring for nonlinear eigenvalue problems," Ph.D. dissertation, Institute of Numerical Simulation, Hamburg University of Technology, 2006.
- [13] W. Gao, X. Li, C. Yang, and Z. Bai, "An implementation and evaluation of the AMLS method for sparse eigenvalue problems," *ACM Trans. Math. Softw.*, vol. 34, 2008.
- [14] M. Kaplan, "Implementation of automated multilevel substructuring for frequency response analysis of structures," Ph.D. dissertation, Dept. of Aerospace Engineering & Engineering Mechanics, University of Texas at Austin, 2001.
- [15] J. Yin, H. Voss, and P. Chen, "Improving eigenpairs of automated multilevel substructuring with subspace iteration," Institute of Numerical Simulation, Hamburg University of Technology, Tech. Rep. 158, 2011, submitted to *Computers & Structures*.
- [16] K. Bathe and E. Wilson, "Large eigenvalue problems in dynamic analysis," *J. Engrg. Div.*, vol. 98, pp. 1471 – 1485, 1972.
- [17] K. Bathe, *Finite Element Procedures*. New Jersey: Prentice-Hall, 1996.
- [18] R. Craig Jr. and M. Bampton, "Coupling of substructures for dynamic analysis," *AIAA J.*, vol. 6, pp. 1313–1319, 1968.
- [19] R. Duffin, "The Rayleigh–Ritz method for dissipative and gyroscopic systems," *Quart. Appl. Math.*, vol. 18, pp. 215 – 221, 1960.

# Incomplete LU Preconditioning and Error Compensation Strategies for Sparse Matrices

Eun-Joo Lee

Department of Computer Science, East Stroudsburg University of Pennsylvania,  
327 Science and Technology Center, East Stroudsburg, PA 18301-2999, USA

**Abstract**—Several preconditioning enhancement strategies for improving inaccurate preconditioners produced by the incomplete LU factorizations of sparse matrices are presented. The strategies employ the elements that are dropped during the incomplete LU factorization and utilize them in different ways by separate algorithms. The first strategy (error compensation) applies the dropped elements to the lower and upper parts of the LU factorization to compute a new error compensated LU factorization. Another strategy (inner-outer iteration), which is a variant of the incomplete LU factorization, embeds the dropped elements in its iteration process. Experimental results show that the presented enhancement strategies improve the accuracy of the incomplete LU factorization when the initial factorizations found to be inaccurate. Furthermore, the convergence cost of the preconditioned Krylov subspace methods is reduced on solving the original sparse matrices with the proposed strategies.

**Keywords:** ILU factorization, preconditioning, error compensation, inner-outer iteration, sparse matrices

## 1. Introduction

Preconditioned Krylov subspace methods are generally considered as a class of the most promising techniques [1], [2], [3], [8] for solving very large sparse linear systems of the form

$$Ax = b, \quad (1)$$

where  $A$  is a sparse matrix of order  $n$ . Although the most popular preconditioners nowadays are constructed from the incomplete lower-upper (ILU) factorizations of the coefficient matrix,  $A$ , in one way or another, a common problem with such ILU preconditioners still exists in that their accuracy may be insufficient to yield an adequate rate of convergence. This problem is usually caused by the difficulty in determining even a small number of parameters or threshold values for the ILU factorizations for some particular matrices. This still remains as a tradeoff problem between the factorization accuracy and the costs of the computation and memory associated with the factorization.

In order to clarify the low accuracy problem, we first account for the aspects of advantages and disadvantages of ILU(0) and ILUT. ILU(0) (incomplete LU factorization with

zero fill-in) preconditioning is rather easy to implement and computationally inexpensive than other ILU preconditioners, but may require many iterations to converge due to its coarse approximation to the original matrix  $A$  [7], [8]. ILUT incomplete factorization is considered as one of the alternative ways of ILU(0). ILUT approach relies on numerical values for dropping elements, and it has a limit on the number of allowed fill-in elements. It develops incomplete factorization preconditioners with improved accuracy if the dropping tolerance and the fill-in parameters are chosen properly. On the other hand, ILUT with large dropping tolerance and small fill-in does not constitute a reliable approach, and generally gives low accuracy ILU factorizations [7], [8].

In response to these problems associated with the ILU factorizations, we consider the fact that the size of the error matrix  $E$  directly affects the convergence rate of the preconditioned iterative methods [5]. Furthermore, the quality of a preconditioning step is directly related to the size of both  $(LU)^{-1}$  and  $E$ , that is, a high quality preconditioner must have an error matrix that is small in size [9]. In the form of an incomplete LU factorization by incorporating the error or residual matrix,  $E$ , we have

$$A = LU + E, \quad (2)$$

where the  $L$  and  $U$  are the lower and upper parts of  $A$ , respectively. In accordance with Saad [8], the elements of the error matrix  $E$  are exactly those that are dropped during the ILU factorization, whereas in the standard ILU factorizations, such elements are just discarded. The new preconditioning accuracy enhancement strategies that we propose in this paper are inspired by understanding these facts and exploiting the error values dropped in the factorization to improve the accuracy of the incomplete LU factorizations. This then leads us to the idea of the error compensation strategy that utilizes the error matrix before the start of the preconditioned iteration process. After acquiring the ILU factorization with a preconditioner, we add the error matrix  $E$  to  $L$  and  $U$  to obtain an error compensated factorization  $\tilde{L}\tilde{U}$ . The accuracy of using  $\tilde{L}\tilde{U}$  is increased significantly in some cases, compared with that of using  $LU$  directly, in the sense that  $\|\tilde{E}(= A - \tilde{L}\tilde{U})\| < \|E(= A - LU)\|$ .

Another preconditioning accuracy enhancement strategy uses the error matrix during the preconditioned iteration process. It can be considered as a variant of the LU solver with

an error embedded LU approach. This results in an inner-outer iteration process. Numerical results of the inner-outer iteration strategies are provided to show that this algorithm requires only fewer iterations to converge. Note that in order to collect the elements during the ILU factorization process, additional computation cost and some memory spaces are required to keep these elements. This, however, can be negligible as low cost storages become available.

This paper is organized as follows: In Section 2, we present preconditioning enhancement strategies arising from ILU preconditioning, and their analysis. The experimental results are in Section 3, and the conclusion and future work are discussed in Section 4.

## 2. Preconditioning Accuracy Enhancement Strategies

ILU factorizations are made incomplete by dropping carefully chosen nonzero elements to make the factorizations more economical to store, compute, and solve with it. Each nonzero element that is dropped contributes to the “error” in the factorization [3]. In this section, we assume that the ILU factorizations in question are stable, and propose an error compensation algorithm and an inner-outer iteration algorithm to improve the accuracy of the (inaccurate) incomplete LU factorizations. ILU factorizations with stability problems can be better handled by using reordering strategies [6].

### 2.1 Matrix Error Compensation Strategies

Let  $L$  and  $U$  be the triangular factors of the input matrix  $A$  from an ILU factorization. Then  $A$  can be written as in Equation (2), in which  $E$  is the error matrix that represents the dropped elements during the incomplete LU factorization process [8]. Since the size of the  $E$  matrix directly affects the convergence rate of the preconditioned iterative methods, and the quality of a preconditioning step is directly related to the size of both  $(LU)^{-1}$  and  $E$  [5], [9], we propose an algorithm to improve the preconditioning accuracy by reducing the size of the error matrix  $E$  and  $(LU)^{-1}E$ . By switching the matrix  $A$  and  $LU$ , Equation (2) can be written as:

$$E = A - LU. \quad (3)$$

Then the error matrix  $E$  can be separated into two parts, corresponding to nonzero structures of  $L$  and  $U$ , in the following way:

$$E = E_l + E_u,$$

where  $E_l$  and  $E_u$  are the lower and upper triangular parts of  $E$ . In order to acquire error compensated  $\tilde{L}$  and  $\tilde{U}$ , the separated error matrices,  $E_l$  and  $E_u$ , are now combined with the original lower part ( $L$ ) and upper part ( $U$ ) of the preconditioned matrix:

$$\tilde{L} = L + E_l, \quad \tilde{U} = U + E_u.$$

These new lower part  $\tilde{L}$  and upper part  $\tilde{U}$  are then used as the preconditioner instead of the original  $L$  and  $U$ . In other words,  $\tilde{L}$  and  $\tilde{U}$  are used in the preconditioning steps, i.e., we solve  $(\tilde{L}\tilde{U})e = r$  instead of  $(LU)e = r$  at each preconditioning step. Here  $e$  and  $r$  are the error and residual vectors of the current approximate solution. In the experimental results presented in Section 3, we will see that replacing LU solver with the new  $\tilde{L}\tilde{U}$  results in a more accurate preconditioner. In many cases, the size of the error matrix with the new  $\tilde{L}\tilde{U}$  is smaller than that with the original  $LU$ .

In order to provide a better picture of the algorithm, an ILU factorization in which the error compensated  $L$  and  $U$  improves the accuracy of the original factorization is given in the following: The ILU(0) factors  $L$  and  $U$  of a given matrix

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix}$$

are

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & 0 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.5 \end{pmatrix},$$

from which we can compute the error matrix  $E = A - LU$  as

$$E = A - LU = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -0.5 \\ 0 & -0.5 & 0 \end{pmatrix}.$$

The error matrix  $E$  is then split into two parts,  $E_l$  and  $E_u$ . We add  $E_l$  and  $E_u$  to the original incomplete lower and upper parts of the  $A$ . This results in a new incomplete LU factors

$$\tilde{L} = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & -0.5 & 1 \end{pmatrix}, \quad \tilde{U} = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1.5 & -0.5 \\ 0 & 0 & 1.5 \end{pmatrix}.$$

From the above  $A$ ,  $\tilde{L}$ , and  $\tilde{U}$ , we can obtain a new error matrix  $\tilde{E}$  as:

$$\tilde{E} = A - \tilde{L}\tilde{U} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.25 & -0.25 \end{pmatrix}.$$

Table 1 presents the comparisons of the sizes of the error matrices in the example given above. The sizes of the preconditioned error matrix for the error compensated preconditioner and the original preconditioned matrix are measured in the 2-norm and the Frobenius norm. The case using error compensated preconditioned matrix reduces the error norm by 36% on average. The error compensation algorithm can have four different implementations. The above description assumes that both the  $L$  and  $U$  factors are compensated. If none of them is compensated, we have the standard ILU preconditioner. If either one of them, but not both factors, is compensated, we have two forms of partial error compensations.



Table 1: Comparison of the size of preconditioned error matrices  $E$  and  $\tilde{E}$ .

Norm	$E$	$\tilde{E}$	$(LU)^{-1}E$	$(\tilde{L}\tilde{U})^{-1}\tilde{E}$
2-norm	0.5	0.3536	0.4082	0.2940
Frobenius norm	0.7071	0.3536	0.5270	0.2940

## 2.2 An Inner-Outer Iteration Process

In this subsection, we describe an error embedded variant of ILU preconditioning process to compensate dropped elements during the ILU factorization. In solving Equation (1) with a standard preconditioned iterative solver, with an ILU factorization type preconditioner, the preconditioning step is an LU solution process, i.e., to solve  $(LU)e = r$  for the residual vector  $r$  of the current approximate solution and to get an approximate correction vector  $e$ . For notational convenience, we denote the current approximate solution as  $\tilde{x}$ . The current residual vector is  $\tilde{r} = b - A\tilde{x}$ . The ideal preconditioning step is to solve  $A\tilde{e} = \tilde{r}$  and to correct the approximate solution  $\tilde{x}$  to get the exact solution as  $(\tilde{x} + \tilde{e})$ . Since solving  $A\tilde{e} = \tilde{r}$  is as expensive as solving the original system  $Ax = b$  and is thus impractical in a preconditioning step, we can again use an iteration procedure to approximately solve  $A\tilde{e} = \tilde{r}$  to achieve a good preconditioning effect. If the error matrix is available, we can split the matrix  $A$  as in Equation (2), so, for an idealized preconditioning step, we have

$$(LU + E)e = r, \quad (4)$$

or

$$LUe + Ee = r. \quad (5)$$

As a stationary iteration process, Equations (4) and (5) can be rewritten by:

$$LUe^{(k+1)} = r - Ee^{(k)},$$

and

$$e^{(k+1)} = (LU)^{-1}(r - Ee^{(k)}) \quad (6)$$

respectively. This iteration process starts with  $e^{(0)} = 0$ , and  $LUe^{(1)} = r$  is just the standard one step ILU preconditioning procedure. Better preconditioning effect can be achieved by performing a few iterations at each preconditioning step. This will constitute an inner-outer iteration scheme and is well-known for the preconditioned Krylov method, as each outer Krylov iteration step is preconditioned by a series of inner stationary iteration steps.

The necessary and sufficient condition to converge for the inner iteration process, given by Equation (6), is that the spectral radius of the iteration matrix  $(LU)^{-1}E$  is smaller than 1. This is, of course, not automatically guaranteed. However, we found that the inner iteration process usually converges, when the ILU factorization of the original matrix  $A$  is stable and the LU factor is not an extremely poor approximation to the original matrix  $A$ . We point out that

the inner-outer iteration strategy is not a new one, but here we study it in a systematic way.

## 2.3 Analysis

A five-point matrix (of a finite difference discretization of a Poisson equation),  $A$ , of size  $n = 400$  corresponding to an  $n_x \times n_y = 20 \times 20$  mesh is used for analysis to validate the presented algorithms. The sizes of the error matrix,  $E$ , from the original preconditioner and the new error matrix,  $\tilde{E}$ , from the error compensated preconditioner are measured in the 2-norm and the Frobenius norm. For the case using the error compensated algorithm, the norms (2-norm and Frobenius norm) in columns 2 and 3 in Table 2 show the fact that the norms of the new error matrix are smaller than the norms of the original error matrix, and also the norms (2-norm and Frobenius) in columns from 4 to 7 show that all the norms (2-norm and Frobenius) of the corresponding matrices in columns from 5 to 7 are smaller than the norms in column 4. In addition to that, from the values in the last row of Table 2, we can see that the inner iteration process converges, since the spectral radii of the  $(LU)^{-1}E$  and  $(\tilde{L}\tilde{U})^{-1}\tilde{E}$  satisfy the necessary and sufficient condition for convergence.

## 3. Experimental Results

We present numerical results from the experiments by solving sparse matrices with our algorithms and some preconditioners. The preconditioned iterative solver we employed was GMRES(20), with ILU type preconditioners, ILU(0) and ILUT [7]. For all linear systems, the right-hand side was generated by assuming that the solution is a vector of all ones. The initial guess was a zero vector. We set the iteration to be terminated when the  $l_2$ -norm of the initial residual is reduced by at least seven orders of magnitude, or when the number of iterations reaches 200. The computations were carried out in double precision arithmetics on a 64-bit Sun-Blade-100 workstation with a 500 MHz UltraSPARC III CPU and 1 GB of RAM. The set of sparse matrices that we used for our experiments are from the Harwell-Boeing Sparse Matrix Test Collection [4]. A brief description of each matrix of the experiment set is given in Table 3. In the table, the order, the number of nonzero entries, and the condition number of the matrix are denoted by  $n$ ,  $nnz$ , and  $cond$  respectively. All tested matrices do not have zero diagonals and the ILU factorizations are stable.

Table 2: Comparison of the size of preconditioned error matrices  $E$  and  $\tilde{E}$ .

Norm	$E$	$\tilde{E}$	$(LU)^{-1}E$	$(\tilde{L}\tilde{U})^{-1}\tilde{E}$	$(\tilde{L}\tilde{U})^{-1}\tilde{E}$	$(\tilde{L}\tilde{U})^{-1}\tilde{E}$
2-norm	0.5788	0.3582	0.9276	0.8889	0.9109	0.9106
Frobenius norm	7.7958	3.2058	3.6129	2.4812	3.1488	3.1447
Spectral radius			0.9276	0.8885	0.9098	0.9098

Table 3: Description of the test matrices.

Matrix	Description	$n$	$nnz$	$cond$
add32	Computer component design 32-bit adder	4960	23884	2.1E+02
bfw398a	Bounded Finline Dielectric Waveguide	398	3678	7.6E+03
cdde1	Model 2-D Convection Diffusion Operator p1=1, p2=2, p3=30	961	4681	4.1E+03
cdde3	Model 2-D Convection Diffusion Operator p1=1, p2=2, p3=80	961	4681	1.0E+04
hor131	Flow network problem	434	4710	1.3E+05
jpwh991	Circuit physics modeling	991	6027	7.3E+02
lop163	LOPSI Stochastic Test Matrix	163	935	3.4E+07
mhd3200b	Alfven Spectra in Magneto hydrodynamics	3200	18316	2.0E+13
mhd416b	Alfven Spectra in Magneto hydrodynamics	416	2312	5.1E+09
mhd4800b	Alfven Spectra in Magneto hydrodynamics	4800	27520	1.0E+14
orsirr1	Oil reservoir simulation - generated problems	1030	6858	1.0E+02
orsirr2	Oil reservoir simulation - generated problems	886	5970	1.7E+05
orsreg1	Oil reservoir simulation - generated problems	2205	14133	1.0E+02
pde225	Partial Differential Equation	225	1065	1.0E+02
pde2961	Partial Differential Equation	2961	14585	9.5E+02
pde900	Partial Differential Equation	900	4380	2.9E+02
pores2	Reservoir modeling Reservoir simulation	1224	9613	3.3E+08
pores3	Reservoir modeling Reservoir simulation	532	3474	6.6E+05
saylr1	Saylor's petroleum engineering/reservoir simulation matrices	238	1128	1.6E+09
saylr4	Saylor's petroleum engineering/reservoir simulation matrices	3564	22316	1.0E+02
sherman1	Oil reservoir simulation challenge matrices	1000	3750	2.3E+04
sherman4	Oil reservoir simulation challenge matrices	1104	3786	7.2E+03
sherman5	Oil reservoir simulation challenge matrices	3312	20793	3.9E+05
watt1	Petroleum engineering	1856	11360	5.4E+09

### 3.1 Preconditioning with Error Compensation Strategies

Numerical experiments of testing the error compensation algorithm with different implementations are presented. ILU(0) and ILUT preconditioners are used. In ILUT, we chose the dropping tolerance to be 0.1 and the fill-in parameter to be 5 for all test problems.

Table 4 reports the number of preconditioned GMRES (PGMRES) iterations with and without different error compensation strategies. The data in the columns marked "No" are those of the standard preconditioning strategy, i.e., no error compensation strategy was used. The columns marked "Full" indicate that both the L and the U parts were compensated. Similarly, the columns marked "Upart" means that only the U part was compensated, and the columns marked "Lpart" means that only the L part was compensated. The value "-1" in the table indicates the failure of convergence within the maximum number (200) of allowed iterations.

Each column of Table 4 shows the number of PGMRES iterations. The number of (preconditioned) iterations, especially for solving the matrices add32, bfw398a, hor131, lop163, pde2961, sherman1, sherman5, and watt1, are greatly reduced with the error compensation strategies in

the ILU(0) preconditioner, compared with the standard preconditioner ILU(0). In addition, the number of iterations for solving the matrices mhd and sherman5 are reduced with the error compensation strategies in ILUT, compared with the standard preconditioner ILUT. We can see that for most of the test matrices, the use of error compensation strategies reduces the number of PGMRES iterations. Furthermore, the full error compensation strategy is shown to be more robust than both the L part and the U part partial error compensation strategies. For the two partial error compensation strategies, their performance seems to be comparable.

Table 5 presents the total CPU processing time which includes the preconditioner construction time with different error compensation strategies. The listed results in Table 5 are concerned with the results listed in Table 4. "N/A" indicates the failure of computation since the GMRES iteration number is reached the maximum number(200). The total CPU time for solving the matrices (add32, bfw398a, hor131, lop163, mhd3200b, pde2961, sherman1, and watt1) is reduced by applying using the error compensation strategies with ILU(0)/ILUT. This implies that as the number of overall iterations are greatly decreased, the CPU time is also reduced even though we need extra processing time for implementing the error compensation strategies.

Table 4: Comparison of the number of PGMRES iterations with and without different error compensation strategies.

Matrix	ILU(0)				ILUT			
	No	Full	Upart	Lpart	No	Full	Upart	Lpart
add32	81	39	40	40	16	15	-1	15
bfw398a	-1	150	-1	196	-1	-1	-1	-1
hor131	-1	183	-1	-1	-1	-1	-1	-1
jpwh991	29	20	24	24	32	21	25	24
lop163	-1	88	98	176	18	19	18	19
mhd3200b	5	4	4	4	164	33	76	101
mhd416b	4	4	4	4	137	32	63	77
mhd4800b	4	3	4	3	148	33	76	107
pde225	30	16	21	22	16	11	15	13
pde2961	-1	91	103	126	85	66	74	77
pde900	61	46	53	55	33	22	30	27
sherman1	116	62	76	75	107	87	-1	94
sherman4	143	107	124	128	136	101	117	116
sherman5	101	73	77	98	-1	145	-1	169
watt1	178	114	115	95	108	85	94	121

Table 5: Comparison of CPU time in seconds with the full error compensation strategies.

Matrix	ILU(0)				ILUT			
	No	Full	Upart	Lpart	No	Full	Upart	Lpart
add32	1.1E+00	1.0E+00	9.9E-01	9.8E-01	1.2E+00	1.2E+00	N/A	1.1E+00
bfw398a	N/A	1.9E-01	N/A	1.9E-01	N/A	N/A	N/A	N/A
hor131	N/A	2.1E-01	N/A	N/A	N/A	N/A	N/A	N/A
jpwh991	7.0E-02	8.9E-02	7.9E-02	7.9E-02	1.1E-01	1.5E-01	1.3E-01	1.3E-01
lop163	N/A	2.9E-02	3.9E-02	5.0E-02	6.0E-02	2.9E-02	4.9E-02	5.0E-02
mhd3200b	1.8E-01	2.0E-01	2.2E-01	2.1E-01	8.8E-01	4.0E-01	5.9E-01	6.9E-01
pde225	1.9E-02	2.9E-02	1.9E-02	3.9E-02	1.9E-02	2.9E-02	1.9E-02	4.9E-02
pde2961	N/A	5.6E-01	5.8E-01	7.1E-01	1.2E+00	1.4E+00	1.3E+00	1.4E+00
sherman1	1.5E-01	1.1E-01	1.1E-01	1.2E-01	2.4E-01	1.9E-01	N/A	2.1E-01
watt1	5.1E-01	4.9E-01	4.3E-01	3.9E-01	7.7E-01	7.7E-01	6.9E-01	7.4E-01

### 3.2 Preconditioning with Inner-Outer Iteration Strategies

In this subsection, we present numerical experiments of the inner-outer iteration strategy with the different settings of the number of inner iterations. Table 6 reports the number of preconditioned GMRES (PGMRES) iterations with respect to each setting. The data in the columns marked “1-Its” are those of the standard preconditioning strategy, i.e., no additional iteration beyond the standard preconditioning procedure was used, whereas each column marked with from “2-Its” to “4-Its” indicates inner iteration steps of 2 to 4, respectively that are performed at each preconditioning step. Also, the value “-1” in the table denotes the failure of convergence within the maximum number (200) of allowed iterations.

Table 6 shows that the inner-outer iteration strategies reduce the number of PGMRES iterations for most of the tested matrices. Especially, ILU(0) with the inner-outer iteration strategies is shown to be effective for solving add32, bfw398a, lop163, pde225, pde900, sherman, and watt1 matrices. ILUT with different number of inner-outer iteration strategies works well on the pde and sherman matrices. In general, larger number of inner iterations reduces the number of outer iterations needed for convergence. However, the

reduction of the number of outer iterations does not always lead to the reduction of the total CPU time, as shown in Table 7. This is because that the cost of employing more inner iterations sometimes outweighs the savings obtained in the reduction of the number of outer iterations, as each outer iteration becomes more expensive to conduct. For the purpose of maintaining robustness, however, two or three inner iterations may be used in each outer iteration.

### 3.3 Comprehensive Results: Comparisons between Error Compensation and Inner-Outer Iteration Strategies

Table 8 shows the comparison of the number of PGMRES iterations with different enhanced preconditioning strategies. The data in the columns marked “Both” and “BothL” mean that applying “Full” and “Lpart” followed by “2-Its”.

The data in the Table 8 indicate that most of the proposed strategies are of benefit to reduce the number of PGMRES iterations, and show that “Both” (the error compensation strategy with the L and U part followed by the inner-outer iteration strategy) has best results for almost all tested matrices; 12 (8) matrices have best results with the “Both,” and 3 (2) matrices with the “2-Its” (inner-outer iteration strategy), in the context of ILU(0) (ILUT). Also, “BothL”

Table 6: Comparison of the number of PGMRES iterations with different number of inner iterations.

Matrix	ILU(0)				ILUT			
	1-Its	2-Its	3-Its	4-Its	1-Its	2-Its	3-Its	4-Its
add32	81	38	43	27	16	9	7	6
bfw398a	-1	117	116	50	-1	-1	-1	-1
jpwh991	29	15	13	10	32	18	13	11
lop163	-1	110	98	66	18	8	7	5
orsirr1	41	22	20	16	50	25	30	17
pde225	30	16	10	8	16	8	7	5
pde900	61	31	18	15	33	16	11	9
sherman1	116	52	57	32	107	50	47	30
sherman4	143	65	46	33	136	60	36	26
sherman5	101	54	39	29	-1	87	72	57
watt1	178	66	53	36	108	49	39	31

Table 7: Comparison of CPU time in seconds of ILU(0) and ILUT with different number of inner iterations.

Matrix	ILU(0)				ILUT			
	1-Its	2-Its	3-Its	4-Its	1-Its	2-Its	3-Its	4-Its
add32	1.1E+00	1.1E+00	1.4E+00	1.2E+00	1.2E+00	1.2E+00	1.5E+00	1.3E+00
bfw398a	N/A	1.7E-01	2.4E-01	1.6E-01	N/A	N/A	N/A	N/A
jpwh991	7.0E-02	7.0E-02	8.9E-02	7.0E-02	1.1E-01	1.1E-01	1.3E-01	1.2E-01
lop163	N/A	4.9E-02	5.9E-02	3.9E-02	6.0E-02	4.9E-02	7.0E-02	3.9E-02
orsirr1	9.0E-02	1.3E-01	1.1E-01	1.2E-01	1.4E-01	1.7E-01	1.7E-01	1.7E-01
pde225	1.9E-02	1.9E-02	1.9E-02	1.9E-02	1.9E-02	2.9E-02	2.9E-02	2.9E-02
sherman1	1.5E-01	1.0E-01	1.3E-01	1.2E-01	2.4E-01	1.7E-01	2.1E-01	1.8E-01
watt1	5.1E-01	3.9E-01	3.7E-01	3.6E-01	7.7E-01	5.9E-01	5.9E-01	5.8E-01

Table 8: Comparison of the number of PGMRES iterations with and without different error compensation strategies.

Matrix	ILU(0)					ILUT				
	No	Full	2-Its	Both	BothL	No	Full	2-Its	Both	BothL
bfw398a	-1	150	117	81	88	-1	-1	-1	-1	-1
cdde1	-1	102	94	55	77	164	102	66	44	54
cdde3	-1	-1	-1	185	-1	-1	-1	196	198	175
orsirr1	41	40	22	22	22	50	41	25	22	25
orsirr2	40	41	22	22	22	51	43	25	24	25
orsreg1	51	50	22	24	25	52	52	26	22	25
pde2961	-1	-1	-1	-1	-1	85	66	51	31	47
pde900	61	46	31	22	27	33	22	16	16	15
pores2	117	148	-1	-1	82	-1	-1	-1	-1	-1
pores3	108	64	52	44	54	121	127	88	53	82
saylr1	33	33	16	16	15	75	-1	24	31	24
saylr4	-1	-1	185	163	-1	-1	-1	-1	187	-1
sherman1	116	62	52	43	52	107	87	50	51	50
sherman4	143	107	65	43	55	136	101	60	43	37
sherman5	101	73	54	44	50	-1	145	87	76	79
watt1	178	114	66	40	69	108	85	49	37	49

(the error compensation strategy with the L part only followed by the inner-outer iteration strategy) is best to 4 (5) matrices with ILU(0) (ILUT), respectively. As a result, the error compensation strategy followed by the two inner-outer iteration strategy outperformed the other error compensation strategies based on this experiment.

Table 9 and Table 10 present the total CPU processing time which includes the preconditioner construction time with different error compensation strategies. The listed results in Table 9 and Table 10 are concerned with the results listed in Table 8. "N/A" indicates the failure of computation since the GMRES iteration number is reached the maximum

number(200). This implies that as the number of overall iterations are greatly decreased, and the CPU time is also reduced even though we need extra processing time for implementing the error compensation strategies.

#### 4. Conclusion and Future Work

We proposed preconditioning accuracy enhancement strategies to augment the ILU factorizations in case that the initial factorizations are found to be inaccurate. The strategies recompense the dropped elements during the incomplete factorization before/during the preconditioned iteration processes. Results of extensive experiments demonstrated that

Table 9: Comparison of CPU time in seconds with the full error compensation strategies.

Matrix	ILU(0)				
	No	Full	2-Its	Both	BothL
bfw398a	N/A	1.9E-01	1.7E-01	1.9E-01	1.6E-01
cdde1	N/A	1.6E-01	2.1E-01	1.5E-01	1.8E-01
cdde3	N/A	N/A	N/A	3.4E-01	N/A
orsirr1	9.0E-02	1.3E-01	1.3E-01	1.4E-01	1.5E-01
orsirr2	8.9E-02	1.1E-01	9.9E-02	1.2E-01	1.1E-01
orsreg1	2.6E-01	3.9E-01	2.7E-01	3.3E-01	3.1E-01
pde2961	N/A	N/A	N/A	N/A	N/A
pde900	9.0E-02	1.0E-01	8.9E-02	9.0E-02	8.9E-02
pores2	2.7E-01	4.5E-01	N/A	N/A	3.9E-01
pores3	7.9E-02	7.9E-02	9.0E-02	1.0E-01	1.0E-01
saylr1	2.9E-02	1.9E-02	1.9E-02	2.9E-02	1.9E-02
saylr4	N/A	N/A	2.1E+00	2.4E+00	N/A
sherman1	1.5E-01	1.1E-01	1.0E-01	1.1E-01	1.2E-01
sherman4	1.6E-01	1.6E-01	1.3E-01	1.2E-01	1.4E-01
sherman5	6.9E-01	7.4E-01	7.3E-01	7.7E-01	7.6E-01
watt1	5.1E-01	4.9E-01	3.9E-01	3.4E-01	4.5E-01

Table 10: Comparison of CPU time in seconds with the full error compensation strategies.

Matrix	ILUT				
	No	Full	2-Its	Both	BothL
bfw398a	N/A	N/A	N/A	N/A	N/A
cdde1	3.6E-01	2.7E-01	3.2E-02	2.4E-01	2.8E-01
cdde3	N/A	N/A	5.9E-01	6.8E-01	6.4E-01
orsirr1	1.4E-01	1.9E-01	1.7E-01	2.0E-01	2.0E-01
orsirr2	1.3E-01	1.7E-01	1.4E-01	1.7E-01	1.6E-01
orsreg1	3.9E-01	5.8E-01	3.9E-01	4.7E-01	4.4E-01
pde2961	1.2E+00	1.3E+00	1.4E+00	1.6E+00	1.6E+00
pde900	1.2E-01	1.4E-01	1.3E-01	1.4E-01	1.4E-01
pores2	N/A	N/A	N/A	N/A	N/A
pores3	1.4E-01	1.6E-01	1.6E-01	1.6E-01	1.8E-01
saylr1	3.9E-02	N/A	2.9E-02	3.9E-02	2.9E-02
saylr4	N/A	N/A	N/A	4.2E+00	N/A
sherman1	2.4E-01	1.9E-01	1.7E-01	1.9E-01	1.2E-01
sherman4	2.9E-01	2.8E-01	2.2E-01	2.0E-01	2.0E-01
sherman5	N/A	1.6E+00	1.3E+00	1.5E+00	1.4E+00
watt1	7.7E-01	7.7E-01	5.9E-01	5.6E-01	6.9E-01

the strategies are quite promising and effective in helping reduce the number of PGMRES iterations, and a noticeable improvement in enhancing the accuracy of the incomplete LU factorizations. Furthermore, the total computation time for solving sparse linear system was reduced even though extra processing time was used for the proposed strategies.

Our subsequent work will be to extend the strategies to solving other classes of general sparse matrices and sparse matrices from particular application areas.

## References

- [1] M. Benzi, *Preconditioning techniques for large linear systems: a survey*. J. Comput. Phys., 182:418–477 (2002).
- [2] T.F. Chan, and H.A. van der Vorst, *Approximate and incomplete factorizations*. Parallel Numerical Algorithms, ICASE/LaRC Interdisciplinary Series in Science and Engineering, 4:91–118 (1997). (<http://www.math.uu.nl/people/vorst/publ.html#publ94>)
- [3] E. Chow, and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*. J. Comput. Appl. Math., 86:387–414 (1997).
- [4] I.S. Duff, R.G. Grimes, and J.G. Lewis, *Users' Guide for the Harwell-Boeing Sparse Matrix Collection (Release I)*. Technical Report RAL-92-086, Rutherford Appleton Laboratory, Oxfordshire, England (1992).
- [5] I.S. Duff, and G.A. Meurant, *The effect of reordering on preconditioned conjugate gradients*. BIT, 29:635–657 (1989).
- [6] E.-J. Lee, and J. Zhang, *Hybrid reordering strategies for ILU preconditioning of indefinite sparse matrices*, J. Appl. Math. Comput., 22:307–316 (2006).
- [7] Y. Saad, *ILUT: a dual threshold incomplete LU factorization*. Numer. Linear Algebra Appl., 14:387–402 (1994).
- [8] Y. Saad, *Iterative Methods for Sparse Linear Systems*. PWS, New York, 1996.
- [9] J. Zhang, *A multilevel dual reordering strategy for robust incomplete LU factorization of indefinite matrices*. SIAM J. Matrix Anal. Appl., 22:925–947 (2001).
- [10] R.S. Varga. *Matrix Iterative Analysis*. Springer Verlag, Berlin, New York, 2000.

# General Algorithms for Computing Derivatives of Repeated Eigenvalues and Eigenvectors of Symmetric Quadratic Eigenvalue Problems

Delin Chu<sup>1</sup>, Jiang Qian<sup>2,1</sup>, and Roger C.E. Tan<sup>1</sup>

<sup>1</sup>Department of Mathematics, National University of Singapore, Singapore

<sup>2</sup>School of Sciences, Beijing University of Posts and Telecommunications, Beijing, China

**Abstract**—*The numerical computation of derivatives of eigenvalues and eigenvectors has been an active research topic due to its wide applications in engineering and the physical sciences. There are many numerical methods available in the literatures for computing derivatives of eigenvalues and eigenvectors for standard eigenvalue problems and quadratic eigenvalue problems. However, almost all existing methods for quadratic eigenvalue problems have a common limitation, that is, they are based on the assumption that repeated eigenvalues have distinct first order derivatives. In this paper, we lift this assumption and develop general algorithms for computing derivatives, of any arbitrary order, of repeated eigenvalues and corresponding eigenvectors of quadratic eigenvalue problems, under much more general conditions than existing methods. The effectiveness of our algorithms are illustrated by some numerical examples.*

**Keywords:** derivatives of eigenvalues and eigenvectors, repeated eigenvalues, quadratic eigenvalue problems

## 1. Introduction

Sensitivity analysis of eigenvalues problems is essential in many practical applications, such as dynamical response analysis, structural analysis and design, model updating, damage detection, and diffraction grating theory. For example, in the analysis, design and reconstruction of diffractive grating structures, the sensitivity of the physical quantities including the reflected and transmitted amplitudes of the electric field or the diffracted field and the shape of the diffraction grating with respect to different physical parameters play critical roles for measuring how these quantities behavior for small changes in the parameters. To find sensitivity of these physical quantities, derivatives of resulting eigenvalues and eigenvectors with respect to these parameters must be computed, see [6], [11] for more details. Sensitivity analysis consists of two aspects: theoretical analysis and numerical computation. Theoretical analysis focuses on existence of derivatives of eigenvalues and eigenvectors with respect to parameters, and numerical computation is on numerical methods for computing these derivatives.

In this paper we consider the following symmetric quadratic eigenvalue problem depending on a parameter

$\rho \in \mathbf{R}$ :

$$(\lambda^2(\rho)M(\rho) + \lambda(\rho)C(\rho) + K(\rho))x(\rho) = 0, \quad (1)$$

where  $M(\rho) = M(\rho)^\top$ ,  $C(\rho) = C(\rho)^\top$ ,  $K(\rho) = K(\rho)^\top \in \mathbf{R}^{n \times n}$  are analytic functions of  $\rho$  throughout some open interval  $D_0 \subset \mathbf{R}$  containing  $\rho_0$ , and  $\lambda(\rho) \in \mathbf{C}$  and  $x(\rho) \in \mathbf{C}^n$  are called eigenvalues and eigenvectors of the symmetric quadratic eigenvalue problem (1), respectively. Quadratic eigenvalue problems arise frequently in areas such as applied mechanics, electrical oscillation, vibro-acoustics, fluid dynamics, signal processing, and finite element model of some critical partial differential equations [10]. We consider the case when the symmetric quadratic eigenvalue problem (1) has a semi-simple eigenvalue  $\lambda_1$  of multiplicity  $r$  at  $\rho = \rho_0$ . Assume that there exist  $r$  analytic functions  $\lambda_1(\rho), \dots, \lambda_r(\rho)$  that are eigenvalues of (1) throughout some open interval  $D \subset D_0$  such that  $\rho_0 \in D$ , and  $\lambda_1(\rho_0) = \dots = \lambda_r(\rho_0) = \lambda_1$ , and we also assume that the eigenvector functions  $x_i(\rho)$  of (1) corresponding to  $\lambda_i(\rho)$  are analytic at  $\rho = \rho_0$ . We are interested in computing derivatives of these analytic functions  $\lambda_1(\rho), \dots, \lambda_r(\rho), x_1(\rho), \dots, x_r(\rho)$ .

Repeated eigenvalues arise in many applications, for example, model updating problems and optimization problems. Since the eigenvalues  $\lambda_1(\rho), \dots, \lambda_r(\rho)$  are analytic in  $D$ , it follows that for every pair of eigenvalues  $\lambda_i(\rho), \lambda_j(\rho)$ , either  $\lambda_i(\rho) = \lambda_j(\rho)$  for all values of  $\rho$  in  $D$  (often in model updating problems), or there is no open neighborhood of  $\rho_0$  throughout which  $\lambda_i(\rho) = \lambda_j(\rho)$ . In the first case, the eigenvector derivatives are not uniquely defined, which will not be considered in our paper. This paper concerns only with the more common second case, which often arises in optimization problems, where repeated eigenvalues often occur at the optimum point but not at any neighboring point. Another important reason for proposing methods for the case when derivatives are repeated is that, while derivatives of repeated eigenvalues are not themselves likely to be equal, they could well be very close, and in the presence of round-off, there is no clear distinction between quantities which are exactly equal and those which are merely very nearly equal. Methods designed for problems where the derivatives are distinct will normally break down when the derivatives are sufficiently close.

Many efforts have been devoted to numerical computation of derivatives eigenpairs of various eigenvalue problems. The existing approaches can be classified into the modal methods, the direct methods and the iterative methods. The modal method [12] is based on expanding the derivatives of eigenvectors as linear combinations of modes (eigenvectors). But this approach requires knowledge of all eigenvectors, which is, however, often difficult in engineering applications. Generally, only the low-frequency modes can be obtained and are used as the basis vectors of eigenvector derivatives. The direct method tends to find derivatives of eigenvalues and eigenvectors by solving some linear systems or eigenvalue problems using only the knowledge of eigenvalues and eigenvectors being considered. Friswell and Adhikari [4], Guedria et al. [5] generalized Nelson's method [7] to compute the first and second order derivatives of eigenvectors corresponding to distinct eigenvalues. Tang et al. [9] investigated methods for computing derivatives with repeated eigenvalues for general asymmetric systems. But they only consider the case when the repeated eigenvalues have distinct first order derivatives. The iterative methods [1], [3] can be used to find derivatives of several eigenpairs of a matrix simultaneously, and they can be accelerated by using the techniques described in [8]. However, we are not aware of any iterative methods for computing derivatives of eigenpairs of quadratic eigenvalue problems.

To the best of our knowledge, all these existing direct methods for compute derivatives of repeated eigenvalues and corresponding eigenvectors of quadratic eigenvalue problems are based on the assumption that these repeated eigenvalues have distinct first order derivatives. In this paper, we shall extend the ideas in [2] to symmetric second order systems, which is to compute derivatives, of arbitrary order, of repeated eigenvalues and corresponding eigenvectors, under the more general cases when repeated eigenvalues may have repeated derivatives. Specifically, we consider the case when for all  $j \neq k$ , with  $1 \leq j, k \leq r$ ,

$$\left. \begin{aligned} \lambda_j^{(i)}(\rho_0) &= \lambda_k^{(i)}(\rho_0), & i = 0, 1, \dots, m-1 \\ \lambda_j^{(m)}(\rho_0) &\neq \lambda_k^{(m)}(\rho_0), \end{aligned} \right\} \quad (2)$$

hold for some integer  $m$ . Based on these assumptions, Algorithm 1 in Section 2 can be applied to compute derivatives of eigenvalues of order 1 to  $m$  and the first order derivative of eigenvectors. Algorithm 2 in Section 2 then enables us to successively compute derivatives, of any arbitrary higher order, of eigenvalues and eigenvectors. The more general nonsymmetric cases with proofs will be considered in a subsequent paper elsewhere. Numerical examples show that the algorithms developed here are robust and reliable even for close derivatives of eigenvalues.

## 2. Algorithms

Let  $\Lambda(\rho) = \text{diag}(\lambda_1(\rho), \dots, \lambda_r(\rho))$ ,  $X(\rho) = [x_1(\rho) \ \dots \ x_r(\rho)]$ . Then from (1) we have

$$M(\rho)X(\rho)\Lambda(\rho)^2 + C(\rho)X(\rho)\Lambda(\rho) + K(\rho)X(\rho) = 0. \quad (3)$$

To ensure the uniqueness of eigenvectors, we take the following normalization condition for  $i = 1, 2, \dots, r$ ,

$$x_i^\top(\rho_0)(2\lambda_1(\rho_0)M(\rho_0) + C(\rho_0))x_i(\rho) = 1. \quad (4)$$

At  $\rho = \rho_0$ , we further take the normalization condition

$$X^\top(\rho_0)(2\lambda_1(\rho_0)M(\rho_0) + C(\rho_0))X(\rho_0) = I. \quad (5)$$

The computed eigenvalues of the quadratic eigenvalue problem (1) at  $\rho = \rho_0$  are  $\Lambda(\rho_0) = \lambda_1(\rho_0)I$ , while the corresponding eigenvectors may not be  $X(\rho_0)$  described as above. Let the computed linearly independent eigenvectors corresponding to  $\lambda_1(\rho_0)$  be the columns of the  $n \times r$  matrix  $X_0$ . Then there exists a nonsingular  $r \times r$  matrix  $C_0$  such that

$$X(\rho_0) = X_0C_0. \quad (6)$$

Without loss of generality, we assume that the computed  $X_0$  satisfies

$$X_0^\top(2\lambda_1(\rho_0)M(\rho_0) + C(\rho_0))X_0 = I. \quad (7)$$

Then combining (5), (6) and (7) shows that  $C_0^\top C_0 = I$ . A main task for computing derivatives of eigenvectors is to find  $C_0$ .

Throughout the rest of the paper, all functions are assumed to be evaluated at  $\rho = \rho_0$  unless stated otherwise. For simplicity, we will take the following notation:

$$\begin{aligned} \lambda &= \lambda_1, \quad Q(\lambda) = \lambda^2 M + \lambda C + K, \quad Q'(\lambda) = 2\lambda M + C, \\ W_p &= \sum_{q=0}^p \binom{p}{q} \sum_{s=0}^q \binom{q}{s} \lambda^{(q-s)} \lambda^{(s)} M^{(p-q)} \\ &+ \sum_{q=0}^p \binom{p}{q} \lambda^{(q)} C^{(p-q)} + K^{(p)}, \quad p = 0, \dots, m-1, \quad (8) \\ Z_i &= \sum_{q=0}^{i-1} \binom{i}{q} \sum_{s=0}^q \binom{q}{s} \lambda^{(q-s)} \lambda^{(s)} M^{(i-q)} \\ &+ \sum_{s=1}^{i-1} \binom{i}{s} \lambda^{(i-s)} \lambda^{(s)} M + \sum_{q=0}^{i-1} \binom{i}{q} \lambda^{(q)} C^{(i-q)} + K^{(i)}, \\ & \quad i = 1, \dots, m. \quad (9) \end{aligned}$$

The following Algorithm 1 can be applied to find  $\Lambda^{(i)}$  ( $i = 1, \dots, m+1$ ) and  $X'$ , where  $\Lambda^{(i)}$  denotes the  $i$ -th derivative of  $\Lambda$ .

### Algorithm 1:

1. Set  $i = 1$ ,  $V_0 = X_0$ , compute  $Z_1 = \lambda^2 M' + \lambda C' + K'$  and  $M_1 = -X_0^\top Z_1 X_0$ .
2. Compute the average of the diagonal elements of  $M_i$  as  $\lambda^{(i)}$ . If  $\|M_i - \lambda^{(i)}\| \geq \epsilon$ , set  $m = i$  and go to 5.

3. Compute  $W_i = Z_i + \lambda^{(i)}Q'(\lambda)$ , find a solution  $V_i$  to the equation

$$Q(\lambda)V_i = -\sum_{p=1}^i \binom{i}{p} W_p V_{i-p},$$

compute  $Z_{i+1}$  by (9) and compute

$$M_{i+1} = -X_0^\top \left( \sum_{p=1}^i \binom{i+1}{p} W_p V_{i+1-p} + Z_{i+1} X_0 \right).$$

4. Set  $i = i + 1$  and go to 2.

5. Find eigenvalues and eigenvectors of the matrix  $M_m$ . Let  $\Lambda^{(m)} = \text{diag}(l_1, \dots, l_r)$  be the diagonal matrix with  $l_i$  being eigenvalues of  $M_m$ , and form  $C_0$  whose columns are corresponding eigenvectors of  $M_m$  and are normalized such that  $C_0^\top C_0 = I$ . Compute  $X = X_0 C_0$ .

6. Find a solution  $V_m$  to the equation

$$Q(\lambda)V_m = -\sum_{p=1}^{m-1} \binom{m}{p} W_p V_{m-p} - Z_m X_0 - Q'(\lambda)X\Lambda^{(m)}C_0^\top,$$

and compute

$$\begin{aligned} M_{m+1} &= -X^\top \left( \sum_{p=1}^{m-1} \binom{m+1}{p} W_p V_{m+1-p} C_0 \right. \\ &+ (m+1)Z_m V_1 C_0 + (m+1)Q'(\lambda)V_1 C_0 \Lambda^{(m)} \\ &+ \left( \sum_{p=0}^{m-1} \binom{m+1}{p} \sum_{q=0}^p \binom{p}{q} \lambda^{(p-q)} \lambda^{(q)} M^{(m+1-p)} \right. \\ &+ (m+1) \sum_{p=1}^{m-1} \binom{m}{p} \lambda^{(m-p)} \lambda^{(p)} M' \\ &+ \sum_{p=2}^{m-1} \binom{m+1}{p} \lambda^{(m+1-p)} \lambda^{(p)} M \\ &+ \sum_{p=0}^{m-1} \binom{m+1}{p} \lambda^{(p)} C^{(m+1-p)} + K^{(m+1)} \left. \right) X \\ &+ (m+1)(2\lambda M' + C' + 2\lambda' M)X\Lambda^{(m)} \\ &+ 2(m+1)MX\Lambda'\Lambda^{(m)}. \end{aligned}$$

7. Set  $\Lambda^{(m+1)} = \text{diag}(M_{m+1})$ , that is,  $\Lambda^{(m+1)}$  is a diagonal matrix whose diagonal elements are corresponding diagonal elements of  $M_{m+1}$ .

8. Compute the matrix  $C_1$ , whose off-diagonal elements  $c_{ij}$  are given by

$$c_{ij} = \frac{m_{ij}}{(m+1)(l_j - l_i)}, \quad i \neq j,$$

where  $m_{ij}$  is the  $(i, j)$  element of the matrix  $M_{m+1}$ , and  $l_i$  is the  $i$ -th diagonal element of the diagonal matrix  $\Lambda^{(m)}$  computed from Step 5, and whose diagonal elements

are the corresponding diagonal elements of the matrix  $-X^\top Q'(\lambda)V_1 C_0$ .

9. Compute  $X' = V_1 C_0 + X C_1$ .

**Remark 1.** The tolerance  $\epsilon$  in Step 2 is to determine  $m$ , since it is not known a priori.

**Remark 2.** Although we can prove that  $M_i = \Lambda^{(i)} = \lambda^{(i)}I$  for  $i < m$  in exact arithmetic, the diagonal elements of computed  $M_i$  may not be equal. So we compute the average of the diagonal elements of  $M_i$  as  $\lambda^{(i)}$  to minimize the effect of roundoff.

**Remark 3.** There are several methods for solving the equations in Step 3 and Step 4 for  $V_i$ . For example, we can use the QR decomposition or SVD of the singular coefficient matrix.

Algorithm 1 enables us to compute  $\Lambda, \Lambda', \dots, \Lambda^{(m+1)}$  and  $X, X'$ . The following Algorithm 2 then enables us to successively compute derivatives of arbitrary order. Precisely, for arbitrary positive integer  $k$ , we can compute  $\Lambda^{(m+k)}$  and  $X^{(k)}$  by using the values of  $\Lambda, \Lambda', \dots, \Lambda^{(m+k-1)}$  and  $X, X', \dots, X^{(k-1)}$ , where  $X^{(i)}$  denotes the  $i$ -th derivative of  $X$ .

**Algorithm 2:**

1. For  $i = k, \dots, m+k-1$ , compute a solution  $V_{ik}$  to the equation

$$\begin{aligned} Q(\lambda)V_{ik} &= -\sum_{p=0}^{k-1} \binom{i}{p} \sum_{q=0}^{i-p} \binom{i-p}{q} C^{(i-p-q)} X^{(p)} \Lambda^{(q)} \\ &- \sum_{p=0}^{k-1} \binom{i}{p} K^{(i-p)} X^{(p)} - \sum_{p=k}^{i-1} \binom{i}{p} W_{i-p} V_{pk} \\ &- \sum_{p=0}^{k-1} \binom{i}{p} \sum_{q=0}^{i-p} \binom{i-p}{q} M^{(i-p-q)} X^{(p)} \sum_{s=0}^q \binom{q}{s} \Lambda^{(q-s)} \Lambda^{(s)}. \end{aligned}$$

2. Compute

$$\begin{aligned} M_{m+k} &= -X^\top \left( MX \sum_{s=1}^{m+k-1} \binom{m+k}{s} \Lambda^{(m+k-s)} \Lambda^{(s)} \right. \\ &+ \sum_{q=0}^{m+k-1} \binom{m+k}{q} M^{(m+k-q)} X \sum_{s=0}^q \binom{q}{s} \Lambda^{(q-s)} \Lambda^{(s)} \\ &+ \sum_{q=0}^{m+k-1} \binom{m+k}{q} C^{(m+k-q)} X \Lambda^{(q)} + K^{(m+k)} X \\ &+ \sum_{p=1}^{k-1} \binom{m+k}{p} \sum_{q=0}^{m+k-p} \binom{m+k-p}{q} \\ &\left. M^{(m+k-p-q)} X^{(p)} \sum_{s=0}^q \binom{q}{s} \Lambda^{(q-s)} \Lambda^{(s)} \right) \end{aligned}$$



$$\begin{aligned}
& + \sum_{p=1}^{k-1} \binom{m+k}{p} \sum_{q=0}^{m+k-p} \binom{m+k-p}{q} C^{(m+k-p-q)} X^{(p)} \Lambda^{(q)} \\
& + \sum_{p=1}^{k-1} \binom{m+k}{p} K^{(m+k-p)} X^{(p)} \\
& + \binom{m+k}{k} \left( Z_m V_{kk} + Q'(\lambda) V_{kk} \Lambda^{(m)} \right) \\
& + \sum_{p=k+1}^{m+k-1} \binom{m+k}{p} W_{m+k-p} V_{pk} \Big).
\end{aligned}$$

3. Set  $\Lambda^{(m+k)} = \text{diag}(M_{m+k})$ , that is,  $\Lambda^{m+k}$  is a diagonal matrix whose diagonal elements are corresponding diagonal elements of  $M_{m+k}$ .

4. Compute the matrix  $C_{kk}$ , whose off-diagonal elements  $c_{ij}$  are given by

$$c_{ij} = \frac{m_{ij}}{\binom{m+k}{k} (l_j - l_i)}, \quad i \neq j,$$

where  $m_{ij}$  is the  $(i, j)$  element of  $M_{m+k}$  and  $l_i$  is the  $i$ -th diagonal element of  $\Lambda^{(m)}$ , and whose diagonal elements are the corresponding diagonal elements of the matrix  $-X^\top Q'(\lambda) V_{kk}$ .

5. Compute  $X^{(k)} = V_{kk} + X C_{kk}$ .

By Algorithm 1 and Algorithm 2 we can successively compute derivatives, of any arbitrary order, of the repeated eigenvalues and corresponding eigenvectors.

### 3. Numerical examples

In this section, we will give some examples to illustrate the performance of the proposed algorithms. All examples were carried out using MATLAB 7.5.0, with machine epsilon  $\varepsilon \approx 2.2 \times 10^{-16}$ .

**Example 1.** Consider the following second order system depending on the real parameter  $\rho$ , where

$$\begin{aligned}
M &= S \text{diag}(\rho + 1, \rho - 1, 2\rho^2 - 3\rho - 2, \rho^2 + 4) S^\top, \\
C &= S \text{diag}(-\rho^3 - 2\rho^2 - 3\rho - 3 - \delta\rho(\rho + 1), 2\rho^2 + \rho - 1, \\
&\quad -5\rho^2 - \rho - 3, -5\rho^3 - \frac{1}{2}\rho^2 - 20\rho - 3) S^\top, \\
K &= S \text{diag}(\rho^2 + \rho + 2 + \delta\rho, \rho^3 + 2\rho^2 - \rho + 6, \\
&\quad 2\rho^2 - \rho - 1, 5\rho + \frac{1}{2}) S^\top. \tag{10}
\end{aligned}$$

Here

$$S = \frac{1}{\sqrt{3}} \begin{bmatrix} \cos \rho & 1 & \sin \rho & -1 \\ -\sin \rho & -1 & \cos \rho & -1 \\ 1 & -\sin \rho & 1 & \cos \rho \\ -1 & \cos \rho & 1 & \sin \rho \end{bmatrix}$$

is an orthogonal matrix.

When  $\delta = 0$ , the eigenvalues  $\lambda_1(\rho) = \rho^2 + \rho + 2$  and  $\lambda_2(\rho) = -\rho - \frac{2}{\rho-1}$  become a pair of repeated eigenvalues with multiplicity 2 at  $\rho = 0$ . Specifically,

$$\begin{aligned}
\lambda_1(0) &= \lambda_2(0) = 2, & \lambda_1'(0) &= \lambda_2'(0) = 1, \\
\lambda_1''(0) &= 2 \neq \lambda_2''(0) = 4, & (m=2) \\
\lambda_1'''(0) &= 0, & \lambda_2'''(0) &= 12, & \lambda_1^{(4)} &= 0, & \lambda_2^{(4)} &= 48.
\end{aligned}$$

Denote the exact derivatives of eigenvalues and eigenvectors by  $\Lambda', \Lambda'', \Lambda''', \Lambda^{(4)}$  and  $X', X''$ . Then Algorithm 1 with  $\varepsilon = 10^{-10}$  gives the correct  $m = 2$ , and the computed  $\tilde{\Lambda}', \tilde{\Lambda}'', \tilde{\Lambda}''', \tilde{X}'$  satisfy

$$\begin{aligned}
\frac{\|\tilde{\Lambda}' - \Lambda'\|_F}{\|\Lambda'\|_F} &= 4.44\text{E-}16, & \frac{\|\tilde{\Lambda}'' - \Lambda''\|_F}{\|\Lambda''\|_F} &= 1.26\text{E-}15, \\
\frac{\|\tilde{\Lambda}''' - \Lambda'''\|_F}{\|\Lambda'''\|_F} &= 2.87\text{E-}15, & \frac{\|\tilde{X}' - X'\|_F}{\|X'\|_F} &= 1.15\text{E-}14.
\end{aligned}$$

We can then apply Algorithm 2 ( $m = 2, k = 2$ ) to find  $\Lambda^{(4)}$  and  $X''$ . The computed  $\tilde{\Lambda}^{(4)}$  and  $\tilde{X}''$  satisfy

$$\frac{\|\tilde{\Lambda}^{(4)} - \Lambda^{(4)}\|_F}{\|\Lambda^{(4)}\|_F} = 6.82\text{E-}15, \quad \frac{\|\tilde{X}'' - X''\|_F}{\|X''\|_F} = 2.89\text{E-}14.$$

These show that Algorithm 1 and Algorithm 2 are feasible for computing derivatives of repeated eigenvalues and corresponding eigenvectors of symmetric second order systems.

**Example 2.** Next, consider the system as in (10), while now  $\delta$  is small but nonzero. The eigenvalues  $\lambda_1$  and  $\lambda_2$  are still a pair of repeated eigenvalues with multiplicity 2 at  $\rho = 0$ , while their derivatives are  $\lambda_1'(0) = 1 + \delta, \lambda_2'(0) = 1$ , which are close rather than identical. We apply Algorithm 1 with different tolerance  $\varepsilon$ , which enables us treat the close derivatives as repeated ones ( $m = 2$ ) or distinct ones ( $m = 1$ ). The following Table 1 shows the relative errors of the computed derivatives of eigenvectors  $\|X' - \tilde{X}'\|_F / \|X'\|_F$  for different  $\delta$ , where  $X'$  and  $\tilde{X}'$  are exact and computed derivatives of eigenvectors, respectively.

Table 1: Relative errors of derivatives of eigenvectors

$\delta$	$m = 1$	$m = 2$
E-14	5.38E+14	1.15E-14
E-12	6.64E+9	5.38E-12
E-10	8.57E+5	5.40E-10
E-8	8.00E+1	5.40E-8
E-6	6.11E-3	5.40E-6
E-4	6.34E-7	5.40E-4

Table 1 shows that for small  $\delta$ , if we regard the close derivatives of eigenvalues as distinct ones, the computed derivatives of eigenvectors will be very ill-conditioned. Note that the error by Algorithm 1 with  $m = 2$  is approximately  $\delta$ , whereas by Algorithm 1 with  $m = 1$ , it is close to  $10^{-15}/\delta^2$ . That is, when  $\delta < 10^{-5}$ , it is much better to treat the close derivatives as repeated ones, rather than distinct ones.

## References

- [1] A.L. Andrew, Iterative computation of derivatives of eigenvalues and eigenvectors, *Journal of the Institute of Mathematics and its Applications*, 24(1979), 209–218.
- [2] A.L. Andrew and R.C.E. Tan, Computation of derivatives of repeated eigenvalues and the corresponding eigenvectors of symmetric matrix pencils, *SIAM Journal on Matrix Analysis and Applications*, 20(1998), 78–100.
- [3] A.L. Andrew and R.C.E. Tan, Iterative computation of derivatives of repeated eigenvalues and the corresponding eigenvectors, *Numerical Linear Algebra with Applications*, 7(2000), 151–167.
- [4] M.I. Friswell and S. Adhikari, Derivatives of complex eigenvectors using Nelson's method, *AIAA Journal*, 38(2000), 2355–2357.
- [5] N. Guedria, M. Chouchane and H. Smaoui, Second-order eigensensitivity analysis of asymmetric damped systems using Nelson's method, *Journal of Sound and Vibration*, 300(2007), 974–992.
- [6] M.G. Moharam, D.A. Pommet and E.B. Grann, Stable implementation of the rigorous coupled-wave analysis for surface-relief gratings: enhanced transmittance matrix approach, *Journal of the Optical Society of America A*, 12(1995), 1077–1086.
- [7] R.B. Nelson, Simplified calculation of eigenvector derivatives, *AIAA Journal*, 14(1976), 1201–1205.
- [8] R.C.E. Tan, Some acceleration methods for iterative computation of derivatives of eigenvalues and eigenvectors, *International Journal for Numerical Methods in Engineering*, 28(1989) 1505–1519.
- [9] J. Tang, W.M. Ni and W.L. Wang, Eigensolutions sensitivity for quadratic eigenproblems, *Journal of Sound and Vibration*, 196(1996), 179–188.
- [10] F. Tisseur and K. Meerbergen, The quadratic eigenvalue problem, *SIAM Review*, 43(2001), 235–286.
- [11] N.P. Van Der Aa, *Sensitivity Analysis For Grating Reconstruction*, Eindhoven University of Technology, 2007.
- [12] H.Q. Xie and H. Dai, Calculation of derivatives of multiple eigenpairs of unsymmetrical quadratic eigenvalue problems, *International Journal of Computer Mathematics*, 85(2008), 1815–1831.

**SESSION**  
**PARALLEL COMPUTING AND HPC + CLOUD**  
**COMPUTING**

**Chair(s)**

**TBA**



# Hybrid parallelization of a pure Eulerian finite volume solver for multi-material fluid flows

M. Peybernes<sup>1</sup>, J.-Ph. Braeunig<sup>1</sup>, and J.M. Ghidaglia<sup>2</sup>

<sup>1</sup>CEA, DAM, DIF, F-91297 Arpajon Cedex, France

<sup>2</sup>CMLA, ENS Cachan, 61, Avenue du Président Wilson, 94235 Cachan Cedex, France

**Abstract**—The FVCF-NIP method has been developed since the work of Braeunig et al. [1] for compressible multi-material fluid flows simulation. The main property of this pure Eulerian method is the sliding condition at the interface between materials, which is an improvement in the consistency of the discretization with respect to the Euler equations model. In this paper, we propose a parallelization of this method using a domain decomposition in slices associated with a transposition using the MPI library, and not in blocks as in usual domain decomposition techniques. This is a convenient and efficient choice for this totally directionally splitted method. Then a hybrid parallel algorithm is introduced using multithreading (with OpenMP) and GPU migration (with HMPP) into each slice.

**Keywords:** Finite volume; Multi-material; parallel computing; MPI; OpenMP

## 1. Introduction: the FVCF-NIP method

This method is termed as pure Eulerian Finite Volumes in the sense it does not use an operator splitting as in the Lagrange-Remap scheme for instance. This method is built for compressible multi-material fluid flows simulation. The underlying single-phase scheme is FVCF [2]. The interface between materials is sharp and is approximated by a piecewise linear curve. In a mixed cell, *i.e.* a cell containing more than one material, the interface is then a straight line in 2D, which separates partial volumes containing each a pure material, with no mixing at all. The method is totally cell centred, so each partial volume, as a pure cell, has its own volume centred pressure, velocity vector, density, energy and corresponding EOS. This method conserves locally the mass, the momentum and the total energy, since we write conservation laws on pure cells and even on each partial volume. This is made possible by introducing a 1D data structure called *condensate*, in the context of a directional splitting, which is a merge of neighboring mixed cells in one direction of the mesh, see Figure 1. The evolution of this set of cells is computed no more considering cell faces but considering interfaces between materials as Lagrangian interfaces. A conservation law is written on partial volumes in the *condensate* by computing fluxes through these Lagrangian interfaces. It should be noticed that these fluxes are computed in such a way a

sliding condition is imposed between materials, by writing a Riemann problem in the interface normal vector direction. The new state of the *condensate* is then remapped on the mesh. This technique is restricted to rectangular mesh since the directional splitting is necessary to compute the interface motion using *condensates*.

One interesting feature of this method for parallel computing is that each phase of the directional splitting in a generic direction  $x$  does not need informations from other directions (however, the 2D interfaces normal vectors are used to compute the fluxes and to impose the sliding condition at interfaces). That makes the computation of each cell line of the mesh independent from the others during one step of the directional splitting, so an interesting property for parallel computing. Indeed the parallel algorithm presented here takes advantage of these 1D features using a domain decomposition in slices.

On the other hand, to take full advantage of the largest and fastest computers employing both shared and distributed memory architectures, we propose a hybrid model which combines message passing and shared memory programming.

The remainder of this article is organized as follows. In section 2, we present the MPI parallel algorithm with the associated *transposition*, and the hybrid parallel algorithm. Then we propose a significative multi-material test in section 3 to evaluate the efficiency of the parallel code.

## 2. MPI + (OpenMP or GPU) parallelization

### 2.1 MPI Decomposition in slices

With classical rectangular subdomain decomposition parallelization, a condensate (defined in section 1) can cross several subdomains and then it has to be computed by several processors. Therefore, this method FVCF-NIP using condensates is not well adapted to this kind of parallelization.

The parallel algorithm presented here allows to compute each condensate by a single processor and to take advantage of the 1D directional splitting used in the FVCF-NIP method. During the computation of the  $x$  step of the directional splitting, the 2D domain is decomposed in horizontal slices (Figure 2). Each slice is computed by a processor  $P_i$  on a distributed memory system. In the same way, we use vertical

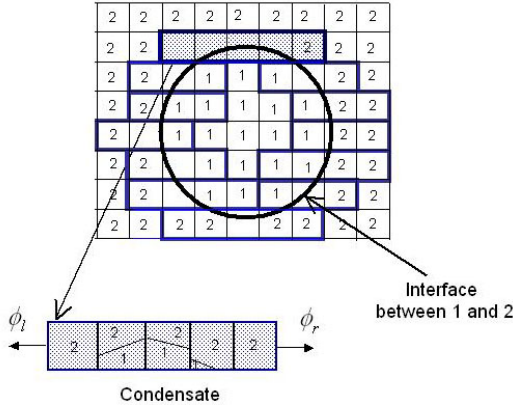


Fig. 1: Extraction of neighboring mixed cells from the grid to become a *condensate* during  $x$  direction step.

slices to decompose the domain during the  $y$  step. Thus, each slice contains the same number of cells equal to the total number of cells ( $NbCells$ ) divided by the number of processors ( $n$ ), what allows a good load balancing between processors.

However, with this kind of parallel algorithm, a processor (attached to a MPI processus) has to compute a different subset of cells (vertical or horizontal slice) at each step ( $x$  or  $y$ ) of the directional splitting. Thus, to allow a given processor to compute the appropriate subset of cells following the direction change, we propose a “transposition algorithm” (see section 2.2). This transposition allows to transfer the necessary data using the Message Passing Interface (MPI).

### 2.2 Transposition

Between two steps of the directional splitting, the transposition allows to move from a horizontal slice decomposition to a vertical slice decomposition and reciprocally. The data transfer is performed using MPI communications. During transposition  $x \rightarrow y$  (Figure 4), each slice is splitted in  $n$  blocks. Then each processor communicates each block  $j$ ,  $j \neq i$ , to processor  $i$ . MPI nonblocking messages are used to obtain an overlap of communications and computations.

To limit the communication cost of the transposition, we communicate the smallest amount of data needed by the algorithm: for each cell, the number of materials, the material volumes, and the conservative quantities per material  $V_m := (\rho, \rho u_x, \rho u_y, \rho E)$ .

Other necessary quantities to the algorithm are deduced from the restricted set of communicated ones thanks to equations of state. Fortunately, this supplementary computations are scalable since they are proportional to the number of cells in a slice ( $NbCells/n$ ).

### 2.3 Hybrid parallel algorithm: multithreading or manythreading into a slice

Modern multicore HPC clusters have hierarchical levels of parallelism. They contain multicore processors (for instance, processor intel Xeon 7500 Nehalem in the TERA100<sup>1</sup> cluster is a eight-cores) with shared memory and graphics processors (GPU’s).

Then, to improve performance of our full MPI version of VFFC-NIP code on these clusters, we have introduced a shared memory threading into each slice using OpenMP. Thus each slice is attached to a processor though a MPI processus and each sub-slice is computed by a core though a thread.

On the other hand, to take advantage of GPU’s and following the previous OpenMP algorithm using directives, we use HMPP directives to allow the migration of a slice computation to a GPU. HMPP is a Heterogeneous Multicore Parallel Programming workbench with compilers, developed by CAPS entreprise, that allows the integration of heterogeneous hardware accelerators in a non-intrusive manner while preserving legacy codes.

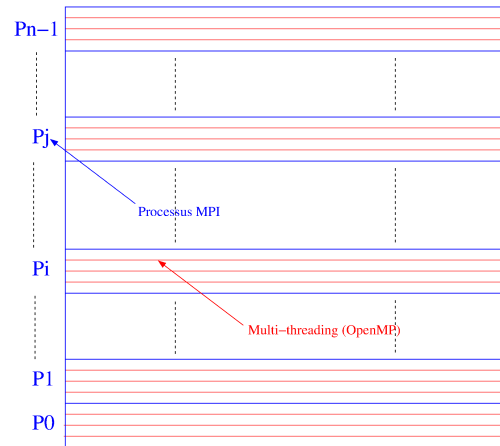


Fig. 2: Decomposition in  $n$  horizontal slices for the  $x$ -step, for  $n$  processors on a distributed memory system

This hybrid parallel algorithm is efficient since MPI communications are all localized in the transposition, between the two steps of the directional splitting (Figure 5). Actually, each cell line can be computed independently from the others, as it has been said in section 1. So during each directional step, a processor makes all the computations for the slice needed by this finite volume method (flux computation, interfaces motion computation...) independently from the others, so without any communication. Then, the “OpenMP region” and “HMPP region” do not contain any MPI communication as we can see in figure 5. Moreover,

<sup>1</sup>The TERA100 cluster is located at Bruyeres-le-Chatel, CEA/DAM-Ile de France center. Built by Groupe Bull, it has a peak processing speed of 1050 teraflops, making it the fastest supercomputer in Europe as of 2011.

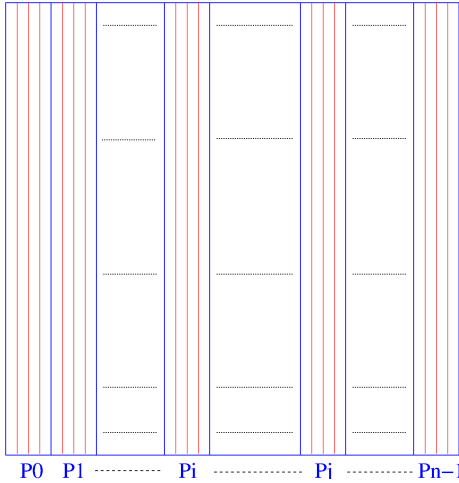


Fig. 3: Decomposition in  $n$  vertical slices for the  $y$ -step, for  $n$  processors on a distributed memory system

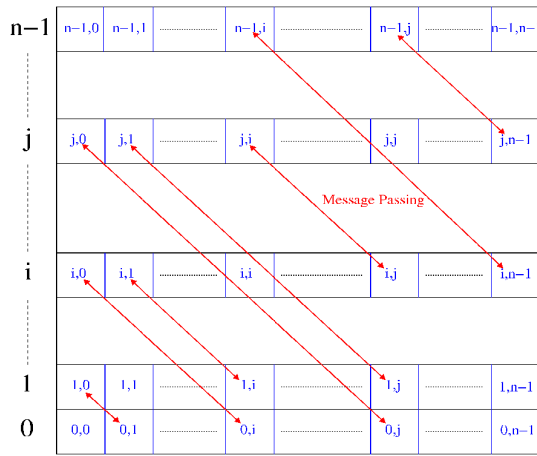


Fig. 4: Transposition  $x \rightarrow y$  : blocks communications

in opposition to classical subdomain parallelization, this method does not need ghost cells. This memory saving is particularly appreciable for supercomputers that tend to have less memory per core.

### 3. Numerical results and performance analysis

The numerical simulations presented in this paper have been performed on the Titane cluster (CCRT: Centre de Calcul Recherche et Technologie, located at Bruyeres-le-Chatel, CEA/DAM-Ile de France center). This massively hybrid parallel computer is composed of a fast network (Infiniband) connecting classical nodes with multicore processors Intel Xeon 5570 (quadri-cores) and accelerators (GPUs Tesla from NVIDIA).

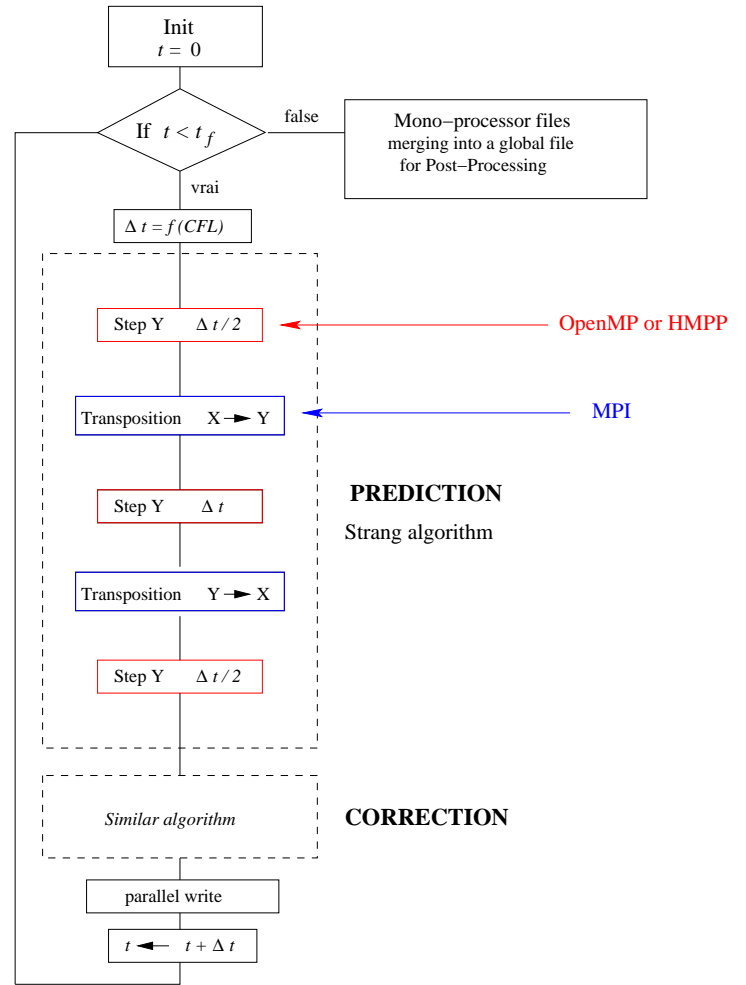


Fig. 5: Parallel algorithm using transposition (MPI).

The test we have chosen to evaluate the efficiency of the parallelization algorithm is a two material “SOD like” shock tube with an initial geometry containing a triple point and three different states. A vortex will be created by the difference of shock waves velocity between horizontal layers in such a way materials will roll up. Equations of state are of type perfect gas with different gamma coefficients between layers. This test is convenient for parallel performance analysis since it creates enough mixed cells to be demonstrative for the interface capturing cost.

In tables 1 and 2, one can see that the efficiency is good ( $> 0.6$ ) up to 64 cores for a 1 million cells test. The data size of blocks communicated during transpositions is decreasing fast by a  $n^2$  factor. Of course, the number of communications follows a  $n^2$  factor as well, so a good balance should be found function of the number of cores. When the data size of blocks decreases, the use of OpenMP allows to keep a

good efficiency up to 256 cores, compared to the full MPI algorithm which has a bad efficiency under 16 000 cells per block. This improvement with MPI+OpenMP is illustrated in Table 3 too, for 4.2 million cells and 512 cores.

Nb cores	1	2	16	32
	1MPI	2MPI	16MPI	32MPI
Elapsed time	1800	910	150	80
Efficiency		0.98	0.75	0.70
Cells/core	1048K	524K	65K	32K

Table 1: Efficiency with 1 million cells

Nb cores	64	256	256
	64MPI	256MPI	64MPI +4Th.
Elapsed time	47	54	17
Efficiency	0.60	0.13	0.41
Cells/core	16K	4K	4K

Table 2: Efficiency with 1 million cells

Nb Cores	512	512	512
	512MPI	256MPI +2Th.	128MPI +4Th.
Elapsed time	650	440	320
Cells/core	8K	8K	8K

Table 3: Efficiency with 4.2 million cells

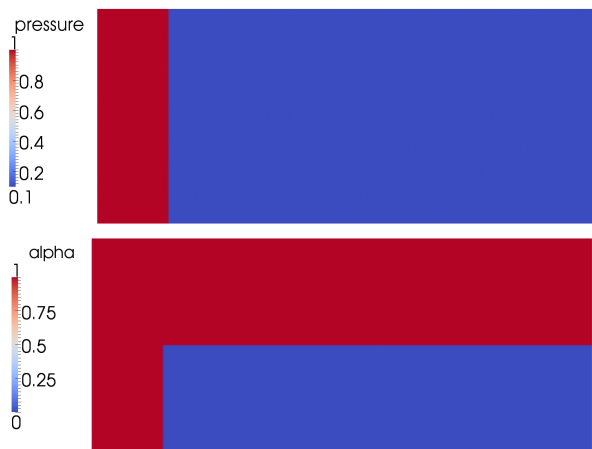


Fig. 6: Pressure (up) and volume fraction (down) at initial time, triple point shock tube.

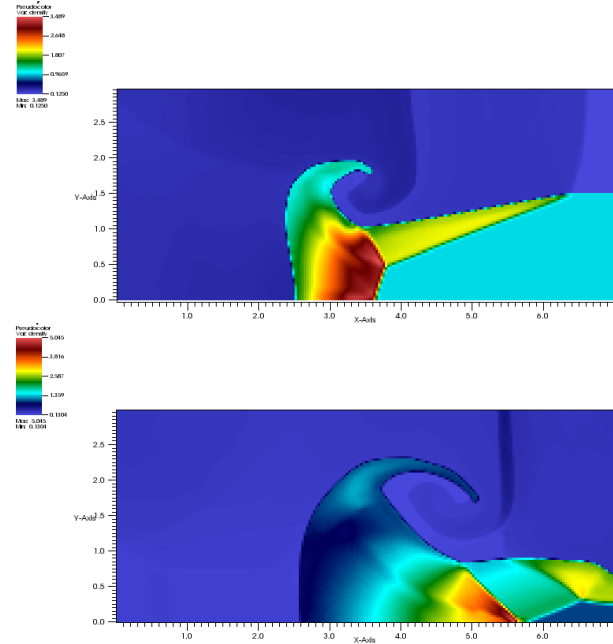


Fig. 7: Density at time 3.3 (up) and at final time 5 (down), triple point shock tube with 19 000 cells.

## 4. Conclusion

We have described a MPI parallelization of the FVCF-NIP method based on a domain decomposition in slices associated with a transposition. It is well adapted for methods using a directional splitting. The independence (without any communications) of each MPI process during a directional step allows to easily add in it multi-threaded shared memory parallelism. The resulting hybrid code (MPI+OpenMP and MPI+GPUwith HMPP) allows to take full advantage of these method features and of heterogeneous architectures of our current and future HPC clusters (massively hybrid parallel clusters).



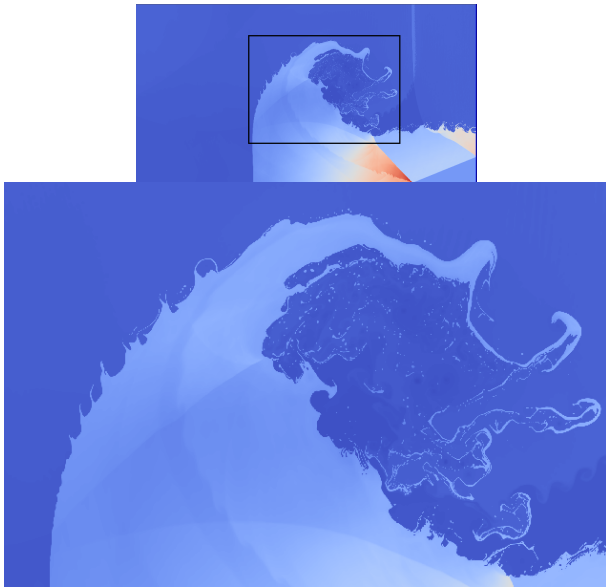


Fig. 8: Density full geometry (up) and zoom on small structures (down) with 12 Million cells, triple point shock tube.

## References

- [1] J.P. Braeunig, B. Desjardins, J.-M. Ghidaglia, "totally Eulerian finite volume solver for multimaterial fluid flows," *European Journal of Mechanics B/Fluids*, vol. 28, No4, pp. 475-485, 2009.
- [2] J.M. Ghidaglia, A. Kumbaro, G. LeCoq, "On the numerical solution to two fluid models via a cell centred finite volume method," *Eur. J. Mech.*, Vol. 20 pp. 841-867, 2001.

# Hybrid Update Algorithms for Regular Lattice and Small-World Ising Models on Graphical Processing Units

A. Leist, K.A. Hawick and D.P. Playne

Computer Science, Institute for Information and Mathematical Sciences,  
Massey University, North Shore 102-904, Auckland, New Zealand

{ a.leist, k.a.hawick, d.p.playne }@massey.ac.nz

Tel: +64 9 414 0800 Fax: +64 9 441 8181

**Abstract**—*Local and cluster Monte Carlo update algorithms offer a complex tradeoff space for optimising the performance of simulations of the Ising model. We systematically explore tradeoffs between hybrid Metropolis and Wolff cluster updates for the 3D Ising model using data-parallelism and graphical processing units. We investigate performance for both regular lattices as well as for small-world perturbations when the lattice becomes a generalised graph and locality can no longer be assumed. In spite of our use of customised Compute Unified Device Architecture (CUDA) code optimisations to implement it, we find the Wolff cluster update loses out in computational performance efficiency over the localised Metropolis algorithm systemically as the small-world rewiring parameter is increased. This manifests itself as a phase transition in the computational performance.*

**Keywords:** GPU; CUDA; Ising model; Wolff; Metropolis

## 1. Introduction

The Ising model [1] is a thoroughly studied model of a computational ferromagnet. Its popularity over the last decades comes from the fact that it is one of the simplest models of a system of interacting particles with some physically realistic features [2]. Like metal alloys that are magnetic only below a certain critical temperature  $T_c$ , also known as the Curie temperature, the Ising model undergoes a phase transition when the system temperature transitions from "hot" to "cold".

When the system is in a "hot" state, then there is no order in the spin values of its cells. But as the system approaches the critical temperature, clumps of like-like spins begin to form, creating order in the formerly unordered system. However, it has been shown that the phase transition only occurs in dimensions greater than one [1]. The cells in the Ising model know the two states "up" and "down",

but extensions to more states can be analysed using the Q-state Potts model [3]. While analytical methods to determine the critical temperature are known for the two-dimensional case [4], no such methods are known for the three or more dimensional cases. Instead, Monte Carlo simulations are often used to approximate the solution using random sampling.

The interactions between neighbouring cells in the ferromagnetic Ising model are defined by a Hamiltonian of the form [5]:

$$H = - \sum_{\langle i,j \rangle} J_{ij} \sigma_i \sigma_j, \quad (1)$$

where  $\sigma_i = \pm 1$ ,  $i = 1, 2, \dots, N$  sites.  $J_{ij}$  is  $|J| = 1/k_B T$  is the ferromagnetic coupling over neighbouring sites  $i$  and  $j$  on the network,  $T$  is the temperature and  $k_B$  is the Boltzmann constant. The total energy  $E$  of a single configuration is obtained from the Hamiltonian. The magnetisation  $M$  is measured from a single configuration as [5]:

$$M = \frac{1}{N} \left| \sum_i \sigma_i \right| \quad (2)$$

The Ising model can also be used to study various other kinds of systems with pairwise correlations between neighbouring nodes, like the propagation of opinions in a social community [6]. However, these models are often more accurately described using irregular graphs instead of the traditionally regular lattices used for the Ising model. Of particular interest are complex structures observed in many natural and social networks. Such networks often exhibit properties that classify them as small-world [7], [8], [9] or scale-free [5] graphs. We are particularly interested in small-world graph structures, and even though Ising model simulations on these networks have been performed in the past [10], [11], [6], the system sizes were rather limited due to the computational complexity

involved. But to determine the scaling of various system properties as the rewiring probability  $p$  – which is used to rewire a fraction of the edges of the originally regular lattice to obtain a network structure similar to that generated by the Watts-Strogatz small-world model [8] – approaches zero, very large systems need to be analysed. Of special interest is the change in the critical temperature with respect to the rewiring probability,  $T_c(p)$ , over several length scales of  $p$ .

To tackle this problem, we have parallelised two commonly used Markov chain Monte Carlo algorithms to run on the highly data-parallel architecture of modern graphics processing units (GPUs), specifically those based on the Compute Unified Device Architecture (CUDA) [12]. The first one is the Metropolis-Hastings algorithm [13], [14], which selects a random cell at every time step and proposes to flip its spin. The new configuration is always accepted if it has a lower energy than the current configuration,  $\Delta E \leq 0$ . Otherwise, it is accepted with probability  $\exp(-\Delta E/k_b T)$ , which depends on the change in energy and the system temperature  $T$ . The parallel implementation of this algorithm is described in [15], where all  $N$  cells of the system are updated during each simulation step.

The Metropolis algorithm is shown to perform very well on the parallel architecture of the GPU and significant speed-ups compared to a sequential CPU implementation are achieved. However, this does not improve upon an issue known as the critical slow down, which is inherent to the local update dynamics of this algorithm. The localised nature of the interactions between individual cells and the size of correlated regions near the critical temperature make it necessary to perform many system updates to obtain two system configurations that are sufficiently decorrelated to be considered as independent.

Cluster updates like those performed by the Wolff algorithm [16] do not suffer from the critical slow down, as they update entire clusters of cells with like spins during every simulation step. The algorithm works by picking a random cell  $i$  and marking it as the first site of a cluster. It then visits all neighbouring cells  $j$  and adds them to the cluster if their spins  $\sigma_i$  and  $\sigma_j$  are equal and if the bond is activated, which happens with random probability  $p(\langle i, j \rangle) = 1 - e^{-2\beta}$ , where  $\beta$  is the reciprocal temperature. The algorithm continues iteratively until no new cells are added to the

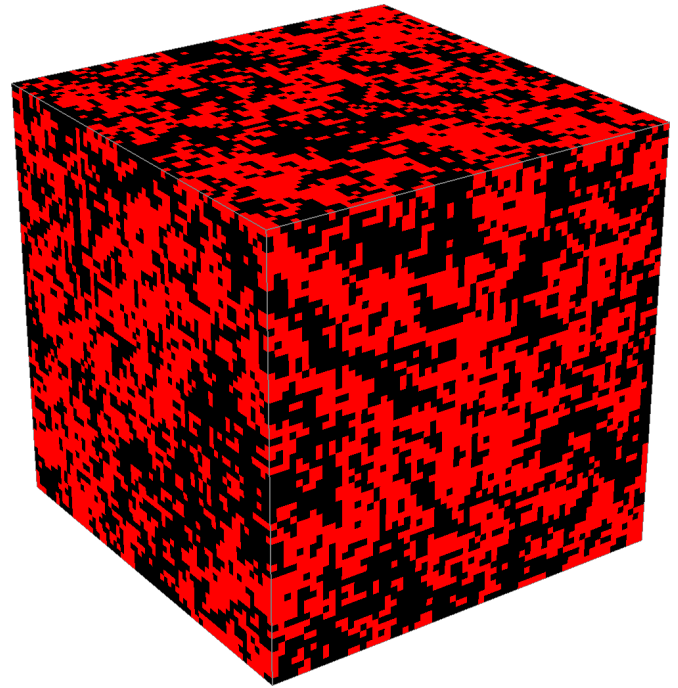


Fig. 1: A  $64 \times 64 \times 64$  Ising model near criticality.

cluster. Then the spin values of the entire cluster are flipped and the simulation step is completed.

While [17] describes how the Wolff algorithm for both regular lattice and small-world Ising models can be parallelised on the GPU, the performance is significantly lower than that of the Metropolis algorithm on the GPU. It is therefore interesting to investigate whether the higher performance of the Metropolis updates or the avoidance of the critical slow down with the Wolff algorithm can produce decorrelated system configurations more quickly and what effect the different graph structures have on the results. Going one step further, we also investigate whether interleaving both algorithms at various ratios offers a performance advantage.

The Ising model simulation has two distinct phases. The system is initialised randomly and must be allowed to settle around its equilibrium energy before its properties are representative for the the given parameters. This first phase is discussed in section 2. Once equilibrated, the actual simulation begins, where decorrelated system configurations are generated and various system properties can be obtained from these independent configurations. This is discussed in section 3. Finally, section 4 offers a discussion and conclusions.

## 2. System Equilibration

Starting from a random configuration, the system is quenched until it reaches its equilibrium state. We look at the performance of the Metropolis algorithm on its own, but do not consider the Wolff algorithm on its own in this phase, as it performs very poorly with the initially random system. The reason for this is that the clusters are very small as a result of the randomness, which is not a good situation for the parallel implementations of the Wolff algorithm. However, two hybrid combinations **M:W** 1:1 and 4:1, are tested. The latter test case is used to investigate whether a relatively small number of the slower Wolff updates is enough to counter the effects of the critical slow down that reduces the efficiency of the fast Metropolis updates. These algorithmic combinations are tested on regular lattice and small-world rewired lattice structures in three-dimensions. Only the results for rewiring probability  $p = 10^{-4}$  are shown for the small-world structures, as they are representative for all other tested rewiring probabilities in the range  $p = 10^{-2}$  to  $p = 10^{-7}$  in this phase.

The system temperature  $T$  is set to the best approximation of the critical temperature  $T_c(p)$  for the given configuration. From previous studies [18], [19], [20], this is known to be  $T_c(p = 0) \approx 4.5115$  for the regular lattice Ising model. But there are no good estimates for the small-world lattice with rewiring probabilities as small as those considered here available at this time. It is the eventual aim of this study to determine these critical temperatures to a high precision. Therefore, we obtain initial rough estimates by computing Binder's fourth-order magnetisation cumulant [21] over a relatively large temperature range with a large step size. The result from this is  $T_c(p = 10^{-4}) \approx 4.514$ , which is then used to calibrate the equilibration phase for more thorough simulation runs.

Essentially, the aim is to ensure that the system is equilibrated but to avoid an unnecessarily large number of system updates. Figure 2 illustrates the change in the system energy over a number of simulation steps and with respect to time for the different algorithmic combinations. The generally more important result is the time to equilibrate the system, as it directly affects the wall clock time required to execute the simulation.

The results for the regular lattice Ising simulation show that all tested algorithmic ratios equi-

brate the system in approximately the same number of steps. It is important to note that only the bottom end of the energy range, where the system settles into its equilibrium state, is shown in the graph. The results of the change in energy with respect to time, on the other hand, show the measurements over the entire energy spectrum. The step functions visible for ratios 4:1 and 1:1 on this plot also reinforce the before mentioned issues of the Wolff cluster updates with the initially random system configuration. During the first few update steps, the Wolff algorithm has almost no effect on the system energy and all the work is performed by the Metropolis updates. The Metropolis algorithm on its own thus equilibrates the system in less time than the hybrid ratios.

The results for the small-world rewired lattice also show that all tested combinations of Metropolis and Wolff updates equilibrate the system in nearly the same number of simulation steps. Although the mean energy measured for the 1:0 ratio is slightly higher than that for the hybrid ratios, the difference is within the region of uncertainty illustrated by the inset to the plot. The plot of the energy with respect to time clearly emphasises the performance difference between the Metropolis and Wolff algorithms. Once again, the step functions visible in the hybrid update measurements illustrate that the Wolff algorithm is not a good choice for this phase.

To verify that the system is indeed equilibrated, the histogram of the energy distribution over a number of simulation steps is plotted as shown in Figure 3. The approximately normal distribution of the values is characteristic for random fluctuations around a mean, in this case the equilibrium energy of the system. From these results, we can be confident that the system with  $p = 10^{-4}$  is equilibrated after 3000 simulation steps of either the 1:0, 1:1 or 4:1 ratios.

## 3. Decorrelated Measurements

Only when the system has reached its equilibrium state for a given temperature can meaningful measurements of its characteristics be taken. But another factor needs to be considered to obtain statistically relevant results too, namely the correlation between successive system states. As demonstrated in Figure 4, every Metropolis or Wolff step only truly updates a fraction of the cells. This is due to the correlated regions of like-like bonds caused by the local update mechanics of the Metropolis

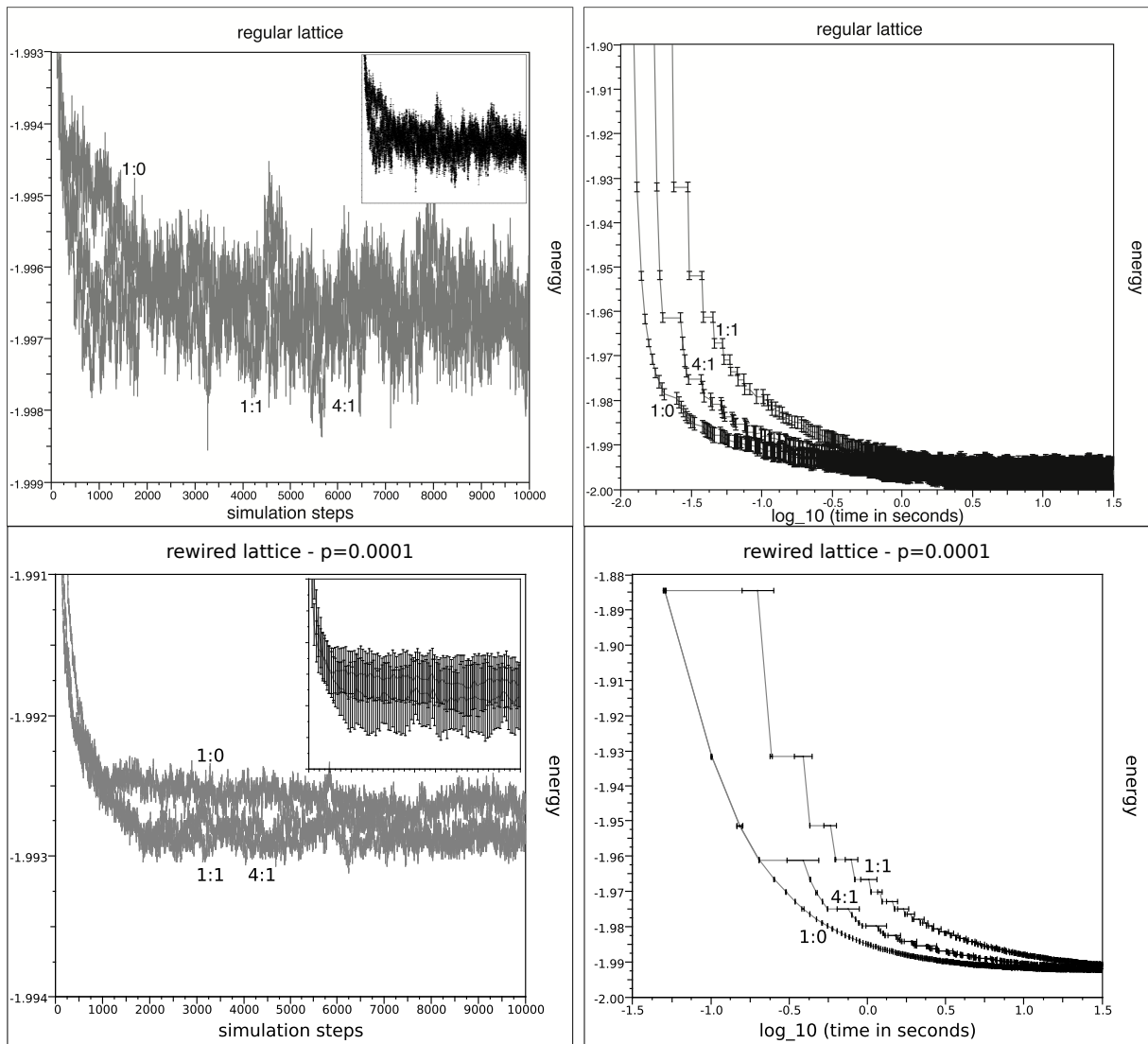


Fig. 2: The plots illustrate the equilibration phase of the Ising model on a regular lattice (top) and rewired lattice with  $p = 10^{-4}$  (bottom). The system size is  $N = 128^3$  for the regular lattice and  $N = 384^3$  for the small-world graph. The change in energy is shown with respect to the number of simulation steps (left) and with respect to time (right). Each data point is averaged over 30 simulation runs. The inset to the plots on the left illustrate the standard deviations for ratios 1:0 and 1:1 to give an idea of the extent of the uncertainty at this level of magnification. The uncertainty in the energy measurements of the timing results is negligible relative to the given energy range, but therefore the actual time measurements show a significant error for some of the data points as illustrated by the horizontal error bars.

algorithm and the fact that only a limited number of cells are part of any one Wolff cluster, especially in the region around the critical temperature. It is therefore necessary to perform enough update steps between measurements to sufficiently decorrelate the system. We use the Pearson product-moment correlation coefficient  $\rho$  [22] to quantify the correlation between two populations. It is defined as:

$$\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}, \quad (3)$$

where  $E$  is the expected value operator,  $\mu_X$  and  $\mu_Y$  are the expected values and  $\sigma_X$  and  $\sigma_Y$  are the standard deviations of populations  $X$  and  $Y$  respectively. The coefficient  $\rho$  is  $+1$  in the case of perfect correlation and  $-1$  in the case of perfect anticorrelation. Here, either case is equally undesirable and, thus, the results presented in this paper use the absolute value of the correlation coefficient  $|\rho|$ . While a coefficient of zero indicates that the metric does not detect any correlation between the two system states, trying to obtain

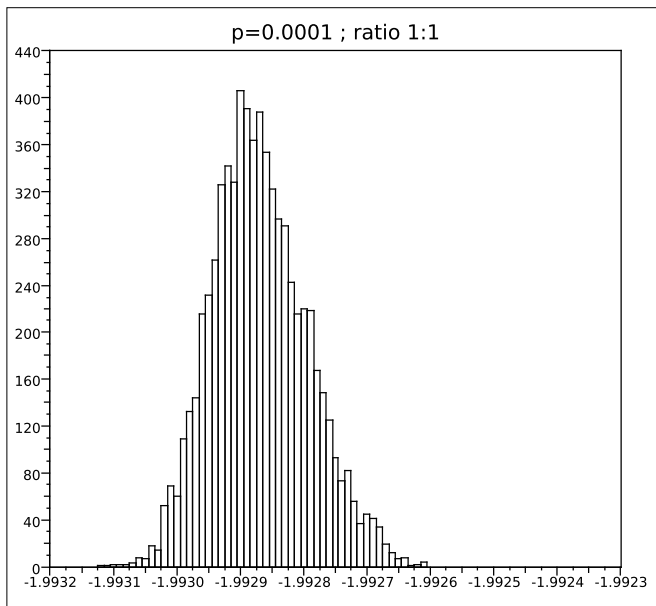


Fig. 3: The histogram shows the energy distribution for simulation steps 3000 to 10000 of the 1:1 hybrid Metropolis-Wolff update ratio and rewiring probability  $p = 10^{-4}$ . The approximately normal distribution of the values indicates that the system is equilibrated.

this level of decorrelation would be inefficient. Therefore, two system states are deemed to be sufficiently decorrelated to be used for independent measurements of system properties when  $|\rho| \leq 10^{-2}$ .

Figure 4 shows how the correlation decreases with the number of system updates and, once again more relevant to the actual execution of the simulation, how it changes with respect to time. The same hybrid algorithmic ratios used during the equilibration phase are tested again. Like before, no results for the Wolff algorithm on its own are shown. The reason to omit them in the already equilibrated system is that the difference in performance compared to the Metropolis algorithm is too great to even consider using only Wolff updates when running the simulations on the GPU. Instead, additional results for the small-world rewired lattice with  $p = 10^{-2}$ ,  $T_c \approx 4.592$  and system size  $N = 352^3$  are given. As demonstrated in the results of the equilibration phase, the Metropolis algorithm parallelises significantly better than the Wolff cluster updates. It is therefore interesting to observe if any of the hybrid algorithmic ratios performs better than the Metropolis algorithm on its own and whether the results are the same for all tested graph structures.

The plots for the results of the regular lattice Ising simulation illustrate that the hybrid updates with both Metropolis and Wolff steps interleaved at different ratios indeed require fewer simulation steps to decorrelate the system configuration to a given degree than the Metropolis algorithm on its own. This is the expected result as – due to the critical slow down effect – the local Metropolis updates are less efficient near the phase transition. However, as the measurements with respect to the wall clock time show, the sheer speed advantage of the data-parallel CUDA implementation of the Metropolis algorithm over the respective implementation of the Wolff algorithm more than makes up for the reduced efficiency.

The results for the small-world Ising simulation give an interesting insight into the effects of the rewired lattice structure on the dynamics of the system. With the small rewiring probability  $p = 10^{-4}$ , the Wolff algorithm still reduces the number of simulation steps required to sufficiently decorrelate the system. However, the results are reversed for  $p = 10^{-2}$ , where the Metropolis algorithm on its own decorrelates the system in fewer steps than the Wolff algorithm. All of the tested ratios produce independent measurements in significantly fewer steps when compared to  $p = 10^{-4}$  and even reach an apparently minimum correlation value of approximately  $10^{-3.6}$ , after which additional simulation steps no longer produce a noticeable change. The explanation for this is that the perturbations to the lattice created during the rewiring procedure help reduce the effects of the critical slow down, as they facilitate long distance interactions that are otherwise not possible. While this benefits both algorithms, the Metropolis updates clearly profit more. The results for the execution time once again show that the much higher performance of the Metropolis updates offsets any advantage offered by the Wolff algorithm for the small-world simulations.

## 4. Discussion & Conclusions

To determine the scaling of the critical temperature with respect to the rewiring probability, tens of millions of system updates have to be performed for every combination of system size, rewiring probability and temperature. And many such combinations are necessary to compute a good approximation of the critical temperature from the cross over of several Binder cumulant curves. To get a single data point on a Binder cumulant curve,

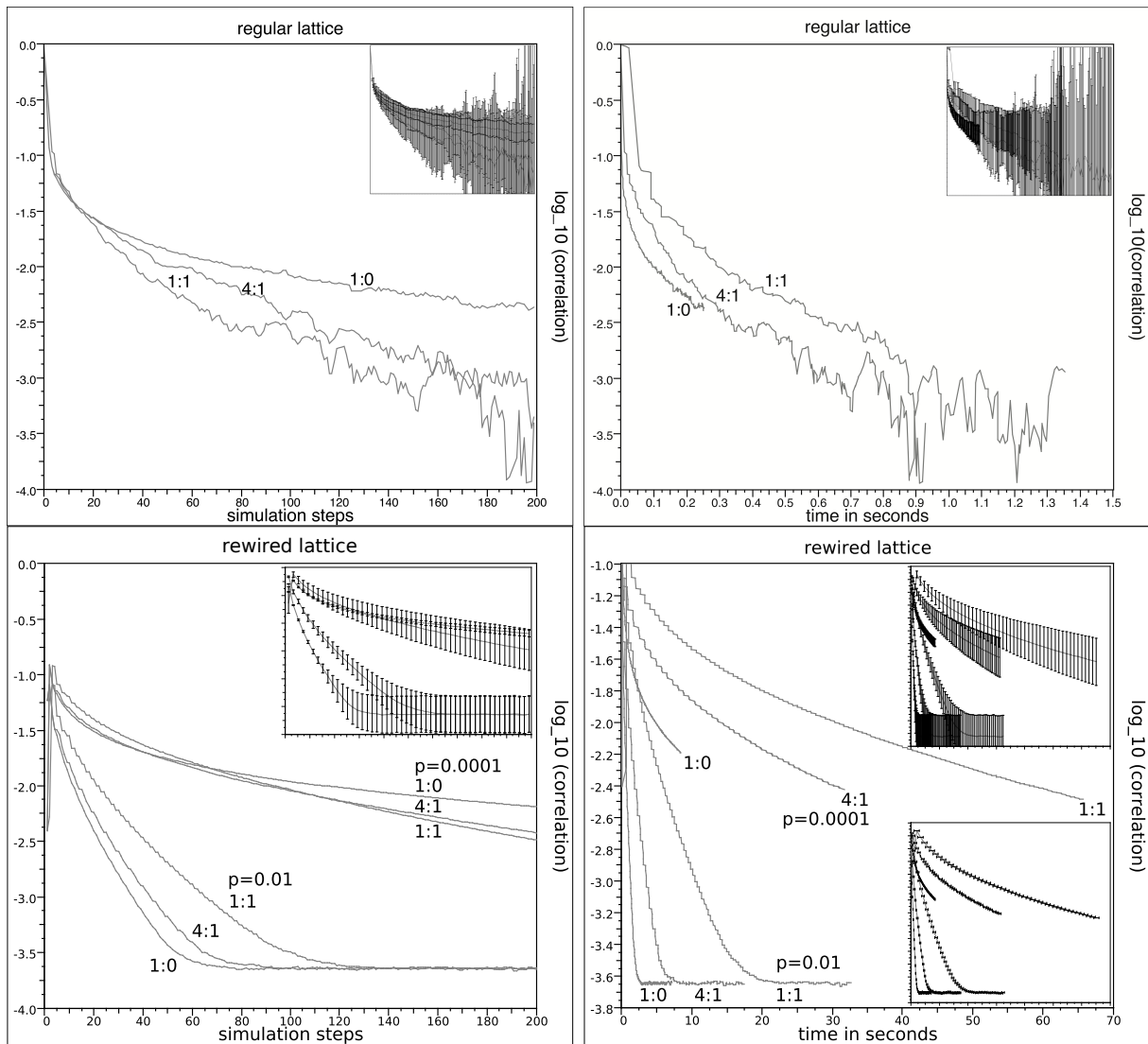


Fig. 4: The graphs show how the correlation between system configurations of the Ising model changes with respect to the number of system updates (left) and with respect to time (right). Different ratios of the Metropolis and Wolff (M:W) update algorithms are compared. The results for simulations on regular lattices (top) and small-world rewired lattices with  $p = 10^{-4}$  and  $p = 10^{-2}$  (bottom) are given. The system size is  $N = 128^3$  for the regular lattice, and  $N = 384^3$  and  $N = 352^3$  for the small-world systems with  $p = 10^{-4}$  and  $p = 10^{-2}$  respectively. Each data point is averaged over 3000 measurements from 30 independent simulation runs. Measurements are taken at intervals of 200 simulation steps (the reference configuration), to which the subsequent 200 system configurations are compared. The insets illustrate the standard deviations from the correlation measurements as error bars. In addition, the second inset to the plot on the lower right shows the error in the time measurements.

a large number of magnetisation values need to be computed from independent system configurations. To make matters even more difficult, the system size required to reliably support a given rewiring probability and remain in the small-world regime must be much larger than the reciprocal of  $p$  [23].

Highly parallel architectures are needed to complete the enormous amount of computation demanded by these simulations in a realistic amount

of time. The GPU has proven to be an excellent choice for affordable, high performance simulations of a large number of complex models. The Metropolis algorithm discussed here can utilise the data-parallel architecture of the GPU particularly well, making it a good choice for the Ising model system updates. Although the Wolff cluster update algorithm can decorrelate system configurations in fewer steps when the rewiring probability is very



small, it can not keep up with the Metropolis updates in terms of raw performance. We also show that for larger values of  $p$ , the avoidance of the critical slow down – usually one of the big advantages of the Wolff cluster algorithm over the localised nature of the Metropolis updates – loses significantly in relevance. The random shortcuts introduced to the lattice structure essentially have the same effect, as they enable long distance interactions between different parts of the system.

We have demonstrated that the interleaving of different update algorithms can offer improvements to the number of simulation updates needed to decorrelate the system configuration. But which algorithm or combination thereof gives the best performance is highly dependent on the specifics of the hardware architecture and system parameters used for the simulation. For the Fermi-architecture based GPUs used during our tests, the Metropolis algorithm always offers a better performance – in terms of the wall clock time to complete the simulation – over any hybrid combination with the Wolff algorithm.

Independent of the algorithmic decisions, it is worthwhile to invest some time into the optimisation of the different phases of the simulation. Any savings in the time required to generate independent system configurations add up particularly quickly with the large number of measurements typically needed for statistically reliable results.

With the parallel CUDA implementations of the small-world rewired lattice Metropolis and Wolff algorithms [15], [17] and based on the optimisations of the execution configurations for the different phases of the simulation described in this paper, it was possible to improve the estimate of the critical temperature for rewiring probabilities as small as  $p = 10^{-7}$  and system sizes of up to  $N = 512^3$  cells [24].

## References

- [1] E. Ising, "Beitrag zur Theorie des Ferromagnetismus," *Zeitschrift fuer Physik A Hadrons and Nuclei*, vol. 31, no. 1, pp. 253–258, 1925.
- [2] G. F. Newell and E. W. Montroll, "On The Theory Of The Ising Model Of Ferromagnetism," *Reviews of Modern Physics*, vol. 25, no. 2, pp. 353–389, 1953.
- [3] R. B. Potts, "Some Generalized Order-Disorder Transformations," in *Proceedings of the Cambridge Philosophical Society*, vol. 48, no. 1, 1952.
- [4] L. Onsager, "Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition," *Physical Review*, vol. 65, no. 3-4, pp. 117–149, 1944.
- [5] J. J. Binney, N. J. Dowrick, A. J. Fisher, and M. E. J. Newman, *The Theory of Critical Phenomena - An Introduction to the Renormalization Group*. Oxford University Press, 1992.
- [6] P. Svenson, "Damage spreading in small world Ising models," *Physical Review E*, vol. 65, no. 3, p. 036105, 2002.
- [7] S. Milgram, "The Small-World Problem," *Psychology Today*, vol. 1, pp. 61–67, 1967.
- [8] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, June 1998.
- [9] M. E. J. Newman, "Models of the Small World," *Journal of Statistical Physics*, vol. 101, no. 3-4, pp. 819–841, November 2000.
- [10] A. Barrat and M. Weigt, "On the properties of small-world network models," *The European Physical Journal B*, vol. 13, no. 3, pp. 547–560, 2000.
- [11] C. P. Herrero, "Ising model in small-world networks," *Physical Review E*, vol. 65, no. 6, p. 066110, 2002.
- [12] NVIDIA® Corporation, "The Compute Unified Device Architecture (CUDA)," <http://developer.nvidia.com/> (last accessed March 2012).
- [13] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.
- [14] W. K. Hastings, "Monte-Carlo Sampling Methods Using Markov Chains And Their Applications," *Biometrika*, vol. 57, no. 1, pp. 97–107, 1970.
- [15] K. A. Hawick, A. Leist, and D. P. Playne, "Regular Lattice and Small-World Spin Model Simulations using CUDA and GPUs," *International Journal of Parallel Programming*, vol. 39, no. 2, pp. 183–201, April 2011.
- [16] U. Wolff, "Comparison Between Cluster Monte Carlo Algorithms in the Ising Model," *Physics Letters B*, vol. 228, no. 3, pp. 379–382, 1989.
- [17] K. A. Hawick, A. Leist, and D. P. Playne, "Cluster and Fast-Update Simulations of Regular and Rewired Lattice Ising Models Using CUDA and Graphical Processing Units," Massey University, Tech. Rep. CSTN-104, 2010. [Online]. Available: <http://www.massey.ac.nz/~kahawick/cstn/>
- [18] A. M. Ferrenberg and D. P. Landau, "Critical behavior of the three-dimensional Ising model: A high-resolution Monte Carlo study," *Physical Review B*, vol. 44, no. 10, pp. 5081–5091, September 1991.
- [19] G. S. Pawley, R. H. Swendsen, D. J. Wallace, and K. G. Wilson, "Monte Carlo renormalization-group calculations of critical behavior in the simple-cubic Ising model," *Physical Review B*, vol. 29, no. 7, pp. 4030–4040, 1984.
- [20] C. F. Baillie, R. Gupta, K. A. Hawick, and G. S. Pawley, "Monte Carlo renormalization-group study of the three-dimensional Ising model," *Physical Review B*, vol. 45, no. 18, pp. 10 438–10 453, 1992.
- [21] K. Binder, "Finite Size Scaling Analysis Of Ising Model Block Distribution Functions," *Zeitschrift fuer Physik B*, vol. 43, no. 2, pp. 119–140, 1981.
- [22] K. Pearson, "Note on Regression and Inheritance in the Case of Two Parents," *Proceedings of the Royal Society*, vol. 58, pp. 240–242, 1895.
- [23] M. Barthélemy and L. A. N. Amaral, "Small-World Networks: Evidence for a Crossover Picture," *Physical Review Letters*, vol. 82, no. 15, pp. 3180–3183, 1999.
- [24] A. Leist, "Experiences in Data-Parallel Simulation and Analysis of Complex Systems with Irregular Graph Structures," Ph.D. dissertation, Massey University, Auckland, New Zealand, November 2011. [Online]. Available: <http://hdl.handle.net/10179/2992>



# A Hybrid MPI/OpenMP 3D FFT for Plane Wave First-principles Materials Science Codes

A. Canning<sup>1</sup>, J. Shalf<sup>1</sup>, N. J. Wright<sup>1</sup>, S. Anderson<sup>2</sup>, and M. Gajbe<sup>3</sup>

<sup>1</sup>Lawrence Berkeley National Laboratory, Berkeley CA, USA.

<sup>2</sup>Cray Inc., Saint Paul MN, USA.

<sup>3</sup>National Center for Supercomputing Applications, Urbana IL, USA.

**Abstract**—*First principles electronic structure calculations based on a plane wave expansion of the wavefunctions are the most commonly used approach for electronic structure calculations in materials and nanoscience. In this approach the electronic wavefunctions are expanded in Fourier components and 3D FFTs are used to construct the charge density in real space. Efficient parallel 3D FFTs are required for many other application codes such as in fluid mechanics, climate research accelerator design, etc. Due to the large amount of communications required in 3D parallel FFTs the scaling of these application codes on large parallel machines depends critically on having a 3D FFT that scales efficiently to large processor counts. With the recent increase in the number of cores per chip/node the simple model of running one MPI process per core on large node counts results in a large number of small messages causing contention in the network and latency issues. In this paper we show that a hybrid MPI/OpenMP implementation of our 3D FFT on the Cray XT5 can significantly outperform the pure MPI version, particularly on large processor counts, by sending fewer larger messages. Our Hybrid 3D FFT has been implemented in the electronic structure code PETot and allowed us to perform simulations of 4000 atom PbSe quantum rods on up to 21,600 cores on the Cray XT5.*

**Keywords:** materials science, parallel computing, 3D FFTs, MPI, OpenMP

## 1. Introduction

In electronic structure calculations in materials science first-principles methods based on Density Functional Theory (DFT) in the Kohn-Sham (KS) formalism [1] are the most widely used approach. In this approach the wave functions are usually expanded in plane waves (Fourier components) and pseudopotentials replace the nucleus and core electrons. This implementation requires parallel 3D FFTs to transform the electronic wavefunctions from Fourier space to real space to construct the charge density. This gives a computationally very efficient approach with a full quantum mechanical treatment for the valence electrons, allowing the study of systems containing hundreds of atoms on modest-sized parallel computers. Taken as a method DFT-based codes are one of the largest consumers of scientific computer

cycles around the world with theoretical chemists, biologists, experimentalists etc. now becoming users of this approach.

Parallel 3D FFTs are very demanding on the communication network of parallel computers as they require global transpositions of the FFT grid across the machine. The ratio of calculations to communications for 3D FFTs is of order  $\log N$  where  $N$  is the grid dimension (compared to a ratio of  $N$  for a distributed matrix multiply of matrix size  $N$ ) which makes it one of the most demanding algorithms to scale on a parallel machine. A scalable parallel 3D FFT is critical to the overall scaling of plane wave DFT codes. Efficient parallel 3D FFTs are also required for many other application codes such as in fluid mechanics, climate research accelerator design, etc.

In plane wave codes we have many electronic wavefunctions where each one is represented in Fourier space so unlike spectral type codes we are typically performing many moderate sized 3D FFTs rather than one large 3D FFT. This has the disadvantage from the scaling point of view that it is difficult to efficiently scale up a moderate sized 3D FFT on a large number of processors but it has the advantage that for all-band codes we can perform many 3D FFTs at the same time to aggregate the message sizes and reduce latency issues. The wavefunctions are also represented by a sphere of points in Fourier space and a standard grid in real space where the sphere typically has a diameter about half the size of the grid. This means we can also reduce the amount of message passing and calculations required compared to using a standard 3D FFT where the number of grid points is the same in Fourier and real space. We have therefore written our own specialized 3D FFTs for plane wave codes that can run faster than using public domain 3D FFT libraries such as FFTW or P3DFFT and have no restrictions on what grid sizes can be run. Our specialized 3D FFTs are used in many widely used materials science codes such as PARATEC, PETot, ESCAN [3].

Present supercomputers such as the Cray XT/XE range have increasingly more cores per chip and node so to scale to large node counts efficiently it is increasingly important to have efficient parallelism at the node as well as internode level. Each node on the Cray XT5 is constructed from two AMD Istanbul 6 core chips while the Cray XE6 has two AMD Magny-Cours 12 core chips per node. Future AMD

chips are projected to have more cores per chip and recent chips from IBM such as those used in the Blue Gene (BG) supercomputers have a similar number of cores. The present model for running many large first principles scale materials science codes on these machines has often been to just run an MPI process on each core. An alternative approach is to use OpenMP for the parallelism at the node/chip level and MPI between the nodes. This hybrid OpenMP/MPI approach has the advantage of using less memory per core and can result in fewer and larger messages compared to a pure MPI implementation. This is particularly important for 3D FFTs where the transpose of the grid results in all-to-all type communications between the MPI processes. This can result in limited scaling due to the large number of small messages in the network resulting in contention as well as latency issues. A pure MPI version also results in many wasteful memory copies on the node corresponding to the MPI communications between MPI processes mapped to the cores on a given node. A hybrid OpenMP/MPI code has the disadvantages of making the code more complex and introducing overheads due to creation and destruction of threads as well as possible load imbalance between threads.

Previous studies of hybrid OpenMP/MPI 3D FFTs have focused on small core counts where there was little or no improvement over a pure MPI version [4], [5]. In this work we show how a hybrid OpenMP/MPI can have improved performance over a pure MPI code for large core counts on moderate sized FFT grids.

## 2. Theoretical Background: Plane Wave First-principles Materials Science Codes

DFT using the Local Density Approximation (LDA) for the exchange-correlation potential requires that the wavefunctions of the electrons  $\{\psi_i\}$  satisfy the Kohn-Sham equations

$$\left[-\frac{1}{2}\nabla^2 + \sum_R v_{ion}(r-R) + \int \frac{\rho(r')}{|r-r'|} d^3r' + \mu_{xc}(\rho(r))\right]\psi_i = \varepsilon_i \psi_i \quad (1)$$

where  $v_{ion}(r)$  is the ionic pseudopotential,  $\rho(r)$  is the charge density and  $\mu_{xc}(\rho(r))$  is the LDA exchange-correlation potential. We use periodic boundary conditions, expanding the wavefunctions in plane waves (Fourier components),

$$\psi_{j,\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{g}} a_{j,\mathbf{k}}(\mathbf{g}) e^{i(\mathbf{g}+\mathbf{k})\cdot\mathbf{r}}. \quad (2)$$

The selection of the number of plane waves is determined by a cutoff  $E_{cut}$  in the plane-wave kinetic energy  $\frac{1}{2}|\mathbf{g} + \mathbf{k}|^2$  where  $\{\mathbf{g}\}$  are reciprocal lattice vectors. This means that the representation of the wavefunctions in Fourier space is a sphere or ellipsoid with each  $\mathbf{g}$  vector corresponding to a Fourier component (see Figure 1). The  $\mathbf{k}$ 's are vectors sampling the first Brillouin Zone (BZ) of the chosen unit cell (or supercell). The Kohn-Sham equations are usually solved

by minimizing the total energy with an iterative scheme, such as conjugate gradient (CG), for a fixed charge density and then updating the charge density until self-consistency is achieved (for a review of this approach see reference [2]). Some parts of the calculation are done in Fourier space and some in real space transforming between the two using 3D FFTs. In particular the charge density is constructed in real space by transforming each of the wavefunctions from Fourier space to real space then squaring and summing them.

## 3. Communication Structure and Parallel Data Decomposition for 3D FFTs

A 3D FFT consists of three sets of 1D FFTs in the x,y and z directions with transpositions of the data between each set of 1D FFTs. Only two transposes are needed if the final data layout is not required to have the same x,y,z order in both spaces as is the case for our application. Since the  $\mathbf{g}$  vectors (Fourier coefficients) are distributed across the nodes these two transposes can require global communications across the parallel computer.

As mentioned in the previous section the data for a given wavefunction forms a sphere of points in Fourier space and a standard grid in real space (see Figure 1). The data distribution for the sphere is driven by 1) the need to have complete columns of data on a given processor to perform the first set of 1D FFTs 2) other parts of the full materials science code require intensive calculations related to the number of Fourier components each processor holds so to load balance this part of the calculation we require a similar number of Fourier components on each processor. The data layout we use in Fourier space is to order the columns of the sphere in descending order and then to give out the individual columns to the processors such that each new column is given to the processor with the fewest number of Fourier components. In this way each processor holds sets of complete columns and approximately the same number of Fourier components (see Figure 1 for an example of the layout on three processors). In real space we consider the grid as a one dimensional set of columns with (x,y,z) ordering and then give out contiguous sets of columns to each processor giving as closely as possible the same number of columns to each processor. In this way each processor will hold complete planes or sections of planes of the three dimensional grid (see Figure 1). With this data layout we have no restrictions on the number of processors required for a given sphere or grid size. Also this data layout means that the first transpose in the 3D FFT typically requires all processors communicating with every other processor while the second transpose may require no communications (if the each processor has complete planes) or limited local communications if each processor has a section of a plane. In the case of SMP nodes complete planes can still reside on a node even if each processor is performing calculations

on sections of a plane. A more detailed description of each step in our specialized 3D FFT can be found in reference [7].

In order to have a scalable parallel 3D FFT we therefore need to have as efficient as possible an implementation of the communications in the two parallel transposes. As mentioned above it is the first transpose that typically involves all the processors communicating with each other while the second transpose will typically involve limited local communications. In our original pure MPI implementation we found that using `MPI_ALLTOALLV` gave the best performance and scaling on Cray XT and IBM BG platforms (see reference [8] for details of the performance of the pure MPI 3D FFT code). In the pure MPI version we also have implemented what we will refer to as blocked versions of the 3D FFTs where we perform a number of 3D FFTs at the same time (typically 40) and so can aggregate the message sizes to reduce latency problems. In our particular application we are performing a large number of moderate sized 3D FFTs (our tests will be for  $144^3$  grids) so it is important to take advantage of this blocking to reduce latency issues on large processor counts.

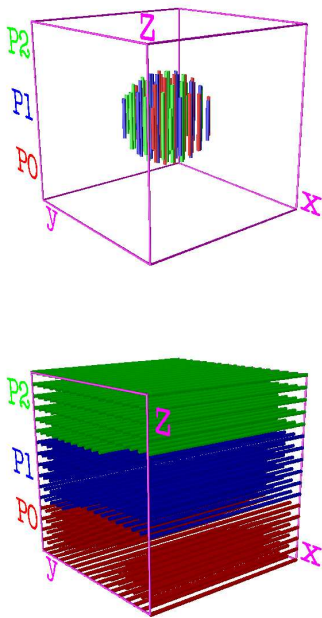


Fig. 1: A three processor example of the parallel data layout for the wavefunctions of each electron in Fourier space (left) and real space (right). The different colors correspond to the data held by processors P0, P1 and P2.

## 4. Hybrid OpenMP/MPI Implementation

In the pure MPI parallel 3D FFT there are basically three types of code section. The first is the three sets of one dimensional FFTs for the x,y and z direction FFTs. The second is the local gather and scatter type operations before and after the communications. Each MPI process gathers the data together into an array for sending to each of the other MPI processes. The scatter type operation then unpacks the received data to form contiguous columns before the next set of one dimensional FFTs are performed. The third type of code section is the actual communications which are performed using an `MPI_ALLTOALLV`. The gather and scatter operations combined with the communications are performing the parallel transposes. In the pure MPI code the parallelism over the the one dimensional FFTs is very efficient as these are completely independent and there is a large number of one dimensional FFTs to load balance over the MPI processes. In a standard  $N^3$  grid there are  $N^2$  one dimensional FFTs to be performed in each dimension although for our specialized FFTs there are fewer than this for the first two dimensions. In the pure MPI code the gather and scatter operations are local to the MPI process and load balance well since each MPI process is doing similar operations. It is the all-to-all communication step in the first transpose that greatly limits the scaling of 3D FFTs on parallel computers. In this step each MPI process will send a message of approximately the same size to all the other MPI processes. If we have  $n_{proc}$  MPI processes then we will have a total of  $n_{proc}^2$  messages of size  $N^3/n_{proc}^2$ . Therefore if we scale to large values of  $n_{proc}$  we will have a very large number of small messages being sent through the network. By using our blocked 3D FFTs we can increase the size of the messages to reduce latency issues. It should also be noted that since we are going from a sphere in Fourier space to a cube in real space the forward and reverse FFTs are different so we have different routines for the forward and reverse 3D FFTs.

In the OpenMP version of the code we basically need to introduce another level of parallelism in the code below the MPI parallelism to divide the work for each MPI process among the OpenMP threads. In the case of the three sets of one dimensional FFTs this is straightforward as for each MPI process we still have a fairly large number of one dimensional FFTs to divide among the OpenMP threads. In the case of the gather/scatter routines it was necessary to restructure the loops associated with each MPI process to exploit the parallelism. The gather/scatter operations to pack and unpack the arrays used for the communications are always operating on independent elements so this work can be divided among the different OpenMP threads and load balanced reasonably well by allowing each thread to deal with roughly the same number of elements. In the MPI

communication steps we only perform the communications using one of the OpenMP threads. The basic structure of the hybrid version of the 3D FFT code is that we create  $n$  threads (which we will denote as  $n_{thr}$ ) at the start of the routine which are then used for the parallelization of the one dimensional FFTs and gather/scatter operations and the MPI communications are then done in a master region. In this way we avoid the cost of creation and destruction of threads in the subroutine. The threads are only destroyed at the end of the subroutine. If we now consider the difference between running the pure MPI and the hybrid MPI/OpenMP code on the same number of cores and nodes then the hybrid code will aggregate the messages associated with all the threads per MPI process. So the hybrid code running on the same number of cores will send a factor of  $n_{thr}^2$  fewer messages that are  $n_{thr}^2$  larger in size. This can greatly reduce latency issues as well as contention in the communication network.

## 5. Results on the Cray XT5

Our simulations were performed on the Cray XT5 machine Jaguar at the Oak Ridge Leadership Computing Facility (OLCF). Each node of this machine consists of two AMD Istanbul 2.6 GHz six core chips connected by a Seastar2 interconnect. The latency of the network is  $7 \mu s$  with an internode bandwidth of 1.6 GBytes/s. There are 1.3 Gbytes of memory per core and the complete machine has 224,256 cores (18,688 nodes) with a peak speed of 2.33 Petaflops.

The size of grid for our tests was  $144^3$  which is the size required for the simulation of about 500 silicon atoms in the crystal structure and is typical of the grid size required for a moderate sized simulation for these types of first principles electronic structure codes. A larger grid would give better scaling to larger core counts but the purpose of our simulations is to show that for a typical sized simulation the hybrid code can give significant improvements over the pure MPI code.

The data in Table 1 and Figure 2 are for runs on 288 cores (24 nodes) with 1 to 12 OpenMP threads per node. The nodes are always run fully packed, using all the cores per node. The results are for four hundred 3D FFTs to average over fluctuations and the blocking factor used for the 3D FFTs is 40. Table 2 and Figure 3 are the same runs but for 576 cores (48 nodes). We can see from Tables 1 and 2 that as we increase the number of threads the code runs faster with the fastest speed being for 6 threads for the runs on 576 cores. The speed drops slightly for 12 threads which we believe is due to NUMA effects on the node as each chip has six cores and we cannot pin the local memory to the thread when we have more than 6 threads per node. Looking at Figure 3 we can see that all the gain in speed of the hybrid code over the pure MPI code (that corresponds to running with one thread) is in the all-to-all communication steps ( $t_{1,comm}$  and  $t_{5,comm}$ ). The timings for 1 to 6 threads for the other parts of the code are similar. As previously

mentioned this is a result of the reduction in the number of messages by a factor of  $n_{thr}^2$  that are now  $n_{thr}^2$  larger. The results on 288 cores for Figure 2 are similar although the gain in performance is not as large as on the larger core counts and the fastest run is with 3 threads although the time for 6 threads is similar. It should also be noted that running with 6 threads the speedup in going from 288 to 576 cores is almost a factor of two with the hybrid code but running on 576 core the pure MPI version was actually slower than on 288 cores.

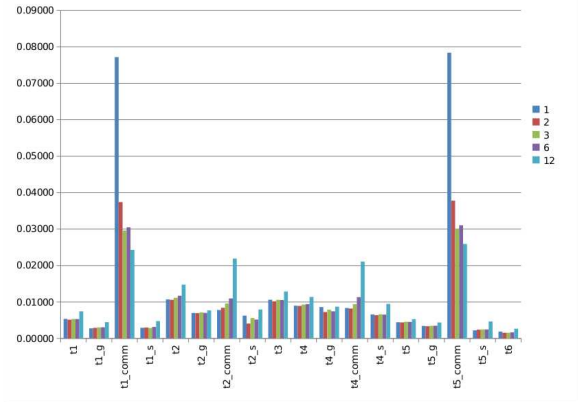


Fig. 2: Timings for 400 3D FFTs (grid size  $144^3$ ) on 288 cores and different OpenMP thread counts for the different sections of the code. The nodes are always run fully packed ie. using all the cores. These results are for both the forward and reverse 3D FFTs. The  $t$  is for the 1D FFTs,  $t_g$  and  $t_s$  are for the gather scatter operations and  $t_{comm}$  is for the communication steps.  $t1$  to  $t3$  are the steps for the forward 3D FFT and  $t3$  to  $t6$  are for the reverse 3D FFT. There are only two communication steps for each of the forward and reverse 3D FFTs

mpi procs.	Threads	Time 3D FFTs
288	1	0.253
144	2	0.169
96	3	0.160
48	6	0.165
24	12	0.199

Table 1: Timings for 400 3D FFTs (grid size  $144^3$ ) on 288 cores and different OpenMP thread counts. Mpi procs is the number of mpi processes and Threads is the number of OpenMP threads. This data corresponds to the same run as Figure 2.

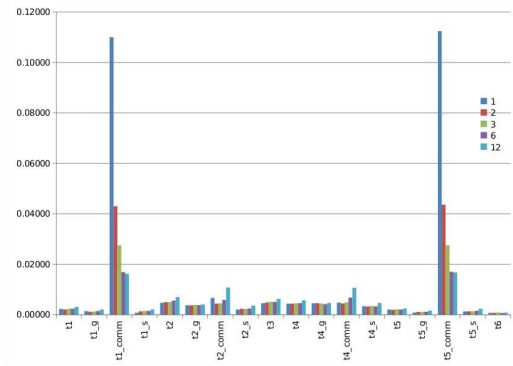


Fig. 3: Timings for 400 3D FFTs (grid size  $144^3$ ) on 576 cores and different OpenMP thread counts for the different sections of the code. The nodes are always run fully packed i.e. using all the cores. These results are for both the forward and reverse 3D FFTs. The  $t$  is for the 1D FFTs,  $t_g$  and  $t_s$  are for the gather scatter operations and  $t_{comm}$  is for the communication steps.  $t_1$  to  $t_3$  are the steps for the forward 3D FFT and  $t_3$  to  $t_6$  are for the reverse 3D FFT. There are only two communication steps for each of the forward and reverse 3D FFTs

mpi procs.	Threads	Time 3D FFTs
576	1	0.271
288	2	0.134
192	3	0.104
96	6	0.087
48	12	0.106

Table 2: Timings for 400 3D FFTs (grid size  $144^3$ ) on 576 cores and different OpenMP thread counts. Mpi procs is the number of mpi processes and Threads is the number of OpenMP threads. This data corresponds to the same run as Figure 3.

## 6. Conclusion and Future Work

The basic conclusion of our studies is that by introducing another level of parallelism in the code at the node level with OpenMP our hybrid MPI/OpenMP 3D FFTs can scale efficiently to larger core counts than the pure MPI version. This gain comes from the large reduction in the number of messages as well as their increase in size, reducing latency effects. For the final version of the paper we plan to have data on more grid sizes as well as on other multicore platforms.

We implemented our hybrid 3D FFT in the electronic structure code PEtot which allowed us to perform simulations of 4000 atom PbSe quantum rods on up to 21,600 cores on the Cray XT5. The full code has other higher

levels of parallelism such as over states and k-points which allows us to scale to much larger numbers of cores than the scaling of the 3D FFTs. The new hybrid 3D FFTs allow us to have another level of on-node parallelism at the lowest level. Threading at the node level also allows us to reduce the memory footprint of the full code. The performance of the full code will be the subject of a future publication.

## Acknowledgment

The authors would like to thank Lin-Wang Wang and Harvey Wasserman for useful discussions. This work was carried out at the Lawrence Berkeley National Laboratory under U.S. Department of Energy Contract No. DE-AC02-05CH11231. This work used resources at the Oak Ridge Leadership Computing Facility (OLCF) which is supported by the Office of Science of the U.S. Department of Energy.

## References

- [1] W. Kohn and L.J. Sham, Phys. Rev. **140**, A1133 (1965).
- [2] M. Payne, M.P. Teter, D.C. Allan, T.A. Arias and J.D. Joannopoulos, Rev. Mod. Phys. **64**, 1045 (1992).
- [3] PARATEC (PARAllel Total Energy Code) [www.nersc.gov/projects/paratec/](http://www.nersc.gov/projects/paratec/) by B. Pfrommer, D. Raczowski, A. Canning, S.G. Louie, Lawrence Berkeley National Laboratory (with contributions from F. Mauri, M. Côté, Y. Yoon, C. Pickard and P. Haynes). PEtot code <https://hpcrd.lbl.gov/linwang/PEtot/PEtot.html>
- [4] D. Takahashi, PPAM 2005, R. Wyrzkowski et al. (Eds), p 970, Publisher: Springer-Verlag (2006)
- [5] G. Luecke, O. Weiss, M. Kraeva, J. Coyle and J. Hoekstra, Proceedings of the Cray User Group Conference (2010).
- [6] Matteo Frigo and Steven G. Johnson, Proceedings of the IEEE 93 (2), 216-231 (2005). <http://www.fftw.org/>
- [7] A. Canning, L.W. Wang, A. Williamson and A. Zunger, J. of Comput. Phys. **160**, 29 (2000).
- [8] A. Canning, J. Shalf, H Wassermann and M. Gajbe, Parallel Computing: From Multicores and GPUs to Petascale, IOS Press, B. Chapman (Eds.) (2010).
- [9] A. Grama, A. Gupta, G. Karypis and V. Kumar, Introduction to Parallel Computing, Addison-Wesley, second edition, (2003).
- [10] [www.sdsc.edu/us/resources/p3dfft.php](http://www.sdsc.edu/us/resources/p3dfft.php)
- [11] A. Canning, High Performance Computing for Computational Science, Springer, J.M.L.M Palma et. al. (Eds.) proceedings of VECPAR 2008, p280 (2008).

# ***Involving Kalman filter technique for increasing the reliability and efficiency of cloud computing***

Mehdi Darbandi, Mohammad Abedi, Saeed Fard, Sanaz Nakhodchi

Department of Electrical Engineering and Computer Science at Iran University of Science and Technology (IUST); Tehran, Iran

*Abstract— Cloud is a virtual image about some amount of undefined powers, that is widespread and had unknown power and inexact amount of hardware and software configurations, and because of we haven't any information about clouds location and time dimensions and also the amounts of its sources we tell that Cloud Computing [1]. This technology presents lots of abilities and opportunities such as processing power, storage and accessing it from everywhere, supporting, working – team group - with the latest versions of software and etc., by the means of internet. Cloud computing is reliable services that presenting by next generation data centers and from internet, they based on virtual technologies and computing methods [4]. In recent decades the internet has very deep influences in human lives and also in offer and demand markets; lots of people for trading, reading the news, seeing the movies or to play online games go through the internet. As our daily needs from the internet became more, so that the processing power and amount of storage data and files should increase significantly [8]. With remarkable developments of communication and information technology in recent days, we consider processing and processing power as the fifth vital component in human lives (after water, electricity, gas and telephone) [9]. These days' lots of technologies migrate from traditional systems into cloud and similar technologies; also we should note that cloud can be used for military and civilian purposes [3]. On the other hand, in such a large scale networks we should consider the reliability and powerfulness of such networks in facing with events such as high amount of users that may login to their profiles simultaneously, or for example if we have the ability to predict about what times that we would have the most crowd in network, or even users prefer to use which part of the Cloud Computing more than other parts – which software or hardware configuration. With knowing such information, we can avoid accidental crashing or hanging of the network that may be cause by logging of too much users. In this paper we propose Kalman Filter that can be used for estimating the amounts of users and software's that run on cloud computing or other similar platforms at a certain time. After introducing this filter, at the end of paper, we talk about some potentials of this filter in cloud computing platform. In this paper we demonstrate about how we can use Kalman filter in estimating and predicting of our target, by the means of several examples on Kalman filter.*

*Keywords-* Cloud computing and its influences, Security, Kalman Filter, Estimation and prediction.

## I. INTRODUCTION

History of computer science has had fundamental changes. For example in first generation of computers (from 1945 until 1956), vacuum tube was used in computers. In 2<sup>nd</sup> generation (from 1956 until 1963) by the invention of transistors, these tiny devices were used in computers. After that, integrated circuits were used in 3<sup>rd</sup> generation of computers (1963 to 1971). Finally in 4<sup>th</sup> generation (since 1971 up to now), along with technology advancement, Large Scale Integration (LSI) circuits, Very Large Scale Integration (VLSI) and Ultra Large Scale Integration (ULSI) were used in computers. Nowadays new technology

that is named as Cloud Computing is creating a new era in computer industry and processing power. By reviewing the historical points, we can understand that the idea of Cloud Computing taken from this fact: When the current user or users don't require the processing resources, these resources can be assigned to other users. Most simple definition of Cloud Computing is: Access to enormous resources and processing powers even, through cheap computers.

One of the major benefits of this technology is sharing resources – software and hardware resources - and the ability to simultaneously working on specific projects or files. For example you want to do some processing on DNA of person(s). If you are user of cloud computing, at first you can use processing power of cloud computing that is more reliable and efficient, because your system may not have enough processing power. Secondly, you can do the above operations with many experts – sharing your task - simultaneously and find instant results of each other. Also, before this technology revealed, you were restricted to use only processing power of your own system. Also, regularly you must update the hardware and software of your system, in order to be able to access the latest software version and also have required hardware to support this software, that's very costly. Also if you want to install this software on more computers; you need to purchase more licenses of that software.

We don't have such problems in Cloud Computing, because every time you log out from your account and logging to it next time, you'll see the latest update of the software, without need to developing the hardware of your system, because the required hardware is provided by cloud resources. Also you don't pay additional costs for this software developing – buying licenses or pay for updating the software.

In ordinary systems, if you have high sensitive and important information on your computers, you must always update antivirus and firewall of your own system, to avoid from data loss or manipulation. If you forget doing this action (updating), it is possible to accidentally your system be attacked and your information became destroyed or manipulated. But in Cloud Computing, when you login to your account, you will see security system is up-to-date with the latest version.

Another benefit of cloud computing is, supporting (recognizing) all different formats of the files. For example, in ordinary systems, you may download a file – from your e-mail or from a website - that has unknown format for your machine and you don't know which software can open it; but in Cloud Computing this file will be open with associated and applicant software –automatic recognition of that file type and devoting it the exact software for running is occurred in cloud computing.



Moreover, for instance if you have a company, you can transfer internal network of your company –your server database - on Cloud Computing to enjoy more speed and processing power, and also if you use server, you will economize in budget and only pay power consumption and maintenance costs.

These are just part of the great performance of new technology, known as Cloud computing that is named also as "the next big thing" [1-9].

## II. CONSIDERING HIGH IMPACTS OF CLOUD COMPUTING ON DIFFERENT INDUSTRIES

In two past sections of the paper, we define some of the basic and fundamental principles of cloud and also we tell about some of its advantageous. Now we want imply into, the major applications of this technology. After that when we understand the importance of this technology, we introduce Kalman Filter; which can be used for prediction and estimation of different parameters in cloud platform.

The main reason for using cloud computing is, humans in the current era wants to use wireless and high speed communications [6]. The first commercially provider of cloud computing was Amazon. Another reason for this amount of usage is that, we prefer not to know about the volume of processors, hardware and servers and only wants to face with very gigantic ability in storage and processing capability [9]. The third reason for using of cloud computing instead of traditional systems is that, these days people want to use social sites, and these sites should have enough capability to support the users; we shouldn't have any crash or low speed when we use them. For example Facebook has more than 850 million members these days and the Facebook Team decides recently that it's better to migrate to Cloud Computing platform, because it had more storage and processing power [14]. Another reason is that user(s) can decide with whom and with which places in all over the world, they wants to share his/her information and files; we don't have such abilities in traditional systems, also in traditional systems we have the restriction on the volume of data that's stored [11]. Another reason is that, a programmer doesn't need to program the servers and computers for particular users or organizations, but every people can use cloud computing with his/her special uses that wants. Furthermore, it doesn't need any specialty in using, it doesn't have any software and hardware requirement, and in cloud computing all kind of files were recognizable – automatic format recognition [9].

Google Company wants to publish Chrome operating system, and with that it wants to visit their users in cloud. They believe that by employing that technology they can better satisfy their users. Furthermore, Google recently has published the new version of Google Docs, which is very powerful software package for cloud users. Also Microsoft Company decides to introduce Microsoft Azure – that's an OS which is based on cloud computing, because they fear about losing their users. Also General Service Administration (GSA); because of having so many visitors that visit their sites every day, they fearing about crashing or hanging of it, as a result they decide to migrate to cloud computing. National Aeronautics and Space Administration

(NASA) builds NEBULA, which is cloud computing platform and with using of that platform, people can participate in NASA missions and tell their ideas and even their suggestions; moreover NASA search for better storage and processing ability from this technology. The NASA Headquarter has publishes on their recent report, that they decide to implement International Space Station platform on Cloud Computing. Also Department of Interior, which is provider of lots of services for organizations decide to present some services by cloud computing, because they think it's more convenient. Department of Health and Human Services wants to employ new platform which is based on cloud computing to be able to give better, faster and more efficient services. Census Bur, which is the provider and supporter of SaaS services in Salesforce site - that is giving services to millions of people every day - decide to present new products based on Cloud platform. Also The White House decide to migrate to cloud computing technology because they think in that way they can do their tasks such as e-voting, having conversation with their citizens, and their internal networks; more easily and faster [1, 5,15].

Also United Kingdom Governments run G-Cloud networks for themselves, in order to have more precise in their works. Europeans use cloud computing in public and private sections such as: Management of public sector housing, transportation service networks, Census, Economic development, Health services, and Contracting and education services [1].

Also in Denmark with two pilot projects, Digitalise'r.dk and NemHandel, they evaluate cloud computing for their users and after that when they became satisfy about this technology they decide to establish new platform for their governments based on cloud computing. One of the pioneers in this technology are Japanese with their cloud, that known as Kasumigaseki Cloud and built in government-industry area of Tokyo, they wants to improve cloud for their public uses, also they named this technology and area that implement it as a pilot project as "green environment". In China, especially in north of this country the government with the project that named as: "The Yellow River Delta Cloud Computing Center", wants to establish governmental based services on this technology. Even in Wuxi city, the government established a company for manufacturing resources based on cloud computing technology. In Thailand, The Government Information Technology Services (GITS), design and construct private cloud for public sectors; and they wants to, as soon as possible present so many services for civilians and private organizations by the means of cloud computing. In Vietnam, IBM Company with their government and universities wants to develop the new laws, for presenting cloud for public and private sections. In New Zealand, they search for the better and more efficient uses and potentials of cloud computing [1, 7].

Now we want to consider the impact of this technology on minor companies and organizations. YouTube Company, which is in 2006, has a daily increasing about 30 million in their web-pages, nowadays decide to use cloud computing technology for better and faster searching and hosting, because as they told in their report we have more intelligent algorithms in cloud platform. Although if such company doesn't used cloud computing, they should pay lots of money

for maintaining and operating of their servers, also lots of money for upgrading their software and hardware configurations [13].



Fig. 1: Every People Can Share Their Tasks and Projects with Others, They Can Get Helps and New Ideas.

SmugMug Company which is a site for sharing images; for better and faster services and also because of they doesn't have enough storage space decide to migrate to cloud computing. Or as an example Google Company prepares services such as Google App Engine which promise you to run your applications on Google cloud resources. As such, you can share your files and processes with anyone you want or even you can share your files with all the peoples around the world. Also Google and IBM decide to construct and run new networks for universities based on cloud computing; so that universities can do their researches with help of each other simultaneously, even faster and with more precise; because the students and professors from other universities can take part in other research topics and tell their ideas. The universities which are used this technology are: University of Washington, Carnegie-Mellon University, MIT, Stanford University, The University of California at Berkeley, Maryland University [7].

Nasdaq Company, which is had lots of data about stock and funds, wants to share and sell their information; but they are anxious about using of servers and their storage capacity; thus they decide to use Amazon S3 service, which is cloud based platform. Recently, minor companies such as Nimbus and Eucalyptus present storage and processing powers by getting fees [1, 13].

Sun Micro Systems Company now involved in constructing new data centers for hosting applications and users of cloud computing. This company wants to establish many sites all over the world to hosting user's application and avoiding from failure. Also these days science networks such as My experiment and nanoHub migrates to cloud computing, because they wants to had better and more efficient and convenient communications with their fans. Also cloud computing can present banking services, it means you take some services by paying their fees and after that you improve that service or add some features to that service and sell it to other users. Also, iPhone company build their new Mobile Phone based on Cloud computing platform, the providers of this product think that with using this

technology they can implement chipper Microcontrollers on the board of this device, but instead they can gain more processing power, only because they use cloud computing technology. Finally, writers of this paper predict in near future lots of people involved in this technology. Engineers and designers must play their role more accurately to get the interest of more users and provide them their needs [1 to 5].

In the next section we discuss about basic concepts of Kalman Filter and introduce it briefly. We can use this algorithm for estimation and prediction the amount of users that logging into their profiles at certain time; furthermore we can use it for security purposes.

### III. AN INTRODUCTION INTO KALMAN FILTER

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem. Since that time, due in large part to advances in digital computing; the Kalman filter has been the subject of extensive research and application, particularly

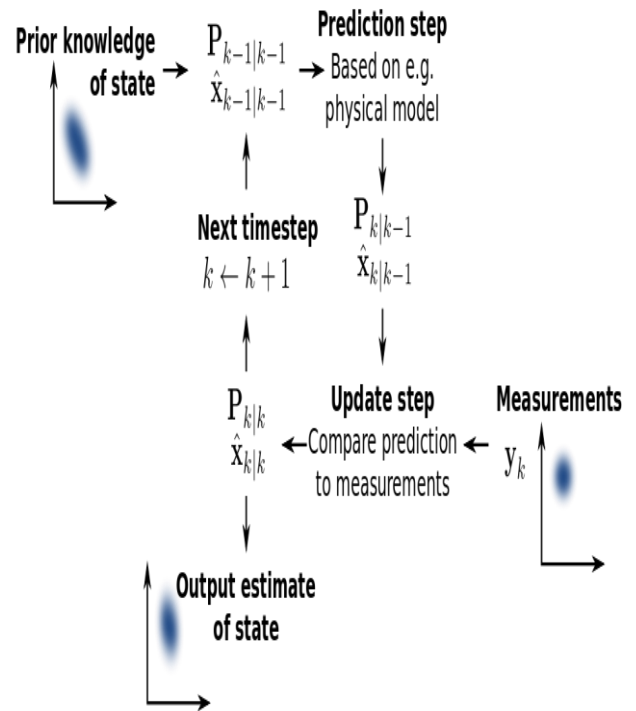


Fig. 2: The Kalman filter keeps track of the estimated state of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements.  $\hat{x}_{k|k-1}$  Denotes the estimate of the system's state at time step  $k$  before the  $k$ -th measurement  $y_k$  has been taken into account;  $P_{k|k-1}$  is the corresponding uncertainty.

in the area of autonomous or assisted navigation. The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown. The Kalman filter has numerous applications in technology.



A simple and ubiquitous example is the phase-locked loop, used in FM radios and most electronic communications equipment. Another common application is for guidance, navigation and control of vehicles, particularly aircraft and spacecraft.

The Kalman filter uses a system's dynamics model (i.e., physical laws of motion), known control inputs to that system, and measurements (such as from sensors) to form an estimate of the system's varying quantities (its state) that is better than the estimate obtained by using any one measurement alone. As such, it is a common sensor fusion algorithm.

All measurements and calculations based on models are estimates to some degree. Noisy sensor data, approximations in the equations that describe how a system changes, and external factors that are not accounted for introduce some uncertainty about the inferred values for a system's state. The Kalman filter averages a prediction of a system's state with a new measurement using a weighted average. The purpose of the weights is that values with better (i.e., smaller) estimated uncertainty is "trusted" more. The weights are calculated from the covariance, a measure of the estimated uncertainty of the prediction of the system's state. The result of the weighted average is a new state estimate that lies in between the predicted and measured state, and has a better estimated uncertainty than either alone. This process is repeated every time step, with the new estimate and its covariance informing the prediction used in the following iteration. This means that the Kalman filter works recursively and requires only the last "best guess" - not the entire history - of a system's state to calculate a new state.

When performing the actual calculations for the filter (as discussed below), the state estimate and covariance's are coded into matrices to handle the multiple dimensions involved in a single set of calculations. This allows for representation of linear relationships between different state variables (such as position, velocity, and acceleration) in any of the transition models or covariances.

The Kalman filter is used in sensor fusion and data fusion. Typically, real-time systems produce multiple sequential measurements rather than making a single measurement to obtain the state of the system. These multiple measurements are then combined mathematically to generate the system's state at that time instant.

As an example application, consider the problem of determining the precise location of a truck. The truck can be equipped with a GPS unit that provides an estimate of the position within a few meters. The GPS estimate is likely to be noisy; readings 'jump around' rapidly, though always remaining within a few meters of the real position. The truck's position can also be estimated by integrating its speed and direction over time, determined by keeping track of wheel revolutions and the angle of the steering wheel. This is a technique known as dead reckoning. Typically, dead reckoning will provide a very smooth estimate of the truck's position, but it will drift over time as small errors accumulate. Additionally, the truck is expected to follow the laws of physics, so its position should be expected to change proportionally to its velocity.

In this example, the Kalman filter can be thought of as operating in two distinct phases: predict and update. In the prediction phase, the truck's old position will be modified according to the physical laws of motion (the dynamic or

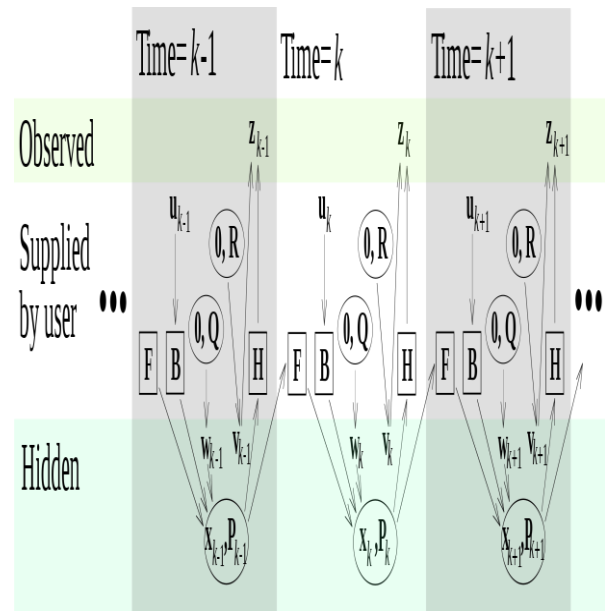


Fig. 3: Model underlying the Kalman filter. Squares represent matrices. Ellipses represent multivariate normal distributions (with the mean and covariance matrix enclosed). Unenclosed values are vectors. In the simple case, the various matrices are constant with time, and thus the subscripts are dropped, but the Kalman filter allows any of them to change each time step.

"state transition" model) plus any changes produced by the accelerator pedal and steering wheel. Not only will a new position estimate be calculated, but a new covariance will be calculated as well. Perhaps the covariance is proportional to the speed of the truck because we are more uncertain about the accuracy of the dead reckoning estimate at high speeds but very certain about the position when moving slowly. Next, in the update phase, a measurement of the truck's position is taken from the GPS unit. Along with this measurement come some amount of uncertainty, and its covariance relative to that of the prediction from the previous phase determines how much the new measurement will affect the updated prediction. Ideally, if the dead reckoning estimates tend to drift away from the real position, the GPS measurement should pull the position estimate back towards the real position but not disturb it to the point of becoming rapidly changing and noisy. As another example application consider the use of Kalman filter in computer vision, Data fusion using a Kalman filter can assist computers to track objects in videos with low latency (not to be confused with a low number of latent variables). The tracking of objects is a dynamic problem, using data from sensor and camera images that always suffer from noise. This can sometimes be reduced by using higher quality cameras and sensors but can never be eliminated, so it is often desirable to use a noise reduction method. The iterative predictor-corrector nature of the Kalman filter can be helpful, because at each time instance only one constraint on the state variable need be considered. This process is repeated, considering a different constraint at every time instance. All the measured data are accumulated over time

and help in predicting the state. Video can also be pre-processed, perhaps using a segmentation technique, to reduce the computation and hence latency.

#### IV. CONCLUSION

In this article, we tell about basic definitions and concepts of cloud computing. Also we tell about uses of this technology in different societies and industries and also about how they use it and about their future plans. An important factor in using cloud computing and migrating into this network is its security and durability. In this paper the authors propose Kalman filtering for increasing security and durability of such networks for the first time. If we implement this algorithm on such networks – for example on the edge of such networks, we can estimate and predict the amount of users that use the resources – software and hardware resources – at anytime. Also we can estimate and predict the amount of users that logging onto a certain account, and by the means of that we can avoid surveillance entering of bad users – we estimate the location of user by the previous location and its background data. Also we can use it for estimating the amount of user that use a certain application on such networks, and by knowing that amount we can improve power of our network to be able to support our users. Furthermore by using this algorithm we can increase the security of this technology, by estimating and predicting the point of presence of bad users. In this paper we demonstrate about how can we use Kalman filter in estimating and predicting of our target, by the means of several examples on Kalman filter.

#### REFERENCES

- [1] David C. Wyld; "the cloudy future of government IT: cloud computing and the public sector around the world", *IJWesT*, Vol. 1, Num. 1, Jan. 2010.
- [2] Jean-Daniel Cryans, Alain April, Alain Abran; "criteria to compare cloud computing with current database technology", R. Dumke et al. (Eds.): *IWSM / MetriKon / Mensura 2008*, LNCS 5338, pp. 114-126, 2008.
- [3] Anil Madhavapeddy, Richard Mortier, Jon Crowcroft, Steven Hand; "multiscale not multicore: efficient heterogeneous cloud computing", published by the British Informatics Society Ltd. *Proceedings of ACM-BCS Visions of Computer Science 2010*.
- [4] Harold C. Lim, Shivnath Babu, Jeffrey S. Chase, Sujay S. Parekh; "automated control in cloud computing: challenges and opportunities", *ACDC'09*, June 19, Barcelona, Spain.
- [5] N. Sainath, S. Muralikrishna, P.V.S. Srinivas; "a framework of cloud computing in the real world"; *Advances in Computational Sciences and Technology*, ISSN 0973-6107, Vol. 3, Num. 2, (2010), pp. 175-190.
- [6] Kyle Chard, Simon Caton, Omer Rana, Kris Bubendorfer; "social cloud: cloud computing in social networks"
- [7] G. Bruce Berriman, Eva Deelman, Paul Groth, Gideon Juve; "the application of cloud computing to the creation of image mosaics and management of their provenance"
- [8] Roy Campbell, Indranil Gupta, Michael Heath, Steven Y. Ko, Michael Kozuch, Marcel Kunze, Thomas Kwan, Kevin Lai, Hing Yan Lee, Martha Lyons, Dejan Milojicic, David O'Hallaron, Yeng Chai Soh; "open cirrus™ cloud computing testbed: federated data centers for open source systems and services research"
- [9] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic; "cloud computing and Emerging IT platforms: Vision, Hype, and Reality for delivering computing as the 5<sup>th</sup> utility"

- [10] Lamia Youseff, Maria Butrico, Dilma Da Silva; "toward a unified ontology of cloud computing"
- [11] Daniel A. Menasce, Paul Ngo; "understanding cloud computing: experimentation and capacity planning"; *Proc. 2009, Computer Measurement Group Conf. Dallas, TX. Dec. 2009*.
- [12] Won Kim; "cloud computing: today and tomorrow"; *JOT*, Vol. 8, No. 1, Jan-Feb 2009.
- [13] Richard Chow, philippe Golle, Markus Jakobsson, Elaine Shi, Jessica Staddon, Ryusuke Masuoka, Jesus Molina; "controlling data in the cloud: outsourcing computation without outsourcing control"; *CCSW'09*, Nov. 13, 2009, Chicago, Illinois, USA.
- [14] Bo Peng, Bin Cui, Xiaoming Li; "implementation issues of a cloud computing platform"; *Bulletin of the IEEE computer society technical committee on data engineering*.
- [15] Daniel Nurmii, Rich Wolski, Chris Grzegorzczak, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov; "the eucalyptus open-source cloud computing system"

#### BIOGRAPHIES



**Mehdi Darbandi** received his B.Sc. degree in Electrical Engineering from University of Mashhad in 2012. His research areas are Kalman Filter, Matlab Simulink, Evolutionary Algorithms, and Cloud Computing. Now he is master student in Iran University of Science and Technology (IUST); Tehran, Iran. His e-mail address is: [mahdidarbandi@hotmail.com](mailto:mahdidarbandi@hotmail.com)



**Mohammad Abedi** received his B.Sc. degree in Computer Engineering (with Software tendency) from University of Mashhad in 2005. His research areas are Neural Network, Design and Implementation of Distributed networks. Now he is master student in Iran University of Science and Technology (IUST); Tehran, Iran. His e-mail address is: [m.abedi\\_ict@yahoo.com](mailto:m.abedi_ict@yahoo.com)



**Saeed Fard** received his B.Sc. degree in Computer Engineering (with Software tendency) from University of Mashhad in 2005. His research areas are Network Analysis and Computer Architecture. His e-mail address is: [ttm.Saeed.@gmail.com](mailto:ttm.Saeed.@gmail.com)



**Sanaz Nakhodchi** received her B.Sc. degree in Information Technology from University of Mashhad in 2010. Her interest areas are Artificial Intelligence, Soft Computing and Game Theory. Now she is master student in Iran University of Science and Technology (IUST); Tehran, Iran. Her e-mail address is: [nakhodchi@vu.iust.ac.ir](mailto:nakhodchi@vu.iust.ac.ir)

**SESSION**  
**NOVEL APPLICATIONS AND ALGORITHMS**

**Chair(s)**

**TBA**



# Inclined flow of a heated fluid film with temperature dependent fluid properties

N. Gonputh<sup>1</sup>, J.P. Pascal<sup>1</sup> and S.J.D. D'Alessio<sup>2</sup>

<sup>1</sup>Department of Mathematics, Ryerson University, Toronto, Ontario, M5B 2K3, Canada

<sup>2</sup>Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada

**Abstract-***The gravity driven film flow down a heated inclined ramp and how it is affected by temperature dependent fluid properties is examined. The five temperature dependent fluid properties examined were: surface tension, mass density, dynamic viscosity, thermal conductivity and specific heat capacity. The investigation utilized a theoretical model based on the conservation of mass, momentum and energy, including the physically appropriate Newton's Law of Cooling to incorporate temperature changes on the surface of the film. A depth-integrated model was considered and modified Integral Boundary Layer equations were generated. A linear stability analysis was carried out. Nonlinear numerical simulations were also performed in order to verify the predictions of the linear theory and determine the evolution of the unstable flow.*

**Keywords:** inclined flow, thermocapillary effects, temperature dependent fluid properties, weighted residual method

## 1. Introduction

The flows of thin fluid films exist in various aspects of daily life. In engineering applications we see their usage in distillation units, condensers and heat exchangers. In biological systems we find thin fluid films lining the airways in the lungs and thin tear films covering the eye. These are just a few of the many examples of daily occurrences of thin fluid films. In geophysical events we see the flow of shallow layers on a larger scale in the forms of gravity currents, mud, granular and debris flow, snow avalanches and lava flows. Although these occurrences seem very different with little in common with each other, they can all be modeled using the same mathematical principles. Important simplifications in the mathematical formulation can be made on the assumption of small aspect ratios of vertical to lateral length scales.

Shkadov [1] exploited the assumed shallowness of the flow and filtered out the explicit dependence on the depth coordinate by depth integrating the general flow equations. The Integrated Boundary Layer (IBL) equations obtained by Shkadov are however inaccurate for determining the instability threshold. Ruyer-Quil and Manneville [2] used a modified IBL approach, where they combined a gradient

expansion with a weighted residual technique using polynomial test functions. The modified IBL equations are similar to Shkadov's, but correctly predict the instability threshold.

There has been some research done in recent years on the problem of a thin fluid flow down a heated inclined plane [3-6]. This research considered the Marangoni effect associated with the variation in surface tension with temperature, however other fluid properties were assumed to be temperature independent. Goussis and Kelly [7] examined the role of temperature variation in the viscosity only. They performed a linear stability analysis on the Navier-Stokes equations and found that heating a film whose viscosity decreases with temperature has the effect of destabilizing the flow. Hwang and Weng [8] set up a Benney equation and performed a linear and weakly nonlinear stability analysis on it. The limitation of these investigations [7,8] is that a prescribed constant temperature at the surface of the fluid is assumed. As a result, the Marangoni effect does not play a role. In order to capture the Marangoni effect, the model must apply the physically appropriate Newton's Law of Cooling at the liquid-air interface.

The purpose of the present paper will be to examine how the flow of a thin fluid film down a heated inclined plane is affected by temperature dependent fluid properties. As was determined in [3-6], when a gravity-driven fluid film is heated, the variation in its surface tension with temperature can combine with the inertial effects of the flow to generate interfacial instability. We will extend the basic non-isothermal problem with temperature dependent surface tension, to also include temperature variation in mass density, dynamic viscosity, thermal conductivity and specific heat. In obtaining our model, we will use a modified IBL approach, like that used by Ruyer-Quil and Manneville [2] for the isothermal problem and later extended by Trevelyan et al. [6] for the basic non-isothermal problem. This model will be used to perform linear and nonlinear stability analyses on the flow.

## 2. Governing Equations

We model our fluid flow down a heated inclined plane based on the conservation of mass, momentum and energy.

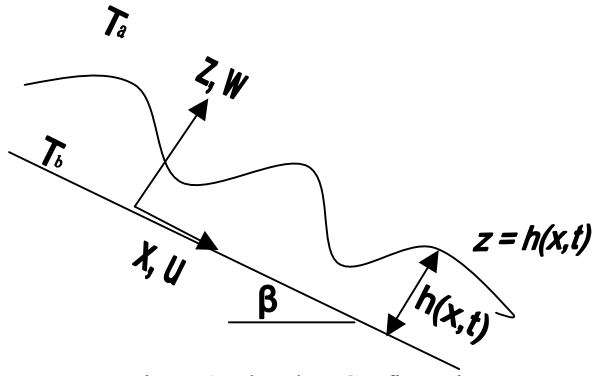


Figure 1: The Flow Configuration

As is illustrated in Figure 1, we assume that the inclined surface along which the fluid is flowing is even and impermeable and that the flow is two-dimensional.  $T_a$  refers to the temperature of the ambient medium and  $T_b (>T_a)$  refers to the temperature of the ramp. The velocity components in the  $x$  and  $z$  directions are denoted by  $u$  and  $w$  respectively.

The general equations for the conservation of momentum, mass and energy then are

$$\frac{D(\rho u)}{Dt} = -p_x + \rho g \sin \beta + [\mu u_x]_x + [\mu u_z]_z + \mu_x u_x + \mu_z w_x$$

$$\frac{D(\rho w)}{Dt} = -p_z - \rho g \cos \beta + [\mu w_x]_x + [\mu w_z]_z + \mu_x u_z + \mu_z w_z$$

$$\frac{D\rho}{Dt} + \rho(u_x + w_z) = 0$$

$$\frac{D}{Dt}(c_p \rho T) = [KT_x]_x + [KT_z]_z$$

where  $\frac{D}{Dt} = \partial_t + u\partial_x + w\partial_z$ ,  $p$  is the pressure,  $\rho$  mass density,  $\mu$  dynamic viscosity,  $g$  acceleration due to gravity,  $c_p$  specific heat,  $T$  temperature and  $K$  thermal conductivity. These equations are general in the sense that they apply to fluid flows with variable fluid properties.

Since we are testing the effects of temperature dependent fluid properties, we allow the properties of the fluid to vary with temperature as follows:

$$\rho = \rho_0 [1 - \hat{\alpha}(T - T_a)], \quad \sigma = \sigma_0 - \gamma(T - T_a), \quad K = K_0 + \hat{\Lambda}(T - T_a), \\ \mu = \mu_0 - \hat{\lambda}(T - T_a), \quad c_p = c_{p_0} + \hat{S}(T - T_a)$$

where  $\sigma$  is the surface tension, and  $\hat{\alpha}, \gamma, \hat{\Lambda}, \hat{\lambda}$  and  $\hat{S}$  are positive parameters measuring the rate of change with respect to the temperature. Note that the expression for  $\mu$  is a simplification of the commonly assumed Arrhenius-type

exponential relation  $\mu = \mu_0 e^{-\frac{\hat{\lambda}(T - T_a)}{\mu_0}}$ , used, for example, to describe the viscosity of lava flows and ice sheets [9]. The linear formulation was initially employed and justified by Reisfeld and Bankoff [10].

The governing equations are simplified by applying the Boussinesq approximation which assumes the mass density to be constant ( $\rho = \rho_0$ ) except where it appears in a gravitational term. We will then make the resulting equations non-dimensional by scaling with the uniform and steady flow. To do this we make the following transformation (where the \*s denote the non-dimensional quantities):

$$x = (H/\delta)x^*, \quad z = Hz^*, \quad h = Hh^*, \quad u = Uu^*, \quad w = \delta U w^*, \quad t = (L/U)t^*, \\ p = \rho_0 U^2 p^*, \quad T - T_a = \Delta T T^*, \quad \text{where } \Delta T = T_b - T_a,$$

$$U = Q/H, \quad H = \left( \frac{3\mu_0 Q}{\rho_0 g \sin \beta} \right)^{1/3}, \quad Q \text{ is the volume flux, and}$$

$\delta$  the aspect ratio. The non-dimensional governing equations can be expressed as (dropping the \*s for notational convenience and discarding the  $O(\delta^2)$  terms):

$$u_x + w_z = 0 \quad (1)$$

$$\text{Re} \delta (u_t + uu_x + ww_z) = -\text{Re} \delta p_x + 3(1 - \alpha T) + [(1 - \lambda T)u_z]_z \quad (2)$$

$$0 = -\text{Re} p_z - 3 \cot \beta (1 - \alpha T) + \delta [(1 - \lambda T)w_z]_z + \quad (3)$$

$$\delta (1 - \lambda T)_x u_z + \delta (1 - \lambda T)_z w_z \quad (4)$$

$$\delta \text{Pr Re} \frac{D}{Dt} \left[ (1 + S/\Delta T_r) T + ST^2 \right] = [(1 + \Lambda T)T_z]_z$$

where  $\text{Re} = \frac{\rho_0 U H}{\mu_0}$  is the Reynolds number,  $\text{Pr} = \frac{\mu_0 c_{p_0}}{K_0}$

is the Prandtl number,  $\Delta T_r = \Delta T / T_a$  is the relative temperature difference of the ramp and ambient medium, and  $\alpha, \lambda, \Lambda$ , and  $S$  are the scaled rates of change with temperature of the density, viscosity, thermal conductivity and specific heat respectively.

At the bottom ( $z=0$ ) we have the conditions  $u=w=0$ ,  $T=1$ , while at the surface ( $z=h(x,t)$ ) the normal and tangential stress conditions, respectively are

$$p + \frac{2\delta}{\text{Re}} (1 - \lambda T)u_x + \delta^2 \text{We} h_{xx} = 0 \quad (5)$$

$$(1 - \lambda T)u_z + \text{Ma Re} \delta (T_x + h_x T_z) = 0 \quad (6)$$

where  $\text{We} = \frac{\sigma_0}{\rho_0 U^2 H}$  is the Weber number (surface tension

parameter) and  $\text{Ma} = \frac{\gamma \Delta T}{\rho_0 U^2 H}$  is the Marangoni number

(scaled surface tension gradient). The Weber number is large for most fluids and we thus assume that  $\delta^2 \text{We}$  is not negligible.

The energy condition at the surface is assumed to be given by Newton's Law of Cooling, which in scaled form is

$$(1 + \Lambda T)T_z = -\text{Bi}T \quad (7)$$

where  $\text{Bi} = \frac{\alpha_g H}{K_0}$  is the Biot number, with  $\alpha_g$  being the heat transfer coefficient.

Finally, the kinematic condition at the surface is

$$w = h_t + u h_x \quad (8)$$

### 3. Depth integrated model

Our aim now is to reduce the space dimensionality of the governing equations by eliminating the explicit variation with the depth coordinate  $z$ . We begin by depth integrating the conservation of mass equation (1) to obtain:

$$h_t + q_x = 0 \tag{9}$$

where  $q = \int_0^h u dz$  is defined as the flow rate. Integrating the

$z$ -momentum equation (3) we obtain the expression for the pressure, which is then substituted into the  $x$ -momentum equation (2). This gives us:

$$\begin{aligned} \text{Re} \delta (u_t + uu_x + wu_z) &= 3(1 - \alpha T) + [(1 - \lambda T)u_z]_z \\ &+ 3\delta \cot \beta \alpha \theta_x + \delta^3 \text{We} h_{xxx} - 3\delta \cot \beta h_x \\ &- 3\delta \cot \beta \alpha \int_h^z T_x dz \end{aligned} \tag{10}$$

where  $\theta$  denotes the surface temperature, i.e.  $\theta(x, t) = T(x, h(x, t), t)$ .

We recognize that our new  $x$ -momentum equation (10) and our temperature equation (4) cannot be converted to our new variables  $h, q$  and  $\theta$  via direct integration. We will thus proceed by employing a weighted residual method. The general approach is to expand  $u$  and  $T$  in linear combinations of  $z$  dependent test functions. The coefficients are then determined by equating the weighted residuals to zero.

For the basic non-isothermal problem, Trevelyan et al. [6] extended the method proposed by Ruyer-Quil and Manneville [2] for the isothermal flow and assumed the following profiles for  $u$  and  $T$ :

$$\begin{aligned} u(x, z, t) &= \frac{3}{2h^3(x, t)} q(x, t) b_0(x, z, t) \\ &+ \frac{\delta \text{Ma} \text{Re} \theta_x(x, t)}{4h(x, t)} b_1(x, z, t) \end{aligned} \tag{11}$$

$$T(x, z, t) = 1 + \frac{\theta(x, t) - 1}{h(x, t)} z \tag{12}$$

where  $b_0(x, z, t) = 2hz - z^2$ , and  $b_1(x, z, t) = 2hz - 3z^2$ .

It should be pointed out that the velocity profile (11)

satisfies the condition  $q = \int_0^h u dz$ , the bottom conditions

( $u=w=0$ ) as well as the surface tangential stress condition (6) which for the basic non-isothermal problem is given by  $u_z = -\text{Ma} \text{Re} \delta \theta_x$  at  $z=h(x, t)$ . The temperature profile (12) satisfies the bottom condition ( $T=1$ ) but not the surface condition (7)  $T_z = -\text{Bi} \theta$  at  $z=h(x, t)$ . However this condition can be incorporated into the residual by implementing integration by parts in the integration process.

For the isothermal problem Ruyer-Quil and Manneville [11] formally analyzed the accuracy of employing the velocity profile (11) (with  $\text{Ma}=0$ ). They show that employing more elaborate expansions leads to formulations

which ultimately converge to the modified IBL equations. For the basic non-isothermal problem, Trevelyan et al. [6] demonstrate the efficacy of the temperature profile (12). Indeed, the linear stability analysis of the modified IBL equations predicts the correct critical Reynolds number for the onset of instability as obtained from the full equations for the isothermal problem and the basic non-isothermal problem.

For the current problem with variation in all fluid properties we again resort to the profiles (11) and (12). No adjustment is made to the profiles to account for the extra temperature variations. Various options were considered but none improved the agreement with the full equations, they only complicated the governing equations. It turns out that for the current problem the profile for  $u$  in (11) does not satisfy the surface tangential force condition which can be stated as  $u_z = (-\text{Ma} \text{Re} \delta \theta_x) / (1 - \lambda \theta)$  at  $z=h(x, t)$ . However, like with the temperature profile, we can include the correct condition into the integrated momentum equation.

In accordance with the Galerkin method, test functions are used to weight the residuals. The temperature equation (4) is weighted with  $z$  and then depth integrated. Since we would like to satisfy the boundary condition (7), we apply integration by parts to the term arising from  $[(1 + \lambda T)T_z]_z$  and replace the boundary term. We thus obtain the following equation in  $h, q$  and  $\theta$ :

$$\begin{aligned} h(3S\theta + S + 2\left(1 + \frac{S}{\Delta T_r}\right)\theta_t) &= \frac{3}{10} S q_x - \frac{6}{5} S q \theta_x \\ &- \frac{21}{5} S q \theta_{xx} - \frac{2}{5} S \theta^2 q_x + \frac{1}{10} S \theta q_x \\ &+ \frac{7}{20} \left(1 + \frac{S}{\Delta T_r}\right) (1 - \theta) q_x - \frac{27}{10} \left(1 + \frac{S}{\Delta T_r}\right) q \theta_x \\ &+ \frac{6(1 - \theta(1 + \text{Bi}h))}{\text{Pr} \text{Re} \delta h} - \frac{3 \Lambda (\theta^2 - 1)}{\text{Pr} \text{Re} \delta h} \end{aligned} \tag{13}$$

The  $x$ -momentum equation (10) is weighted with  $b_0$  and then depth integrated. Since the profile for  $u$  in (11) does not satisfy the surface tangential force condition we apply integration by parts twice to the term arising from  $[(1 - \lambda T)u_z]_z$  and include the boundary conditions to obtain:

$$\begin{aligned} \alpha_t - \delta &\left( \frac{9 q^2 h_x}{7 h^2} - \frac{17 q_x q}{7 h} - \frac{5}{4} \text{Ma} \theta_x - \frac{5}{48} \lambda \text{Ma} \theta_x \theta \right. \\ &+ \frac{5}{48} \lambda \text{Ma} \theta_x - \frac{5 \text{h} \cot \beta h_x}{2 \text{Re}} + \frac{29 \text{h} \cot \beta h_x \alpha \theta}{16 \text{Re}} \\ &\left. + \frac{5}{6} \text{h} \text{We} \delta^2 h_{xxx} + \frac{11 h^2 \cot \beta \alpha \theta_x}{16 \text{Re}} + \frac{11 \text{h} \cot \beta \alpha h_x}{16 \text{Re}} \right) \\ &- \frac{5 h}{2 \text{Re}} + \frac{25 \text{h} \alpha \theta}{16 \text{Re}} + \frac{15 \text{h} \alpha}{16 \text{Re}} - \frac{5 \lambda q \theta}{8 h^2 \text{Re}} - \frac{15 \lambda q}{8 h^2 \text{Re}} \\ &+ \frac{5 q}{2 h^2 \text{Re}} = 0 \end{aligned} \tag{14}$$

These three equations (9), (13) and (14) govern the unknowns  $h, q$  and  $\theta$  and constitute “modified” IBL equations. If we allow  $\alpha = \lambda = S = 0$ , these equations reduce



to those used by Trevelyan et al. [6] for the basic non-isothermal problem.

For the first step of our stability analysis, we compute the steady state, by setting all time derivatives to zero. From the continuity equation (9) we realize that  $q$  is a constant. Thus, our steady state solution is  $h=1$ ,  $q=q_s=\text{constant}$  and  $\theta=\theta_s=\text{constant}$ , where  $q_s$  and  $\theta_s$  are obtained from:  $8 - 5\alpha\theta_s - 3\alpha + 2\lambda\theta_s q_s + 6\lambda q_s - 8q_s = 0$  and

$$-2 + 2\theta_s \text{Bi} + 2\theta_s + \Lambda \theta_s^2 - \Lambda = 0.$$

We then introduce a perturbed steady flow into the modified IBL equations by letting  $h=1+\tilde{h}$ ,  $q=q_s+\tilde{q}$  and  $\theta=\theta_s+\tilde{\theta}$ , and then linearize with respect to the perturbations. Into these linearized equations, we introduce the normal modes:  $(\tilde{h}, \tilde{q}, \tilde{\theta}) = (\hat{h}, \hat{q}, \hat{\theta})e^{ct+ikx}$ . This results in a

3x3 system of linear (homogeneous) equations for  $\hat{h}$ ,  $\hat{q}$  and  $\hat{\theta}$ . Solving the characteristic equation we get a dispersion relation, which we solve for  $c$ . For neutral stability, the growth rate is zero. Setting  $\Re(c)=0$  gives us our neutral stability curve in the  $\text{Re}-k$  plane. The critical Reynolds number for the onset of interfacial instability,  $\text{Re}_{\text{CRIT}}$ , is the intercept of this curve with the  $\text{Re}$  axis. So, we set  $k=0$  and solve for  $\text{Re}$ . This relation reveals the conditions under which the steady flow is unstable. All calculations were done analytically using Maple. However, the expressions for the neutral stability curve and the critical Reynolds number are too long to give. It is however apparent from this formula that  $\text{Re}_{\text{CRIT}}$  reduces (if we let

$$\alpha=\lambda=A=S=0) \text{ to the familiar } \frac{10(1+Bi)^2 \cot \beta}{5MaBi + 12(1+Bi)^2}, \text{ which}$$

is the  $\text{Re}_{\text{CRIT}}$  for the basic non-isothermal problem obtained by D'Alessio et al. [12] and coincides with that obtained by Trevelyan et al. [6] if the difference in scaling is taken into account. Furthermore, if we set all the temperature variations to zero, including  $Ma$  and  $Bi$ , the result for  $\text{Re}_{\text{CRIT}}$  reduces to  $(5/6)\cot\beta$ , which is the well known result for isothermal flow [13,14]. This is exactly what should happen, and helps to validate our result.

It turns out that the general expression for the critical Reynolds number is independent of the Weber number, as is the case for the isothermal and basic nonisothermal problems. We point out that the Weber number is the scaled surface tension at the reference temperature  $T_b$  (the prescribed temperature of the bottom surface). The Marangoni number, on the other hand, is the scaled gradient of the surface tension with temperature, and measures the effect of thermocapillary forces, which do affect the onset of instability.

In order to determine how the onset of instability is affected by the variation in fluid properties we examine plots of  $\text{Re}_{\text{CRIT}}$  as a function of various parameters. In Figures 2 and 3 we display the effect of the density variation parameter,  $\alpha$ ,

on the stability of the flow. The considered values of  $\alpha$  are less than 1 in order to obtain positive values for the density. It is evident from the results that, depending on the value of other parameters, increasing  $\alpha$  can result in an increase or decrease in  $\text{Re}_{\text{CRIT}}$ . In general, a decrease in mass density reduces inertia and stabilizes the flow. However, for our problem the vertical temperature gradient in the fluid results in a top-heavy density stratification. The density differences associated with depth fluctuations resulting from surface waves can combine with thermocapillary forces and destabilize the flow. In Figure 2 we see that if the specific heat variation parameter  $S$  is sufficiently large, then the density variation acts to destabilize the flow. This is explained by the fact that an increase in the specific heat of the fluid decreases thermal diffusivity and thus steepens temperature gradients and consequently accentuates the density stratification. The results in Figure 3 reveal that the same effect occurs if  $Ma$  is sufficiently large. In other words, with substantial thermocapillary action, increasing the density variation destabilizes the flow.

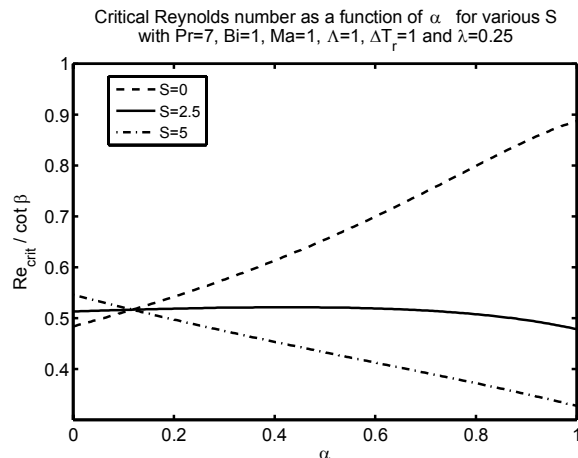


Figure 2

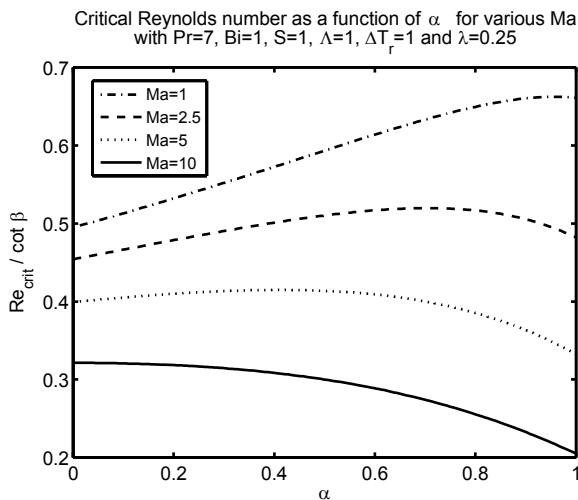


Figure 3



In Figure 4 we present the variation of  $Re_{CRIT}$  with  $Ma$  for different values of  $\lambda$ . It should be pointed out that since we assume the viscosity to decrease with temperature we must restrict the range of  $\lambda$  to nonnegative values less than 1 in order to maintain a positive value for the viscosity. In all cases  $Re_{CRIT}$  decreases with  $Ma$  in accordance with the expectation that strengthening the thermocapillary effects acts to destabilize the flow. Another interesting observation however, is that as  $\lambda$  increases there is less variation in  $Re_{CRIT}$  with  $Ma$ . Indeed, in the case with  $\lambda = 0.75$ ,  $Re_{CRIT}$  is essentially independent of  $Ma$ . We can then conclude that if the viscosity is sufficiently reduced the resulting increase to flow inertia is the dominant instability mechanism and the contribution from the Marangoni effect is negligible.

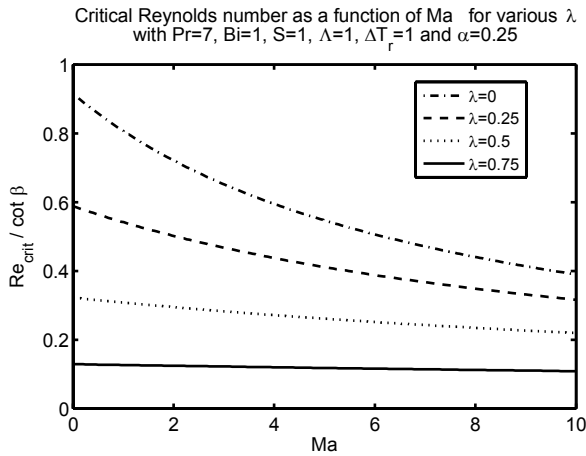


Figure 4

### 4. Nonlinear simulations

In order to solve the system (9), (13) and (14) numerically, we express these equations in a form suitable for applying numerical methods. Instead of working with  $\theta$ , we introduce  $\Phi$ , related to  $\theta$  by the equation  $\Phi = (\theta - 1)h$ . From the relation  $(T - 1)h = (\theta - 1)z$ , it follows that the variable  $\Phi$  is related to  $T$  through:  $\int_0^h (T - 1)dz = \frac{\Phi}{2}$  and thus,  $\Phi$  is proportional to the lineal heat content stored in the fluid layer. In terms of  $\Phi$  the  $x$ -momentum equation (14) is expressed as:

$$q_t + \frac{\partial}{\partial x} \left[ \frac{9q^2}{7h} + \frac{5 \cot \beta}{4 \text{Re}} (1 - \alpha)h^2 + \frac{5Ma\Phi}{4h} + \frac{5}{96} \lambda Ma \frac{\Phi^2}{h^2} - \frac{11\alpha \cot \beta}{16\text{Re}} h\Phi \right] = \frac{qq_x}{7h} + \frac{5}{6} \delta^2 Weh h_{xxx} + \frac{7\alpha \cot \beta}{16\text{Re}} h_x \Phi + \frac{5}{2\text{Re}\delta} \left( h - \frac{q}{h^2} \right) - \frac{5\alpha}{16\text{Re}\delta} (8h + 5\Phi) + \frac{5\lambda}{8\text{Re}\delta} \frac{q}{h^3} (4h + \Phi) \quad (16)$$

while introducing  $\Phi$  into the temperature equation (13) and discarding  $O(\delta^2)$  terms we obtain:

$$\Phi_t + \frac{\partial}{\partial x} \left[ \frac{27\Phi q}{20h} + \frac{3}{80} S \frac{q\Phi^2}{h^2} \right] = \frac{7}{40} \frac{\Phi q_x}{h} + \frac{S}{10} \frac{\Phi^2 q_x}{h^2} - \frac{3(Bih + Bi\Phi + \Phi/h)}{\delta \text{Pr Re } h\Theta} - \frac{3\Lambda(2\Phi h + \Phi^2)}{2\delta \text{Pr Re } h^3 \Theta} \quad (17)$$

$$\text{where } \Theta = 1 + \frac{S}{\Delta Tr} + 2S + \frac{3}{2} S \frac{\Phi}{h}$$

We will use the Fractional-Step Method [15] in obtaining a numerical method for solving (9), (16) and (17). The idea is to split the equations into two systems that can be solved in an alternating manner.

For the first step we discard the derivative terms that can not be expressed in conservation form and solve:

$$h_t + q_x = 0$$

$$q_t + \frac{\partial}{\partial x} \left[ \frac{9q^2}{7h} + \frac{5 \cot \beta}{4 \text{Re}} (1 - \alpha)h^2 + \frac{5Ma\Phi}{4h} + \frac{5}{96} \lambda Ma \frac{\Phi^2}{h^2} - \frac{11\alpha \cot \beta}{16\text{Re}} h\Phi \right] =$$

$$\frac{5}{2\text{Re}\delta} \left( h - \frac{q}{h^2} \right) - \frac{5\alpha}{16\text{Re}\delta} (8h + 5\Phi) + \frac{5\lambda}{8\text{Re}\delta} \frac{q}{h^3} (4h + \Phi)$$

$$\Phi_t + \frac{\partial}{\partial x} \left[ \frac{27\Phi q}{20h} + \frac{3}{80} S \frac{q\Phi^2}{h^2} \right] = - \frac{3(Bih + Bi\Phi + \Phi/h)}{\delta \text{Pr Re } h\Theta} - \frac{3\Lambda(2\Phi h + \Phi^2)}{2\delta \text{Pr Re } h^3 \Theta}$$

over a time step  $\Delta t$ . In the second step we focus on the terms not included in the previous step and solve

$$q_t = \frac{qq_x}{7h} + \frac{5}{6} \delta^2 Weh h_{xxx} + \frac{7\alpha \cot \beta}{16\text{Re}} h_x \Phi$$

$$\Phi_t = \frac{7}{40} \frac{\Phi q_x}{h} + \frac{S}{10} \frac{\Phi^2 q_x}{h^2}$$

using the solution obtained from the first step as the initial condition. The second step then returns the solution for  $q$  and  $\Phi$  at the new time  $t + \Delta t$ , with the solution for  $h$  being that obtained in the first step.

The system considered in the first step consists of nonlinear hyperbolic conservation equations with source terms. While there are several methods available to solve this system, its complicated eigenstructure makes the use of eigen-based methods impractical. We resort to MacCormack's method since it can be applied component-wise and does not require the eigenstructure of the system. MacCormack's method is a conservative second-order accurate finite difference scheme, which correctly captures discontinuities and converges to the physical weak solution of the problem. LeVeque & Yee [16] extended MacCormack's method to include source terms via an explicit predictor-corrector scheme.

In the second step we discretize the equations by means of the Crank-Nicolson scheme and using the output from the

first step as an initial condition, leads to a nonlinear system of algebraic equations, which was solved iteratively. A robust algorithm, taking advantage of the structure and sparseness of the resulting linearized systems, was used to speed up the iterative process. It was found that convergence was reached quickly, typically in less than five iterations.

To perform a stability analysis of the flow we begin by solving our equations on a periodic spatial domain, from 0 to  $L$ . As the initial condition, we use the base flow, with a small amplitude sinusoidal perturbation of length  $L$  added to  $h$ :

$$h = 1 + 0.0001 \sin\left(\frac{2\pi}{L}x\right), q = q_s, \theta = \theta_s$$

With this small perturbation now in our system, we calculate the evolution and determine if it is amplified or dampened. If the evolution of the wave is amplified, then the system is unstable. By iterating over  $Re$  we can determine the value for which a perturbation with wavenumber  $k=2\pi/L$  is neutrally stable and thus by considering different  $L$  values we obtain points on the neutral stability curve. For longer  $L$  values ( $L>3$ ) we use a mesh width of  $\Delta x=0.02$  which required, for numerical stability, a time step of  $\Delta t=10^{-4}$ . For the shorter  $L$  values we used  $\Delta x=0.005$  and  $\Delta t=7 \times 10^{-6}$ . We found the results to be in excellent agreement with those from the linear analysis.

For supercritical conditions the nonlinear simulations on a sufficiently long domain can be used to determine the evolution of the unstable flow. The advantage of the nonlinear simulations is that they include the nonlinear interactions of the perturbations and thus capture the entire instability mechanism of the flow. Furthermore, for unstable flows, the temporal evolution can be continued until the growth of the disturbances reaches saturation. An illustration of the evolution of an unstable film flow is given in Figure 5. Notice that at time  $t=40$  our small amplitude sinusoidal perturbation makes very small waves in our system. As time goes on that small perturbation becomes larger and larger, until finally we have a permanent wave structure at time  $t=140$ . In other words these solitary waves will not subside or grow in time and will propagate with a constant speed.

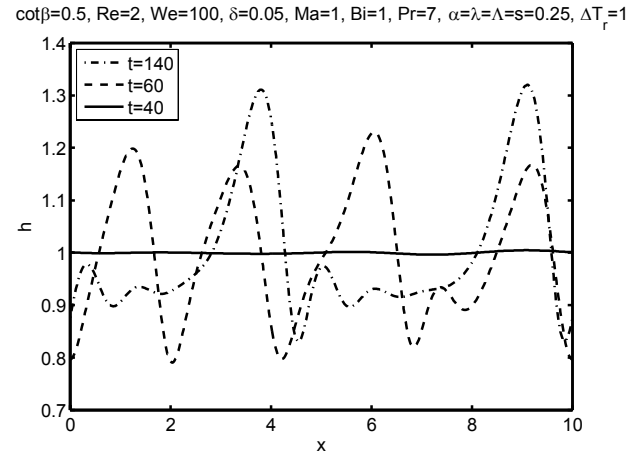


Figure 5: Evolution of the surface profile of an unstable flow

In Figure 6 we compare the permanent surface profiles for unstable flows with different Reynolds numbers. Notice that for the larger  $Re$  values the instability leads to large solitary-wave structures with the height increasing with  $Re$ . However, for smaller  $Re$  values the flow is “less unstable” with the interfacial deflection being almost sinusoidal with small amplitude.

In Figure 7 we consider another plot showing the relation between surface temperature and wave height for a permanent solution. We notice that at the crest of the waves the surface is cooler; and at the troughs the surface is warmer. This makes sense as increasing the distance from the surface of the fluid to the heated ramp will cool the fluid’s surface and vice versa.

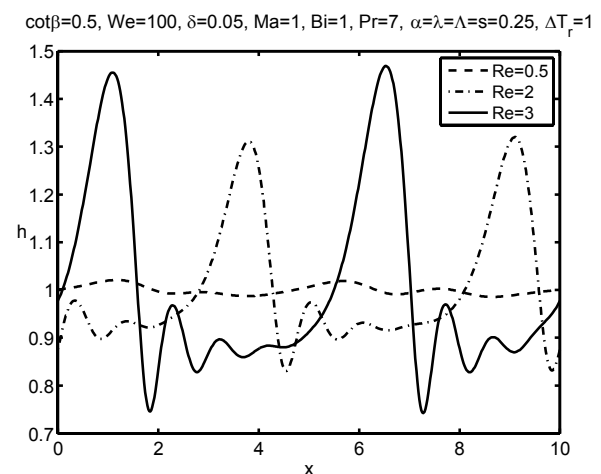


Figure 6: Permanent surface profile for an unstable flow

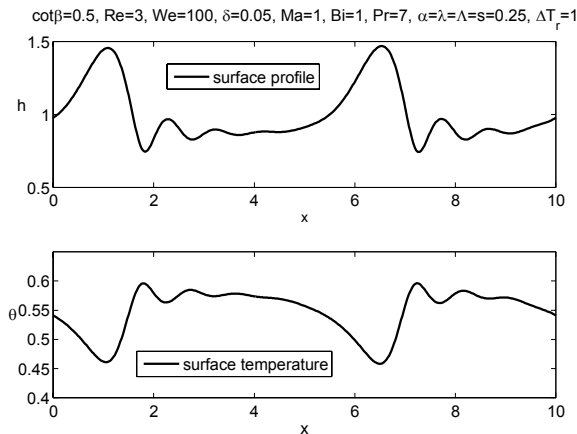


Figure 7: Relation between the permanent surface profile and surface temperature.

## 5. Concluding remarks

The purpose of this paper was to examine how the flow of a fluid film down a heated inclined plane is affected by temperature dependent fluid properties. The effects of five different temperature dependent fluid properties were examined: surface tension, mass density, dynamic viscosity, thermal conductivity and specific heat capacity. Each of these can be significantly affected by changes in temperature and can have either stabilizing or destabilizing effects on the fluid flow. A depth-integrated model was obtained by implementing a weighted residual method. We performed a linear stability analysis and obtained an expression for the critical Reynolds number for the onset of instability, which was analyzed to determine the effect of the variations in fluid properties with respect to temperature. Numerical simulations of the nonlinear equations were employed to calculate the evolutions of unstable flows and examine the effect of the degree of instability on the structure of the permanent interfacial waves.

### Acknowledgement

Funding for this research was provided by the Natural Sciences and Engineering Research Council of Canada.

## References

- [1] Shkadov, V. Y., 1967. Wave conditions in flow of thin layer of a viscous liquid under the action of gravity. *Izvestiya Akademii Nauk SSSR, Mekhanika Zhidkosti i Gaza*. **1**, 43-50.
- [2] Ruyer-Quil, C. & Manneville, P., 2000. Improved modeling of flows down inclined planes. *The European Physical Journal B*. **15**, 357-369.
- [3] Kalliadasis, S., Demekhin, E. A., Ruyer-Quil, C. & Velarde, M., 2003. Thermocapillary instability and wave

formation on a film flowing down a uniformly heated plane. *Journal of Fluid Mechanics*. **492**, 303-338.

- [4] Ruyer-Quil, C., Scheid, B., Kalliadasis, S., Velarde, M.G., & Zeytounian R. Kh., 2005. Thermocapillary long waves in a liquid film flow. Part 1. Low-dimensional Formulation. *Journal of Fluid Mechanics*. **538**, 199-222.
- [5] Scheid, B., Ruyer-Quil, C., Kalliadasis, S., Velarde, M.G., & Zeytounian R. Kh., 2005. Thermocapillary long waves in a liquid film flow. Part 2. Linear stability and nonlinear waves. *Journal of Fluid Mechanics*. **538**, 223-244.
- [6] Trevelyan, P.M.J., Scheid, B., Ruyer-Quil, C., & Kalliadasis, S., 2007. Heated Falling Films. *Journal of Fluid Mechanics*. **592**, 295-334.
- [7] Goussis, D. A., & Kelly, R. E., 1985. Effects of Viscosity Variation on the Stability of Film Flow Down Heated or Cooled Inclined Surfaces: Long-wavelength Analysis. *Physics of Fluids*. **28**, 3207-3214.
- [8] Hwang, C.-C. & Weng, C.-I., 1988. Non-linear stability analysis of film flow down a heated or cooled inclined plane with viscosity variation. *International Journal of Heat and Mass Transfer*. **31**, 1775-1784.
- [9] Craster, R. V. & Matar, O. K., 2009. Dynamics and Stability of Thin Liquid Films. *Reviews of Modern Physics*. **81**, 1131-1198.
- [10] Reisfeld, B., & Bankoff S. G., 1990. Nonlinear stability of a heated thin film with variable viscosity. *Physics of Fluids A*. **2**, 2066-2067
- [11] Ruyer-Quil, C. & Manneville, P., 2002. Further Accuracy and Convergence Results on the Modeling of Flows Down Inclined Planes by Weighted-Residual Approximations. *Physics of Fluids*. **14**, 170-183
- [12] D'Alessio, S. J. D., Pascal, J. P., Jasmine, H. A., & Ogden, K. A., 2010. Film flow over heated wavy inclined surfaces. *Journal of Fluid Mechanics*. **665**, 418-456.
- [13] Benjamin, T.B., 1957. Wave formation in laminar flow down an inclined plane. *Journal of Fluid Mechanics*. **2**, 554-573.
- [14] Yih, C.-S., 1963. Stability of Liquid Flow down an Inclined Plane. *Physics of Fluids*. **6**, 321-334.
- [15] LeVeque, R. J., 2002. *Finite Volume Methods for Hyperbolic Problems*. Cambridge, Cambridge University Press.
- [16] LeVeque, R. J. & Yee, H. C., 1990. A study of numerical methods for hyperbolic conservation laws with stiff source terms. *Journal of Computational Physics*. **86**, 187-210.

# A stabilizing approach to resolve the inconsistent results prevalent when applying the standard time domain BEM to 3D problems

A. Kamali Yazdi\* and M. Abdollahian<sup>+</sup>

\*Department of Civil Engineering, Faculty of Engineering, University of mashhad, mashhad, Iran

<sup>+</sup>School of Mathematical and Geospatial Sciences, RMIT University, Melbourne, Australia

**Abstract** - *The established authority regulating the time domain boundary element method (BEM) suggests that numerical response has been disturbed either due to the presence of noise or a consequence of unstable behavior. Subsequently, there are techniques which can lead to stable time-domain BEM algorithms. But these techniques introduce another time consuming formulation for BEM to overcome instabilities. This paper proposes a new stabilizing approach to resolve the inconsistent results prevalent when applying the standard time domain BEM to 3D problems. The proposed time-weighted time domain BEM could be employed subsequent to the standard BEM process in order to stabilize the unstable computed results of the standard time domain BEM. The proposed approach is tested on three dimensional problems. The results show that the proposed procedure can effectively be used as a stabilizing approach to resolve the inconsistent results prevalent when applying the standard time domain BEM to 3D.*

**Keywords:** BEM, stability, three dimensional problems, time weighted

## 1. Introduction

Time domain BEM is one of the methods applied for dynamic problems in engineering. Despite the extensive studies on time-domain elastodynamic BE formulations [1-2] there are still many unsolved problems. Some studies such as Dominguez's study [2] or Peirce and Siebrits' [3] showed the evidence of instabilities in boundary integral elastodynamic models. The instabilities become particularly evident when finite domains involving reflection of elastic waves are analyzed. A series of numerical simulations concerning a doubly infinite strip in plane strain conditions exploiting symmetries as a finite body is developed by Dominguez [2]. He employs the standard collocation direct BE method and evident instabilities are encountered even for very small values of the time parameter. Choosing a suitable time step  $\Delta t$  for the evaluative analysis is problematic: the range of values yielding stable results is generally narrow and sometimes vanishes, depending on the mesh adopted.

Other evidences of unstable results are provided by Peirce and Siebrits [3]. They define these phenomena as 'intermittent'.

Performing a series of analyses by monotonically incrementing  $\Delta t$ , one may obtain stable and unstable results often in an unpredictable way.

There are techniques which can lead to stable time-domain BEM algorithms. The  $\varepsilon$  scheme and the half-step approach introduced by Siebrits and Peirce [4], the  $\theta$ -scheme presented by Yu et al. [5-7], the time-convolution modification proposed by Soares and Mansur [8-10] are some of the related works. Employing time and space weighting functions is another approach to improve stability [11, 12]. But these techniques introduce another time consuming formulation for BEM to overcome instabilities. They were not introduced in a procedure that could be used subsequent to the standard BEM for stabilizing the unstable computed results with little effort. In other words, if the problem is studied by standard BEM and in the case that the results have instabilities the problem should be studied again with another formulation. It is true to say there is not a BEM formulation which is stable for all time steps. Thus, proposing methodologies which could be examined subsequent to another formulation (without computing new matrix terms) would be extremely desirable. This would lead to the reduction of the computing time cost. This study proposes an approach based on time weighting method which could be used subsequent to the standard BEM to resolve the inconsistent results prevalent when applying the standard time domain BEM to 3D problems.

The integral equations for a time-weighted time domain BEM were presented by Maier et al. [13]. Guoyou yu et al. [11] applied time weighting method for BEM in 2D scalar wave problems. Kamali yazdi et al. [14] utilized time weighting for dual boundary element method (DBEM) in 3D fracture problems.

In this paper the authors applied a weighted formulation of displacement boundary integral equation as a stabilizing approach subsequent to the standard time domain BEM. A weighted BEM formulation is applied which employs only the computed terms of standard BEM and can be utilized subsequent to the conventional method for stabilizing the results. Simplicity and stability can be mentioned as two appropriate characteristics of the proposed formulation when applied to the solution of the three dimensional problems by the BEM.

The proposed approach has a significant advantage over the existing approaches. If the problem is studied by standard

BEM and in the case that the results have instabilities, the proposed approach could be applied subsequent to the conventional method. Indeed it is not required to solve the problem with another time consuming formulation. In other words, the proposed approach could be considered as a stabilizing method for unstable computed results obtained by standard BEM.

The study of 3D transient elastodynamic problems with actual non-zero values of the Poisson's ratio is considered as one of the most challenging situations from the stability point of view of time domain BE approaches. In those cases, the existence of two kinds of waves in the fundamental solution and the reflection of different kinds of waves on the boundaries may cause causality errors and instability problems. Most published works in the field of stability improvement deal with two dimensional models and they rarely address the cases involving three-dimensional models with non-zero Poisson's ratio. The proposed procedure is tested for finite 3D problems with zero and nonzero Poisson's ratio. The results show that the procedure is capable of solving the problems.

## 2. Basic equations

The integral representation for the displacement  $u$  at point  $x'$  on the boundary  $(\Gamma)$  of an elastic body, at time  $t$  with zero body forces and zero initial conditions can be written as [15]

$$c_{ij} u_j(x', t) = \int_0^t \int_{\Gamma} U_{ij}(x', x, t - \tau) t_j(x, \tau) d\Gamma(x) d\tau - \int_0^t \int_{\Gamma} T_{ij}(x', x, t - \tau) u_j(x, \tau) d\Gamma(x) d\tau \quad (1)$$

Where  $u_j$  and  $t_j$  are the displacements and tractions on the boundary respectively;  $U_{ij}(x', x, t - \tau)$  and  $T_{ij}(x', x, t - \tau)$  are the displacement and traction fundamental solutions of elastodynamic, respectively [15]. The coefficient  $c_{ij}$  depends on the geometry at  $x'$  and  $\int$  stands for a Cauchy principal-value integral

In order to implement a numerical scheme to solve Eq. (1), the boundary  $\Gamma$  is divided into a set of discrete points  $x_m$ ,  $m = 1, 2, \dots, M$ , and the observation time  $t$  is divided into  $N$  time steps ( $N\Delta t$ ). So displacements and tractions over the boundary are approximated using interpolation functions,

$$u_j(x, \tau) = \sum_{m=1}^M \sum_{n=1}^N \eta^n(\tau) \phi^m(x) u_j^{nm} \quad (2)$$

$$t_j(x, \tau) = \sum_{m=1}^M \sum_{n=1}^N \mu^n(\tau) \phi^m(x) t_j^{nm}$$

Where  $n$  and  $m$  refer to time and space respectively,  $\phi^m$  and  $\phi^m$  are space interpolation functions whereas  $\eta^n$  and  $\mu^n$  are time interpolation functions. The use of these approximations enables the displacement integral equation to be discretized as:

$$c_{ij} u_j(x', t) = \sum_{n=1}^N \sum_{m=1}^M \int_{\Gamma_m} U_{ij}(x', x, t - \tau) \mu^n(\tau) \phi^m(x) t_j^{nm} d\tau d\Gamma - \int_{\Gamma_m} T_{ij}(x', x, t - \tau) \eta^n(\tau) \phi^m(x) u_j^{nm} d\tau d\Gamma \quad (3)$$

Where  $\Gamma_m$  are the elements to which node  $m$  belongs. Eq. (3) can be written in more compact form as,

$$c_{ij} u_j(x', t) = \sum_{n=1}^N \sum_{m=1}^M [U_{ij}^{Nnm} t_j^{nm} - T_{ij}^{Nnm} u_j^{nm}] \quad (4)$$

Where

$$U_{ij}^{Nnm} = \int_{\Gamma_m} \int_0^t U_{ij}(x', x, t - \tau) \mu^n(\tau) \phi^m(x) d\tau d\Gamma \quad (5)$$

$$T_{ij}^{Nnm} = \int_{\Gamma_m} \int_0^t T_{ij}(x', x, t - \tau) \eta^n(\tau) \phi^m(x) d\tau d\Gamma$$

Once the boundary conditions are considered, Eq. (4) yields a system of equations which can be solved to obtain the time variation of boundary unknowns. Piecewise constant time interpolation function  $\mu^n(\tau)$  and piecewise linear function  $\eta^n(\tau)$  are used for tractions and displacements respectively. This type of interpolation has been tested by different authors in other time domain BE approaches [2]. So, according to Eq. (2), one obtains:

$$\mu^n(\tau) = I_n(\tau) \quad \text{Where, } I_n(t) = H(\tau - (n-1)\Delta t) - H(\tau - n\Delta t)$$

$$\eta^n(\tau) = \frac{\tau - (n-1)\Delta t}{\Delta t} I_n(\tau) + \frac{(n+1)\Delta t - \tau}{\Delta t} I_{n+1}(\tau) \quad \text{for } n < N$$

$$\eta^n(\tau) = \frac{\tau - (n-1)\Delta t}{\Delta t} I_n(\tau) \quad \text{for } n = N \quad (6)$$

Where  $H(\tau)$  stands for the Heaviside step function.

The time integrals given by Eqs. (5) can be evaluated analytically without much difficulty regarding the above approximation. For explicit expression see Aliabadi [15]. In this paper quadratic discontinuous elements with eight nodes are used (Fig. 1).

The parameter  $\lambda$  in the shape functions of discontinuous elements was chosen arbitrarily as  $2/3$ . Cauchy finite part integrals are applied for evaluating spatial singular integrals [15].

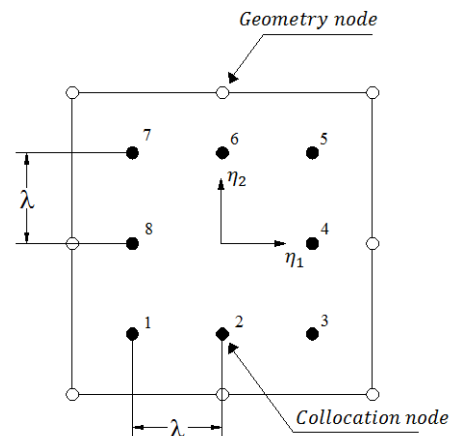


Fig. 1 Eight node discontinuous element

### 3. Weighted integral equations and stabilizing approach

In time-weighted integral equations the average of whole time history of the solution is considered in a weighted residual sense. As a result, although this has no significant effect on early stages, it may have a significant effect on the late time response, especially in problems with unstable behavior. In this section a time-weighted boundary integral equation and the corresponding numerical implementation that leads to a stabilizing approach to resolve the inconsistent results prevalent when applying the standard time domain BEM are presented.

The time-weighted formulation of general displacement boundary integral equation corresponding to Eq. (4), can be written as:

$$\int_0^T W_n(t) \left[ c_{ij} u_j(x', t) - \sum_{n=1}^N \sum_{m=1}^M [U_{ij}^{Nnm} t_j^{nm} - T_{ij}^{Nnm} u_j^{nm}] \right] dt = 0 \quad (7)$$

Where  $W_n(t)$  is a time weighting function, T denotes the total time of analysis ( $T = t_N$ ).

It should be noted that in the case of  $W_n(t) = \eta^n(t) = \mu^n(t)$  Eq. (7) becomes general time-weighted discrete Galerkin type integral equation for three dimensional problems. In this research the time weighting function is chosen as  $W_n(t) = H(t - t_{n-1}) - H(t - t_{n+1})$ . So, Eq. (7) can be written as,

$$\int_{t_{n-1}}^{t_{n+1}} 1 \times \left[ c_{ij} u_j(x', t) - \sum_{n=1}^N \sum_{m=1}^M [U_{ij}^{Nnm} t_j^{nm} - T_{ij}^{Nnm} u_j^{nm}] \right] dt = 0 \quad (8)$$

In this case,  $u^{n+1}$  and  $t^{n+1}$ , are required for computing  $u^n$  and  $t^n$ . So, a prediction of the boundary variables at  $t = t_{n+1}$  is required. This prediction is done by assuming that the velocity change in any boundary variables between  $t_{n-1}$  and  $t_{n+1}$  is constant. So  $u^{n+1} = 2u^n - u^{n-1}$  and  $t^{n+1} = 2t^n - t^{n-1}$  are assumed for computing unknowns at time  $t_{n+1}$ .

For numerical evaluation of time integrals in Eq. (8), only two weighting points are used in each time step ( $t_{n-1}$  and  $t_{n+1}$ )

$$\int_{t_{n-1}}^{t_{n+1}} p(t) dt = \frac{1}{2} \Delta t [p(t_{n-1}) + p(t_n)] + \frac{1}{2} \Delta t [p(t_n) + p(t_{n+1})], \quad \text{the}$$

obtained formulation could be utilized as stabilizing approach equations. In other words, the traditional method could be used and only in the cases that the results have instabilities, the weighted formulation could be applied at that time step with an increase of analysis time which is considerably less than the time of the traditional formulation process. The proposed approach could be considered as a stabilizing method for BEM.

subsequent to the conventional method. By this assumption the formulation becomes:

$$\begin{aligned} & \int_{t_{n-1}}^{t_{n+1}} 1 \times \left[ c_{ij} u_j(x', t) - \sum_{n=1}^N \sum_{m=1}^M [U_{ij}^{Nnm} t_j^{nm} - T_{ij}^{Nnm} u_j^{nm}] \right] dt \\ &= \frac{1}{2} \Delta t \left\{ c_{ij} u_j(x', t_{N+1}) - \sum_{n=1}^{N+1} \sum_{m=1}^M [U_{ij}^{(N+1)nm} t_j^{nm} - T_{ij}^{(N+1)nm} u_j^{nm}] \right. \\ &+ 2 \left[ c_{ij} u_j(x', t_N) - \sum_{n=1}^N \sum_{m=1}^M [U_{ij}^{Nnm} t_j^{nm} - T_{ij}^{Nnm} u_j^{nm}] \right] + c_{ij} u_j(x', t_{N-1}) \\ &\left. - \sum_{n=1}^{N-1} \sum_{m=1}^M [U_{ij}^{(N-1)nm} t_j^{nm} - T_{ij}^{(N-1)nm} u_j^{nm}] \right\} = 0 \quad (9) \end{aligned}$$

Since it was assumed that  $u^{n+1} = 2u^n - u^{n-1}$  and  $t^{n+1} = 2t^n - t^{n-1}$ , Eq(9) can be written as:

$$\begin{aligned} & \int_{t_{n-1}}^{t_{n+1}} 1 \times \left[ c_{ij} u_j(x', t) - \sum_{n=1}^N \sum_{m=1}^M [U_{ij}^{Nnm} t_j^{nm} - T_{ij}^{Nnm} u_j^{nm}] \right] dt \\ &= \frac{1}{2} \Delta t \left\{ 4c_{ij} u_j(x', t_N) - \sum_{m=1}^M [U_{ij}^{(N+1)Nm} + 4U_{ij}^{NNm}] t_j^{Nm} \right. \\ &- \sum_{n=1}^{N-1} \sum_{m=1}^M [U_{ij}^{(N+1)nm} + 2U_{ij}^{Nnm} + U_{ij}^{(N-1)nm}] t_j^{nm} \\ &+ \sum_{m=1}^M [T_{ij}^{(N+1)Nm} + 4T_{ij}^{NNm}] u_j^{Nm} \\ &+ \sum_{n=1}^{N-1} \sum_{m=1}^M [T_{ij}^{(N+1)nm} + 2T_{ij}^{Nnm} + T_{ij}^{(N-1)nm}] u_j^{nm} \\ &\left. + \sum_{m=1}^M [U_{ij}^{NNm} t_j^{(N-1)m} - T_{ij}^{NNm} u_j^{(N-1)m}] \right\} = 0 \quad (10) \end{aligned}$$

Eq. (10) can be used instead of Eq. (4) to obtain the system of equations for  $u^n$  and  $t^n$  which yields the boundary values at that time. The capability of this formulation for stabilizing unstable computed solutions is shown in the next section by numerical examples. It should be noted that only the combination of the terms computed in the conventional method is applied in the weighted algorithm. Therefore if the results have instabilities in conventional method, the weighted formulation could be applied. The evaluating time of terms which expends considerable time in the proposed approach is the same as conventional method and if the integrations are done in the conventional method the results could be directly used in the proposed method to improve the stability. Extra time processing is limited to the combination of terms which had already been computed in the conventional method, producing and solving the system of

It is enough to utilize the stored data of the  $T_{ij}^{Nnm}, U_{ij}^{Nnm}$  terms during the conventional process for producing Eq. (10). This procedure could simply be added to existing BEM codes and only in the case that the results have instabilities could be utilized. The proposed approach in contrary to existing method utilizes only the combination of computed terms of the standard BEM. There is no new term in Eq. (10) that needed to be evaluated after applying standard BEM. It is clear that

the Eq. (10) could be applied for studying problems independently. It was regularized in the form that could also be applied subsequent to the standard form.

## 4. Numerical examples

To assess the efficacy of the proposed approach, two elastodynamic problems have been studied using both the time-weighted formulation and the conventional form. At first, the traditional BEM is applied then the weighted formulation is utilized by using computed terms in conventional method (it is noted that the Eq. (10) could be applied independently). The first problem corresponds to a square bar under impact load. The second problem refers to a short beam under a shear load with a triangular time variation. The Poisson's ratio is different from zero in the second problem which are more susceptible for causality errors and instability problems.

### 4.1. Square bar subjected to dynamic tension loads

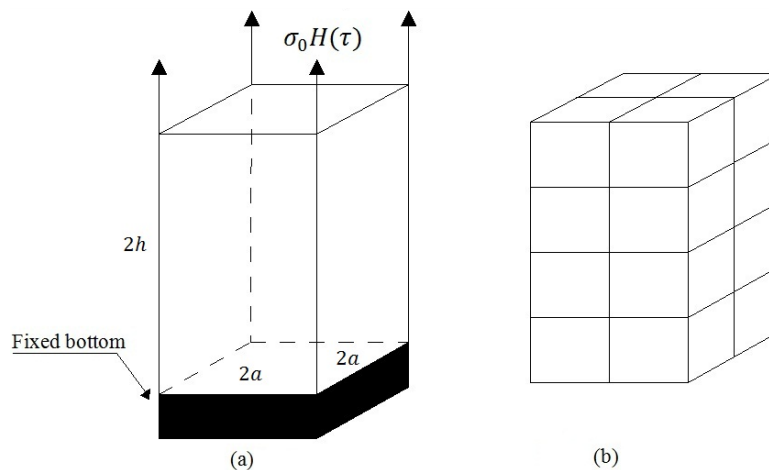


Fig. 2 A bar under uniform load

A square bar ( $2h \times 2a \times 2a$ ) is subjected to a dynamic load  $H(t)$  at top of the bar (Fig. 2a). The ratio  $\frac{a}{h} = 0.5$  is

considered. There are 40 discontinuous elements on the boundary (Fig. 2b). In order to examine the capability of the proposed method, three normalized dimensionless parameters

$\beta$  are chosen ( $\beta = \frac{c_1 \Delta t}{l}$ ). Here,  $l$  is the smallest length of

elements, therefore according to Fig. 2b  $l = a$ . To study the problems both standard and time weighted approach are used.

The dynamic traction at the end of bar for Poisson's ratio equal to zero is shown in Fig.3. The results are acceptable for both traditional and time weighted method when  $\beta = 0.3$ .

Using the traditional method noise in results is observed for  $\beta = 0.2, 0.6$  which is in contrast with the stable results of the proposed method as shown in Fig. 3. It can be seen that the proposed approach has a clear advantage for stabilizing the results and preventing scatter and instability of the computed results.

Applying the standard method, in the early steps the results are in agreement with exact solution; but as the time stepping progresses the cumulative errors of previous steps cause an instability in the results. The proposed approach leads to better and more stable results.

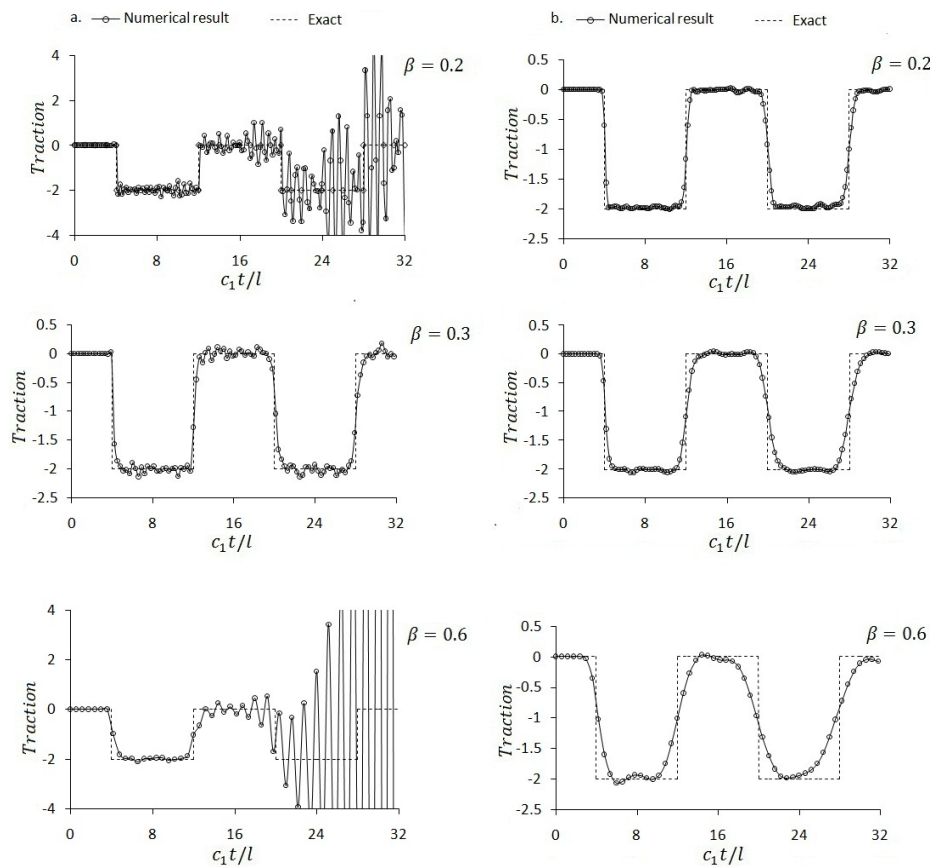


Fig. 3 Dynamic traction for  $\nu = 0$

### 4.2. A Short beam under dynamic shear load

A square bar ( $2h \times 2a \times 2a$ ) is subjected to a dynamic shear load  $\tau(t)$  with triangular time variation at free end (Fig. 4a and 4c). The bar geometry is defined by the ratio  $\frac{a}{h} = 0.5$ , and the value of Poisson's ratio is equal to 0.2. The considered number of elements on the outer boundary is equal to 18 (Fig. 4b). Different normalized dimensionless

parameters  $\beta$  are chosen in order to examine the stability and efficacy of the proposed method. The problem is studied by both standard and time weighted formulation. The horizontal dynamic displacements at point A (Fig.4b) are shown in Fig. 5. For comparison, the result obtained for  $\beta = 0.6$  under the traditional method is plotted in Fig. 5. In this example similar to the previous example using time weighted formulation leads to better results when standard formulation is unstable.

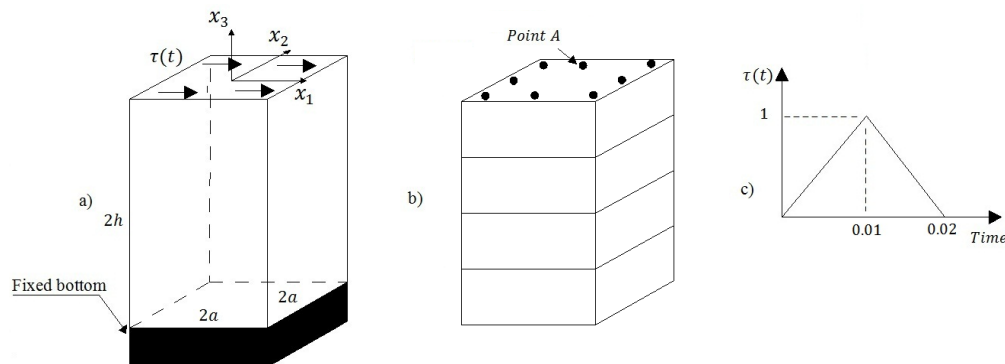


Fig. 4 A short beam under uniform shear load



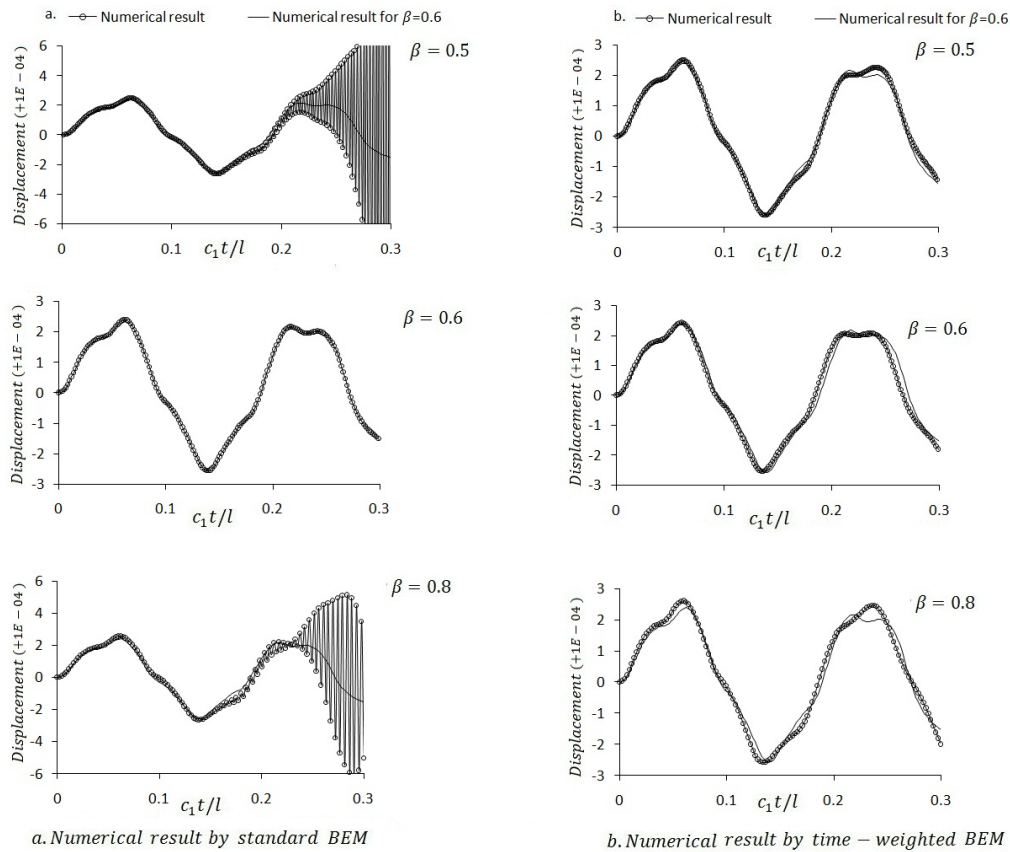


Fig. 5 Horizontal dynamic displacement

## Conclusion

In this paper the time domain BEM formulation for three dimensional problems are revisited. This study proposes a technique to resolve the inconsistent results prevalent when applying the standard time domain BEM to 3D problems. A time weighted formulation of BEM was described and used to improve the stability of the standard time domain BEM. The proposed approach has a significant advantage over the existing approaches. The approach discussed could be used as a tool for stabilizing unstable results subsequent to the standard BEM. Indeed it is not required to solve the problem with another time consuming formulation. The proposed approach has been tested for 3D transient elastodynamic problems. Two examples are investigated for several time steps with both proposed approach and standard form. The following conclusions could be inferred from the numerical analyses presented here:

1. The results show that the proposed procedure can improve stability. Moreover the time weighted method could be preferred to standard method where the time marching process is likely to present instabilities in applications

2. The proposed approach could be used as a stabilizing method subsequent to the conventional BEM.
3. The proposed approach can easily be implemented into the existing computer codes.

## References

- [1] Manolis GD and Beskos DE, Boundary Element Methods in Elastodynamics (Unwin Hyman, London, 1988).
- [2] Dominguez J, Boundary Elements in Dynamics (Computational Mechanics Publications-Elsevier Applied Science, Southampton,1993).
- [3] Peirce A, Siebrits E, Stability analysis and design of time-stepping schemes for general elastodynamic boundary element models, Int J Num Meth Eng 40 (1997) 319-342.
- [4] Siebrits E, Peirce AP, Implementation and application of elastodynamic boundary element discretizations with improved stability properties, Eng Comput 14 (1997) 669–95.
- [5] Yu G, Mansur WJ, Carrer JAM, The linear  $\theta$  method for 2-Delastodynamic BE analysis, Comput Mech 24 (1999) 82–9.

- [6] Yu G, Mansur WJ, Carrer JAM, Gong L, A linear  $\theta$  method applied to 2D time-domain BEM, *Commun Numer Meth Eng* 14 (1998) 1171–9.
- [7] Yu G, Mansur WJ, Carrer JAM, Lie ST, A more stable scheme for BEM/FEM coupling applied to two-dimensional elastodynamics, *Comput Struct* 79 (2001) 811–823
- [8] Soares Jr D, Mansur WJ, An efficient stabilized boundary element formulation for 2D time-domain acoustics and elastodynamics, *Comput Mech* 40 (2007) 355–365
- [9] Soares Jr D, Numerical modelling of acoustic-elastodynamic coupled problems by stabilized boundary element techniques, *Comput Mech* 42 (2008) 787–802.
- [10] Soares Jr D, Mansur WJ, An efficient time-truncated boundary element formulation applied to the solution of the two-dimensional scalar wave equation, *Eng Anal Bound Elem* 33 (2009) 43-53
- [11] Yu G, Mansur WJ, Carrer JAM, Gong L, Time weighting in time domain BEM, *Eng Anal Bound Elem* 22 (1998) 175–181
- [12] Yu G, Mansur WJ, Carrer JAM, Gong L, Stability of Galerkin and collocation time domain boundary element methods as applied to the scalar wave equation, *Comput Struct* 74 (2000) 495–506
- [13] Maier G, Diligenti M, Carini A, A variational approach to boundary element elastodynamic analysis and extension to multidomain problems, *Comp Method Appl Mech Eng* 92 (1991) 193–213.
- [14] Kamali Yazdi A, Omidvar B, Rahimian M, Improving the stability of time domain dual boundary element method for three dimensional fracture problems: A time weighting approach, *Eng Anal Bound Elem* 35 (2011) 1142-48
- [15] Aliabadi MH, *The Boundary Element Method, Applications in Solids and Structures*, Wiley: Chichester, 2002

# Sensitivity of Blood Serum Uric Acid Concentration in HGPRT Deficiency to Initial Conditions of the Purine Metabolism Pathway

Jack K. Horner  
PO Box 266  
Los Alamos NM 87544 USA  
email: jhorner@cybermesa.com

## Abstract

*Purine metabolism is a fundamental component of the production of nucleotides, the building blocks of DNA and RNA. In this pathway, hypoxanthine-guanine phosphoribosyltransferase (HGPRT) catalyzes the conversion of phosphoribosylpyrophosphate to inosine monophosphate and to phosphorylated guanosine. Mild HGPRT deficiency can result in gout and mild hyperuricemia. More severe HGPRT deficiency can result in Lesch-Nyhan syndrome, characterized by spasticity, choreoathetosis, mental retardation, self-mutilation, renal failure, and in the most severe cases, death. For all but the simplest biochemical networks, empirical surveys of system behavior are intractable due to the large size of the state space, and simulation is the only general system-characterization method available. Here I present a simulator-oriented analysis of the sensitivity of blood serum uric acid concentration in HGPRT deficiency to initial conditions of the purine metabolism pathway. The results are consistent with clinical observations.*

**Keywords:** purine metabolism, Lesch-Nyhan syndrome, HGPRT deficiency, S-system

## 1.0 Introduction

Purine metabolism (see Figure 1) is a fundamental component of the production of nucleotides, the building blocks of DNA and RNA. In this pathway, hypoxanthine-guanine phosphoribosyltransferase (HGPRT) catalyzes the conversion of phosphoribosylpyrophosphate (PRPP) to inosine monophosphate (IMP) and to phosphorylated guanosine. Mild HGPRT deficiency can result in gout and mild hyperuricemia. More severe HGPRT deficiency can result in Lesch-Nyhan syndrome (LNS), characterized by spasticity, choreoathetosis, mental retardation, renal failure, self-mutilation, and in the most severe cases, death ([11]).

Can we systematically survey the sensitivity of serum uric acid concentration in HGPRT deficiency to initial conditions in the purine metabolism pathway? One can perform sensitivity studies that vary one quantity at a time, observing the effect of that variation on the system trajectory. This method, however, cannot systematically survey the effects of general variation in system state space of interest unless the trajectories are linear in each of the variables; in highly nonlinear systems, couplings among system variables can prohibit obtaining usable information from the variable-at-a-time variation method.



**Table 1. Purine metabolites and associated simulator variable names for Figure 1 (adapted from [1]).**

Simulator variable name	Biochemical name	Abbreviated name (in Figure 1)	Nominal initial concentration, microMolar
X1	Phosphoribosylpyrophosphate	PRPP	5
X2	Inosine_monophosphate	IMP	100
X3	Adenylosuccinate	S-AMP	0.2
X4	Adenosine	Ado	2500
X4	Adenosine_monophosphate	AMP	2500
X4	Adenosine_diphosphate	ADP	2500
X4	Adenosine_triphosphate	ATP	2500
X5	S-adenosyl-L-methionine	SAM	4
X6	Adenine	Ade	1
X7	Xanthosine_monophosphate	XMP	25
X8	Guanosine_monophosphate	GMP	400
X8	Guanosine_diphosphate	GDP	400
X8	Guanosine_triphosphate	GTP	400
X9	Deoxyadenosine	dAdo	6
X9	Deoxyadenosine_monophosphate	dAMP	6
X9	Deoxyadenosine_diphosphate	dADP	6
X9	Deoxyadenosine_triphosphate	dATP	6
X10	Deoxyguanosine_monophosphate	dGMP	3
X10	Deoxyguanosine_diphosphate	dGDP	3
X10	Deoxyguanosine_triphosphate	dGTP	3
X11	Ribonucleic_acid	RNA	28600
X12	Deoxyribonucleic_acid	DNA	5160
X13	Hypoxanthine	HX	10
X13	Inosine	Ino	10
X13	Deoxyinosine	dIno	10
X14	Xanthine	Xa	5
X15	Guanine	Gua	5
X15	Guanosine	Guo	5
X15	Deoxyguanosine	dGuo	5
X16	Uric_acid	UA	100
X17	Ribose-5-phosphate	R5P	18
X18	Phosphate	P_i	1400

**Table 2. Fluxes and enzymes for Figure 1 (adapted from [1]).**

Abbreviated flux name	Abbreviated enzyme name	Full name of enzyme that catalyzes reaction	E.C. enzyme identifier
vprpps	PRPPS	Phosphoribosylpyrophosphate synthetase	2.7.6.1.
vgprt	HGPRT	Hypoxanthine-guanine phosphoribosyltransferase	2.4.2.8.
vhprt	HGPRT	Hypoxanthine-guanine phosphoribosyltransferase	2.4.2.8.
vaprt	APRT	Adenine phosphoribosyltransferase	2.4.2.7.
vden	ATASE	`De novo synthesis' (Amidophosphoribosyltransferase)	2.4.2.14.
vpyr		`pyrimidine synthesis'	several enzymes
vasuc	ASUC	Adenylosuccinate synthetase	6.3.4.4.
vasli	ASLI	Adenylosuccinate lyase	4.3.2.2.
vimpd	IMPD	IMP dehydrogenase	1.1.1.205.
vgmps	GMPS	GMP synthetase	6.3.4.1.
vampd	AMPD	AMP deaminase	3.5.4.6.
vgmpr	GMPR	GMP reductase	1.6.6.8.
vtrans	MT	`transmethylation pathway' (Protein O-methyltransferase)	2.1.1.24.
vmat	MAT	Methionine adenosyltransferase	2.5.1.6.
vpolyam	SAMD	`Polyamine pathway' (S-adenosylmethionine decarboxylase)	4.1.1.50.
vade		`Adenine oxidation (xanthine oxidase)	1.2.1.37.
Abbreviated flux name	Abbreviated enzyme name	Full name of enzyme that catalyzes reaction	E.C. enzyme identifier

vinuc	5NUC	50-Nucleotidase	3.1.3.5.
vgnuc	5NUC	50-Nucleotidase	3.1.3.5.
varna	RNAP	RNA polymerase (from ATP)	2.7.7.6.
vgrna	RNAP	RNA polymerase (from GTP)	2.7.7.6.
vrnaa	RNAN	RNases (to AMP)	several enzymes
vrnag	RNAN	RNases (to GMP)	several enzymes
vdgnuc	3NUC	50(30) Nucleotidase	3.1.3.31.
vada	ADA	Adenosine deaminase	3.5.4.4.
vdada	ADA	Adenosine deaminase	3.5.4.4.
vadrnr	DRNR	Diribonucleotide reductase	1.17.4.1.
vgdrnr	DRNR	Diribonucleotide reductase	1.17.4.1.
vgua	GUA	Guanine hydrolase	3.5.4.3.
vadna	DNAP	DNA polymerase (from dATP)	2.7.7.7.
vgdna	DNAP	DNA polymerase (from dGTP)	2.7.7.7.
vdnaa	DNAN	DNases (to dAMP)	several enzymes
vdnag	DNAN	DNases (to dGMP)	several enzymes
vhx		`Hypoxanthine excretion'	Non-enzymatic step
vhxd	XD	Xanthine oxidase or xanthine dehydrogenase	1.2.1.37.
vxd	XD	Xanthine oxidase or xanthine dehydrogenase	1.2.1.37.
vx		`Xanthine excretion'	Non-enzymatic step
vua		`Uric acid excretion'	Non-enzymatic step

## 2.0 Method

The HGPRT deficiency model in [4] is an *S-system* ([3]). An *S-system* is a power-law-oriented, finite-difference SODE each of whose dependent variables  $X_i$  is described by an equation of the form

$$\frac{dX_i}{dt} = \alpha_i \prod_j X_j^{g_{i,j}} - \beta_i \prod_j X_j^{h_{i,j}} \quad \text{Eq. 2.1}$$

where

- the left-hand side of Eq. 2.1 is the first derivative of  $X_i$  with respect to time
- $i, j = 1, 2, 3, \dots, N$
- $\{X_i\}$  is the set of real-valued dependent variables of the system
- for any given  $X_i$ , only those independent and dependent variables  $X_j$  that have an action on  $X_i$  are included as factors in the products on the right-hand-side (RHS) of Eq. 2.1. The factors in the first term on the RHS of Eq. 2.1 correspond to just those entities that increase or inhibit the production of  $X_i$ ; the factors in the second term of the RHS of Eq. 2.1 correspond to just those entities that contribute to the consumption of  $X_i$ .
- $\alpha_i, \beta_i > 0$
- $g_{i,j}, h_{i,j}$  are real-valued

There is a natural mapping from a biochemical map,  $K$ , to equations that have the form of Eq 2.1. In particular, let  $K = \langle \{X_k\}, E \rangle$ ,  $E \in \{X_k\} \otimes \{X_k\}$ ,  $k = 1, 2, \dots, N$ , be a directed graph in which each distinct  $X_i \in \{X_k\}$  corresponds to a distinct dependent variable (e.g., the concentration of a distinct chemical species in the map), and  $w \in E$  if and only if  $w = (X_m, X_n)$  is a directed edge in  $K$ ,  $m \neq n = 1, 2, \dots, N$ .

$\alpha_i$  and  $\beta_i$  are called *generalized rate constants* (or just rate constants) for  $X_i$ , and  $g_{i_j}$  and  $h_{i_j}$  are called the *generalized kinetic orders* (or just kinetic orders) for  $X_i$ , on analogy with standard chemical kinetic theory. (Strictly speaking, the S-system appropriation of these terms does not presume anything about the way those terms are used in standard chemical kinetic theory.) The subexpression  $i_j$  indicates the action of  $X_j$  on  $X_i$ .

An S-system has several desirable features, including the fact that it is fully characterized by its rate constants and kinetic orders. Any SODE can be *recast* ([9],[10]) as an S-system without loss of accuracy or precision; the recasting, however, is not in general unique. In addition to biochemical systems, S-systems have been successfully used to model epidemics ([16]), forest diversification, and world dynamics ([15]).

Multi-variate techniques ([7],[12],[13],[14]) can be applied to the HGPRT deficiency model in [1] to help overcome the limitations of single-variable sensitivity analyses. Toward this end, [4] (the PLAS-language ([5]) S-system implementation in [3] of the model in [1]) was parameterized and translated to the *Mathematica* ([6], [17]) language. The resulting implementation ([17]) is an S-system containing 16 state variables and ~145 parameters. In the model, HGPRT deficiency is represented as a factor,  $d$ , whose values lie in the interval [0.01, 0.99], that modulates the kinetic orders of several terms in the system. The larger the value of  $d$ , the more severe the deficiency.

Initial values of the system variables in [4] were randomly drawn from uniform distributions of values of each of the system variables,  $X_i$ . The midpoint,  $\mu_i$ , of the

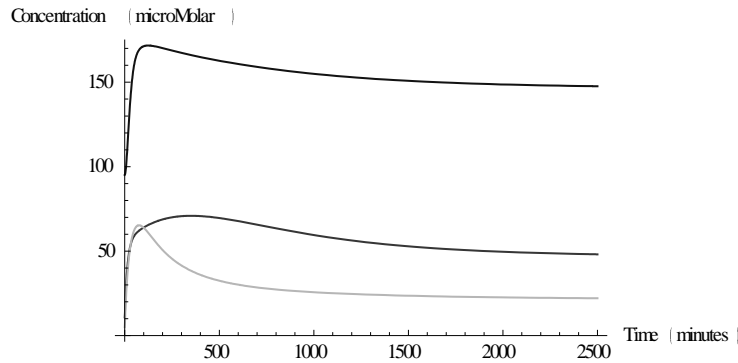
distribution of  $X_i$  was set to the nominal initial value in Table 1 and the range for each  $X_i$  was defined as  $\mu_i \pm 0.1\mu_i$ . (Larger ranges are formally possible but tend to lie outside the calibration range of available data.) The resulting random sampling from the initial-condition space of the model form a set of random vectors of the initial conditions of the system variables. The resulting S-systems were solved using (*Mathematica's* NDSolve function) for 1000 of these random vectors. All software was executed on a Dell Inspiron 545 with an Intel Core2 Quad CPU Q8200 (clocked @ 2.33 GHz) and 8.00 GB RAM, running under the *Windows Vista Home Premium* operating environment.

### 3.0 Results

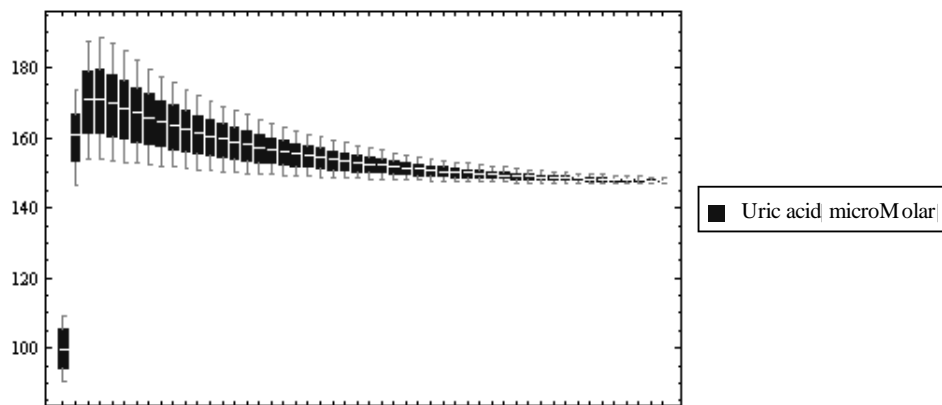
Figure 2 (severe HGPRT deficiency,  $d = 0.99$ ) and 4 ("no" HGPRT deficiency;  $d = 0.01$ ) show nominal hypoxanthine, guanine, and uric acid blood serum concentrations as a function of time. Figure 3 is a box-and-whiskers plot of uric acid concentration under severe HGPRT deficiency, across all randomly drawn initial conditions in the simulation; Figure 5, a corresponding box-and-whiskers plot of uric acid concentration under "no" HGPRT deficiency.

Figures 3 and 5 show that under the conditions described in Section 2.0, the blood serum concentration of uric acid stabilizes at about 1500 minutes after  $t_0$  at ~155 microMolar in severe HGPRT deficiency; at about ~100 microMolar, with "no" HGPRT deficiency. These simulation results are consistent with clinical observations and with the results reported in [3].

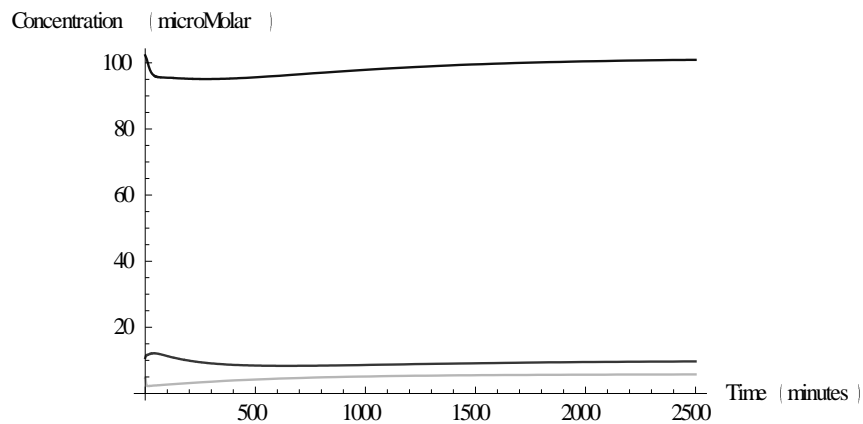
In addition, the figures demonstrate that the initial response (i.e., in the first ~500 minutes) of the purine metabolism network to initial conditions that are not those of the steady state is much less stable in the case of severe HGPRT than in the case of "no" HGPRT. Remarkably, even in the case of severe HGPRT, the network achieves a steady-state (that unfortunately has disastrous physiological consequences).



**Figure 2.** Nominal hypoxanthine (intermediate grey), guanine (lightest grey), and uric acid (darkest grey) blood serum concentrations as a function of time in severe HGPRT deficiency ( $d = 0.99$ ).

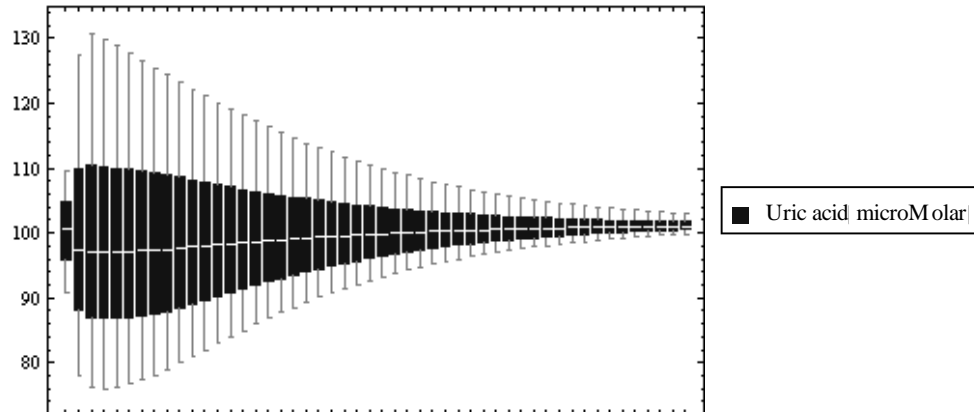


**Figure 3.** Box-and-whiskers plot of blood serum uric acid concentrations (microMolar) as a function of time (0-2500 minutes) in severe HGPRT deficiency ( $d = 0.99$ ). The ends of the blue lines ("whiskers") delimit the range of values across all initial conditions in the study. The solid vertical bars ("boxes") show the mean  $\pm$  one standard deviation of that data. The white horizontal bars in the boxes are the medians of that data.



**Figure 4.** Nominal hypoxanthine (intermediate grey), guanine (lightest grey), and uric acid (darkest grey) blood serum concentrations as a function of time with "no" HGPRT deficiency ( $d = 0.01$ ).





**Figure 5. Box-and-whiskers plot of blood serum uric acid concentration (microMolar), as a function of time (0-2500 minutes), with "no" HGPRT deficiency ( $d = 0.01$ ). The ends of the blue lines ("whiskers") delimit the range of uric acid concentrations generated by all initial conditions in the study. The solid vertical bars ("boxes") show the mean  $\pm$  one standard deviation of that data. The white horizontal bars in the boxes are the respective medians of that data.**

The time-to-solution on the platform described in Section 2.0 was  $\sim 10$  minutes.

In general, there is no guarantee that a random sample of a given size of initial conditions would converge. Figures 3 and 5 strongly suggest, however, that as the system approaches steady state, the convergence is adequate.

## 4.0 References

- [1] Curto R, Voit EO, Sorribas A, and Cascante M. Mathematical models of purine metabolism in man. *Mathematical Biosciences* 151 (1998), 1 - 49.
- [2] Curto R, Voit EO, Sorribas A, and Cascante M. Analysis of abnormalities in purine metabolism leading to gout and to neurological dysfunctions in man. *Biochemistry Journal* 329 (1998), 477 - 487.
- [3] Voit EO. *Computational Analysis of Biochemical Systems*. Cambridge. 2000. See especially Chapter 10.
- [4] File PurineSHGPRT.plc. A PLAS ([5]) script bundled with [3].
- [5] Ferreira AEN. *Power Law Analysis and Simulation (PLAS)* v1.2.120. <http://enzymology.fc.ul.pt/software.htm>. March 2011. Note: as of 16 January 2012, this link is broken; a copy of the tool is available on request from me.
- [6] Wolfram Research. *Mathematica* v8.0.1. 2011.
- [7] Liu JS. *Monte Carlo Strategies in Scientific Computing*. Springer. 2001.

- [8] Chung KL. *A Course in Probability Theory*. Third Edition. Academic Press. 2001.

- [9] Voit EO and Savageau MA. Equivalence between S-systems and volterra systems. *Mathematical Biosciences* 78 (1986), 47-55.

- [10] Voit EO. Recasting nonlinear models as S-systems. *Mathematical and Computer Modelling* 11 (1988), 140-145.

- [11] Harris JC. Disorders of purine and pyrimidine metabolism. In Kliegman RM, Behrman RE, Jenson HB, Stanton BF, eds. *Nelson Textbook of Pediatrics*. 18th ed. Saunders Elsevier. 2007.

- [12] von Neumann J. Various techniques used in connection with random digits. *National Bureau of Standards Applied Mathematics Series* 12 (1951), 36-38.

- [13] Rubinstein RY. *Simulation and the Monte Carlo Method*. Wiley. 1981.

- [14] Robert CP and Casella G. *Monte Carlo Statistical Methods*. Second Edition. Springer. 2004.

- [15] Horner JK. A dynamical implementation of the Stockholm Resilience Centre safe operating space (SOS) model. *Proceedings of the 2010 International Conference on Scientific Computing*. CSREA Press. 2010. 236-242.

- [16] Horner JK. A power-law model of dengue haemorrhagic fever epidemics in Thailand. *Proceedings of the 2004 Conference on Mathematical and Engineering Techniques in Medicine and Biological Sciences*. CSREA Press. 2004. 410-415.

- [17] The source code for the application is available on request from the author.

# Wavelet Galerkin Method for the Solution of Nonlinear Klein-Gordon Equations By Using B-Spline Wavelets

K. Maleknejad<sup>1</sup>, M. Nosrati Sahlan<sup>2</sup>, and A. Ebrahimizadeh<sup>2</sup>

<sup>1</sup>Department of Mathematics, Fandanesht Institute of Higher Education (FDIHE), Saveh, 39157-14774, Iran

<sup>2</sup>Department of Mathematics, Iran University of Science and Technology, Narmak, Tehran 16846-13114, Iran

**Abstract**— This paper aims to obtain approximate solutions of the one-dimensional nonlinear Klein-Gordon equation by employing Cubic B-spline wavelets. Our scheme uses the Galerkin method and approximates the solution in the terms of cubic B-spline scaling and wavelet functions. These wavelets are applied as testing and weighting functions. Because of some properties of these wavelets such as having compact support, vanishing moments and semiorthogonality, operational matrices of these wavelets are very sparse, so implementation of the method is simple and the computational time is low. The results of numerical experiments are presented for showing the accuracy of the method.

**Keywords:** Klein-Gordon equation, cubic B-spline wavelets, Galerkin method, operational matrices

## 1. Introduction

In the present paper, wavelet Galerkin approach based on cubic B-spline wavelets is utilized for the numerical solution of a generalized form of the nonlinear Klein-Gordon equation. For the past years, the Klein-Gordon equation has been the focus of numerous papers that dealt with either proving the existence of solutions or seeking analytical and numerical solutions. Though the equation has been extensively studied, nonetheless it is still a problem of continuing interest because certain physical phenomena are still formulated in terms of the extended form of the Klein-Gordon equation.

The quasilinear Klein-Gordon equation

$$\frac{\partial^2 u(x, t)}{\partial t^2} - \alpha \frac{\partial^2 u(x, t)}{\partial x^2} + \beta u(x, t) - \gamma u(x, t)^3 = 0,$$

and the nonlinear Klein-Gordon equation

$$\frac{\partial^2 u(x, t)}{\partial t^2} - \alpha \frac{\partial^2 u(x, t)}{\partial x^2} + f(u(x, t)) = 0,$$

are used to model many nonlinear phenomena [1], including the propagation of dislocations in crystals and the behavior of elementary particles and the propagation of fluxons in Josephson junctions. The function  $f(u(x, t))$  takes many forms such as:

$\sin(u)$ : Sine-Gordon equation,

$\sinh(u)$ : Sinh-Gordon equation,

$e^u$ : Liouville equation,

$e^u + e^{-2u}$ : Dodd-Bullough-Mikhailov equation,

$e^{-u} + e^{-2u}$ : Tzitzeica-Dodd-Bullough equation.

Also  $f(u(x, t))$  appears as a polynomial such as  $f(u) = bu - ku^n$ .

In this work we consider the nonlinear one-dimensional Klein-Gordon equation

$$\frac{\partial^2 u(x, t)}{\partial t^2} + \alpha \frac{\partial^2 u(x, t)}{\partial x^2} + \beta u(x, t) + g(u(x, t)) = F(x, t),$$

$$(x, t) \in [a, b] \times [0, T] \quad (1)$$

with specified initial and boundary conditions given by

$$u(x, 0) = f_0(x), \quad u_t(x, 0) = f_1(x), \quad (2)$$

$$u(a, t) = \phi_0(t), \quad u(b, t) = \phi_1(t). \quad (3)$$

The function  $g(u(x, t))$  is a nonlinear function in  $u$ ,  $a$  and  $b$  are real constants, and  $F(x, t)$  is a known function.

Equation (1) is one of the important mathematical models in quantum mechanics, it occurs in relativistic physics as a model of dispersive phenomena. Information about the literature of such applications is given in [2]- [3] and the other references therein. There are numerous papers dealing with the existence, uniqueness of the smooth and weak solutions of equation (1)- (3), and with the numerical solutions using radial basis functions, finite difference, finite element or collocation methods such as in [3]- [6]. Wazwaz [7] uses the tanh and sine-cosine methods for compact and noncompact solutions of the nonlinear Klein-Gordon equation. Chowdhury and Hashim [8] employed the homotopy perturbation method to obtain approximate solutions of the Klein-Gordon and Sine-Gordon equations. Deeba and Khuri [2] presented a decomposition scheme for obtaining approximate solutions to the Klein-Gordon equation. Wong et al. [3] presented a fully implicit and discrete energy conserving finite difference scheme for the solution of an initial-boundary value problem of the nonlinear Klein-Gordon equation and a theoretical analysis was performed. In [6], Yücel obtained approximate analytical solution for the Sine-Gordon equation using the homotopy analysis method. Ming and Guo [5] utilized a Fourier collocation method for solving the nonlinear Klein-Gordon equation. Most recently, Dehghan and Shokri [4] proposed a numerical scheme to solve the one-dimensional nonlinear Klein-Gordon equation with quadratic and cubic

nonlinearity using the collocation points and approximating the solution by Thin Plate Splines radial basis functions. Rashidinia and Mohammadi [9] derived a three-level spline-difference scheme to solve the one dimensional Klein-Gordon equation which is based on using the finite difference approximation for the time derivative and the spline approximation for the second-order spatial derivative. Several second order finite difference schemes have been presented (see [10] and references therein). We note that alternative approaches using spectral and pseudo-spectral methods have recently been presented [11]. Four finite difference schemes for approximating the nonlinear Klein-Gordon equation were discussed in [12].

In this work we reduce the problem to a set of algebraic equations by expanding the unknown function as cubic B-spline scaling and wavelet functions specially constructed on bounded interval, with unknown coefficients. The operational matrix of derivative is given. This matrix together with the cubic B-spline scaling and wavelet functions are then utilized to evaluate the unknown coefficients.

## 2. B-spline Scaling and Wavelet Functions

The general theory and basic concepts of the wavelet theory and MRA is given in [13]-[18]. B-spline wavelets can be used to expand any function in  $L^2(R)$ . These functions are defined on the entire real lines, so that they could be outside of the domain of the problem. This behavior may require an explicit enforcement of the boundary conditions. In order to avoid this occurrence, semiorthogonal compactly supported spline wavelets, constructed for the bounded interval  $[0, 1]$ , have been taken into account in this paper. These wavelets satisfy all the properties verified by the usual wavelets on the real line.

Let  $B_m(x)$  be the  $m$ th-order cardinal B-spline function [19]:

$$B_m(x) = \int_0^1 B_{m-1}(x-t)dt, \quad m \geq 2,$$

where  $B_1(x) = \chi_{[0,1]}(x)$ , the characteristic function of  $[0, 1]$  and  $Supp[B_m(x)] = [0, m]$ .

It is well known that these compactly supported functions generate an MRA [17] with two scale relations

$$B_m(x) = \sum_{k=0}^m 2^{-m+1} \binom{m}{k} B_m(2x-k).$$

Then the corresponding  $m$ th-order cardinal semiorthogonal compactly supported B-wavelet function  $\psi_m(x)$  is given by

$$\psi_m(x) = \sum_{k=0}^{3m-2} q_k B_m(2x-k),$$

with

$$q_k = \frac{(-1)^k}{2^{m-1}} \sum_{l=0}^m \binom{m}{l} B_{2m}(k-l+1).$$

Some of the important properties relevant to the present work are given below:

1) Vanishing moments: A wavelet  $\psi(x)$  is said to have a vanishing moments of order  $m$  if

$$\int_{-\infty}^{\infty} x^p \psi(x) dx = 0; \quad p = 0, 1, \dots, m-1.$$

All wavelets must satisfy the above condition for  $p = 0$ . Cubic B-spline wavelet has 4 vanishing moments. That is

$$\int_{-\infty}^{\infty} x^p \psi_4(x) dx = 0; \quad p = 0, 1, 2, 3.$$

2) Semiorthogonality: The wavelets  $\psi_{j,k}$  form an semiorthogonal basis if

$$\langle \psi_{j,k}, \psi_{s,i} \rangle = 0; \quad j \neq s; \quad \forall j, k, s, i \in \mathbb{Z}.$$

Cubic B-spline wavelet are semiorthogonal.

### 2.1 Cubic B-spline Scaling and Wavelet Functions

When semiorthogonal wavelets are constructed from B-splines of order  $m$ , the lowest octave level  $j = j_0$  is determined in [20] by

$$2^j \geq 2m - 1,$$

so as to give a minimum of one complete wavelet on the interval  $[0, 1]$ . Cubic B-spline scaling function  $B_4(x)$  is given by:

$$\varphi_4(x) = \begin{cases} \frac{1}{6}x^3 & x \in [0, 1) \\ \frac{1}{6}(-3x^3 + 12x^2 - 12x + 4) & x \in [1, 2) \\ \frac{1}{6}(3x^3 - 24x^2 + 60x - 44) & x \in [2, 3) \\ \frac{1}{6}(4-x)^3 & x \in [3, 4) \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

and its two-scale dilation equation defined as follows:

$$\varphi_4(x) = \sum_{k=0}^4 \frac{1}{8} \binom{4}{k} \varphi_4(2x-k).$$

Fig. 1 is helpful to get a geometric understanding of two-scale relation of cubic B-spline scaling function.

In this section, the scaling functions used in this work, for  $j_0 = j = 3$  and  $m = 4$ , are reported :

#### Boundary scalings

Three left boundary cubic B-spline scaling functions are constructed by the following formula:

$$\varphi_{4,k}^{(3)}(x) = \varphi_4^{(3)}(8x-k) \cdot \chi_{[0,1]}(x),$$

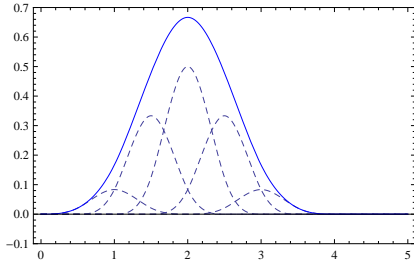


Fig. 1: Two scale relation of Cubic B-spline scaling function

$$k = -3, -2, -1, \tag{5}$$

and for other levels of  $j$ , we have:

$$\begin{aligned} \varphi_{4,k}^{(j)}(x) &= \varphi_{4,k}^{(3)}(2^{j-3}x), \\ k &= -3, -2, -1, \quad j = 4, 5, \dots \end{aligned} \tag{6}$$

left and right boundary scaling functions are symmetric with respect to 0, so we have:

$$\varphi_{4,5}^{(3)}(x) = \varphi_{4,-1}^{(3)}(1-x), \tag{7}$$

$$\varphi_{4,6}^{(3)}(x) = \varphi_{4,-2}^{(3)}(1-x), \tag{8}$$

$$\varphi_{4,7}^{(3)}(x) = \varphi_{4,-3}^{(3)}(1-x), \tag{9}$$

and for other levels of  $j$ , we have:

$$\begin{aligned} \varphi_{4,2^j-k-3}^{(j)}(x) &= \varphi_{4,k}^{(3)}(2^{j-3}x), \\ k &= -3, -2, -1, \quad j = 4, 5, \dots \end{aligned} \tag{10}$$

**Inner scalings**

Five inner cubic B-spline scaling functions are constructed by the following formula:

$$\begin{aligned} \varphi_{4,k}^{(3)}(x) &= \varphi_4^{(3)}(8x - k) \cdot \chi_{[0,1]}(x), \\ k &= 0, 1, 2, 3, 4, 5, \end{aligned} \tag{11}$$

and for other levels of  $j$ , we get:

$$\begin{aligned} \varphi_{4,k}^{(j)}(x) &= \varphi_{4,k}^{(3)}(2^{j-3}x - k), \\ k &= 0, 1, \dots, 2^j - 4, \quad j = 4, 5, \dots \end{aligned} \tag{12}$$

Two scale relation equation for cubic B-spline wavelet is given by:

$$\psi_4(x) = \sum_{k=0}^{10} \frac{(-1)^k}{8} \sum_{l=0}^4 \binom{4}{l} \varphi_8(k-l+1) \varphi_4(2x-k). \tag{13}$$

Other inner and boundary wavelets are made similarly [20]. Fig. 3 is helpful to get a geometric understanding of inner and boundary cubic B-spline wavelets.

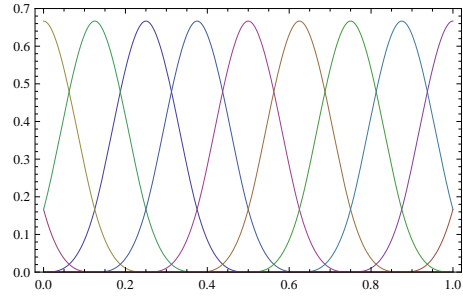


Fig. 2: Inner and boundary cubic B-spline scaling functions

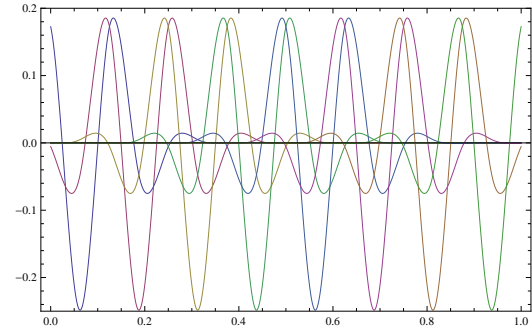


Fig. 3: Cubic B-spline inner and boundary wavelet

**3. Function approximation**

A function  $f(x) \in L^2(\mathbb{R})$  defined over  $[0, 1]$  may be approximated by cubic B-spline wavelets as:

$$f(x) = \sum_{i=-3}^{2^{j_0}-1} c_{j_0,i} \varphi_{j_0,i}^{(3)}(x) + \sum_{j=j_0}^{\infty} \sum_{k=-3}^{2^j-4} d_{j,k} \psi_{4,k}^{(j)}, \tag{14}$$

where  $\varphi_{j_0,i}$  and  $\psi_{4,k}^{(j)}$  are scaling and wavelets functions, respectively. If the infinite series in (14) is truncated, then it can be written as:

$$\begin{aligned} f(x) &\simeq \sum_{i=-3}^{i=2^{j_0}-1} c_{j_0,i} \varphi_{j_0,i}^{(3)}(x) + \sum_{j=j_0}^{j_u} \sum_{k=-3}^{2^j-4} d_{j,k} \psi_{4,k}^{(j)}(x) \\ &= C^T \Upsilon(x) = \Upsilon^T(x)C, \end{aligned} \tag{15}$$

where  $C$  and  $\Upsilon$  are  $2^{(j_u+1)} + 3$  column vectors given by

$$C = (c_{j_0,-3}, \dots, c_{j_0,2^{j_0}-1}, d_{j_0,-3}, \dots, d_{j_u,2^{j_u}-4})^T, \tag{16}$$

$$\Upsilon = (\varphi_{4,-3}^{(3)}, \dots, \varphi_{4,7}^{(3)}, \psi_{4,-3}^{(3)}, \dots, \psi_{j_u,2^{j_u}-4}^{(3)})^T, \tag{17}$$

with

$$c_{j_0,i} = \int_0^1 f(x) \tilde{\varphi}_{j_0,i}^{(3)}(x) dx, \quad i = -3, \dots, 2^{j_0} - 1,$$

$$d_{j,k} = \int_0^1 f(x) \tilde{\psi}_{4,k}^{(j)}(x) dx,$$

$$j = j_0, \dots, j_u, \quad k = -3, \dots, 2^j - 4,$$

and  $\tilde{\varphi}_{j_0,i}^{(3)}$  and  $\tilde{\psi}_{4,k}^{(j)}$  are dual functions of  $\varphi_{j_0,i}^{(3)}$ ,  $i = -3, \dots, 2^{j_0} - 1$  and  $\psi_{4,k}^{(j)}$ ,  $j = j_0, \dots, j_u$ ,  $k = -3, \dots, 2^j - 4$ , respectively. These can be obtained by linear combinations of  $\varphi_{j_0,i}^{(3)}$  and  $\psi_{4,k}^{(j)}$ . Let

$$\varphi(x) = \left( \varphi_{4,-3}^{(3)}(x), \varphi_{4,-2}^{(3)}(x), \dots, \varphi_{4,7}^{(3)}(x) \right)^T, \quad (18)$$

$$\psi(x) = \left( \psi_{4,-3}^{(3)}(x), \dots, \psi_{4,4}^{(3)}(x), \dots, \psi_{4,2^{j_u}-4}^{(j_u)}(x) \right)^T. \quad (19)$$

Using (5)-(10) and (18) we get

$$\int_0^1 \varphi(x)\varphi^T(x)dx = P_1, \quad (20)$$

similarly for cubic B-spline wavelets, product matrix is:

$$\int_0^1 \psi(x)\psi^T(x)dx = P_2, \quad (21)$$

where  $P_1$  and  $P_2$  are  $11 \times 11$  and  $(2^{j_u+1} - 8) \times (2^{j_u+1} - 8)$  matrices, respectively. Suppose  $\tilde{\varphi}(x)$  and  $\tilde{\psi}(x)$  are the dual functions of  $\varphi(x)$  and  $\psi(x)$ , respectively, given by

$$\tilde{\varphi}(x) = \left( \tilde{\varphi}_{4,-3}^{(3)}(x), \tilde{\varphi}_{4,-2}^{(3)}(x), \dots, \tilde{\varphi}_{4,7}^{(3)}(x) \right)^T, \quad (22)$$

$$\tilde{\psi}(x) = \left( \tilde{\psi}_{4,-3}^{(3)}(x), \dots, \tilde{\psi}_{4,4}^{(3)}(x), \dots, \tilde{\psi}_{4,2^{j_u}-4}^{(j_u)}(x) \right)^T. \quad (23)$$

Using (18)-(19), (22) and (23) we have

$$\int_0^1 \tilde{\varphi}(x)\varphi^T(x)dx = I_{11},$$

$$\int_0^1 \tilde{\psi}(x)\psi^T(x)dx = I_{2^{j_u+1}-8}.$$

where  $I_{11}$  and  $I_{2^{j_u+1}-8}$  are  $11 \times 11$  and  $(2^{j_u+1} - 8) \times (2^{j_u+1} - 8)$  identity matrices, respectively.

Thus we get

$$\tilde{\varphi} = P_1^{-1}\varphi, \quad \tilde{\psi} = P_2^{-1}\psi.$$

Thus, the dual function of  $\Upsilon$ , that is:

$$\tilde{\Upsilon} = \left( \tilde{\varphi}_{j_0,-3}^{(3)}, \dots, \tilde{\varphi}_{j_0,2^{j_0}-1}^{(3)}, \tilde{\psi}_{j_0,-3}, \dots, \tilde{\psi}_{j_u,2^{j_u}-4} \right)^T, \quad (24)$$

can be constructed as:

$$\tilde{\Upsilon}(x) = P^{-1}\Upsilon(x), \quad (25)$$

where

$$P = \begin{pmatrix} P_1 & \\ & P_2 \end{pmatrix}.$$

**Theorem 1** ([20]) : We assume that  $f \in C^4[0,1]$  is represented by cubic B-spline wavelets as equation (15), where  $\psi$  has 4 vanishing moments, then

$$|d_{j,k}| \leq \alpha\beta \frac{2^{-5j}}{4!}, \quad (26)$$

where

$$\alpha = \max |f^{(4)}(t)|_{t \in [0,1]}$$

and

$$\beta = \int_0^1 |x^4 \tilde{\psi}_4(x)| dx \quad \square$$

**Theorem 2** ([20]): Consider the previous theorem assume that  $e_j(x)$  be error of approximation in  $V_j$ , then

$$|e_j(x)| = O(2^{-4j}). \quad \square$$

Thus, order of error depend on the level  $j$ . Obviously, for larger level of  $j$ , the error of approximation will be smaller.

### 3.1 Operational Matrix of Derivative

The differentiation of vectors  $\Upsilon$  in equation (17) can be expressed as:

$$\Upsilon'(x) = D\Upsilon(x), \quad (27)$$

where  $D$  is  $(2^{j_u+1} + 3) \times (2^{j_u+1} + 3)$  operational matrix of derivative for cubic B-spline scaling and wavelet functions on  $[0, 1]$ . The matrix  $D$  can be obtained by considering:

$$D = \int_0^1 \Upsilon'(x)\tilde{\Upsilon}^T(x)dx = \left( \int_0^1 \Upsilon'(x)\Upsilon^T(x)dx \right) (P^{-1})^T = E(P^{-1})^T, \quad (28)$$

where  $E$  is  $(2^{j_u+1} + 3)$ -dimensional square matrix defined as follows:

$$E = \int_0^1 \Upsilon'(x)\Upsilon^T(x)dx = \begin{pmatrix} E_1 & E_2 \\ E_3 & E_4 \end{pmatrix},$$

with

$$E_1 = \left\langle \varphi_{j_0,i}^{(3)}, \varphi_{j_0,r}^{(3)} \right\rangle = \left( \int_0^1 \varphi_{j_0,i}^{(3)} \cdot \varphi_{j_0,r}^{(3)} dx \right)_{i,r},$$

$$E_2 = \left\langle \psi_{j,k}^{(3)}, \varphi_{j_0,r}^{(3)} \right\rangle = \left( \int_0^1 \psi_{j,k}^{(3)} \cdot \varphi_{j_0,r}^{(3)} dx \right)_{j,k,r},$$

$$E_3 = \left\langle \varphi_{j_0,i}^{(3)}, \psi_{l,s}^{(3)} \right\rangle = \left( \int_0^1 \varphi_{j_0,i}^{(3)} \cdot \psi_{l,s}^{(3)} dx \right)_{i,l,s},$$

$$E_4 = \left\langle \psi_{j,k}^{(3)}, \psi_{l,s}^{(3)} \right\rangle = \left( \int_0^1 \psi_{j,k}^{(3)} \cdot \psi_{l,s}^{(3)} dx \right)_{j,k,l,s},$$

and the subscripts  $i, r, k, j, l$  and  $s$  assume values as given below:

$$i, r = -3, -2, \dots, 2^{j_0}-1,$$

$$k, s = -3, \dots, 2^j - 2,$$

$$j, l = j_0, \dots, j_u.$$

### 4. Numerical Method for the Klein-Gordon Equation

In this section, we present new efficient and accurate numerical methods for the Klein-Gordon equation. Consider the one-dimensional nonlinear Klein-Gordon equation:

$$\frac{\partial^2 u(x, t)}{\partial t^2} + \alpha \frac{\partial^2 u(x, t)}{\partial x^2} + \beta u(x, t) + g(u(x, t)) = F(x, t),$$

$$(x, t) \in [a, b] \times [0, T] \tag{29}$$

with specified initial and boundary conditions given by

$$u(x, 0) = f_0(x), \quad u_t(x, 0) = f_1(x), \tag{30}$$

$$u(a, t) = \phi_0(t), \quad u(b, t) = \phi_1(t). \tag{31}$$

By expanding the unknown functions  $u(x, t)$  and  $g(u(x, t))$  in terms of cubic B-spline scaling and wavelet functions, we get:

$$u(x, t) = \Upsilon^T(t)C\Upsilon(x), \tag{32}$$

$$g(u(x, t)) = Z(x, t) = \Upsilon^T(t)A\Upsilon(x), \tag{33}$$

where  $C$  and  $A$  are  $(2^{j_u+1} + 3)$ -dimensional square coefficient matrices defined as follows:

$$C = \left( \int_0^1 \int_0^1 u(x, t)\Upsilon_i(t)\Upsilon_j(x)dt dx \right)_{i,j},$$

$$A = \left( \int_0^1 \int_0^1 Z(x, t)\Upsilon_i(t)\Upsilon_j(x)dt dx \right)_{i,j},$$

$$i, j = 1, 2, \dots, 2^{j_u+1} + 3.$$

By applying equations (27) and (32), we get:

$$\frac{\partial^2 u(x, t)}{\partial x^2} = \Upsilon^T(t)CD^2\Upsilon(x), \tag{34}$$

and

$$\frac{\partial^2 u(x, t)}{\partial t^2} = \Upsilon^T(t)(D^2)^T C\Upsilon(x). \tag{35}$$

Substituting equations (32)-(35) in equation (29), we have the following system of equations:

$$\begin{aligned} &\Upsilon^T(t)(D^2)^T C\Upsilon(x) + \alpha\Upsilon^T(t)CD^2\Upsilon(x) \\ &+ \beta\Upsilon^T(t)C\Upsilon(x) + \Upsilon^T(t)A\Upsilon(x) = F(x, t), \end{aligned} \tag{36}$$

for solving the current system, Galerkin method is used, cubic B-spline scaling and wavelet functions are used as weighting functions.

Multiplying equations (36) in  $\tilde{\Upsilon}^T(x)$  and integrating 0 to 1,

$$\begin{aligned} &\Upsilon^T(t)(D^2)^T C + \alpha\Upsilon^T(t)CD^2 \\ &+ \beta\Upsilon^T(t)C + \Upsilon^T(t)A = \int_0^1 F(x, t)\tilde{\Upsilon}^T(x)dx, \end{aligned} \tag{37}$$

now, multiplying equations (37) in  $\tilde{\Upsilon}(t)$  and integrating 0 to 1, we get:

$$\begin{aligned} &(D^2)^T C + \alpha CD^2 + \beta C + A \\ &= \int_0^1 \left( \int_0^1 F(x, t)\tilde{\Upsilon}^T(x)dx \right) \tilde{\Upsilon}(t)dt, \end{aligned} \tag{38}$$

Using the initial conditions (30) we obtain:

$$\begin{aligned} u(x, 0) &= \Upsilon^T(0)C\Upsilon(x) = f_0(x), \\ u_t(x, 0) &= \Upsilon^T(0)D^T C\Upsilon(x) = f_1(x), \end{aligned} \tag{39}$$

Also the Dirichlet boundary condition (31) gives:

$$\begin{aligned} u(a, t) &= \Upsilon^T(t)C\Upsilon(a) = \phi_0(t), \\ u(b, t) &= \Upsilon^T(t)C\Upsilon(b) = \phi_1(t), \end{aligned} \tag{40}$$

Multiplying equations (39) in  $\tilde{\Upsilon}^T(x)$  and integrating from 0 to 1;

$$\begin{aligned} \int_0^1 u(x, 0)\tilde{\Upsilon}^T(x)dx &= \Upsilon^T(0)C = \int_0^1 f_0(x)\tilde{\Upsilon}^T(x)dx, \\ \int_0^1 u_t(x, 0)\tilde{\Upsilon}^T(x)dx &= \Upsilon^T(0)D^T C = \int_0^1 f_1(x)\tilde{\Upsilon}^T(x)dx, \end{aligned} \tag{41}$$

and multiplying equations (40) in  $\tilde{\Upsilon}(t)$  and integrating from 0 to 1;

$$\begin{aligned} \int_0^1 u(a, t)\tilde{\Upsilon}(t)dt &= C\Upsilon(a) = \int_0^1 \phi_0(t)\tilde{\Upsilon}(t)dt, \\ \int_0^1 u(b, t)\tilde{\Upsilon}(t)dt &= C\Upsilon(b) = \int_0^1 \phi_1(t)\tilde{\Upsilon}(t)dt, \end{aligned} \tag{42}$$

Equations (38), together with equations (41) and (42) give a system of nonlinear equations with  $3 \times (2^{j_u+1} + 3) \times (2^{j_u+1} + 3)$  equations and unknowns, which can be solved to find the unknown coefficient  $C_{i,j}$  and  $A_{i,j}$ .

### 5. Numerical Examples

In this section, we will apply cubic B-spline wavelets via Galerkin method to obtain numerical solutions for certain cases of the generalized form of the Klein-Gordon equation. Some examples are analyzed to illustrate the viability and efficiency of the proposed method.

*Example 1:* [21] Consider the following special case of equation (1):

$$\frac{\partial^2 u(x, t)}{\partial t^2} + \frac{\partial^2 u(x, t)}{\partial x^2} + 2u(x, t) = 2 \sin(x) \sin(t),$$

subject to the following initial and boundary conditions

$$\begin{aligned} u(x, 0) &= 0, \quad u_t(x, 0) = \sin(x), \\ u(0, t) &= 0, \quad u\left(\frac{\pi}{2}, t\right) = \sin(t), \end{aligned}$$

where the exact solution is  $u(x, t) = \sin(x) \sin(t)$ . The solution for  $u(x, t)$  is obtained by the method in Section 4 at the octave level  $j_0 = 3$  and at the levels  $j_u = 4$  and 5.

In Table 1, we present  $L_2$ ,  $L_\infty$  and RMS errors of numerical solutions of Example 1 in some arbitrary points.

Example 2: [22] Consider the following special case of

Table 1: Errors of numerical solution of example 1 for  $0 \leq x \leq \frac{\pi}{2}$

$t$	$\ e\ _{L_\infty}$	$\ e\ _{L_2}$	RMS
$j_u = 4$			
1	$4.71 \times 10^{-8}$	$5.72 \times 10^{-7}$	$7.36 \times 10^{-8}$
2	$3.89 \times 10^{-8}$	$6.63 \times 10^{-7}$	$7.08 \times 10^{-8}$
3	$7.24 \times 10^{-8}$	$5.96 \times 10^{-7}$	$7.74 \times 10^{-8}$
4	$6.31 \times 10^{-8}$	$6.26 \times 10^{-7}$	$5.97 \times 10^{-8}$
5	$4.55 \times 10^{-8}$	$5.06 \times 10^{-7}$	$6.02 \times 10^{-8}$
$j_u = 5$			
1	$5.92 \times 10^{-14}$	$3.14 \times 10^{-12}$	$2.39 \times 10^{-14}$
2	$7.19 \times 10^{-14}$	$5.38 \times 10^{-12}$	$6.01 \times 10^{-14}$
3	$4.79 \times 10^{-14}$	$5.72 \times 10^{-12}$	$2.12 \times 10^{-14}$
4	$4.07 \times 10^{-14}$	$3.66 \times 10^{-12}$	$3.30 \times 10^{-14}$
5	$6.55 \times 10^{-14}$	$7.27 \times 10^{-12}$	$4.52 \times 10^{-14}$

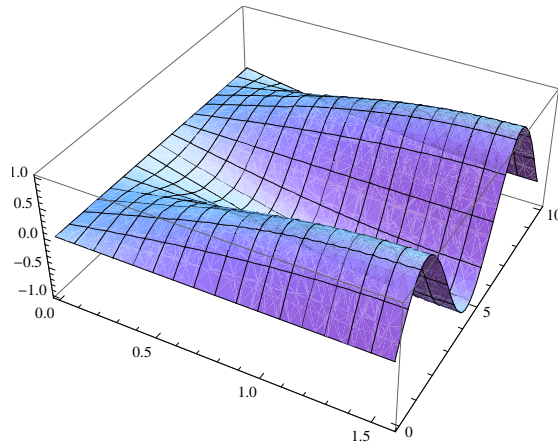


Fig. 4: The space-time graph of estimated solution of example 1

equation (1):

$$\frac{\partial^2 u(x, t)}{\partial t^2} - \frac{\partial^2 u(x, t)}{\partial x^2} + u^2(x, t) = -x \cos(t) + x^2 \cos^2(t),$$

subject to the following initial conditions

$$\begin{aligned} u(x, 0) &= x, & u_t(x, 0) &= 0, \\ u(0, t) &= 0, & u(1, t) &= \cos(t), \end{aligned}$$

where  $0 \leq x \leq 1$ , and the exact solution is  $u(x, t) = x \cos(t)$ . The solution for  $u(x, t)$  is obtained by the method in Section 4 at the octave level  $j_0 = 3$  and at the levels  $j_u = 4$  and 5. Results are shown in Tables 3 and 4. In Tables 3 and 4, we present  $L_2$  and  $L_\infty$  errors of numerical solutions of Example 2 in some arbitrary points.

Example 3: Consider the following nonlinear Klein-Gordon

Table 2: Errors of numerical solution of example 2 for  $0 \leq x \leq 1$

$t$	$\ e\ _{L_\infty}$	$\ e\ _{L_2}$	RMS
$j_u = 4$			
1	$3.92 \times 10^{-8}$	$2.57 \times 10^{-7}$	$3.77 \times 10^{-8}$
2	$4.37 \times 10^{-8}$	$6.49 \times 10^{-7}$	$4.51 \times 10^{-8}$
3	$8.04 \times 10^{-8}$	$5.83 \times 10^{-7}$	$3.44 \times 10^{-8}$
4	$7.60 \times 10^{-8}$	$1.55 \times 10^{-7}$	$7.72 \times 10^{-8}$
5	$6.32 \times 10^{-8}$	$7.62 \times 10^{-7}$	$8.26 \times 10^{-8}$
$j_u = 5$			
1	$2.25 \times 10^{-14}$	$3.58 \times 10^{-12}$	$4.67 \times 10^{-14}$
2	$8.91 \times 10^{-14}$	$3.37 \times 10^{-12}$	$6.29 \times 10^{-14}$
3	$4.09 \times 10^{-14}$	$7.28 \times 10^{-12}$	$2.64 \times 10^{-14}$
4	$3.27 \times 10^{-14}$	$8.04 \times 10^{-12}$	$5.72 \times 10^{-14}$
5	$4.71 \times 10^{-14}$	$5.73 \times 10^{-12}$	$3.65 \times 10^{-14}$

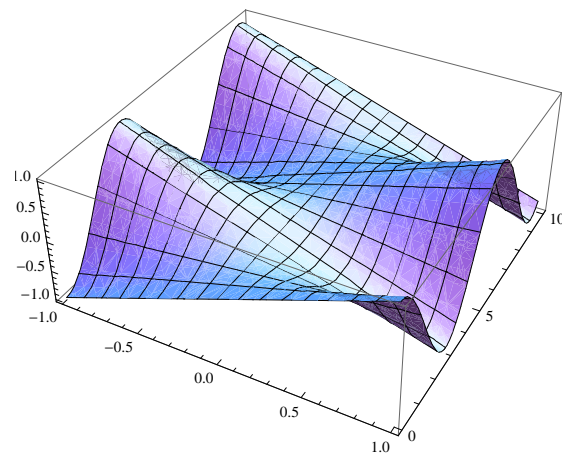


Fig. 5: The space-time graph of estimated solution of example 2

equation:

$$\begin{aligned} \frac{\partial^2 u(x, t)}{\partial t^2} + \frac{\partial^2 u(x, t)}{\partial x^2} - 4u(x, t) + \ln(u(x, t)) \\ = 6xe^{2t} + 3 \ln(x) + 2t, \end{aligned} \tag{43}$$

subject to the following initial conditions

$$\begin{aligned} u(x, 0) &= x^3, & u_t(x, 0) &= 2x^3, \\ u(0, t) &= 0, & u(1, t) &= e^{2t}, \end{aligned}$$

where  $-1 \leq x \leq 1$ , and the exact solution is  $u(x, t) = x^3 \exp(2t)$ . The solution for  $u(x, t)$  is obtained by the method in Section 4 at the octave level  $j_0 = 3$  and at the levels  $j_u = 4$  and 5. Results are shown in Tables 3 and 4. In Tables 3 and 4, we present  $L_2$  and  $L_\infty$  errors of numerical solutions of Example 2 in some arbitrary points. Symbols

Table 3: Errors of numerical solution of example 3 for  $-1 \leq x \leq 1$

$t$	$\ e\ _{L_\infty}$	$\ e\ _{L_2}$	RMS
$j_u = 4$			
1	$6.71 \times 10^{-8}$	$8.22 \times 10^{-7}$	$5.49 \times 10^{-8}$
2	$3.64 \times 10^{-8}$	$5.18 \times 10^{-7}$	$7.43 \times 10^{-8}$
3	$5.11 \times 10^{-8}$	$3.46 \times 10^{-7}$	$2.89 \times 10^{-8}$
4	$2.97 \times 10^{-8}$	$5.14 \times 10^{-7}$	$3.46 \times 10^{-8}$
5	$4.52 \times 10^{-8}$	$4.81 \times 10^{-7}$	$6.55 \times 10^{-8}$
$j_u = 5$			
1	$1.47 \times 10^{-14}$	$2.77 \times 10^{-12}$	$2.09 \times 10^{-14}$
2	$3.27 \times 10^{-14}$	$4.41 \times 10^{-12}$	$4.32 \times 10^{-14}$
3	$2.84 \times 10^{-14}$	$3.15 \times 10^{-12}$	$2.86 \times 10^{-14}$
4	$3.49 \times 10^{-14}$	$4.30 \times 10^{-12}$	$3.93 \times 10^{-14}$
5	$1.84 \times 10^{-14}$	$1.57 \times 10^{-12}$	$3.49 \times 10^{-14}$

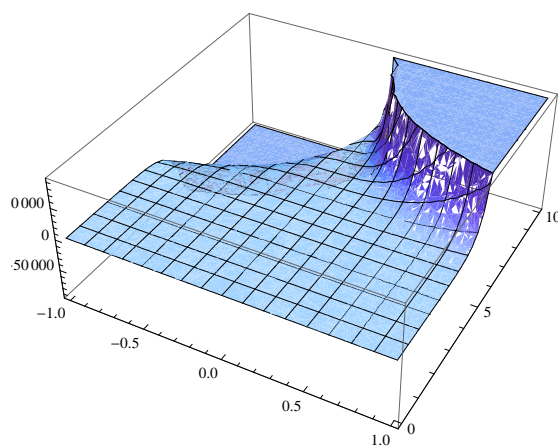


Fig. 6: The space-time graph of estimated solution of example 2

used in Tables defined as follow:

$$e_j = (u_{exact})_j - (u_{approx})_j, \quad e = u_{exact} - u_{approx},$$

$$\|e\|_{L_\infty} = \max_j |e_j|,$$

$$\|e\|_{L_2} = \sqrt{\sum_{j=1}^N |e_j|^2},$$

$$RMS = \frac{\|e_j\|}{\sqrt{N}}.$$

### 6. Conclusions

In this paper we presented a numerical scheme for solving the nonlinear Klein-Gordon equation. The cubic B-spline scaling functions and wavelets were employed as testing and weighting functions. This method can be employed for solving other kinds as differential equations. Because of some properties of these wavelets, the operational matrices are very sparse. Test problems show that the method has high accuracy.

### References

- [1] D. B. Duncan, "Symplectic finite difference approximation of the nonlinear Klein-Gordon equation," *SIAM J. Numer. Anal.*, vol. 34, pp. 1742-1760, 1997.
- [2] E. Deeba and S.A. Khuri, "A decomposition method for solving the nonlinear Klein-Gordon equation," *Journal of Computational Physics*, vol. 124, pp. 442-448, 1996.
- [3] Y. S. Wong, Q. Chang and L. Gong, "An initial-boundary value problem of a nonlinear Klein-Gordon equation," *Applied Mathematics and Computation*, vol. 84, pp. 77-93, 1997.
- [4] M. Dehghan and A. Shokri, "Numerical solution of the nonlinear Klein-Gordon equation using radial basis functions," *Journal of Computational and Applied Mathematics*, vol. 230, pp. 400-410, 2009.
- [5] W. Ming and B.Y. Gue, "Fourier collocation method for solving nonlinear Klein-Gordon equation," *Journal of Computational Physics*, vol. 108, pp. 296-305, 1993.
- [6] U. Yucel, "Homotopy analysis method for the sine-Gordon equation with initial conditions," *Applied Mathematics and Computation*, vol. 203, pp. 387-395, 2008.
- [7] A. M. Wazwaz, "The tanh and sine-cosine methods for compact and noncompact solutions of the nonlinear Klein-Gordon equation," *Applied Mathematics and Computation*, vol. 167, pp. 1179-1195, 2006.
- [8] M. S. H. Chowdhury and I. Hashim, "Application of homotopy-perturbation method to Klein-Gordon and sine-Gordon equations," *Chaos, Solitons and Fractals*, vol. 39, pp. 1928-1935, 2009.
- [9] J. Rashidinia, R. Mohammadi, "Tension spline approach for the numerical solution of nonlinear Klein-Gordon equation," *Computer Physics Communications*, vol. 181, pp. 78-91, 2010.
- [10] M. A. M. Lynch, "Large amplitude instability in finite difference approximations to the Klein-Gordon equation," *Appl. Numer. Math.*, vol. 31, pp. 173-182, 1999.
- [11] B. Y. Guo, X. Li and L. Vazquez, "A Legendre spectral method for solving the nonlinear Klein-Gordon equation," *Math. Appl. Comput.*, vol. 15, pp. 19-36, 1996.
- [12] S. Jimenez and L. Vazquez, "Analysis of four numerical schemes for a nonlinear Klein-Gordon equation," *Appl. Math. Comput.*, vol. 35, pp. 61-94, 1990.
- [13] C. Chui, *An introduction to wavelets*, New York: Academic press, 1992.
- [14] I. Daubechies, *Ten lectures on wavelets*, Philadelphia, PA: SIAM, 1992.
- [15] G. Strang and T. Nguyen, *Wavelets and filter banks*, Cambridge, MA: Wellesley-Cambridge, 1997.
- [16] L. Telesca, G. Hloupis, I. Nikolintaga, F. Vallianatos, "Temporal patterns in southern Aegean seismicity revealed by the multiresolution wavelet analysis," *Commun Nonlinear Sci Num Simul*, vol. 12, pp. 1418-1426, 2007.
- [17] C. Chui, *Wavelets: a mathematical tool for signal analysis*, Philadelphia, PA: SIAM, 1997.
- [18] S.G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans, Pattern Anal. Mach. Intell.* vol. 11, pp. 674-693, 1989.
- [19] C. de Boor, *A practical guide for splines*, Springer-Verlag, New York, 1978.
- [20] K. Maleknejad, K. Nouri and M. Nosrati Sahlan, "Convergence of approximate solution of nonlinear Fredholm-Hammerstein integral equations," *Commun. Non. Sci. Num. Simul.*, vol. 15, pp. 1432-1443, 2010.
- [21] S. A. Khuri, A. Sayfy, "A spline collocation approach for the numerical solution of ageneralized nonlinear Klein-Gordon equation," *Applied Mathematics and Computation*, vol. 216, pp. 1047-1056, 2010.
- [22] A. M. Wazwaz, "The modified decomposition method for analytic treatment of differential equations," *Applied Mathematics and Computation*, vol. 173, pp. 165-176, 2006.



# SamACO Based Optimal Placement of Phase Shifter Transformers for Power Loss Reduction

**H. A. Shayeghi**

*Technical Eng. Department*

*University of Mohaghegh Ardabili  
Ardabil, Iran*

**S. Hadi**

*Technical Eng. Department*

*University of Mohaghegh Ardabili  
Ardabil, Iran*

**H. A. Shayanfar**

*Center of Excellence for Power System*

*Automation and Operation*

*Iran University of Science and Technology,  
Tehran, Iran*

**M. Ghasemi**

*Technical Eng. Department*

*University of Mohaghegh Ardabili  
Ardabil, Iran*

hshayeghi@gmail.com, hashayanfar@yahoo.com, sasan\_h67@yahoo.com, mghasemi1986@yahoo.com

**Abstract-** Control of power transmission and loss causes complications in the large power systems. Phase Shifter Transformers (PST) according to their capabilities can be appropriate devices for power controlling in the large power transmission systems. Various criteria such as decreasing of loss, saving of generator annual economic cost, improving system static and dynamic behaviors and reducing of congestion have been individually studied in the literature to placement problem of phase shifter transformers (PST). In this paper, loss reduction, voltage profile and congestion improvement indices are assessed and the optimal locations of PST devices evaluated by using Sample Ant Colony Optimization (SamACO) algorithm. The effectiveness of the proposed ABC based method is demonstrated on IEEE 30-Bus network through some performance indices in comparison with the genetic algorithm and particle swarm optimization. Results evaluation show that the SamACO algorithm has a good capability in loss reduction, voltage profile and congestion enhancement than the GA and PSO methods.

**Keywords:** PST, Allocation, SamACO; Power Loss Reduction.

## 1. Introduction

Nowadays, increment of generation quantity becomes a necessity relating to the ongoing industrials, electrical consumptions growing and consequently the consecutive load increasing. On the other hand, operating of power system must be closed to its nominal capacity considering high development costs of

networks and their related devices and environmental concerns. Liberalization of the electricity market and utilities tendency in getting more profits also compel power systems to operate close to their rate capacities and sometimes in over load conditions. Moreover, variable distribution of load and generation resources based on their conditions make network operate in heavy congested load conditions in some parts and in light load conditions in others. Accurate evaluating of power transmission and determining its level is investigated in congestion indices concepts. Many various indices already have been represented to quantifying congestion values of transmission lines [1]. Network operation close to its nominal capacity may appear as over load state in some sections and can lead to partial outages. Continuation of this state results in blackout condition possibly. Thus, proper management of network power flow is a main necessity along with holding the operation constraints. Controlling and managing the power flow in network lines can be done by using the various methods and some controlling actions and devices such as Generation Rescheduling (GR), series capacitors, FACTS controlling device and sude-FACTS devices [2-4]. Phase Shifter Transformer (PST) is one of the sude-FACTS devices which can replace the power in related and next lines by changing the transmitting power value that may cause to relieve congestion [5-7]. Phase shifter transformer installation considering its advantages can control the value of line power flows obviously. However, because of investment limitations, the installation and usage of phase shifter transformers only based on their advantages will not be economic.

Proper location and sizing of PST should be noticed and studied because its installation at wrong places and capacities can cause various problems in the operation conditions. In this paper, a Sample Ant Colony

---

\*Corresponding Author (hshayeghi@gmail.com)

Optimization (SamACO) algorithm is proposed for finding optimal location of PST aimed at reducing the power loss and improving voltage profile and power congestion. The SamACO algorithm is a typical swarm-based approach to optimization, in which the search algorithm is inspired by the intelligent foraging behavior of a ant colony process [8]. It is based on the idea of sampling candidate values for each variable and selecting the values to form solutions. In SamACO technique, the sampling method possesses the feature of balancing memory, exploration and exploitation. By preserving variable values from the best solutions constructed by the previous ants, promising variable values are inherited from the last iteration. Thus, high solution accuracy is achieved by exploiting new variable values surrounding the best-so-far solution by a dynamic exploitation process.

IEEE 30 Bus network has been used as a test system to demonstrate the effectiveness and robustness of the proposed SamACO algorithm and their ability to provide efficient loss reduction and voltage profile improvement. To show the superiority of the proposed approach, the simulations results are compared with the particle swarm optimization and genetic algorithm through some performance indices. The results evaluation shows that the proposed method achieves good robust performance and is superior to the other methods.

**2. PST Model [9]**

An ideal voltage source with voltage  $V_s$  and reactance  $X_s$  that is connected in series between nodes  $i$  and  $j$  is shown in Fig. 1.

$V_i$  is an imaginary voltage source which can be defined as:

$$\bar{V}'_i = \bar{V}_s + \bar{V}_i \tag{1}$$

$$\bar{V}_s = r \bar{V}_i e^{j\gamma} \text{ and } 0 \leq r \leq 1 \tag{2}$$

Figure 2 shows a phasor diagram that is used to represent the voltage of Fig. 1 with regulating magnitude and angle.

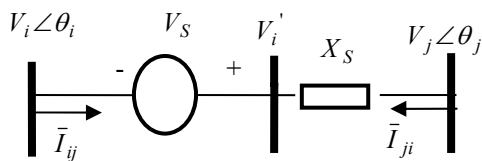


Fig. 1. Series voltage source between buses  $i$  and  $j$

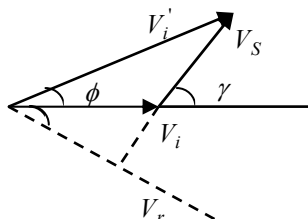


Fig. 2. Voltages phasor diagram of Fig. 1

Figure 3 shows the current source model of PST (the Norton model of voltage source) where,  $b_s = \frac{1}{X_s}$  and  $I_s = -jb_s V_s$ .

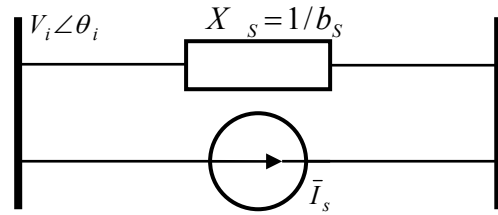


Fig. 3. Norton model of series voltage source

Current source is dependent on the usage of nodes  $i$  and  $j$  for transmitting power, then  $\bar{S}_{is}$  and  $\bar{S}_{js}$  are expressed as follows:

$$\bar{S}_{is} = \bar{V}_i (-\bar{I}_s)^* \tag{3}$$

$$\bar{S}_{js} = \bar{V}_j (\bar{I}_s)^* \tag{4}$$

After replacing relations (1) and (2) to (3) and (4) and simplifying, the injected active and reactive powers are calculated using the following equations [10].

$$P_{is} = r b_s V_i^2 \sin \gamma \tag{5}$$

$$Q_{is} = r b_s V_i^2 \cos \gamma \tag{6}$$

$$P_{js} = -r b_s V_i V_j \sin(\theta_{ji} + \gamma) \tag{7}$$

$$Q_{js} = -r b_s V_i V_j \cos(\theta_{ji} + \gamma) \tag{8}$$

The powers injection model of PST has been shown in Fig. 4 as a series voltage resource.

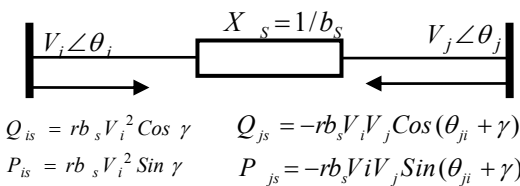


Fig. 4. Injection model of series voltage source

By using Eqs. (5) through (8) and applying a phase shifter transformer, angle  $\gamma$  can be changed and then the value of line power flow will be vary. If a PST is installed between buses  $i$  and  $j$ , the new admittance matrix formed considering the impedance  $X_s$  in the network admittance matrix. The Jacobean matrix is given in Table 1 [9]. Powers injection in PST model can be added to Jacobean matrix elements by a particular sign.

Table 1. Modified Jacobian matrix

$H_{(i,j)} = H_{(i,j)}^0 - Q_{sj}$	$N_{(i,i)} = N_{(i,i)}^0 - P_{sj}$
$H_{(i,j)} = H_{(i,j)}^0 + Q_{sj}$	$N_{(i,j)} = N_{(i,j)}^0 - P_{sj}$
$H_{(j,i)} = H_{(j,i)}^0 + Q_{sj}$	$N_{(j,i)} = N_{(j,i)}^0 + P_{sj}$
$H_{(j,j)} = H_{(j,j)}^0 - Q_{sj}$	$N_{(j,j)} = N_{(j,j)}^0 + P_{sj}$
$J_{(i,i)} = J_{(i,i)}^0$	$L_{(i,i)} = L_{(i,i)}^0 + 2Q_{sj}$
$J_{(i,j)} = J_{(i,j)}^0$	$L_{(i,j)} = L_{(i,j)}^0$
$J_{(j,i)} = J_{(j,i)}^0 - P_{sj}$	$L_{(j,i)} = L_{(j,i)}^0 + 2Q_{sj}$
$J_{(j,j)} = J_{(j,j)}^0 + P_{sj}$	$L_{(j,j)} = L_{(j,j)}^0 + 2Q_{sj}$

### 3. SamACO Algorithm

The SamACO algorithm describes the foraging behavior of ant colony for numerical optimization problems. It is a very simple, robust and population based stochastic optimization algorithm [10]. Unique characteristics of SamACO are the cooperation of a new sampling method for discretizing the continuous search space and an efficient method for incremental solution construction based on the sampled variable values. In SamACO, the sampling method possesses the feature of balancing memory, exploration, and exploitation. By preserving variable values from the best solutions constructed by the previous ants, promising variable values are inherited from the last iteration. Diversity of the variable values is maintained by exploring a small number of random variable values. Thus, high solution accuracy is achieved by exploiting new variable values surrounding the best-so-far solution by a dynamic exploitation process. [11].

The success of SamACO in solving continuous optimization problems depends on an effective variable sampling method and an efficient solution construction process. The variable sampling method in SamACO maintains promising variable values for the ants to select, including variable values selected by the previous ants, diverse variable values for avoiding trapping, and variable values with high accuracy. The ants' construction process selects promising variable values to form high-quality solutions by taking advantage of the traditional ACO method in selecting discrete components [1]. Each decision variable  $X_i$  has  $k_i$  sampled values  $x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(k)}$  from the continuous domain  $[l_i, u_i]$ ,  $i = 1, 2, \dots, n$ . The sampled discrete variable values are then used for optimization by a traditional ACO process (see Ref. [12] for more details about [12] traditional ACO) as in solving DOPs (discrete optimization problems). The flowchart of the SamACO algorithm is depicted.

#### A. Generation of the Candidate Variable Values

The generation processes of the candidate variable values in the initialization step and in the optimization

iterations are different. Initially, the candidate variable values are randomly sampled in the feasible domain as:

$$x_i^{(j)} = l_i + \frac{u_i - l_i}{m + \mathcal{G}} (j - 1 + rand_i^{(j)}) \quad (9)$$

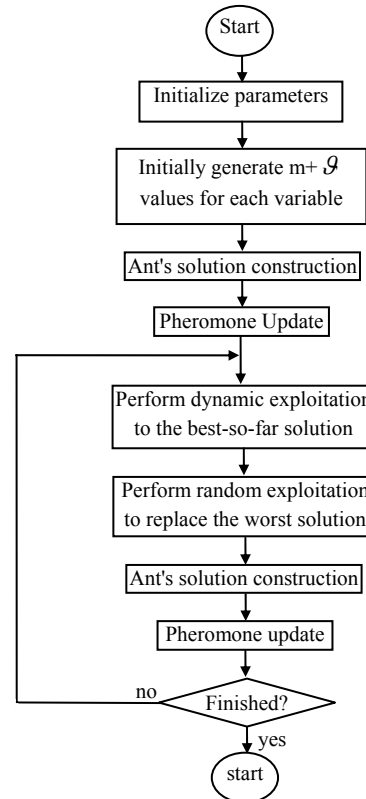


Fig. 5. Flowchart of the SamACO algorithm for continuous optimization

Where,  $(m + \mathcal{G})$  is the initial number of candidate values for each variable  $i$ , and  $rand_i^{(j)}$  is a uniform random number in  $[0,1]$ ,  $i=1, 2, \dots, n, j = 1, 2, \dots, m + \mathcal{G}$ .

During the optimization iterations, candidate variable values have four sources, i.e., the variable values selected by ants in the previous iteration, a dynamic exploitation, a random exploration, and a best-so-far solution. In each iteration,  $m$  ants construct  $m$  solutions, resulting in  $m$  candidate values for each variable for the next iteration. The best-so-far solution is then updated, representing the best solution that has ever been found. The dynamic exploitation is applied to the best-so-far solution, resulting in  $g_i$  new candidate variable values near the corresponding variable values of the best-so-far solution for each variable  $X_i$ . Furthermore, a random exploration process generates  $\Theta$  new values for each variable by discarding the worst  $\Theta$  solutions that are constructed by the ants in the previous iterations. Suppose that the worst  $\Theta$  solutions are denoted by  $x^{(m-\Theta+1)}, x^{(m-\Theta+2)}, \dots, x^{(m)}$ . The new variable values for the solution  $x^{(j)}$  are randomly generated as:

$$x_i^{(j)} = l_i + (u_i - l_i) \times \text{rand}_i^{(j)} \quad (10)$$

Where,  $i = 1, 2, \dots, n$  and  $j = m - \Theta + 1, \dots, m$ . New values, thus, can be imported in the value group. There are a total of  $(m + g_i + 1)$  candidate values for each variable  $X_i$  [11].

For the improvement of SamACO, an efficient dynamic exploitation, ants solution construction and pheromone update is used .

### B. Dynamic Exploitation Process

The dynamic exploitation proposed in this study is effective for introducing fine-tuned variable values into the variable value group. We use a radius  $r_i$  to confine the neighborhood exploitation of the variable value  $x_i$ ,  $i = 1, 2, \dots, n$ . The dynamic exploitation is applied to the best-so-far solution  $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})$ , aiming at searching the vicinity of the variable value  $x_i^{(0)}$  in the interval  $[x_i^{(0)} - r_i, x_i^{(0)} + r_i]$ ,  $i = 1, 2, \dots, n$ . The values of the variables in the best-so-far solution are randomly chosen to be increased, unchanged, or reduced as:

$$\hat{x}_i = \begin{cases} \min(x_i^{(0)} + r_i \cdot \sigma_i, u_i), & 0 \leq q < 1/3 \\ x_i^{(0)} & 1/3 \leq q < 2/3 \\ \max(x_i^{(0)} - r_i \cdot \sigma_i, l_i), & 2/3 \leq q < 1 \end{cases} \quad (11)$$

Where,  $\sigma_i \in (0, 1]$  and  $q \in (0, 1]$  are uniform random values,  $i = 1, 2, \dots, n$ . Then, the resulting solution  $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)$  is evaluated. If the new solution is no worse than the recorded best-so-far solution, we replace the best-so-far solution with the new solution. The above exploitation repeats for  $\mathcal{G}$  times. The new variable values that are generated by increasing or reducing a random step length are recorded as

$$x_i^{(j)}, j = m + 1, m + 2, \dots, m + g_i, i = 1, 2, \dots, n, \quad (12)$$

Where,  $g_i$  counts the number of new variable values in the dynamic exploitation process. In each iteration, the radiuses adaptively change based on the exploitation result. If the best exploitation solution is no worse than the original best-so-far solution (case 1), the radiuses will be extended. Otherwise (case 2), the radiuses will be reduced, *i.e.*:

$$r_i \leftarrow \begin{cases} r_i \cdot v_e, & \text{case 1} \\ r_i \cdot v_r, & \text{case 2} \end{cases} \quad (13)$$

Where,  $v_e$  ( $v_e \geq 1$ ) is the radius extension rate, and  $v_r$  ( $0 < v_r \leq 1$ ) is the radius reduction rate. The initial radius value is set as  $r_i = (u_i - l_i) / (2m)$ ,  $i = 1, 2, \dots, n$ . The extension of the radiuses can import values in a distance further away from the original value, whereas the reduction of the radiuses can generate values with high accuracy near to the original value. The extension and the reduction of radiuses aim at introducing values with different accuracy levels according to the optimization situation.

### C. Ants' Solution Construction

After the candidate variable values are found out,  $m$  ants are dispatched to construct solutions. Each candidate variable value is associated with pheromones, which bias the ants' selection for solution construction. The index  $l_i^{(k)}$  of the variable value selected by ant  $k$  for the  $i$ th variable is given by:

$$l_i^{(k)} = \begin{cases} \arg \max \{ \tau_i^{(1)}, \tau_i^{(2)}, \dots, \tau_i^{(m)} \}, & \text{if } q < q_0 \\ L_i^{(k)}, & \text{otherwise} \end{cases} \quad (14)$$

Where,  $q$  is a uniform random value in  $[0, 1)$ ,  $i = 1, 2, \dots, n$ ,  $k = 1, 2, \dots, m$ . The parameter  $q_0 \in [0, 1)$  controls whether an ant will choose the variable value with the highest pheromone from the  $m$  solutions generated in the previous iteration, or randomly choose an index  $L_i^{(k)} \in \{0, 1, \dots, m + g_i\}$  according to the probability distribution given by

$$p_i^{(j)} = \frac{\tau_i^{(j)}}{\sum_{u=0}^{m+g_i} \tau_i^{(u)}}, \quad j = 0, 1, \dots, m + g_i \quad (15)$$

The constructed solution of ant  $k$  is denoted  $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})$ .

### D. Pheromone Update

Initially, each candidate variable value is assigned an initial pheromone value  $T_0$ . After evaluating the  $m$  solutions constructed by ants, the solutions are sorted by their objective function values in an order from the best to the worst. Suppose that the sorted solutions are arranged as  $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}$ . The pheromones on the selected variable values are evaporated as:

$$\tau_i^{(j)} \leftarrow (1 - \rho) \cdot \tau_i^{(j)} + \rho \cdot T_{\min} \quad (16)$$

Where,  $0 < \rho < 1$  is the pheromone evaporation rate, and  $T_{\min}$  is the predefined minimum pheromone value,  $i = 1, 2, \dots, n$ ,  $j = 1, 2, \dots, m$ . The variable values in the best  $\Psi$  solutions have their pheromones reinforced as:

$$\tau_i^{(j)} \leftarrow (1 - \alpha) \cdot \tau_i^{(j)} + \alpha \cdot T_{\max} \quad (17)$$

Where,  $0 < \alpha < 1$  is the pheromone reinforcement rate, and  $T_{\max}$  is the predefined maximum pheromone value,  $i = 1, 2, \dots, n$ ,  $j = 1, 2, \dots, \Psi$ , with  $\Psi$  being the number of the high-quality solutions that receive additional pheromones on the variable values. The pheromones on the variable values that are generated by the exploration process and the dynamic exploitation process are assigned as  $T_0$ . In each iteration, pheromone values on the variable values of the best-so-far solution are assigned equal to the pheromone values on the iteration best solution.

## 4. Problem Formulation

The main goal of this paper is loss reduction, voltage profile and congestion improvements via optimal allocation of PST. Decreasing the amount of loss and boosting the voltage profile are serious issues in new and modern power networks, but the necessity of having

an acceptable security margin in network operation is also very important. For obtaining these purposes, utilizing of phase shifter transformers is essentially required. Thus, for balancing lines power flow, it is necessary to assess congestion problem in relevant indices firstly. To investigating the congestion parameter of test system, the  $PI_i$  index proposed in reference [13] has been used which is expressed as follows:

$$PI_i = 1 - \frac{1}{N} \sum_{i=1}^N \frac{\omega_i}{\sqrt{2\pi}\sigma} e^{-\frac{(P_i - \mu)^2}{2\sigma^2}} \quad (18)$$

Where,

$\sigma$ : Lines power standard deviation from nominal values

$\mu$ : Equals 70 percent of line nominal power (P.U)

$\omega_i$ : Weight factor of line  $i$

$N$ : Number of lines  $N$

$P_i$ : Power of each line (p.u.)

For calculating  $PI_i$ :  $\omega_i$  and  $\sigma$  is considered 1 and 0.3, respectively.

If all lines are loaded at their nominal value,  $PI_i$  index has a low value and if overload condition occurs in networks,  $PI_i$  will be have a large value. Thus, optimal location of PST will be evaluated to reduce lines loss and improve congestion and voltage profile indices by ABC algorithm. The objective function used for phase shifter transformers placement is given by:

$$PI = W_1 PI_i + W_2 P_{loss} + W_3 \frac{1}{m} \sum_{i=1}^m |V_i - 1| \quad (19)$$

Where,

$m$ : Total number of buses

$V_i$ :  $i$ th bus voltage in p.u.

$PI_i$ : Congestion indices in p.u.

$P_{loss}$ : Total value of system losses in p.u

$w_1, w_2, w_3$ : Weight coefficients related to congestion, loss and bus voltage indices, respectively

Minimizing the objective function that is composed of loss, congestion and voltage profile indices will leads to finding PST optimal location. Thus, the allocation problem can be formulated as the following optimization problem, where the constraints are the buses voltage magnitude limits, lines active power transmitting capabilities and generated active and reactive power of generators limitations [9]:

$$\text{Minimize } PI \quad (20)$$

The proposed approach employs SamACO algorithm to solve this optimization problem and search for optimal placement of PST by evaluating the objective cost function as given in Eq. (20) using load flow of power system. The goal is determining the installation place and angle setting of phase shifter transformers. The weight factors  $w_1, w_2$  and  $w_3$  are included in objective function according the importance and effects of  $P_{loss}$ , congestion index  $PI_i$  and buses voltage magnitudes. It is necessary to mention that in this paper they have set to 20, 1 and 1, respectively.

## 5. Simulation Results

The proposed method is applied to the electrical network on IEEE 30 bus including six thermal generating units as shown in Fig. 6 to assess the suitability of the algorithm. The system data extracted from [13]. The MARTPOWER-4 toolbox of MATLAB software is used for load flow running.

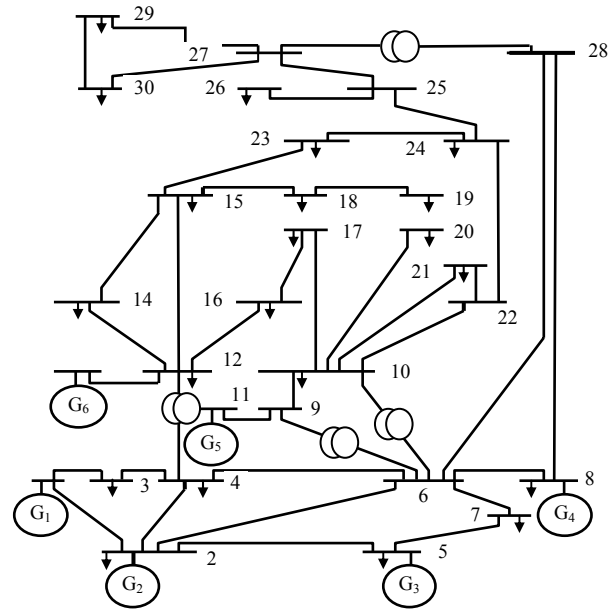


Fig. 6. IEEE 30 bus power system

The goal is determining the installation place and angle setting of phase shifter transformers. The obtained results using SamACO is compared with PSO and GA methods in order to illustrate its robust performance and effectiveness for the solution of optimal allocation of PST problem.

Results of the PST placement based on the objective function  $PI$ , by applying AC power flow using the proposed SamACO, PSO and GA algorithms are given in Table 2. Figure 7 shows the minimum fitness functions evaluating process.

Table 2. Optimal PST parameters

Algorithm	Optimized Place of PST	Angle of PST	The best cost Function
GA	11	-9.8787	0.9166
PSO	14	-8.7775	0.9153
SamACO	14	-8.7775	0.9153

It can be seen that from Table 1 installing a phase shifter transformer in line 14 between buses 9 and 10 can be reduced loss and improved voltage profile and congestion indices of network clearly than the GA method. Table 3 presents the lines and network loss,  $PI_i$  index values before and after installing of PST using three methods. It is evident that the SamACO and PSO

based solution is identical. Using the GA the reduction at total loss of network is 7.5%, whereas it is 8.5 % using the proposed SamACO algorithm.

Figures 7 and 8 show the power loss of network lines and  $PI_1$  index. It can be seen that the proposed SamACO method has good performance and power loss

reduction and  $PI_1$  index improvement is significantly occurred after PST placement using SamACO and PSO techniques. Voltage profile of network before and after PST installation is shown in Fig. 9. Using the proposed SamACO algorithm voltage profile is considerably improved. Moreover, it is superior to the GA method.

Table 3. The lines and network loss,  $PI_1$  index values before and after installing of PST

NO #	From Bus	To Bus	GA		PSO & SamACO			
			$P_{loss}(MW)$ Without PST	$PI_1(p.u)$ Without PST	$P_{loss}(MW)$ With PST	$PI_1(p.u)$ With PST	$P_{loss}(MW)$ With PST	$PI_1(p.u)$ With PST
1	1	2	0.026	0.9995	0.29	0.9994	0.029	0.9786
2	1	3	0.128	0.9998	0.118	0.9998	0.118	0.9754
3	2	4	0.178	0.9878	0.163	0.9882	0.163	0.8391
4	3	4	0.018	0.9872	0.017	0.9877	0.017	0.8337
5	2	5	0.110	0.9866	0.120	0.9870	0.118	0.8935
6	2	6	0.286	0.9741	0.326	0.9694	0.321	0.8239
7	4	6	0.066	0.9670	0.127	0.9587	0.119	0.7887
8	5	7	0.120	0.9682	0.130	0.9496	0.129	0.7689
9	6	7	0.031	0.9677	0.028	0.9492	0.028	0.7927
10	6	8	0.128	0.9400	0.124	0.9314	0.124	0.7604
11	6	9	0	0.9327	0	0.9147	0	0.7045
12	6	10	0	0.9180	0	0.8985	0	0.6751
13	9	11	0	0.9124	0	0.8929	0	0.6823
14	9	10	0	0.9050	0	0.8760	0	0.6581
15	4	12	0	0.8988	0	0.8654	0	0.6542
16	12	13	0	0.8817	0	0.8487	0	0.6321
17	12	14	0.037	0.8668	0.025	0.8340	0.026	0.6180
18	12	15	0.065	0.8604	0.018	0.8190	0.021	0.6047
19	12	16	0.008	0.8342	0.009	0.8045	0.012	0.5942
20	14	15	0.003	0.8175	0.009	0.7876	0.008	0.5798
21	16	17	0.031	0.8000	0.001	0.7710	0	0.5674
22	15	18	0.097	0.7823	0.037	0.7536	0.041	0.5545
23	18	19	0.022	0.7650	0.003	0.7368	0.04	0.5435
24	19	20	0.090	0.7503	0.020	0.7153	0.018	0.5359
25	10	20	0.052	0.7350	0.099	0.6990	0.092	0.5280
26	10	17	0.023	0.7200	0.042	0.6826	0.039	0.5207
27	10	21	0.044	0.7062	0.045	0.6687	0.044	0.5157
28	10	22	0.062	0.6915	0.056	0.6547	0.056	0.5142
29	21	22	0.093	0.6717	0.081	0.6373	0.081	0.5063
30	15	22	0.109	0.6540	0.142	0.6200	0.138	0.4985
31	22	23	0.078	0.6373	0.039	0.6023	0.037	0.4920
32	23	24	0.066	0.6200	0.042	0.5850	0.044	0.4855
33	24	25	0.035	0.6025	0.003	0.5684	0.001	0.4804
34	25	26	0.046	0.5855	0.046	0.5514	0.046	0.4750
35	25	27	0.063	0.5879	0.022	0.5344	0.023	0.4699
36	28	27	0	0.5626	0	0.5253	0	0.4745
37	27	29	0.108	0.5350	0.108	0.5077	0.108	0.4692
38	27	30	0.127	0.5176	0.127	0.4902	0.127	0.4642
39	29	30	0.013	0.5030	0.013	0.4754	0.013	0.4624
40	8	28	0.036	0.4882	0.043	0.4600	0.042	0.4601
41	6	28	0.001	0.4667	0.005	0.4450	0.005	0.4585
TOTAL			2.395		2.215		2.193	



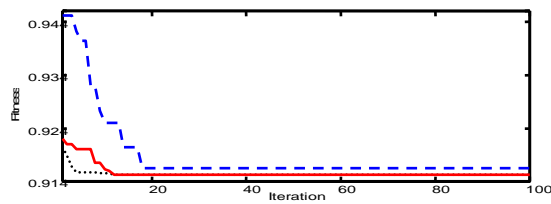


Fig. 7: Fitness convergence, Dashed (GA), Dotted (PSO) and Solid (SamACO).

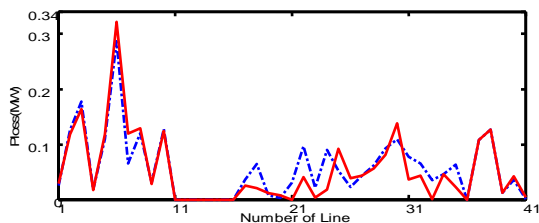


Fig. 7.  $P_{loss}$  index before and after PST installation, Dashed (before PST installation), Solid (after PST installation and using SamACO)

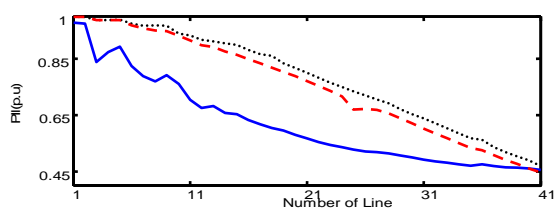


Fig. 8.  $P_{li}$  congestion index, Dashed (without PST), Dotted (with PST using GA), Solid (with PST using SamACO)

## 6. Conclusions

This paper presents an appropriate method based on SamACO algorithm to improve line loss, voltage profile and congestion indices through optimal sitting of a phase shifter transformers. Due to consideration to some practical issues and by defining a new performance index, the optimum allocation of a single PST and its controlling phase angles can be determined. In the SamACO algorithm, the sampling method possesses the feature of balancing memory, exploration and exploitation. By preserving variable values from the best solutions constructed by the previous ants, promising variable values are inherited from the last iteration. Thus, high solution accuracy is achieved by exploiting new variable values surrounding the best-so-far solution by a dynamic exploitation process. The performance of the proposed SamACO based method is tested on IEEE 30-Bus network and the proper location for installing phase shifter transformers is obtained by minimizing the objective function in short evaluating time. Results evaluation show significant reduction in power loss in addition to voltage profile and congestion improvement than the GA method one.

## References

[1] S. N. Singh and A. K. David, "Congestion Management by Optimal FACTS Device Location", Proc. of Int. IEEE Conf. on Elect. Utility Deregulation and Restructuring and Power Technologies, pp. 23-28, Apr. 2000.

- [2] G. Wu, A. Yokoyama, J. He, and Y. Yu, "Allocation and Control of FACTS Devices for Steady-State Stability Enhancement of Large Scale Power Systems," Int. Conf. on Power Syst. Tech. POWERCON '98, Vol. 1, pp. 357-361, 1998.
- [3] S. Gerbex, R. Cherkaoui, and A. J. Germond, "Optimal Location of Multi-Type FACTS Devices in a Power System by Means of Genetic Algorithms," IEEE Trans. Power Syst., Vol. 6, pp. 537-544, 2001.
- [4] H. Mori and Y. Goto, "A Parallel Tabu Search Based Method for Determining Optimal Allocation of FACTS in Power Systems", Proc. of Int. IEEE Conf. on Power System Technology, Vol. 2, pp. 1077-1082, Dec. 2000.
- [5] P. D. Kamjorn, P. K. Arcot, and P. Dcouth, "A Screening Technique for Optimally Locating Phase Shifters in Power Systems," IEEE PES, Chicago, IL, pp. 233-238, 1994.
- [6] A.K. Kazerooni, H. Seifi, M.S. Sepasian, H. Haghghat, "New Approach for Phase Shifter Transformer Allocation", The 8<sup>th</sup> IEE International Conference, pp. 94-98, 2006.
- [7] P. Paterni, S. Vitet, M. Bena, A. Yokoyama, Optimal Location of Phase Shifters in the French Network by Genetic Algorithm. IEEE Trans Power Syst.; Vol. 14; N0 1; pp. 37-42; 1999.
- [8] S. Gerbex, R. Cherkaoui, A.J. Germond. Optimal Location of Multi-type FACTS Devices in a Power System by Means of Genetic Algorithms. IEEE Trans Power Syst. Vol. 16; N0 3; pp. 537-544; 2001.
- [9] M. Gitizadeh, H. Khalilnezhad, "Phase Shifter Transformers Optimum Allocation in Power Systems Using a Combinational Method", Proc. of Int. IEEE Conf. on Power System Technology, pp. 886-890, 2010.
- [10] X.-M. Hu, J. Zhang, "SamACO: Variable Sampling Ant Colony Optimization Algorithm for Continuous Optimization", IEEE Trans. on Systems, Man and Cybernetics, Part B, Vol. 40, No. 6, pp. 1555-1565, 2010.
- [11] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," Artificial Life, Vol. 5, no. 2, pp. 137-172, Apr. 1999.
- [12] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., Vol. 1, No. 1, pp. 53-66, 1997.
- [13] R. D. Zimmerman, C. E. Murillo-sanchez, R. J. Thomas, Matpower: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education, IEEE Trans. on Power Systems, Vol. 26, No. 1, pp. 12-19, Feb. 2011.

# A New Fuzzy-neural Approach to forecast the Exchange Rate for Taiwanese Semiconductor Industry

Tin-Chi Chen<sup>1</sup>, Hsiao-Ting Lu<sup>1</sup> and Yu-Cheng Lin<sup>2\*</sup>

<sup>1</sup>School of Management Development, Feng Chia University, Taichung City, Taiwan

<sup>2\*</sup>Department of Industrial Engineering and Management, Overseas Chinese University, Taichung City, Taiwan

**Abstract** - Accurately forecasting the foreign exchange rate is critical to enterprises like the semiconductor industry in Taiwan. For this purpose, a fuzzy-neural approach is proposed. In the proposed methodology, a committee of virtual experts is organized instead, and then they are asked to give opinions about the fuzzy forecasts. A corresponding FLR equation is constructed to forecast the foreign exchange rate for each virtual expert. To aggregate these fuzzy foreign exchange rate forecasts, a two-step aggregation mechanism is applied. First, partial-consensus fuzzy intersection is applied to aggregate the fuzzy forecasts into a polygon-shaped fuzzy number, in order to improve the precision. Then a radial basis function network (RBF) is constructed to defuzzify the polygon-shaped fuzzy number and to generate a representative/crisp value, so as to enhance the accuracy. To evaluate the effectiveness of the proposed methodology, the practical case of forecasting the foreign exchange rate of NTD for USD is used. According to the experimental results, the proposed methodology improved.

**Keywords:** Exchange Rate, Fuzzy-neural

## 1 Introduction

The foreign-exchange rate between 2 currencies specifies how much one currency is worth in terms of the other. Accurately forecasting the foreign exchange rate is very important for export-oriented enterprises. To the semiconductor industry in Taiwan, the depreciation of Taiwan's currency, NTD, leads to the increase in the costs of raw material imported from overseas, and the decrease in gross margin that is usually measured in terms of NTD. Besides, semiconductor component distributors in Taiwan sell products for world-renowned manufacturers. When there is excessive volatility in the foreign exchange rate of NTD, it becomes very difficult for these distributors to control product costs. Therefore, the manufacturers need to adopt some hedging actions which are usually based on a precise forecast of the foreign exchange rate. Some of the DRAM makers in Taiwan their gross margin might be reduced by half with just a 3% increase in the foreign exchange rate of NTD.

The fluctuation in the foreign exchange rate can be treated as a type of time series [5]. A sequential learning neural network is used in Hu et al. [3] to forecast the exchange rate between USD and Mark. However, most crisp approaches suffer from serious drawback due to inherent uncertainty and data acquisition problems. For this reason, fuzzy or probabilistic approaches have been in the literature. Tseng et al. [7] proposed the fuzzy ARIMA (FARIMA) approach to predict the foreign exchange rate of NTD for USD by determining the upper and lower bounds for the forecasts generated by ARIMA.

In this study, a virtual-expert partial-consensus fuzzy-neural approach is proposed for the foreign exchange rate forecasting problem. The FLR-BPN approach is a general approach like Mamdani's or Takagi-Sugeno's fuzzy inference systems, and potentially can be applied to forecast any phenomena in various fields of research or applications, e.g. semiconductor yield forecasting [1], etc.

Compared with the original FLR-BPN approach, the proposed methodology has the following innovative treatments:

(1) In case that the number of domain experts is not sufficient, opinions about the fuzzy foreign exchange rate forecasts will be solicited from a committee of virtual experts instead in the proposed methodology.

(2) A radial basis function network (RBF) is constructed to defuzzify the polygon-shaped fuzzy number and to generate a representative/crisp value instead of the original BPN defuzzifier.

## 2 Methodology

### 2.1 Data preprocessing

Before applying the proposed methodology, the collected data are scaled into smaller values through the partial normalization approach [2]:

$$N(x) = N_L + \frac{x - x_{\min}}{x_{\max} - x_{\min}} \cdot (N_U - N_L) \quad (1)$$



where  $x$  indicates the original data;  $N(x)$  is the normalized value of  $x$ ;  $N_L$  and  $N_U$  indicate the lower and upper bounds of the range of the normalized value.  $x_{\min}$  and  $x_{\max}$  are the minimum and maximum of  $x$ , respectively. Forecasts generated by the proposed methodology,  $f(x)$ , will be converted back to the un-normalized values as follows:

$$x = \frac{f(x) - N_L}{N_U - N_L} \cdot (x_{\max} - x_{\min}) + x_{\min} \quad (2)$$

### 2.2 Step 1: Generating the opinions of virtual experts

Domain expert are asked to give their opinions about the following terms, so that the fuzzy forecasts can satisfy the requirements of these experts:

- 1) The sensitivity to the uncertainty ( $o_l$ ): A large value of  $o_l$  means that the expert is very sensitive to the uncertainty of the fuzzy forecasts which is usually represented in terms of the average range of the fuzzy forecasts.
- 2) The desired range of every fuzzy foreign exchange rate forecast ( $d_l$ ): It usually depends on their purposes of application and might not be equal.
- 3) The required satisfaction level ( $s_l$ ): A large value of  $s_l$  implies that the core (i.e. values with membership equal to 1) of the fuzzy forecast is very representative.
- 4) The relative unimportance of the outliers of the sample data ( $m_l$ ): A large value of  $m_l$  means that the outliers are not important in fitting the FLR equation.

A real-valued set containing these 4 terms  $OS_l = \{o_l, d_l, s_l, m_l\}$  is therefore called an opinion set;  $l = 1 \sim L$  (the number of experts). A committee of virtual experts provides opinion sets for implementing the fuzzy and neural approach. However, an opinion set has to satisfy the following constraints. First, an opinion set (e.g. with very small  $d_l$  or very large  $s_l$ ) might lead to no feasible solutions of the nonlinear programming problems or increase the difficulties in solving them. Second, opinions that are close to each other might result in the same or similar fuzzy forecasts. Third, increasing the difference between 2 opinion sets does not guarantee that the fuzzy forecasts generated by these 2 opinion sets will be less similar. Fourth, an opinion set is favored if it generates a fuzzy forecast that is distinct from those of the other opinion sets. Finally, an opinion set is favored if it generates a fuzzy forecast that has more intersection points with those of the other opinion sets.

### 2.3 Step 2: Constructing multiple FLR equations

In the proposed methodology multiple experts construct their own FLR equations to forecast the foreign exchange rate of a future day.

$$\tilde{y}_n = \tilde{w}_0 (+) \sum_{k=1}^K \tilde{w}_k \cdot y_{n-k} \quad (3)$$

$$y_n = D(\tilde{y}_n) \quad (4)$$

where  $\tilde{y}_n$  is the fuzzy foreign exchange rate forecast of day  $n$ ;  $\tilde{w}_k$  are constants or coefficients,  $k = 0 \sim K$ ; (+) denotes fuzzy addition;  $D(\cdot)$  is the defuzzification function. Assuming all variables are given in triangular fuzzy numbers (TFNs).

Some virtual experts are asked to submit their opinions about the fuzzy forecasts. Afterwards, equation (3) can be fitted by solving the following NP models:

$$\text{Min } Z_1 = \sum_{n=1}^N d_n^{o_l} \quad (5)$$

subject to

$$d_n = w_{03} + \sum_{k=1}^K w_{k3} y_{n-k} - w_{01} - \sum_{k=1}^K w_{k1} y_{n-k} \quad (6)$$

$$y_n \geq w_{01} + \sum_{k=1}^K w_{k1} y_{n-k} + s_l (w_{02} + \sum_{k=1}^K w_{k2} y_{n-k} - w_{01} - \sum_{k=1}^K w_{k1} y_{n-k}) \quad (7)$$

$$y_n \leq w_{03} + \sum_{k=1}^K w_{k3} y_{n-k} + s_l (w_{02} + \sum_{k=1}^K w_{k2} y_{n-k} - w_{03} - \sum_{k=1}^K w_{k3} y_{n-k}) \quad (8)$$

$$n = 1 \sim N, w_{k1} \leq w_{k2} \leq w_{k3}, k = 0 \sim K \quad (9)$$

NP model I is based on Tanaka and Watada's philosophy [6]. The objective function is to minimize the high-order sum of the ranges of fuzzy foreign exchange rate forecasts. On the other hand, NP model II is based on Peters's philosophy [4]. Theoretically, these nonlinear objective functions and constraints can be easily converted into quadratic ones that are more tractable. In either model, every fuzzy foreign exchange rate forecast contains the actual value. As a result, the intersection of the 2 fuzzy foreign exchange rates forecasts generated by the model also contains the actual value. Besides, the intersection has a range narrower than those of the 2 regions. Therefore, the forecasting precision measured in terms of the average range is improved after intersection. Eventually, there will be at most  $2L$  FLR equations.

### 2.4 Step 3: Applying partial-consensus FI to aggregate

**the foreign exchange rate forecasts into a polygon-shaped fuzzy number**

Partial-consensus FI is applied to aggregate the fuzzy foreign exchange rates into a polygon-shaped fuzzy number, in order to improve the precision of foreign exchange rate forecasting. The intersection of the opinions by several experts embodies the consensus of all these experts.

**Definition**

The  $h/L$  among fuzzy foreign exchange rate forecasts  $\tilde{y}_i(1)$ , ..., and  $\tilde{y}_i(L)$  is indicated with  $I^{h/L}(\tilde{y}_i(1), \tilde{y}_i(2), \dots, \tilde{y}_i(L))$  such that:

$$\mu_{I^{h/L}(\tilde{y}_i(1), \dots, \tilde{y}_i(L))}(x) = \max_{\text{all } g}(\min(\mu_{\tilde{y}_i(g(1))}(x), \dots, \mu_{\tilde{y}_i(g(h))}(x))) \tag{10}$$

where  $g() \in Z^+$ ;  $1 \leq g() \leq L$ ;  $g(i) \cap g(j) = \emptyset$ ;  $h \geq 2$ .

The result of partial-consensus FI is a polygon-shaped fuzzy number (see Fig. 1). Compared with the original TFNs, the partial-consensus FI result has a narrower range while still containing the actual value. Therefore, the precision of foreign exchange rate forecasting will be improved after applying partial-consensus FI.

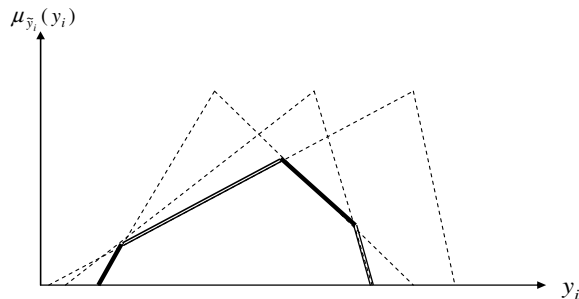


Fig. 1. The polygon-shaped fuzzy foreign exchange rate

The output of this step is a polygon-shaped fuzzy number that specifies the narrowest range of the foreign exchange rate forecast. A crisp foreign exchange rate forecast has to be generated from the polygon-shaped fuzzy number based on the requirement of practical applications. After obtaining the defuzzified value, it is compared with the actual foreign exchange rate to evaluate the accuracy. However, among the existing defuzzification methods, no one can surpass all the others in every case. These phenomena provide a motive to propose a tailored defuzzification method. In this study, a RBF is applied for this purpose.

**2.5 Step 4: Constructing a RBF to defuzzify the polygon-shaped fuzzy number**

The RBF network consists of 3 layers; the input, hidden and output layers. The input variables include  $2v$  parameters corresponding to the  $v$  corners of the polygon-shaped fuzzy number and the membership function values of these corners. All input parameters have to be normalized before they are fed into the network. The input variables are each assigned to a node in the input layer and pass directly to the hidden layer without weights. The hidden-layer nodes contain RBFs, also called transfer functions, and are analogous to the sigmoid function commonly used in a BPN. In the proposed methodology, the Gaussian transfer function is used:

$$h_i(X_i) = e^{-\sum_{j=1}^r (x_{ij} - \hat{x}_{ij})^2 / \sigma_i^2} \tag{11}$$

where  $h_i(X_i)$  is the output from the  $i$ -th hidden-layer node;  $X_i$  is the incoming vectors with components  $x_{i1}, \dots, x_{ir}$ .  $\hat{x}_{ij}$  is the centre of the  $i$ -th RBF unit for input variable  $j$ ;  $\sigma_i$  is the width of the  $i$ -th RBF unit. The processing at the hidden layer is to compute the distance from the input vector to the corresponding center, then apply the transfer function to generate the output. The output layer is linear in that we just take the linear combination of the values of the hidden layer:

$$o(X_i) = \sum_{i=1}^I w_i h_i(X_i) + w_0 \tag{12}$$

The parameters of the RBF units are determined in some way. For example, the unit centres can be determined by some clustering algorithm. Then, the widths are determined by the nearest-neighbour method. Finally, weights connecting the RBF units and the output units are calculated using multiple linear regression techniques. After training and testing, the RBF can be applied to defuzzify a polygon-shaped fuzzy number.

**3 An experiment**

To demonstrate the application of the fuzzy and neural approach, the problem of foreign exchange rate forecasting in Tseng et al. [7] is used (see Fig. 2). After applying moving average (MA) to the collected data, the number of moving period was determined to be 8 days. The original values were normalized into [0.1, 0.9] before applying the proposed methodology.

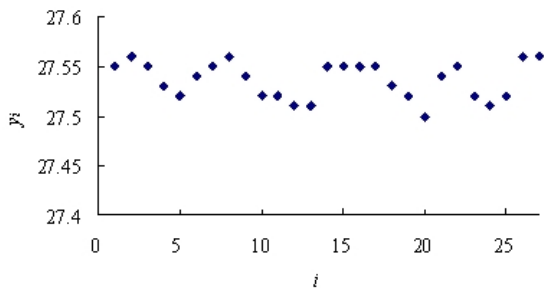


Fig. 2. The exchange rate forecasting problem

The number of virtual experts was determined to be 3, so there were 6 NP problems. From the optimization result of each NP problem, 6 corresponding FLR equations were constructed and applied to forecast the foreign exchange rate. Subsequently, the partial-consensus FI of the forecasting results was obtained, which was a polygon-shaped fuzzy number for each period. The data of the corners of all polygon-shaped fuzzy numbers were used to train and/or test the RBF defuzzifier. Finally, the RBF was applied to defuzzify a polygon-shaped fuzzy number input to the network to generate the crisp foreign exchange rate forecast.

According to experimental results, the following points can be made:

- 1) The accuracy of foreign exchange rate forecasting, measured in terms of MAPE, of the proposed methodology, was significantly better than those of the other approaches by achieving an 83% reduction in MAPE over the comparison basis – MA. The advantage over ARIMA was 64%. The performance of the proposed methodology with respect to MAE or RMSE was also significantly better than those of the other approaches.
- 2) The precision of the proposed methodology was significantly better than those of the other approaches. The advantage over the comparison basis was up to 81%.
- 3) The advantage of the proposed methodology came from some sources. First, it was shown to be a successful way of hybridizing linear and nonlinear approaches (FLR and BPN).
- 4) For testing data, the probability that all actual values were contained in the fuzzy forecasts was 93% with the overall consensus ( $I^{3/3}$ ). With little expansion in the spreads (only 0.005 on average) of the fuzzy forecasts, all actual values could be contained in the fuzzy forecasts with the partial-consensus ( $I^{2/3}$ ).

## 4 Conclusions

It is important for the semiconductor industry in Taiwan to forecast the foreign exchange rate accurately and precisely. For this purpose, a virtual-expert partial-consensus fuzzy-neural approach is applied in this study. In the proposed

methodology, multiple virtual experts construct their own FLR equations from various viewpoints to forecast the foreign exchange rate. A two-step aggregation mechanism is applied to aggregate the fuzzy foreign exchange rate forecasts. Partial-consensus FI is applied to aggregate the fuzzy foreign exchange rate forecasts into a polygon-shaped fuzzy number. The polygon-shaped fuzzy number contains the actual value. After that, considering the special shape of the polygon-shaped fuzzy number, a RBF is constructed to defuzzify the polygon-shaped fuzzy number and to generate a representative/crisp value to enhance the accuracy.

A practical case containing the historical data of the foreign exchange rate from NTD to USD was used to demonstrate the application of the methodology. According to experimental results, we found that the forecasting accuracy of the proposed methodology was significantly better than those of some existing crisp approaches.

More sophisticated fuzzy and neural approaches can be proposed for the foreign exchange rate forecasting in future studies. To facilitate the applications of the fuzzy and neural approach, a new MATLAB toolbox can also be developed in the next work.

## 5 Acknowledgment

This Study was financially supported by National Science Council of Taiwan.

## 6 References

- [1] T. Chen and Y. C. Lin, "A fuzzy-neural system incorporating unequally important expert opinions for semiconductor yield forecasting," *International Journal of Uncertainty, Fuzziness, and Knowledge-based Systems*, vol. 16, no. 1, pp. 35-58, 2008.
- [2] T. Chen, Y. C. Wang and H. R. Tsai, "Lot cycle time prediction in a ramping-up semiconductor manufacturing factory with a SOM-FBPN-ensemble approach with multiple buckets and partial normalization," *International Journal of Advanced Manufacturing Technology*, vol. 42, pp. 1206-1216, 2009.
- [3] M. Hu, P. Saratchandran and S. Narasimhan, "A sequential learning neural network for foreign exchange rate forecasting," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 4, pp. 3963-3968, 2003.
- [4] G. Peters, "Fuzzy linear regression with fuzzy intervals," *Fuzzy Sets and Systems*, vol. 63, pp. 45-55, 1994.
- [5] F. Strozzi and J. M. Zaldivar, "Nonlinear forecasting in high frequency financial time series," *Physica*, no. 353, pp. 463-479, 2005.
- [6] H. Tanaka and J. Watada, "Possibilistic linear systems and their application to the linear regression model," *Fuzzy Sets and Systems*, no. 272, pp. 275-289, 1988.

- [7] F. M. Tseng, G. H. Tzeng, H. C. Yu and B. J. C. Yuan, "Fuzzy ARIMA model for forecasting the foreign exchange market," *Fuzzy Sets and Systems*, no. 118, pp. 9-19, 2001.

# On Godunov Schemes for Combined Longitudinal and Torsional Elastic-Plastic Waves in Thin-Walled Tubes

William W. Dai

Computer, Computational, and Statistical Sciences Division  
Los Alamos National Laboratory  
Los Alamos, New Mexico, USA

**Abstract**—A high-order Godunov scheme is developed and the Riemann problem is discussed for combined longitudinal and torsional elastic-plastic waves in thin-walled tubes. The scheme is based on characteristic theory of the system, uses uniform formulation for all kinds of loading paths, and thus is easy to be implemented in an actual simulation. The scheme is tested through numerical examples including steepening of waves and step loading. It is shown that the scheme keeps the principle advantages of Godunov schemes, i.e., the robust operation in the presence of strong waves, thin fronts of discontinuities with little attendant noise generated. It is turned out that the solution of a Riemann problem in the model depends on the internal structure of the initial discontinuity.

**Keywords:** elastic-plastic waves, finite difference, Riemann problem

## 1. Introduction

Elastic-plastic waves under combined stresses have been studied since the 50's. A test involving combined dynamic stress states beyond the elastic limit of the material is expected to provide useful information regarding the dynamic plastic property of solids under combined stress states. The analysis for the combined longitudinal and torsional impact of thin-walled cylindrical tubes is of fundamental importance in the development of multidimensional theory of plastic wave propagation. A theory was presented for combined longitudinal and torsional elastic-plastic wave propagation in thin-walled tubes. Ting<sup>1</sup> have investigated the wave structures of boundary-value problems with combined stresses.

One feature of elastic-plastic waves is the discontinuity surface generated when solids are subjected impulsive loads. The dynamical structures of discontinuity surfaces are fundamentally important for the yield patterns and fractures of the solid. Analytically, as most nonlinear systems, elastic-plastic waves may not be expected to solve for most practical problems. A practical problem may involve a complicated geometry, sophisticated material properties, and different loading functions, each of which makes a quantitatively analytical analysis very difficult.

Numerically, one of main difficulties is to dynamically treat the discontinuity surfaces. As we know, a numerical

scheme with a first-order accuracy will give numerical results that are very smeared in regions near discontinuities. The reason for the smearing is the large amount of the numerical viscosity that intrinsically exists in a numerical scheme for discontinuities. If we attempt to use a standard scheme with a second-order accuracy, the numerical viscosity will be reduced, but large noise will be introduced in numerical solutions near discontinuities. The amplitudes of the noise increase with mesh refinements. Direct derivatives used in the standard scheme are responsible for the behavior.

A discontinuity is the feature of hyperbolic systems. During last twenty years, quite a few numerical schemes have been developed for hyperbolic systems of conservation laws, especially for gas dynamics. Among them Godunov schemes are believed to most appropriate for discontinuities. The examples of Godunov schemes are Godunov's scheme<sup>2</sup>, MUSCL scheme<sup>3</sup>, Roe's method<sup>4</sup>, PPM method<sup>5</sup>, TVD method<sup>6</sup>, etc. The principle advantages of these schemes are the robust operation in the presence of strong discontinuities, thin discontinuity surfaces with little attendant noise generation.

Several investigators have performed numerical study for elastic-plastic waves through hyperbolic systems (for example, see references<sup>7–13</sup>). Specifically, Nagayama<sup>7</sup> studied the behavior of one-dimensional solid under a high pressure; Trangenstein and Colella<sup>8</sup> considered finite deformation in rate-form for isotropic materials through conservation forms in Lagrangian and Eulerian frames; Lin and Ballmann<sup>12</sup> proposed a second-order Godunov method for elastic-plastic waves in thin-walled tubes through introducing three elementary loading paths on the base of characteristic formulations.

In this paper, A high-order Godunov scheme for combined longitudinal and torsional stress waves in thin-walled tubes is proposed based on characteristic formulations, and the Riemann problem in the model is discussed. The idea in the numerical scheme is borrowed from the piecewise parabolic method<sup>5</sup> except the Riemann solver. We do not use elementary loading paths, since the exact loading paths remain unknown before solutions are obtained and since differently treating different loading paths seems not favorable to the parallel architecture of computers. The idea involved in the scheme is simple and straightforward, and formulations involved in the scheme are uniform for all situations under

the model for thin-walled tubes, which make it very easy to actually implement the scheme. Our investigation shows that the solution of a Riemann problem in the model may depend on the internal structure of the initial discontinuity.

The plan of this paper is as follows. Basic equations are introduced in the second section. The numerical scheme is in Section 3. The results for a few test problems are shown in Section 4, and the conclusion and a brief discussion of the paper may be found in the last section.

## 2. Basic Equations

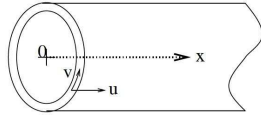


Fig. 1: A thin-walled tube.

Consider a thin-walled cylindrical tube as shown in Fig.1. The tube material is assumed to be isotropic work-hardening. The rate independent theory results in the following set of equations for combined longitudinal and torsional elastic-plastic waves in the tube<sup>4</sup>:

$$\frac{\partial \sigma}{\partial x} = \frac{\partial}{\partial t}(\rho u), \quad (1)$$

$$\frac{\partial \tau}{\partial x} = \frac{\partial}{\partial t}(\rho v), \quad (2)$$

$$\frac{\partial u}{\partial x} = \frac{1}{E} \frac{\partial \sigma}{\partial t} + H(k)[(\sigma/\theta)^2 \frac{\partial \sigma}{\partial t} + \sigma \tau \frac{\partial \tau}{\partial t}], \quad (3)$$

$$\frac{\partial v}{\partial x} = \frac{1}{G} \frac{\partial \tau}{\partial t} + H(k)[\sigma \tau \frac{\partial \sigma}{\partial t} + (\theta \tau)^2 \frac{\partial \tau}{\partial t}]. \quad (4)$$

Here  $\rho$  is the mass density of the tube,  $\sigma$  and  $\tau$  are the longitudinal and torsional stresses, and  $u$  and  $v$  are the longitudinal and circumferential particle velocities.  $E$  (or  $G$ ) is the Young's (or shear) modulus.  $H(k)$  is a function of the yield stress  $k$ ,

$$H(k) \equiv \frac{1}{k^2} \left[ \frac{1}{g(\theta k)} - \frac{1}{E} \right], \quad (5)$$

where the yield condition is given by

$$k^2 = (\sigma/\theta)^2 + \tau^2, \quad (6)$$

$\theta = \sqrt{3}$  for the von Mises yield condition and  $\theta = 2$  for the Tresca yield condition, and  $g(\theta k)$  is the slope of the uniaxial stress-strain curve at  $\sigma = \theta k$ , i.e.,

$$g(\sigma) = \frac{d\sigma}{d\varepsilon} \quad (7)$$

with  $\varepsilon$  the longitudinal stress. In general,  $H(k)$  is a function of  $k$ .

Eqs.(3,4) may be write in the conservation form if the longitudinal and shear strains,  $\varepsilon$  and  $\gamma$ , are introduced:

$$\frac{\partial u}{\partial x} = \frac{\partial \varepsilon}{\partial t}, \quad (8)$$

$$\frac{\partial v}{\partial x} = \frac{\partial \gamma}{\partial t}. \quad (9)$$

Here

$$d\varepsilon = \left[ \frac{1}{E} - H(k)(\sigma/\theta)^2 \right] d\sigma + H(k)\sigma \tau d\tau, \quad (10)$$

$$d\gamma = \left[ \frac{1}{G} - H(k)(\theta \tau)^2 \right] d\tau + H(k)\sigma \tau d\sigma. \quad (11)$$

Let us first look at the basic equations at three special cases. If a specific problem involves only the elastic region,  $g(\theta k) = E$  and  $H = 0$ . Then the basic equations are linear that represent the propagation of linear longitudinal and torsional waves. In the second special case for the linearly isotropic work-hardening material,  $g(\theta k)$  have only two values,  $g(\theta k) = E$  in the elastic region, and  $g(\theta k) = E_p$  (constant) in the plastic region. A problem involving only longitudinal waves,  $\tau = 0$  and  $\gamma = 0$ , is the third special case, in which the set of basic equations is reduced to a system representing the propagation of nonlinear longitudinal waves.

We would like to point out that the system is not, strictly speaking, a system of conservation laws since the equation of state, Eqs.(10,11), are in inexact differential forms. The dynamics of a problem, in principle, is dependent on the internal structure of the discontinuity. We will discuss this point in numerical examples.

Our numerical scheme is based on the characteristic formulations of Eqs.(1-4). The characteristic formulations for Eqs.(1-4) was first discussed by Clifton<sup>4</sup>, and also discussed by several previous investigators<sup>5-8,22</sup>. The set of Eqs.(1-4) allows two kinds of waves, fast and slow waves, with their wave speeds  $C_f$  and  $C_s$  respectively:

$$C_{f,s}^2 = \frac{1}{\rho Q} [A + B \pm \sqrt{(A - B)^2 + 4H\sigma\tau}]. \quad (12)$$

Here  $A$ ,  $B$  and  $Q$  are defined as

$$A \equiv \frac{1}{G} + H(\theta \tau)^2,$$

$$B \equiv \frac{1}{E} + H(\sigma/\theta)^2,$$

$$Q \equiv AB - (H\sigma\tau)^2,$$

and the plus (or minus) sign is for fast (or slow) waves. The wave speeds  $C_f$  and  $C_s$  satisfy the relation

$$C_s \leq C_2 \leq C_f \leq C_0.$$

Here  $C_0$  (or  $C_2$ ) is the elastic longitudinal (or shear) velocity,  $C_0 \equiv \sqrt{E/\rho}$  and  $C_2 \equiv \sqrt{G/\rho}$ .

Each wave speed  $C(x, t)$ , which may be either  $-C_f$ , or  $-C_s$ , or  $C_s$ , or  $C_f$ , defines a characteristic curve in  $(x, t)$ -space,

$$\frac{dx}{dt} = C(x, t).$$

Following the general outline in the book by *Courant and Friedrichs*<sup>20</sup>, for all the characteristic curves, we may find their associated Riemann invariants, which may be expressed only as inexact differentials:

$$dR_{f\pm} = -\frac{1}{\theta\sigma} \{ (\pm\rho C_f du - d\sigma) - \alpha(\pm\rho C_f dv) - d\tau \}, \quad (13)$$

$$dR_{s\pm} = -\frac{1}{\tau} \{ (\pm\rho C_s dv - d\tau) - \beta(\pm\rho C_s du - d\sigma) \}. \quad (14)$$

Here the plus (or minus) sign is for the wave propagating toward the positive (or negative)  $x$ -direction, and  $\alpha$  and  $\beta$  are defined as

$$\alpha \equiv \frac{\theta^2 \tau (C_f/C_0)^2 - 1}{\sigma (C_f/C_2)^2 - 1}, \quad (15)$$

$$\beta \equiv \frac{\sigma (C_s/C_0)^2 - 1}{\theta^2 \tau (C_s/C_2)^2 - 1}. \quad (16)$$

### 3. Numerical Scheme

For the conciseness in formulations, we first write Eqs.(1,2,8,9) in the form

$$\frac{\partial \mathbf{U}}{\partial t} - \frac{\partial \mathbf{F}}{\partial x} = 0, \quad (17)$$

where

$$\mathbf{U} \equiv \begin{pmatrix} \rho u \\ \rho v \\ \varepsilon \\ \gamma \end{pmatrix}, \quad \mathbf{F} \equiv - \begin{pmatrix} \sigma \\ \tau \\ u \\ v \end{pmatrix}.$$

Consider a numerical zone  $x_i < x < x_{i+1}$ , we write Eq.(17) in a difference form:

$$\mathbf{U}_i(\delta t) = \mathbf{U}_i(0) + \frac{\delta t}{\delta x_i} (\bar{\mathbf{F}}_i - \bar{\mathbf{F}}_{i+1}). \quad (18)$$

Here  $\delta t$  is the time step,  $\delta x_i$  is the width of the zone,  $\mathbf{U}_i(t)$  is the average of  $\mathbf{U}$  over the zone at time  $t$ , and  $\bar{\mathbf{F}}_i$  is the time-averaged flux at the interface  $x_i$  during the time step, i.e.,

$$\mathbf{U}_i(\delta t) \equiv \frac{1}{\delta x_i} \int_{x_i}^{x_{i+1}} \mathbf{U}(x, t) dx, \quad (19)$$

$$\bar{\mathbf{F}}_i \equiv \frac{1}{\delta t} \int_{\delta t} \mathbf{F}(x_i, t) dt.$$

The difference scheme, Eq.(18), is called a Godunov scheme for Eq.(17), which may be obtained by integrating Eq.(17) over the rectangular  $x_i < x < x_{i+1}$  and  $0 < t < \delta t$  in the  $(x - t)$ -space. The discretized form Eq.(18) is exact if the time-averaged flux  $\bar{\mathbf{F}}_i$  may be exactly found. Thus one of the key issues for this type of schemes is to obtain the flux at the interfaces of numerical zones.

Before introducing the numerical scheme further, we would like to mention the Riemann problem. A Riemann problem is an initial value problem, Eq.(17), subject to a specific initial condition:

$$\mathbf{U}_0(x) = \begin{cases} \mathbf{U}^{(l)} & \text{if } x < 0 \\ \mathbf{U}^{(r)} & \text{if } x > 0. \end{cases}$$

Here  $\mathbf{U}^{(l)}$  and  $\mathbf{U}^{(r)}$  are any two constant states.

We consider a variable continuous inside each numerical zone, but may have a jump across an interface of numerical zones. We would like to first calculate the time-averaged flux at an interface for the constant left and right states. For given left state  $(\sigma^l, \tau^l, u^l, v^l, \varepsilon^l, \gamma^l)$  and right state  $(\sigma^r, \tau^r, u^r, v^r, \varepsilon^r, \gamma^r)$ , the time-averaged values  $\bar{\sigma}$ ,  $\bar{\tau}$ ,  $\bar{u}$  and  $\bar{v}$  are found through the following set of linear algebra equations:

$$[\bar{\sigma} - \sigma^l] - \rho C_{f0} [\bar{u} - u^l] + \alpha_l [\bar{\tau} - \tau^l] - \rho \alpha_l C_{f0} [\bar{v} - v^l] = 0, \quad (20)$$

$$[\bar{\sigma} - \sigma^l] - \rho C_{s0} [\bar{u} - u^l] + \beta_l [\bar{\tau} - \tau^l] - \rho \beta_l C_{s0} [\bar{v} - v^l] = 0, \quad (21)$$

$$[\bar{\sigma} - \sigma^r] + \rho C_{f0} [\bar{u} - u^r] + \alpha_r [\bar{\tau} - \tau^r] + \rho \alpha_r C_{s0} [\bar{v} - v^r] = 0, \quad (22)$$

$$[\bar{\sigma} - \sigma^r] + \rho C_{s0} [\bar{u} - u^r] + \beta_r [\bar{\tau} - \tau^r] + \rho \beta_r C_{s0} [\bar{v} - v^r] = 0. \quad (23)$$

Here the coefficients are defined as

$$C_{f0} \equiv \frac{1}{2} [C_f^l + C_f^r], \quad C_{s0} \equiv \frac{1}{2} [C_s^l + C_s^r], \quad (24)$$

$$\alpha_l \equiv \frac{1}{2} \text{sign}[\alpha^l] [|\alpha^l| + |\alpha^r|], \quad (25)$$

$$\alpha_r \equiv \frac{1}{2} \text{sign}[\alpha^r] [|\alpha^l| + |\alpha^r|],$$

$$\beta_l \equiv \frac{1}{2} \text{sign}[\beta^l] [|\beta^l| + |\beta^r|],$$

$$\beta_r \equiv \frac{1}{2} \text{sign}[\beta^r] [|\beta^l| + |\beta^r|].$$

The superscript  $l$  or  $r$  stands for the evaluation of at the left (or right) state. We would like to mention two points here. First, Eqs.(20-23) come from the invariance of Riemann invariants along characteristic curves. Second, we always introduce very small values  $\sigma^*$ ,  $\tau^*$  and  $H^*$  for  $\sigma$ ,  $\tau$  and  $H$  when they are vanishing in order to avoid the mathematical singularity in Eqs.(20-23).  $\sigma^*$ ,  $\tau^*$  and  $H^*$  are so small that their influence on the system is negligible and are sufficient large compared to the accuracy of digits for an actual computer. In actual simulations in this paper, we use  $\sigma^* = \sigma^y \times 10^{-8}$ ,  $\tau^* = 0.4 \sigma^*$ , and  $H^* = (1/E \times 10^{-8})^3$ .

From the consideration of a higher order of accuracy, there is an internal structure inside each numerical zone. Our numerical scheme starts from zone-averages  $a_i$  ( $i = 1, 2, \dots, n$ ) of each of the variables  $(\sigma, \tau, u, v, \varepsilon, \gamma)$ , which are defined by Eq.(19). For each of these variables,  $a$ , cubic polynomials are used to interpolate  $a$  to find the values at

interfaces. For a uniform grid, the value at the left interface of the  $i^{th}$  zone,  $a_{i,l}$ , is found to be

$$a_{i,l} = a_{i-1} + \frac{1}{2}(a_i - a_{i-1}) - \frac{1}{6}(\delta a_i - \delta a_{i-1}). \quad (26)$$

Here  $\delta a_i$  is defined as  $\delta a_i \equiv (a_{i+1} - a_{i-1})/2$ . For a nonuniform grid, the value at the interface is

$$a_{i,l} = a_{i-1} + f_a(a_i - a_{i-1}) + f_{da}\delta a_i + f_{dal}\delta a_{i-1}. \quad (27)$$

Here  $\delta a_i$  is defines as

$$\delta a_i \equiv g_{dal}(a_i - a_{i-1}) + g_{dar}(a_{i+1} - a_i),$$

and  $f_a$ ,  $f_{da}$ ,  $f_{dal}$ ,  $g_{dal}$  and  $g_{dar}$  are geometry factors related to the nonuniform grid:

$$g_{dal} = \frac{\delta x_i(2\delta x_{i+1} + \delta x_i)}{(\delta x_{i-1} + \delta x_i)(\delta x_{i+1} + \delta x_i + \delta x_{i-1})},$$

$$g_{dar} = \frac{\delta x_i(2\delta x_{i-1} + \delta x_i)}{(\delta x_i + \delta x_{i+1})(\delta x_{i+1} + \delta x_i + \delta x_{i-1})},$$

$$f_{da} = \frac{-\delta x_{i-1}(\delta x_{i-2} + \delta x_{i-1})}{(2\delta x_{i-1} + \delta x_i)(\delta x_{i-2} + \delta x_{i-1} + \delta x_i + \delta x_{i+1})},$$

$$f_{dal} = \frac{\delta x_i(\delta x_i + \delta x_{i+1})}{(2\delta x_i + \delta x_{i-1})(\delta x_{i-2} + \delta x_{i-1} + \delta x_i + \delta x_{i+1})},$$

$$f_a = \frac{\delta x_{i-1} - 2(\delta x_i f_{da} + \delta x_{i-1} f_{dal})}{\delta x_{i-1} + \delta x_i}.$$

After we obtain the values at interfaces, a monotonicity constraint originally suggested by *Van Leer*<sup>10</sup> is applied to these values at interfaces. As we know, interpolated structures are not always monotone increasing (decreasing) even though they have been constructed from monotone data. The over- and under-shoots in the interpolated internal zone structures would eventually result in to over- and under-shoots in the zone-averaged data. *Van Leer* realized that a scheme could be made to preserve the monotonicity of its initial data if any non-monotone interpolated zone structures are flattened so that they become monotone. This leads the *Van Leer's* monotonicity constraint: no values interpolated within a zone shall lie outside the range defined by the zone averages for this zone and its two neighbors.

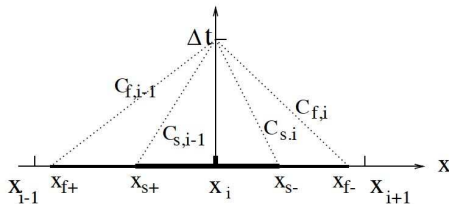


Fig. 2: An illustration for domains of dependence for fast and slow waves.  $(x_i - x_{f+})$  and  $(x_i - x_{s+})$  are the domains of dependence for the fast and slow waves propagating rightward, and  $(x_{f-} - x_i)$  and  $(x_{s-} - x_i)$  are the domains for the fast and slow waves propagating leftward.

Because of the different wave speeds, different waves have different domains of dependence, as shown in Fig.2. For the interface between  $(i-1)^{st}$  and  $i^{th}$  zones, the domain-average of  $a$  over the domain  $|C|dt$  for a wave with the speed  $C$  is found to be following. For  $C > 0$ ,

$$a_d = a_{i-1,r} - \frac{\sigma}{2} \left[ a_{i-1,r} - a_{i-1,l} - a_{i-1,6} \left( 1 - \frac{2\sigma_{i-1}}{3} \right) \right]. \quad (28)$$

For  $C < 0$ ,

$$a_d = a_{i,l} + \frac{\sigma}{2} \left[ a_{i,r} - a_{i,l} + a_{i,6} \left( 1 - \frac{2\sigma_i}{3} \right) \right]. \quad (29)$$

Here  $\sigma_i$  is the Courant number  $|C| \delta t / \delta x_i$  evaluated at the  $i^{th}$  zone.

After obtaining the domain-averages, we may increase the order of accuracy in the calculation of the time-averaged flux, Eqs.(20-23), through the replacement of the left state in Eqs.(20) [or Eq.(21)] by the domain-averages for fast (or slow) waves obtained from Eq.(29), and through the replacement of the right state in Eqs.(22) [or Eq.(23)] by the domain-averages for fast (or slow) waves calculated from Eq.(30). After the calculation of time-averaged flux,  $\sigma$ ,  $\tau$ ,  $\varepsilon$  and  $\gamma$  are updated through Eq.(18) during a time step. The stresses  $\sigma$  and  $\tau$  may be updated through solving the ordinary differential equations which are obtained through dividing Eqs.(10,11) by the time step.

## 4. Numerical Examples

In this section, we will provide a few numerical examples to demonstrate the correctness and robustness of the scheme. The spatial length is normalized by the length of the tube  $L$ , the time is normalized by  $L/u_0$ , where  $u_0 \equiv \sqrt{1(kg/m^3)/1Gpa}$ . A grid with 200 uniform numerical zones is used in all the examples except the last one. The Courant safety number,  $\delta t C_f / \delta x$ , is set to 0.8. Here  $\delta x$  is the width of the numerical zone,  $\delta t$  is the time step. The time step is automatically adjusted to satisfy the Courant safety condition. We use commercially pure aluminum in the simulations, for which  $\rho = 0.0975 \text{ lb/in}^3$ ,  $E = 10^7 \text{ psi}$ ,  $G = 3.7594 \times 10^6 \text{ psi}$ , the yield stress in the simple tension  $\sigma_y = 1750 \text{ psi}$ . The uniaxial stress-strain curve in the simple tension is assumed to be

$$\varepsilon = \begin{cases} \frac{\sigma}{E} & \text{if } \sigma < \sigma_y \\ \frac{\sigma}{E} + 4.03 \times 10^{-9}(\sigma - \sigma_y)^{1.732} & \text{if } \sigma \geq \sigma_y. \end{cases} \quad (30)$$

Here the stress is in the unit of *psi*.

One of important tests for a numerical scheme is for the propagation of a simple wave that is smooth initially. Because of the nonlinearity of Eqs.(1-4), a simple wave with a sufficiently large amplitude will become steepening. During the steepening process, all other Riemann invariants should remain constants in each smooth region. The nonzero of other Riemann invariants in a smooth region is the symptom of numerical errors.



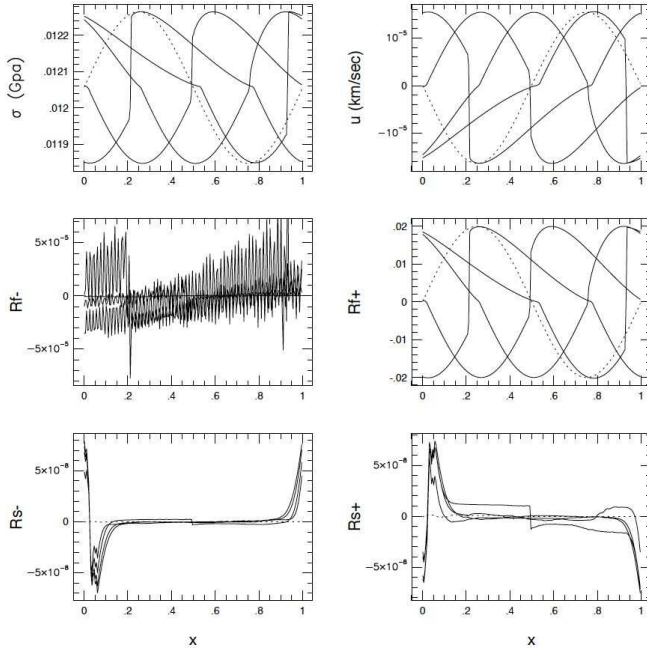


Fig. 3: The propagation of a longitudinal wave. The initial condition are shown by the dashed lines. The solid lines show the profiles at  $t = 0.15, 0.3, 0.45, 0.6$ . During the interval  $\delta t = 0.15$ , the wave traveled a little more than a half wavelength.

The first example is for the propagation of a longitudinal wave without torsion. The initial condition is set through solving the ordinary differential equations:

$$\frac{dR_{f\pm, s\pm}}{dx} = 2\pi A_{f\pm, s\pm} \cos(2\pi x), \quad (31)$$

with  $A_{f+} = 0.02, A_{f-, s\pm} = 0$ , and  $\sigma = \sigma_y, \tau = u = v = 0$  at  $x = 0$ . The initial profiles are shown through the dashed lines in Fig.3, which is partially in the elastic region ( $\sigma < \sigma_y$ ) and partially in the plastic region ( $\sigma > \sigma_y$ ). Since initial profiles for  $R_{f\pm, s\pm}$  shown in the figure are obtained from the approximate calculation from  $\sigma(x)$  and  $\tau(x)$ , they are not exactly constants. The solid lines in Fig.3 show the profiles at  $t = 0.15, 0.3, 0.45, 0.6$ . During the interval  $\delta t = 0.15$ , the wave traveled a little more than a half wavelength. The wave becomes fairly steep after it traveled one wavelength. The Riemann invariants  $R_{f-}$  and  $R_{s\pm}$  should be constant in each smooth region, but they actually not because of numerical errors.

The second example is the propagation of a fast wave in combined compression and torsion. Again, the initial conditions are obtained from Eq.(32) with  $A_{f+} = 0.8, A_{f-, s\pm} = 0$ , and  $\sigma = \sigma_y, \tau = \sigma_y/\theta, u = v = 0$  at  $x = 0$ . Dashed lines in Fig.4 are the initial conditions. The solid lines in the Fig.4 show the profiles at  $t = 0.125, 0.25, 0.375, 0.5$ . During the interval  $\delta t = 0.125$ , the wave traveled about a half wavelength. The Riemann invariants  $R_{f-}$  and  $R_{s\pm}$

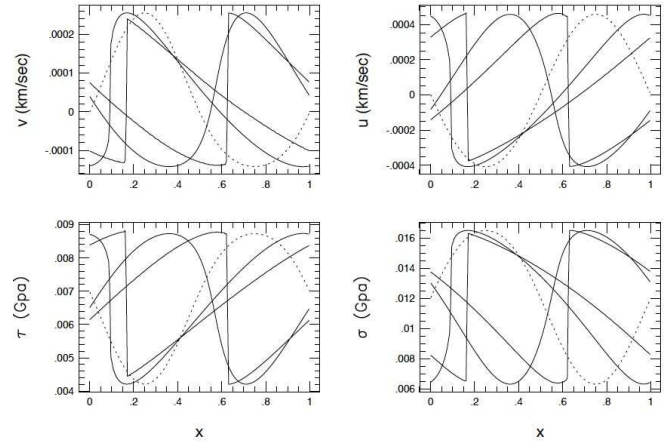


Fig. 4: The propagation of a fast wave. The initial condition are shown by the dashed lines. The solid lines show the profiles at  $t = 0.125, 0.25, 0.375, 0.5$ . During an interval  $\delta t = 0.125$ , the wave traveled about a half wavelength.

remain within the interval 0.001 in each smooth region in this example.

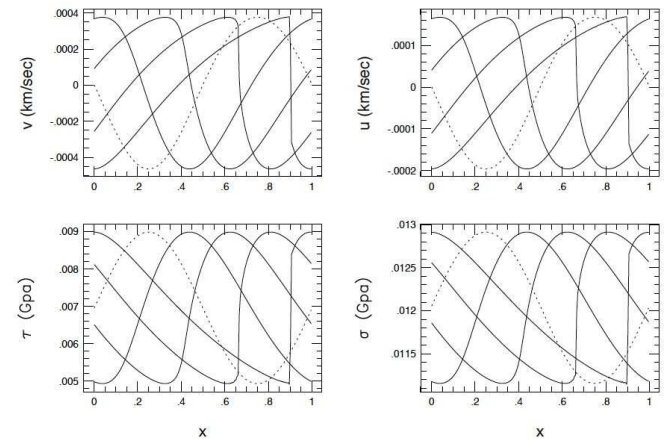


Fig. 5: The propagation of a slow wave. The initial condition are shown by the dashed lines. The solid lines show the profiles at  $t = 0.125, 0.3, 0.375, 0.5$ .

The third example is the propagation of a slow wave in combined compression and torsion. The initial conditions are determined by Eq.(32) with  $A_{s+} = 0.4, A_{f\pm} = A_{s-} = 0$ , and  $\sigma = \sigma_y, \tau = \sigma_y/\theta, u = v = 0$  at  $x = 0$ , which are shown by the dashed lines in Fig.5. The solid lines in Fig.5 show the simulation results at  $t = 0.125, 0.3, 0.375, 0.5$ . The Riemann invariants  $R_{f-}$  and  $R_{s\pm}$  remain within the interval 0.0005 in each smooth region in this example.

Normally, a step loading is considered as the only situation where discontinuous waves will be developed because of the impulsive acceleration at one end of the tube, and because of the fact  $d^2\sigma/d\varepsilon^2 \leq 0$  for solid materials. But, the three examples above show that a smooth elastic-plastic wave may

become discontinuous without any external force. The reason for the steepening of a smooth wave is the relative large wave speed in the valley, and relative small wave speed in the peak of the stress profile. A single simple wave is not frequently the situation in real problems, but real problems frequently contain simple waves.

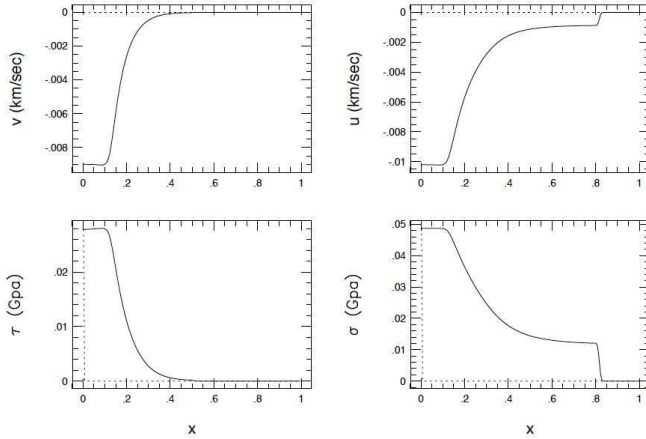


Fig. 6: A problem with a step loading. The fixed stresses at the end  $x = 0$  are  $\sigma_0 = 7\sigma_y/\theta$  and  $\tau_0 = 4\sigma_y/\theta$ . The solid lines are the profiles at  $t = 0.16$ .

Now we turn to simulations for the step loading. Without the complete set of initial and boundary conditions used in numerical simulations, it is difficult to compare results from two numerical schemes. Two examples given here are stimulated by the paper<sup>4</sup>. We consider a tube initially at rest and unstressed which, at the end  $x = 0$ , is simultaneous subjected a constant normal stress  $\sigma_0$  and a constant shear stress  $\tau_0$ . Thus the initial and boundary conditions are

$$\sigma(x, 0) = \tau(x, 0) = 0, \quad (32)$$

$$u(x, 0) = v(x, 0) = 0.$$

$$\sigma(0, t) = \sigma_0 \quad \text{for} \quad 0 < t < \infty, \quad (33)$$

$$\tau(0, t) = \tau_0 \quad \text{for} \quad 0 < t < \infty. \quad (34)$$

In our simulations, the continuation boundary condition are used at the end  $x = 1$ , and the continuation boundary condition for  $u$  and  $v$  and the fixed boundary condition for  $\sigma$  and  $\tau$  are used at the end  $x = 0$ .

In the first step loading problem,  $\sigma_0 = 7\sigma_y/\theta$  and  $\tau_0 = 4\sigma_y/\theta$ . The solution at  $t = 0.16$  is shown in Fig.6. As described by Clifton<sup>4</sup>,  $\sigma$  jumps to  $\sigma_y$  across the wave front propagating with the speed  $C_0$ ;  $\sigma$  increases across the fast wave region to  $\hat{\sigma}$ , while  $\tau$  remains zero. Here  $\hat{\sigma}$  is the stress for which the slope of the uniaxial stress-strain curve is equal to the shear modulus  $G$ , and  $\hat{\sigma} = 1.82588 \times 10^3 \text{ psi}$  for the material;  $\sigma$  and  $\tau$  increase across the slow wave region until the stresses  $\sigma_0$  and  $\tau_0$  are reached;  $\sigma$  and  $\tau$  are constant in the constant state region.

Finally, we would like to discuss Riemann problems in the model. As we said before, since equations of state, Eqs.(10,11), are in inexact differential forms, the solution of a Riemann problem may depend on the internal structure of the initial discontinuity unless the initial stresses are in the elastic region. To show the point, we give two numerical examples here, in which the initial condition for a variable  $a(x)$  is in the form:

$$a(x) = \frac{1}{2} \left\{ \left[ 1 + \tanh \frac{x - 0.5}{w} \right] a_r + \left[ 1 - \tanh \frac{x - 0.5}{w} \right] a_l \right\}. \quad (35)$$

Here  $a_l$  and  $a_r$  are two constant values for the left and right states, and  $w$  is the width of the internal structure.

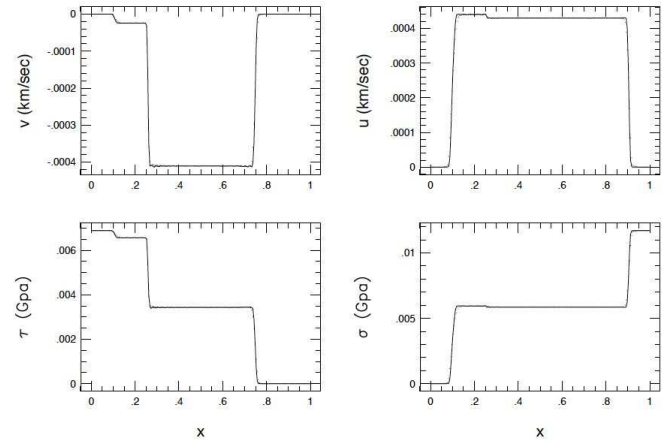


Fig. 7: A Riemann problem. Initially,  $(\sigma, \tau, u, v)$  in the left and right states are respectively  $(0, \tau_l, 0, 0)$  and  $(\sigma_r, 0, 0, 0)$ . Here  $\tau_l = 10^3 \text{ psi}$  and  $\sigma_r = 1700 \text{ psi}$ . The initial stresses are in the elastic region. The solid (or dotted) lines show the simulation results at  $t = 0.08$  for the initial width  $w = 0$  (or  $w = 0.01$ ). Two sets of profiles are fundamentally the same.

In the first Riemann problem, the initial condition for  $(\sigma, \tau, u, v)$  is  $(0, \tau_l, 0, 0)$  for the left state, and is  $(\sigma_r, 0, 0, 0)$  for the right state. Here  $\tau_l = 10^3 \text{ psi} < \sigma_y/\theta$ ,  $\sigma_r = 1700 \text{ psi} < \sigma_y$ . The initial stresses are shown by the dashed lines in Fig.7, which are in the elastic region [ $H(t = 0) = 0$ ]. The solid lines in the figure show the simulation results when the width  $w$  is set to zero, and the dotted lines are the results for a finite width  $w = 0.01$ . Two sets of profile in the figure do not show significant differences since the initial stresses are in the elastic region. Even if we increase the width to  $w = 0.05$ , there is no fundamental difference in the result except the widths of resulting discontinuities. Note that the stresses may be in the plastic region during the evolution as shown by the constant state near  $x = 0.2$  even the initial stresses are in the elastic region.

The second Riemann problem is for the initial condition in which the stresses are in the plastic region. The initial condition for  $(\sigma, \tau, u, v)$  is  $(5\sigma_y, \sigma_y/\theta, 0, 0)$  for the left state, and is  $(\sigma_y, 5\sigma_y/\theta, 0, 0)$  for the right state. We use

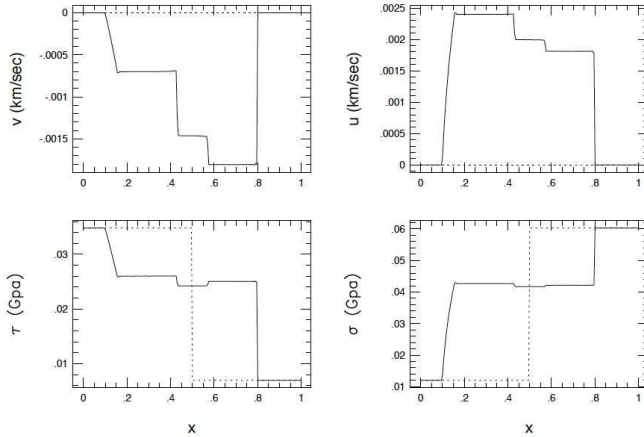


Fig. 8: A Riemann problem. 1000 zones. Initially,  $(\sigma, \tau, u, v)$  in the left and right states are  $(\sigma_y, \sigma_y/2\theta, 0, 0)$  and  $(\sigma_y/2, \sigma_y/\theta, 0, 0)$  respectively, as shown by the dashed lines. The solid lines show the simulation results at  $t = 0.08$  for the vanishing width  $w$  (actually,  $w = 10^{-10}$ )

1000 numerical zones for a better illustration on the Riemann problem. The initial profiles with the vanishing width  $w$  (actually,  $w = 10^{-10}$ ) and the simulation results at  $t = 0.08$  are shown by the dashed and solid lines in Fig.8. The solid (or dotted) lines in Fig.10 show the simulation results at  $t = 0.08$  with the width  $w = 0.002$  (or  $w = 0.005$ ) for the same Riemann problem. Three sets of profiles in Figs. 9 and 10 with different widths are clearly different.

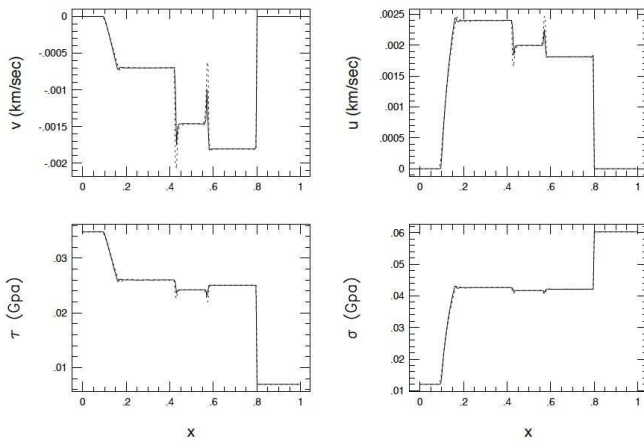


Fig. 9: A Riemann problem described for Fig.8, but with small internal structures of the initial discontinuity. 1000 zones. The solid (or dashed) lines are the results for the initial width  $w = 0.002$  (or  $w = 0.005$ ) at  $t = 0.08$ . The solutions with different internal structures are different.

## 5. Summary

In this paper we have developed a high-order Godunov scheme for combined longitudinal and torsional elastic-

plastic waves in thin-walled tubes. The scheme is based on the characteristic theory of the system. The scheme uses uniform formulations for all loading paths, which make it easy to actually implement the scheme. The scheme has been tested through the steepening of nonlinear waves, the problems for the step loading. The correctness and robustness of the scheme have been demonstrated through these problems. It is shown that the scheme keeps the principle advantages of Godunov schemes, i.e., the robust operation in the presence of strong waves, thin fronts of discontinuous waves with little attendant noise generated. The scheme may be used to investigate one-dimensional combined longitudinal and torsional elastic-plastic waves in thin-walled tubes.

The Riemann problem in the model has been discussed in this paper. It is turned out that the solution of a Riemann problem may depend on the internal structure of the initial discontinuity.

Unlike most numerical schemes for hyperbolic systems, we have not added any additional numerical viscosity in this paper, since we intend to present only the essential part of the scheme. An artificial viscosity may be necessary to reduce the attendant noise generated in post-wave states of discontinuities. It should be pointed out that a Godunov scheme, without any numerical viscosity added, already contains error terms that produce numerical diffusion of this type of schemes.

## References

- [1] T. C. Ting, "Plastic wave propagation in linearly work-hardening materials," *J. of Applied Mech.*, vol. 40, pp. 1045-1049, 1973.
- [2] S. K. Godunov, "Difference methods for the numerical calculation of the equations of fluid dynamics," *Math. Sb.*, vol. 47, pp. 101-136, 1959.
- [3] B. Van Leer "Toward the ultimate conservative difference scheme, V. a second-order sequel to Godunov's method," *J. Comput. Phys.*, vol. 32, pp.101-136, 1979.
- [4] P. L. Roe, "Approximate Riemann solvers, parameters vectors, and difference schemes," *J. Comput. Phys.*, vol. 43, pp. 357-372, 1981.
- [5] P. Colella and P. R. Woodward, "The piecewise parabolic method (PPM) for gas-dynamical simulations," *J. Comput. Phys.*, vol. 54, pp. 174-201, 1984.
- [6] A. Harten, "High resolution schemes for hyperbolic conservation," *J. Comput. Phys.*, vol. 49, pp. 357-393, 1983.
- [7] K. Nagayama, "Solution of the high-pressure Riemann problem for solids including rigity effects," *J. Phys. Soc. Japan.*, vol. 58, pp. 1631-1638, 1989.
- [8] J. A. Trangenstein and R. B. Pember, "The Riemann problem for longitudinal motion in an elastic-plastic bar," *Lawrence Livermore National Laboratory, UCRL-101214.*, 1989.
- [9] J. A. Trangenstein and P. Colella, "A higher-order Godunov method for medeling finite deformation in elastic-plastic solids," *Comm. Pure and Appl. Math.*, vol. XLIV, pp. 41-100, 1991.
- [10] B. J. Plohr and D. H. Sharp, "A conservative formulation for plasticity," *Adv. Appl. Math.*, vol. 13, pp. 462-493, 1992.
- [11] J. G. Glimm, B. J. Plohr, and D. H. Sharp, "A conservative formulation for plasticity," *Appl. Mech. Rev.*, vol. 46, pp. 519-526, 1993.
- [12] X. Lin and J. Ballmann, "A Riemann solver and second-order Godunov method for elastic-plastic wave propagation in solids," *Int. J. Impact Engng.*, vol. 13, pp. 463-478, 1993.
- [13] R. Courant and K. O. Friedrichs, "*Supersonic Flow and Shock Wave, 5th ed.*" Interscience, New York, 1967.
- [14] G. Strang, "On construction and comparison of differential schemes," *SIAM J. of Num. Anal.*, vol. 5, pp. 06-517, 1968.

# A Vocal Tsunami

P. Diez<sup>1</sup>, J. N. Harvey<sup>1</sup>, L. J. Falsetta<sup>2</sup>, S. Boudot<sup>2</sup>, T. Zhang<sup>2</sup>, R. Y. Lee<sup>2</sup>

<sup>1</sup>Department of Mathematics, Central Michigan University, Mount Pleasant, MI, U.S.A.

<sup>2</sup>Department of Computer Science, Central Michigan University, Mount Pleasant, MI, U.S.A.

**Abstract**— *Voice recognition has become one of the leading forms of biometric identification, and it is important to ensure that this technique is as accurate as possible. The main algorithm for single word voice recognition is Dynamic Time Warping (DTW). In this study, we have approached the idea of DTW from a multiple input structure. By performing an analysis on  $n - 1$  control samples and 1 test sample, we propose to take into account subtle variations in speech pattern to increase accuracy of testing. Through the application of the Tsunami Algorithm to an  $n$ -dimensional structure, we compute the shortest weighted path that will identify the relation between the samples. Our results show that comparison of paths with an  $n$ -dimensional diagonal give a more accurate method of single word verification than that of DTW.*

**Keywords:** DTW, nDTW, Tsunami, voice, recognition

## 1. Introduction

In voice recognition, there are several methods requiring a voice recording. A few key approaches are MFCC [1] and HMM [2], which include creating or training a matrix and testing for correctness. The other primary technique is Dynamic Time Warping (DTW), and this is where we will focus. The last approach is using clustering techniques, Vector Quantization [3] can be implemented for speaker identification.

DTW is used to compare a voice test sample and control sample to determine if the samples were recorded by the same person. This analysis is usually conducted by finding a minimal weighted path through two dimensional matrix; however, it has not been analyzed in higher dimensions [4]. This is why we chose to research multidimensional analysis of speech recognition using DTW.

High error rate is a major hurdle in voice recognition. There are several challenges that affect the accuracy of testing, including poor sample quality, background noise, and mood of the speaker [5], speed of the algorithm [6], and other factors. Eliminating false rejections, as well as increasing the accuracy of recognition amongst similar voices or words spoken, would greatly increase the potential for security as well as making speech-to-text programs more viable. To come up with a more accurate interpretation of data involved in voice recognition, we propose a combination of already successful algorithms will improve the accuracy of voice recognition and other applications of DTW, such as

signature-recognition [7]. Through multidimensional extrapolation of DTW, which we call nDTW, this can be achieved. As we will demonstrate, this study provides greater accuracy than classic DTW and, hopefully, brings the field one step closer to finding a unique voice signature for a person.

The paper is organized in the following structure:

- Section 2 provides an survey of our developed program.
- Section 3 presents basic information related to DTW.
- Section 4 lays out our approach using nDTW.
- Section 5 details the results of our approach.
- Section 6 addresses the conclusions of our results.
- Section 7 offers thoughts on how to extend our ideas.

## 2. Architecture of the Program

From Figure 1, the architecture of this project is similar to that of DTW [8], [9]. This should hardly be surprising, since we are just expanding DTW to take into account multiple control samples. The input files are a sequence  $\{V_1, V_2, \dots, V_{n-1}\}$  of voice recordings from the same person saying the same word and a single test  $T$ . For notation purposes throughout the paper, we will use  $V_i(j)$  to indicate the  $j$ th sampling on the  $i$ th input recording.

After loading these samples, we recursively generate an  $n$ -dimensional matrix  $M$ , the process of which will be explained in detail in Section 4.1. The element with indexes all one will be called  $M_0$  and the element with each index being a maximum in that dimension will be  $M_\alpha$ . We then use the Tsunami Algorithm to compute weight and distance from  $M_0$  for each element of  $M$ , which conducts computation in an order that allows for threading. The algorithm is outlined in Section 4.2. A minimal weighted path  $P_w$  is then found from  $M_0$  to  $M_\alpha$  by backtracking from  $M_\alpha$ . This path is compared to a minimal unweighted path  $P_u$  from  $M_0$  to  $M_\alpha$ , or rather the  $n$ -dimensional diagonal between these elements.

By adding up the minimal Euclidean distance from each point of  $P_w$  to a point of  $P_u$ , we create a numerical value  $D$  for testing similarity of voice recordings. If  $D$  is greater than some predetermined cutoff value, the program rejects the test sample.

## 3. Background and Related Work

Before learning about the necessary mathematical and computational structures, we need an understanding of DTW and how it works. This section presents information from



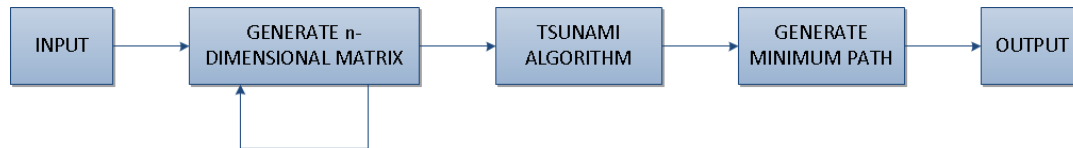


Fig. 1: Refined Structure of nDTW

works related to DTW, as well as some basic information on sound and Euclidean distances.

### 3.1 Aspects of Voice Recognition with DTW

DTW is an algorithm that calculates an optimal warping path between two time series and is used for measuring similarities between sequences which may vary in time or speed. DTW can be applied to more fields than just voice recognition. For example, a race car going around a road course changes speed many times and timing equipment is generally not evenly spaced. Both of these would cause problems if looking for patterns in how consistent a driver is. However, by making use of DTW, these issues are irrelevant and you can quickly see a difference in consistency between an F1 driver and your neighbor, Steve. Any sort of data that can be interpreted or represented in a linear graphical form can have DTW applied to it. For this reason, it makes DTW ideal for the field of voice recognition, where we are concerned with what you say and not how quickly you say it.

Summarizing from Senin [8], the algorithm begins with two time series  $X = (x_1, x_2, \dots, x_N)$  and  $Y = (y_1, y_2, \dots, y_M)$  with  $N, M \in \mathbb{N}$ . For a particular application, a distance function  $d: X \times Y \rightarrow \mathbb{R} \geq 0$  is chosen. A local cost matrix  $C$ , which is a  $N \times M$  matrix with real entries, is calculated with  $C_{i,j} = d(x_i, y_j)$ .

A warping path  $P = (p_1, p_2, \dots, p_L)$  is a sequence of index pairs which meets the following conditions:

- Boundary:  $p_1 = (1, 1)$  and  $p_L = (N, M)$ .
- Monotonicity: If  $p_i = (a, b)$  and  $p_{i+1} = (c, d)$ , then  $a \leq c$  and  $b \leq d$ .
- Step-Size: There is a predefined set of steps  $S$  such that for each  $i$ ,  $p_{i+1} = p_i + \beta$  for some  $\beta \in S$ .

The minimal weight warping path through  $C$  will be denoted  $P_w$  and is often called the Optimal Warping Path. The conditions on warping paths allow us to create a weighted digraph with no loops, one source and one sink and we are tasked with finding the shortest path from source to sink.

Searching through all possible  $P$  for the optimal path would be a daunting task, so DTW employs Dynamic Programming to perform the search in  $O(NM)$  time. Since the elements of  $C$  are nonnegative, Dijkstra's algorithm will allow us to quickly find the optimal path. However, due to the predictable nature of our digraph, modifications can be made to speed this process further.

### 3.2 Properties of Sound

As stated by Rabiner, sound travels through the environment as a longitudinal wave, having a speed that is dependent on the density of the environment. A sinusoidal (defined as having a succession of waves or curves) graphic is the easiest way to represent sounds in this way. The graphic will present a variation of air pressure depending on the time series. Rabiner goes on to explain that the shape of the sound wave depends on these three factors [10]:

- Amplitude: The amplitude is the displacement of the sinusoidal graph above and below the temporal axis ( $y = 0$ ) and it corresponds to the energy the sound wave contains.
- Frequency: The frequency is the number of cycles the sinusoid makes every second.
- Phase: The phase measures the position from the beginning of the sinusoidal curve.

A point of frequency or phase, together with the amplitudes, is called a spectrum. Spectrogram is the way to show it with time, which determines the amplitude intensity. With the feature of the 'naive' DTW algorithm, this can use input source properly, which has a major role in voice recognition. [8]

Myers, along with Rabiner and Rosenberg explain that the majority of modern technologies of recognition for the human voice are to analyze the spoken word. Word identification can be done by doing applying a straight comparison of the numeric representation of the signals spectrogram comparison or with just the signals themselves. In each of these cases the process of comparison needs to compensate for the different length of the sequences and non-linear nature of the sound. The DTW Algorithm succeeds in sorting out these problems by finding the warp-path corresponding to the optimal distances between two series of different lengths [11].

### 3.3 Euclidean Distance

In analyzing optimal warping path, Euclidean distance to the diagonal is the most common comparison. Euclidean distance is the geometric distance in a multidimensional space. For  $x, y \in \mathbb{R}^n$ , the Euclidean distance is defined as:

$$D^2(x, y) = \sum_{i=1}^n (x_i - y_i)^2,$$

and has been used in data mining, especially in speech recognition process. Even though some have shown this is

Word	Subject Relation	DTW Distance	3-DTW Distance	Equivalent
Ball	A, A, A	247.24199	0.00000	Yes
Ball	A, A, A	26.271126	5.000000	Yes
Phone, Phone, Box	B, B, A	15956.785179	18418.914388	No
Cheer, Cheer, Cheer	A, A, A	0.000000	2.828427	Yes

Fig. 2: Results Comparison

not as efficient as the weighted cepstral distance, it is still the standard.

## 4. nDTW

As mentioned Sections 1 and 2, nDTW requires the generation of an  $n$ -dimensional array of sample data, costs and distances. Once generated, the optimal warping path is determined and analyzed.

### 4.1 Generating an $n$ -Dimensional Array

To perform our  $n$ -dimensional DTW, we must first create an  $n$ -dimensional array. To generalize, the number of samples  $n$  to be studied and length of each sample  $\{L_1, L_2, \dots, L_n\}$  are determined at runtime. So, a method of generating a matrix of unknown dimensions is needed.

Algorithm 1 accomplishes this through simple recursion

---

**Algorithm 1** Generates an  $n$ -dimensional array.

---

**Require:** A sequence  $L \subset \mathbb{N}$  of length  $n$

- 1: **function** MATRIXGEN( $L$ )
  - 2:   **if**  $n = 1$  **then**
  - 3:      $v \leftarrow 0$
  - 4:   **else**
  - 5:      $v \leftarrow \text{MATRIXGEN}(L_2, \dots, L_n)$
  - 6:   **return** An array of  $L_1$  copies of  $v$
- 

and we denote this  $n$ -dimensional matrix  $C$ . When writing  $M(X)$ , we will mean  $X$  is a list of indexes, so  $M(X)$  is a particular element of  $M$ .

### 4.2 Tsunami Algorithm

The Tsunami Algorithm is a modification to Dijkstra's algorithm that exploits the predictability of our digraph which, along with threading, allows for speedier computation. Three types of threads are used in this algorithm; a controller thread, an edge thread and a sheet thread. Since each type behaves differently, we must address them as separate algorithms.

Before beginning the algorithm, we should address some notation. Assume we are performing an analysis on samples  $\{V_k\}_{k=1}^n$  with lengths  $\{L_k\}_{k=1}^n$ . Let  $\{e_i\}_{k=1}^n$  be the standard ordered basis of  $\mathbb{R}^n$ . That is, for each  $i$ ,  $e_i$  is an  $n$ -tuple with a 1 in the  $i$ th position and 0 elsewhere. Let  $S$  be the set of possible steps for warping paths, which will be the set of all binary  $n$ -tuples except the zero tuple.

Similar to DTW, we require a distance function

$$d : V_1 \times V_2 \times \dots \times V_n \rightarrow \mathbb{R} \geq 0.$$

For this particular implementation, it will be defined by

$$\begin{aligned} d_0(X) &= \sum |V_i(x_i) - V_j(x_j)|, \\ d_1(X) &= d_0(X) + \min_{s \in S} \{d_0(X - s)\}, \\ d(X) &= d_1(X) + \min_{s \in S} \{d_1(X - s)\}. \end{aligned}$$

Now, with this notation in hand, we are ready to lay out each algorithm.

The simplest method of evaluation relies on incrementing only one dimension.

---

**Algorithm 2** Evaluates from  $v$  along line parallel to  $e_j$

---

**Require:** A vector  $v$  and integer  $j$

- 1: **function** EDGET( $v, j$ )
  - 2:   **for**  $i \leftarrow 1, (L_j - v_j)$  **do**
  - 3:      $M(v + ie_j) \leftarrow d(v + ie_j)$
- 

Instead of incrementing one index, then resetting it to zero and incrementing another by 1 and repeating the process, we can make use of threading technologies to perform computation on independent nodes simultaneously. For a 2D array, we proceed down the main diagonal and spawn edge threads to evaluate in each direction away from the diagonal. This is generalized to a 2D sheet of  $M$  in Algorithm 3.

---

**Algorithm 3** Controller for two dimensional evaluation

---

**Require:** A vector  $v$  and integers  $j, k$

- 1: **function** SHEET( $v, j, k$ )
  - 2:    $u \leftarrow \min\{L_j, L_k\}$
  - 3:    $l \leftarrow \max\{v_j, v_k\}$
  - 4:   **for**  $i \leftarrow 1, (u - l)$  **do**
  - 5:      $M(v + i(e_j + e_k)) \leftarrow d(v + i(e_j + e_k))$
  - 6:     Thread: EDGET( $v + i(e_j + e_k), j$ )
  - 7:     Thread: EDGET( $v + i(e_j + e_k), k$ )
- 

With simple evaluation methods defined, we can generalize to the  $n$ -dimensional structure. Proceeding down the  $n$ -dimensional diagonal, we spawn an edge thread parallel to each  $e_i$ . Once we have begun processing the edges of a 2D sheet, spawn a sheet thread to evaluate within that sheet. This step is given explicitly in Algorithm 4.

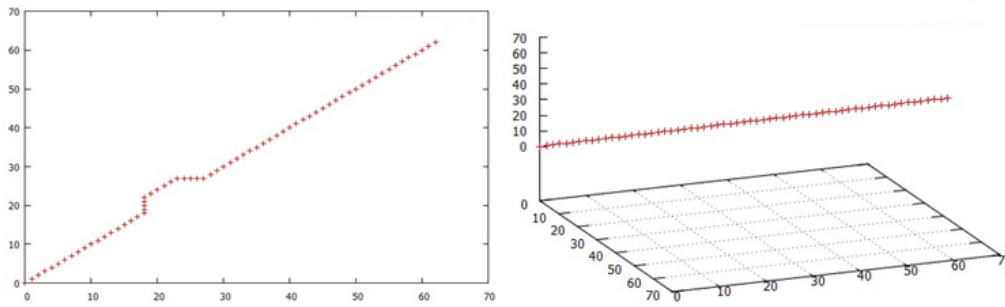


Fig. 3: Visual DTW/3DTW Minimum Path Comparison

**Algorithm 4** Controller for Tsunami algorithm

---

```

1: function MAINT
2:    $v \leftarrow \sum e_i$ 
3:   for  $m \leftarrow 1, \min\{n_i\}$  do
4:      $v' \leftarrow m \cdot v$ 
5:      $M(v') \leftarrow d(v')$ 
6:     for  $j \leftarrow 1, n$  do
7:       Thread: EDGET( $v', j$ )
8:       for  $k \leftarrow 1, (j - 1)$  do
9:         Thread: SHEET( $v', j, k$ )

```

---

Computing distances in this fashion, when reaching  $X$  we have already found  $d(X - s)$  for each  $s \in S$ , so  $d(X)$  can be determined while minimizing excess work. The standard method of evaluation has a single process calls  $d \prod L_k$  times, which can be time consuming. In systems that support threading, these processes are divided among threads which perform at most  $\max\{L_k\}$  evaluations.

Caution should be taken when implementing this algorithm due to the sheer number of threads which can be created. Although we begin with a single thread, a system can quickly be overwhelmed when  $n$  or some  $L_k$  is sufficiently large.

### 4.3 Finding the Minimal Cost Path

After completing the algorithm above,  $M$  contains the minimal distance from the source to each point. By backtracking from the sink to the source and greedily finding the minimal distance of a previous node as in [13], we find  $P_w$ , the optimal warping path. With this path in hand, we can determine if the test sample matches the set of control samples.

### 4.4 Comparison

The purpose of voice recognition is to compare who is speaking a set of control samples to who is speaking a test sample. So, we need to determine if we reject an inaccurate test sample. To do this, we compare  $P_w = (p_1, p_2, \dots, p_L)$

to the  $n$ -dimensional diagonal of  $M$  using a Euclidean distance as detailed in Section 3.3, similar to DTW.

Let  $\Gamma$  be the line through  $M$  with all indexes being the same for each point. Given  $p_i = (t_1, t_2, \dots, t_n)$ , we seek the minimum distance between  $p_i$  and a point on  $\Gamma$ . For each  $p \in \Gamma$ , all the indexes are some  $t$ , so

$$D^2(p_i, p) = \sum_{j=1}^n (t_j - t)^2.$$

Differentiating with respect to  $t$  gives

$$2D(p_i, p) = \sum_{j=1}^n (t - t_j) = nt - \sum_{j=1}^n t_j.$$

Setting to zero and solving for  $t$  gives

$$t = \frac{1}{n} \sum_{j=1}^n t_j.$$

In other words, the point on  $\Gamma$  which is closest to  $p_i$  has indexes equal to the average of the indexes of  $p_i$ .

This gives us a comparison equivalent to standard statistical methods, the mean square error. In fact, we are trying to determine how well  $\Gamma$  (predicted path) models  $P_w$  (actual path).

## 5. Results and Analysis of the Study

Several voice samples from two subjects, A and B, were recorded. Each word was recorded 3 times from each subject and many words were tested.

Several trials are shown in Figure 2, which shows greater accuracy for matches and greater discrepancy for non-matches. Although tabular evidence is nice, 3 sample testing finds a path in 3D space, which lends itself well to plotting.

One trial is shown in Figure 3, which should be a perfect match. The left plot is the path generated by DTW and shows a problematic region around the 20th sampling. Adding a second control sample smooths out the rough edges on positive matches and gives the straight path in 3D space on

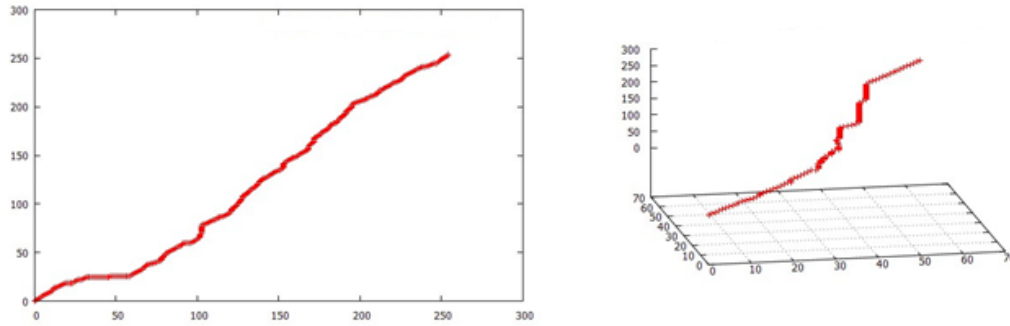


Fig. 4: Visual DTW/3DTW Error Comparison

the right. So, given a small cut off value, we may reject in the case of DTW, but fail to reject when adding additional samples.

In Figure 4, DTW generates the very rough path on the left, which is expected since the test should yield a rejection. However, introducing a second control produces the rougher (and harder to see) plot on the right. So, there are cut off values for which we would fail to reject using DTW, but reject with additional samples. This is the type of error most harmful to security.

## 6. Conclusion

A method to study and evaluate a dimensional expansion of DTW has been introduced. Once an  $n$ -dimensional matrix has been created and the Tsunami Algorithm has been applied to find an optimal warping path, we see how well the diagonal fits this path. The mean square error provides a numerical value for how well the test sample matches the control group. It has been shown that adding more control samples gives an improvement in accuracy.

## 7. Further Study

Additional study and testing is required to determine optimal sample sizes and count. Such knowledge would greatly improve efficiency of voice identification using nDTW.

Also, a relation may be found that relates a test sample to the optimal warping path through a matrix of control samples. This would allow the majority of processing to occur before testing takes place. Current testing of three control samples and one test sample is on the order of minutes, which is hardly reasonable for accessing a secure system.

## References

- [1] R. Hasan, M. Jamil, G. Rabbani, and S. Rahman, "Speaker identification using mel frequency cepstral coefficients," *International Conference on Electrical & Computer Engineering*, pp. 565–568, December 2004.
- [2] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, October 1989.
- [3] H. B. Kekre and V. Kulkarni, "Speaker identification by using vector quantization," *International Journal of Engineering Science and Technology*, vol. 2, no. 5, pp. 1325–1331, 2010.
- [4] M. Slaney, M. Covell, and B. Lassiter, "Automatic audio morphing," *International Conference on Acoustics, Speech, and Signal Processing*, 1996.
- [5] B. Schuller, A. Batliner, S. Steidl, and D. Seppi, "Recognising realistic emotions and affect in speech: State of the art and lessons learnt from the first challenge," *Speech Comm.*, 2011.
- [6] S. Salvador and P. Chan, "Fastdtw: Toward accurate dynamic time warping in linear time and space," *Florida Institute of Technology*.
- [7] R. Niels, "Dynamic time warping: An intuitive way of handwriting recognition?" Master's thesis, Radboud University Nijmegen, Nijmegen, The Netherlands, 2004.
- [8] P. Senin, "Dynamic time warping algorithm review," *Information and Computer Science Department, University of Hawaii at Manoa*, December 2008.
- [9] Z. Fang, W. Wenhui, and F. Ditang, "A log-index weighted cepstral distance measure for speech recognition," *Journal of Computer Science and Technology*, vol. 12, no. 2, pp. 177–184, March 1997.
- [10] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Prentice Hall, 1978.
- [11] C. Myers, L. R. Rabiner, and A. E. Rosenberg, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING*, vol. 28, no. 6, pp. 623–635, December 1980.
- [12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Mit Press, 2001.
- [13] T. F. Furtuna, "Dynamic programming algorithms in speech recognition," *Informatica Economica*, vol. 2, no. 46, pp. 94–99, 2008.



# Fuzzy Inference for Decision Support with fuzzyMorphic.pl

Wagner Arbex<sup>1</sup>, Marta Martins<sup>1</sup>, Marcos Vinícius Silva<sup>1</sup> and Luis Alfredo Carvalho<sup>2</sup>

<sup>1</sup>Brazilian Agricultural Research Corporation, Juiz de Fora, MG, Brazil

<sup>2</sup>Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil

**Abstract**—*Problems when dealing with imprecise or uncertain features, e.g., decision-making problems, can be designed as fuzzy systems, since these systems allow the processing of subjective and qualitative argument, usually intrinsic to such problems. This text presents the fuzzyMorphic.pl – a tool for developing of fuzzy inference systems which enable the specification of the modeling and the implementation of these systems – and its use for fuzzy inference decision models applied to decision support assistance.*

**Keywords:** Fuzzy inference, fuzzy inference system, decision support system, fuzzyMorphic.pl

Several times, the decision support problems cannot be handled by mathematics or logics classical approaches because these traditional methods do not have the necessary techniques or tools for that.

Generally, decision support problems have imprecise or uncertain features - which are sometimes intrinsic - so it becomes necessary to use different approaches with more appropriate tools, like fuzzy inference or fuzzy inference systems.

This text is organized into two parts. Initially, it explains why the fuzzy inference approach rather classical approaches should be used. Therefore, it shows the doubt which can happen in a classification problem and how the fuzzy approach can be more appropriate to deal this kind of problem, when the membership degree concept is used to assign “intensities” to the possible results as a decision support attribute. Next, it presents the use of fuzzy inference from computational modeling for decision support, by means of a set of inference rules.

The second part of this text presents the fuzzyMorphic.pl, a software for developing fuzzy inference systems which enables the specification of modeling of several kinds of fuzzy problems through its own statements and syntax, as briefly described in the fourth section.

The contents of this paper are completed by this introduction, which presents some observations on the problem at hand, and by the conclusion, which summarizes the most important points from text.

## 1. Fuzzy Inference Approach

Classical approaches are insufficient to resolve problems with imprecise or uncertain features, such as problems with results which have values very close to some limit.

Therefore, in circumstances like these, mathematical and logically accurate results, but questionable, can be found.

### 1.1 Classical Approach Fuzzyness Features

A hypothetical classification problem could be to admit as a person of “medium height”, the individual with height between 1.65 *m* and 1.75 *m*. Additionally, individuals which height less than 1.65 *m* will be considered as “short height” and individuals greater than 1.75 *m* will be considered as “higher height”. And, assuming two individuals with heights between 1.66 *m* and 1.74 *m* as classified within the “medium height” range.

This decision, logically and mathematically precise, can be questioned, because of the subjectivity involved. Both values, 1.66 *m* and 1.74 *m*, are much closer to the limits of their class than between them. Furthermore, they are so close to the limits of the class which they belong to those different interpretations can be taken when classifying these values.

However, traditional approaches of logics and mathematics do not have the necessary tools to handle threshold values, or even imprecision or uncertainty. Specifically, threshold values result in doubt in the “decision” to classify the individual with respect to their heights, which suggests a fuzzy inference system for handling this uncertainty.

Usually, the threshold values problem is not as simple as it may seem, was it simple, the classical approaches could easily solve it, but, the closer to the subjective reasoning for the interpretation and the extraction of an answer or a decision, the more complex it becomes and the apparent simplicity is given by fuzzy logic modeling and by its basis in the theory of fuzzy sets.

### 1.2 Fuzzy Approach

The subjectivity inherent to reasoning makes it possible to deal with complex situations, which are based on inaccurate, uncertain or approximate information and, therefore, the strategy is to use human operators of an also imprecise nature, which are expressed in linguistic terms or variables.

For instance, common adjectives representing imprecision or uncertainty, such as *high*, *low*, *more*, *less*, or even cluster relation, as *the tall people set*, cannot be expressed using traditional approaches, unless the concept or value determining, in this case, the threshold for a person’s height has been exactly defined, the persons can be considered “tall”.

However, there is no difficulty in understanding what “being tall” entails if analyzer only two people’s conversation.

Such proposal, essentially human, in order to describe or handle problems, generally, does not enable a solution in terms or exact numbers, but, for instance, leads the solution to a qualitative classification, clustering or aggregating into categories or possible solutions set [1]. These solutions can be seen as a result of the “principle of incompatibility” [2].

Fuzzy approach proposes that “intensities” to the data are determined, when comparing them to the thresholds set out, which is equivalent to establishing a membership degree to the information, ranging between 0 and 1 with respect to the sets which the information may belong to.

## 2. Decision Making with Fuzzy Inference

The subjectivity inherent to reasoning is capable of dealing with complex situations, based on inaccurate, uncertain or approximate information and, therefore, the strategy is to use human operators of an also imprecise nature, which are expressed in linguistic terms or variables.

Such proposal, essentially human, in order to describe or handle problems, generally, does not enable a solution in terms or exact numbers, but, for instance, leads the solution to a qualitative classification, clustering or aggregating into categories or possible solutions sets [1]. These solutions can be seen as a result of the “principle of incompatibility” [2].

The linguistic terms or variables increase the complexity of traditional models and computational systems concerning the ability to handle are numbers, exact and discrete values which sometimes mutually exclusive, suggesting the idea of working with uncertain values, enabling the modeling of complex systems, even if they reduce the accuracy of the result, but not losing credibility.

If uncertainties, when viewed in isolation, are undesirable, when they are associated with other characteristics, they generally allow the reduction of system complexity and increase the credibility of the results [3].

Fuzzy sets theory and fuzzy logics are appropriate to represent, in mathematical terms, the inaccurate information which can be expressed by a set of linguistic rules. And if there is the possibility that human operators are organized as a set of conditional statements of the

*if ANTECEDENT then CONSEQUENT*

form; therefore, subjective reasoning can be expressed in the form of computationally executable algorithms [4] [2] with the ability to imprecisely classify the variables of the antecedents and consequents conditional statements, as qualitative concepts, instead of quantitative, which represents the idea of linguistic variable [1].

Thus, as systems capable of efficiently processing inaccurate and qualitative information, fuzzy inference models are suitable in situations which require decision making [1].

## 3. The fuzzyMorphic.pl

Fuzzy inference systems are implementations of able models to efficiently compute inaccurate and qualitative information, thus, for instance, they are appropriate for problems which require decision making [1].

### 3.1 Introducing the fuzzyMorphic.pl

The fuzzyMorphic.pl is a software tool for modeling and implementation of fuzzy inference systems – developed on Perl language – for which it is possible:

- 1) for fuzzyfication, to represent the membership functions on standard format sets – like trapezoidal or triangular shapes;
- 2) for implementation of inference machine, to use Mamdani’s or Larsen’s models;
- 3) for defuzzyfication, to represent output function on standard format sets; and
- 4) to use Center of Maxima as defuzzyfication method.

Two information sets are needed for fuzzyMorphic.pl to work, which could be in a single flat file or in different files. The first set refers to the description directives of the inference model, as a whole, and the second one is the input data itself.

The fuzzyMorphic.pl has the advantage of not having been developed for any specific problem, so, it is possible to utilize it to develop fuzzy inference systems for several inference problems and models.

Furthermore, its description inference systems way allow an easy the investigation of a problem under several and different respects and it is very important and intended in research procedures.

In the other hand, a constraint of the fuzzyMorphic.pl is to enable defuzzyfication by a single method, the Center of Maxima method.

Originally, the fuzzyMorphic.pl was developed as a part of a research project called “Computational models for the identification of genomic information associated to the resistance to cattle tick” [5] and was used to implement a fuzzy inference system for decision support assistance, from verification and analysis of two previous results.

The fuzzyMorphic.pl is in 1.0 version, release 20090111, and it needs the fuzzyInference.pm module, currently in 1.1 version, and in the same release.

### 3.2 The fuzzyMorphic.pl Structure

Generally, the structure of fuzzy inference systems has three large sections:

- 1) fuzzyfication, to converter of crisps values to fuzzy values;
- 2) inference, to run the machine inference with the inference rules; and
- 3) defuzzyfication, to convert outcome, from fuzzy values to crisps values;

and these structure can be implemented with the fuzzyMorphic.pl through of its descriptions directives, as explain them below.

The description directives are organized in six groups and if they are in the same input data flat file, then they have to occupy the first lines of such file. These directives describe the input data file, define the membership functions, set up relations among crisp values and membership functions, define the inference rules, set up the inference model, and define the output file.

As a result, under these conditions, from flat file with description directives of input data and system model elements, fuzzyMorphic.pl can easily do data mining and discover knowledge. Actually, it can infer knowledge from the inference rules described in the directives.

The six directives groups are explained below:

- 1) the first group has the *\_ID* identifier, is composed of just one directive and must be the first row of the directive file, but it can be preceded by blank or comment lines. This directive shows format file, including field separator, fields themselves, fields which participate of the fuzzy inference model as crisp variables and the result field;
- 2) the second group defines the membership functions. Each row represents one membership function and it must begin with *\_F<sub>i</sub>* (*i* = 1, 2, ...) for each function;
- 3) the third group assigns crisp variables – reported in the first directive group – with their membership function. As is known, the same membership function can be associated with more a one crisp variable. Each row determines one relationship between a crisp variable and a membership function;
- 4) the fourth group lists the inference rules set. Each row brings one rule and must begin with *\_R<sub>i</sub>* (*i* = 1, 2, ...) for each rule;
- 5) the fifth group is composed of just one directive and it defines the inference model. This directive must informs *\_IM : Mamdani* or *\_IM : Larsen*, depending on the choice of inference machine for the model;
- 6) the sixth group is composed of just one directive too and it defines the output function. This row must begin with *\_O* and should be followed by output function description.

The observations below should also be considered when using the fuzzyMorphic.pl:

- if a row has with a “#”, then all text wrote below it will be considered comments and this text will be ignored;
- all blank lines or lines with only space characters, such as tab characters, will be ignored;
- the input file fields must be separated with a field separator, as described in the *\_ID* directive;

- the *\_ID* directive must be the first directive, but the other directives can listed in any order or be mixed and matched;
- if the directives and data are in the same file, then all of the set of description directives – such as the six groups of directives – must be before the first input data record.

## 4. Conclusions

Generally, fixed and precise classification criteria are not suitable when studies show results which are very close to a certain limit, for instance, a division into classes. But, these cases can be approached by fuzzy inference systems, which are also convenient, as well as able, to handle problems characterized by uncertainty and imprecision for decision making actions.

Furthermore, problems featuring imprecision or uncertainty can be designed as fuzzy systems, since these systems allow the processing of subjective and qualitative arguments, usually intrinsic to such problems.

The fuzzyMorphic.pl is a tool for developing fuzzy inference which enables model description and implementation of fuzzy inference systems for solving different problems. Although, originally, the fuzzyMorphic.pl was developed to implement a fuzzy inference model decision support applied to bioinformatics, specifically for the identification of single nucleotide polymorphisms, based on results from two other single nucleotide polymorphisms discovery tools.

More information about this software or the inference model proposed can be seen in the quoted research project “Computational models for the identification of genomic information associated to the resistance to cattle tick” [5].

## References

- [1] P. E. M. de Almeida and A. G. Evsukoff, “Sistemas fuzzy,” in *Sistemas inteligentes: fundamentos e aplicações*, S. O. Rezende, Ed. Barueri: Manole, 2005, pp. 169–202.
- [2] L. A. Zadeh, “Outline of a new approach to the analysis of complex systems and decision processes,” *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 28–44, 1973. DOI: 10.1109/TSMC.1973.5408575. [Online]. Available: <http://www-bisc.cs.berkeley.edu/Zadeh-1973.pdf>.
- [3] G. J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: theory and applications*. Upper Saddle River: Prentice Hall, 1995, p. 592, ISBN: 0131011715.
- [4] R. Tanscheit, “Sistemas fuzzy,” in *Inteligência computacional: aplicada à administração, economia e engenharia em Matlab*, H. A. e Oliveira Júnior, Ed. São Paulo: Thomson Learning, 2007, pp. 229–264.

- [5] W. Arbex, "Computational models for the identification of genomic information associated to the resistance to cattle tick," Systems Engineering and Computer Science Program, PhD thesis, Federal University of Rio de Janeiro, Rio de Janeiro, 2009.

**SESSION**  
**SCIENTIFIC COMPUTING**

**Chair(s)**

**Prof. Hamid R. Arabnia**



# Trading Space for Time: Constant-Speed Algorithms for Managing Future Events in Scientific Simulations

Clarence Lehman<sup>1</sup>, Adrienne Keen<sup>2</sup>, and Richard Barnes<sup>1</sup>

<sup>1</sup>University of Minnesota, 123 Snyder Hall, Saint Paul, MN 55108, USA

<sup>2</sup>London School of Hygiene and Tropical Medicine, Keppel St., London WC1E 7HT, UK

*“It is a mistake to try to look too far ahead. The chain of destiny can only be grasped one link at a time.”*  
—Winston Churchill

**Abstract**—Given vast increases in computing capacity, applications in science and engineering that were formerly interpreted with ordinary or partial differential equations, or by integro-partial differential equations, can now be understood through microscale modeling. Interactions among individual particles—be they molecules, viruses, or individual humans—are modeled directly, rather than first abstracting the interactions into mathematical equations and then simulating the equations. One approach to microscale modeling involves scheduling all events into the future, wherever that is possible. With sufficient space-for-time tradeoffs, this considerably improves the speed of the simulation, but requires scheduling algorithms of high efficiency. In this paper we describe our variation on calendar queues and their usage, presenting detailed algorithms, intuitive explanations of the methods, and notes from our experiences applying them in large-scale simulations. Results can be useful to scientists in ecology, epidemiology, economics, and other disciplines that employ microscale modeling.

**Keywords:** microscale modeling, discrete event simulation, calendar queues, pending events set, space-time tradeoff

## 1. Introduction

The obvious approach to model a large number of discrete interacting entities, hereinafter called “individuals,” is to emulate what is done to model continuous systems with differential equations. That is, select a small time step  $\Delta t$ , compute how the system will change during the interval  $\Delta t$ , update the system with those changes, then advance to the next time step. In ordinary differential equations, as the time step shrinks, the dynamics of the simulated system converge to the correct behavior. This is “macroscale modeling,” following Euler’s method or its many variations [1]. With a model of 100 compartments, representing, for example, 100 age classes in a human population, relatively few dynamical variables must be examined and updated in each time step.

The same approach works with microscale modeling, though with difficulties. At each time step, each individual is examined to determine what interactions will occur during

that time step. The difficulty with this approach is twofold. First, each individual acts as a dynamical variable, so there can be many millions or hundreds of millions of variables to be examined and updated in each time step. Moreover, as the time step shrinks to assure convergence, it becomes exceedingly unlikely that anything will happen to a given individual during the time step. Therefore, in contrast with its macroscale counterpart, that approach to microscale modeling spends most of its time checking and finding nothing to do.

Inspiration for a faster approach comes from an alternative method of solving differential equations. Instead of determining what will happen during the present small time step, an algorithm can determine at what time in the future the next event will occur. This can be determined reliably for the very next event, and the precise process for doing so is called Gillespie’s method [2]. It is the complement of the standard method.<sup>1</sup>

Despite certain epistemological difficulties about projecting the future that are beyond the scope of the present paper, hinted at in Churchill’s statement above, Gillespie’s method can be extended to determine possible times for all future events in many dynamical systems of scientific interest—or at least all events that control the fate of the system. But the number of future events can be large, with many events per individual, and the number of individuals in the simulation may be tens or hundreds of millions or more.

Fortunately, algorithms are known that are extremely efficient at handling schedules of future events. Discovered by Randy Brown in 1988 [3], these are called “calendar queues” or “pending event sets,” and have been undergoing successive refinements ever since (e.g. [4] [5] [6] [7]). They have the desirable—and remarkable—property that their speed is independent of the number of events scheduled.

<sup>1</sup>In simulating  $f(x) = dx/dt \approx \Delta x/\Delta t$ , a small time step  $\Delta t$  can be established, such as 0.01 seconds, and the change in population (or other simulated quantity) can be estimated as  $\Delta x \approx f(x)\Delta t$ . That is Euler’s method. Alternatively, a change in quantity  $\Delta x$  can be specified (such as a population growth of one individual) and the time for that to occur can be estimated as  $\Delta t \approx \Delta x/f(x)$ . That is Gillespie’s method. Thus the mathematics for the two are complementary.

Adding an event, canceling one, or finding the next event about to occur is the same whether the schedule contains 100 events or 100 million. That is, they are “Order 1” algorithms.

In this paper we present our adaptation of calendar queues to large-scale individual-based modeling in epidemiology. Lessons should be applicable to areas including ecology, economics, and other physical and natural sciences. We attempt to make our presentation intuitive for access by scientists and other readers outside computer science. The goals of this paper are to (1) review the idea of space-for-time trade-offs that have become widely useable and applicable to other algorithms (e.g. [8]), (2) explain our variation on calendar queues and their incorporation in microscale simulations, and (3) present our algorithms in full detail for use and adaptation by others.

## 2. Space for time

The persistent increase in random access computer memories has carried algorithms through a “phase change,” wherein a slow continuous advance in memory sizes has resulted in a rapid, almost abrupt, change in some of the rules for constructing algorithms for scientific programs. If it will speed processing, computer algorithms can now afford to allocate hundreds of millions of bytes of empty space—even if that space will never be used. This is a “space-for-time tradeoff.” With large memories now available, such allocation is no longer wasting memory. On the contrary, leaving memory unused, or leaving it applied to insignificant purposes, is wasting it.

A basic space–time tradeoff arises with numerical keys. Suppose we have 10,000 items, each identified by a distinct six-digit “key,” and with keys randomly distributed among values from ‘000000’ to ‘999999’. Suppose each of the 10,000 items occupies 100 memory cells (e.g. 100 bytes). Stored contiguously, this will require  $10^4 \times 10^2 = 10^6$  memory cells. In such a compact arrangement, searching can be relatively fast if the entries are kept in numeric order<sup>2</sup>. However, in this case adding and deleting will be slow, averaging  $N$  or more accesses to keep the list contiguous and in order. On the other hand, if the entries are left in random order, adding and deleting will be fast, 1 to 3 accesses only,<sup>3</sup> but searching will be slow, sequentially checking each entry until the right one is encountered. The point is, this minimal-space approach inevitably results in algorithms that are slow in one respect or another.

An alternative is to “waste” memory by allocating one slot in memory for each of the million entries possible. Now to search for a specific six-digit key, say key ‘314159’, the algorithm merely goes directly to the 314,159th entry of the table. Only one access to the memory array is thus needed

<sup>2</sup>For instance, by using a binary search algorithm, which is of Order  $\log_2 N$  accesses, where  $N$  is the number of items in the list

<sup>3</sup>New entries can be added at the end in 1 access; deleted entries can be swapped with the entry at the end in 3 accesses.

to retrieve, and the same is needed to add or delete. With 10,000 active entries, this space–time tradeoff speeds the algorithm 5,000 fold. However, it comes at the expense of 100-million memory cells, about one-tenth of a gigabyte. Such cavalier abandon in the use of memory would have been unthinkable until recently, but if speed is the utmost criterion, then allocating an extra 1/10 GB to accomplish a multi-thousand-fold increase in speed is the clear and proper choice.

This approach extends to larger keys through the method of “hash coding,” which is directly related to calendar queues. Hash coding is an Order-1 algorithm known at least since Arnold Dumey in 1956 [9]. The key may be an individual’s first, last, and middle name, for which the space required for direct access would be astronomical, beyond the power of any computer presently foreseeable. Even if the key was only a nine-digit social security number, such as 123-45-6789, providing one direct-access entry for all possible social security numbers would be prohibitively large.

The simplest solution merely extracts the rightmost six digits of the social security number and indexes an array of a million entries with those six digits. Of course, as many as 1,000 individuals may share the same last six digits of their social security numbers, so “collisions” can occur. But with only 10,000 entries of a million active, and assuming all possible social security numbers are equally likely, each entry in the array has only a 0.01 chance of being occupied, so the chance that two or more individuals will occupy the same cell is very small. Nonetheless, the possibility of collisions must be provided for, and a variety of practical methods have been devised [10]. Once that is done, locating an individual by social security number, or indeed by first, last, and middle name, can be accomplished in one access, or arbitrarily close to one access, with a sufficiently large space-for-time tradeoff.

Dumey’s scheme [9] was to use a modulus operation by considering the key to be a large number, dividing it by the size of the memory array (number of entries in the array), then discarding the quotient and using the remainder to index the array—as in the social security example. In that case, the rightmost six digits were equivalent to the remainder after division by one million. Essentially the same underlying scheme is applied in calendar queues, dividing the scheduled time by the size of the memory array (one year’s worth of minutes in the intuitive example to follow), and using the remainder to index the array. Therefore, the same space-for-time tradeoffs that make hash-coded accesses maximally fast also can make calendar queues, properly programmed, maximally fast for managing large numbers of future events.

## 3. Future events

Having emphasized the value of spending memory to buy time, we must also say that it is pointless to spend memory



when it does not buy time. The more events that are scheduled at once, the greater the amount of memory that is needed to handle them efficiently, in direct proportion to the number scheduled. Also, the more that the number of events scheduled vary during the simulation, the more frequently the data structures should be optimized by “resizing” [3].

Therefore, to help keep the scheduling algorithms efficient, our microscale simulation programs withhold all but one event per individual from them. Characteristics of individuals are maintained in a large array of data structures,  $A[n]$ , indexed by individual number  $n$ , which ranges from one to some maximum value. This array includes data of two types: (1) information about the individual, such as, in a model of human events, date of birth, sex, geographic location, and so forth, and (2) a list of all future events relevant to that individual. This large array is not processed nor examined by the scheduling routines described in this paper.

Only the earliest among the events pending for each individual is entered into the global schedule, with the data structure  $A[n]$  holding the rest. Such withholding of information has several benefits: (1) the number of events managed by these algorithms is considerably reduced, (2) the number of events that must be canceled and rescheduled is reduced, and (3) the size of the scheduling data structures are predictable, with precisely one event per individual. This partly obviates the need for the scheduling algorithms to maintain separate lists for near, intermediate, and far future events, as in some variations of calendar queues [11], and also eliminates the need for time-consuming “resizing” operations [3].

## 4. Intuitive view

We want to (1) schedule new events, (2) cancel existing events, and (3) notify a dispatcher as the time for each event arrives—all three with maximal efficiency. The coding details can be subtle, but the overall operation is not. It can be understood intuitively through a physical analogy.

Assume, for a specific illustration, that half a million events are to be scheduled over the next five years, and that they appear more-or-less randomly throughout that period. Suppose that each event has a ticket with (1) a unique event number and (2) a scheduled time, represented at least to the nearest second, but possibly much finer.

Now consider a series of pigeon-hole bins to contain the tickets, one bin representing each minute of an entire year. The first bin represents the first minute after midnight on New Year’s Day, the second bin represents the second minute, and so forth to the last bin, which represents the last minute on December 31st. That is  $366 \text{ days} \times 24 \text{ hours/day} \times 60 \text{ minutes/hour} = 527,040$  bins total, each labeled with the month, day, hour, and minute that it represents. Each bin also has a flag that can be lowered or raised according to whether the tickets in the bin are known to be in

chronological order. We assumed half a million events to be scheduled, less than one event per bin on average.

### 4.1 Creating a new event

Events are created as the simulation proceeds, each associated with a particular individual and with a precisely assigned time, usually stochastically assigned. In an ecological model these may represent a time of birth or death, in an epidemiological model they may represent the time of onset of a disease, or the time for transmission to another individual. In any case, new events arise frequently during the simulation. The procedure for scheduling a new event is quite easy:

1. Go to the bin representing the month, day, hour, and minute for the event. Although the year, second, and any fraction of a second are not used to select a bin, they are later used to place events in precise chronological order.
2. Drop the event’s ticket on top of the others in the bin.
3. Raise the flag on the bin to indicate that its tickets may no longer be in chronological order.

That required only a single operation, regardless of how many events were in the bin. We take it to be important merely to drop the ticket atop others in the bin, as above, rather than trying to sort it into place among other tickets in the bin. Earlier implementations of calendar queues [3] keep all bins always sorted, but that can be disabling if a large number of events accumulate in any bin. Such accumulation can occur during testing or simulation.

### 4.2 Canceling an existing event

Once scheduled, events may occasionally have to be canceled. For example, in an epidemiological model, a healthy individual may become the target of an infection. Whatever the next event in their life was, it may have to be rescheduled as the simulated individual progresses toward disease and infectiousness. Therefore, the existing event will be canceled and the earliest of other future events for the individual will be scheduled instead. In the physical analogy, that requires three steps:

1. Go to the bin representing the month, day, hour, and minute for the event. As before, ignore the year, second and any fraction of a second.
2. Flip through them to find the ticket for the event in question.
3. Destroy that ticket.

That required one operation for every ticket in the bin, but on average there is only one ticket in the bin. Canceling an event can be slow if the events cluster badly, because of the need to flip through the tickets in the bin. But canceling is not a usual operation. The two common operations are adding events (described above) and dispatching them (described next).

### 4.3 Dispatching the next event

The simulation proceeds stepwise by locating the earliest among all events in the schedule, removing it, then processing it. This is efficient, but it involves several steps:

1. Go to the bin representing the current day, hour, and minute.
2. If the flag on the bin is raised, arrange the tickets in chronological order and lower the flag.
3. Leave any tickets for future years in the bin.
4. Process any tickets from this year, day, hour, and minute, each to be handled precisely in sequence as the scheduled second and fraction of a second arrives.
5. If any tickets for the current bin arrive while the bin is being handled, put them in their proper position among the other tickets.

This required only one operation for each ticket, plus one or two more per ticket to order them chronologically before dispatching the contents of the bin. Again, on average there is only one ticket in the bin.

This method intentionally does not keep tickets in the bins ordered, using instead “just-in-time sorting.” Usually this will make little difference, since the bins are intentionally designed to be nearly empty. However, as described earlier, if unexpected clustering occurs, this just-in-time sorting will be much faster than keeping the contents of all bins in order each time an event is added.

Within a simulation program using these scheduling algorithms, the individual associated with the ticket being handled will have other pending events in its entry of data structure  $A[n]$ . The simulation program will then pass the earliest of these to the scheduling algorithms, through a call to *EventSchedule*.

The discussion above shows how the algorithms achieve their speed—by maintaining at least as many bins as there are tickets. If there were sixty times as many tickets—thirty million—the same speed of operation could be maintained simply by increasing the number of bins by sixty, to one bin for each second.

## 5. Applications

The algorithms described here have been applied and tested in a large-scale multi-compartmental epidemiological model of tuberculosis transmission developed by one of us (A.K.). That model runs with upwards of  $6 \times 10^7$  individuals (60 million), representing the entire population of the UK, on multiple parallel processors for parameter fitting by simulated annealing. Each individual has many events pending, including, for example, scheduled times of death, emigration, onset of disease for recently infected individuals, next transmission for infectious individuals, potential vaccination for juveniles, and so forth.

In this epidemiological model, typical runs spanned 30 simulated years and used 75 million bins occupied by 60 million individuals. Each run consumed about 80 seconds on a 2.8 GHz processor, using a little over 6 GB of memory

on each of 30 to 50 parallel processors. The average time increment between scheduled events was 14 simulated seconds, with a standard deviation of 12 seconds. The minimum was less than a simulated microsecond, whenever stochastic events appeared by chance close together in time. The maximum time increment was 53 simulated seconds. Thus the time steps are very small compared with a corresponding macroscale model.

In simplified timing tests on the same processor, outside of the operation of the epidemiological model, a list with  $6 \times 10^7$  individuals needed 30 nanoseconds on average to schedule each new event, 18 nanoseconds to cancel an event, and 12 nanoseconds to dispatch each event when its time arrived. This was near-ideal conditions, with new events arising in sequence in a way that minimized clustering in the schedule. Expanding the number of individuals by a factor of more than 16, to  $10^9$  individuals (one billion) required exactly the same amount of time per operation—within small bounds of statistical error—demonstrating the Order-1 behavior of the algorithms.

On the other hand, events arising in random order needed 90 nanoseconds to schedule each new event into a list of 60 million and 180 nanoseconds into a list of one billion. The three to six-fold increase can be attributed to interactions with internal memory caches. Such caches grow less useful as memory accesses become less localized.

## 6. Algorithmic details

The intuitive picture sketched above converts directly into the algorithms displayed in the appendix. As implemented in the algorithms, the bins need not correspond to standard time units such as minutes, but can be any values.

A simulation begins by adding one or more events, typically one event per individual, and ends either at a pre-determined time or when the last event has been dispatched. Array  $A[n]$  would be established earlier with a collection of pending events for each individual  $n$ . The main simulation program would be structured as follows:

- [1] *ProgramInit*();
- [2] **loop for all**  $n$  **in**  $A[n]$  :  
    *EventSchedule*( $n$ , *earliest*( $n$ ));
- [3] **loop for**  $t$  **from** 0 **to**  $t_{max}$ :  
    *Process*(*EventNext*());
- [4] **exit**;

In step 1 above, *ProgramInit* sets the initial conditions for the program, including allocating all individuals that will start the simulation and all future events that are known for each. Step 2 moves through all individuals, selects the earliest event for each (*earliest*( $n$ )), and schedules each event by calling *EventSchedule*. With all events to start the simulation scheduled, step 3 repeatedly asks for the next chronological event by calling *EventNext* and passing the

number of that event to *Process*. In turn, *Process* will call upon *EventSchedule* and possibly *EventCancel* and *EventRenumber* while carrying out the simulation. *ProgramInit*, *Process*, and *earliest*, as well as array  $A[n]$ , are written as part of the simulation program. The rest are scheduler algorithms detailed in the appendix.

The two main data structures organizing the earliest event for each individual are (1) a circular array of integers  $Q[h]$ , each heading a linked list in  $P[n]$  of events scheduled for time bin  $h$ , and (2) an array of integers  $P[n]$ , each continuing the linked list from  $Q[n]$ . The number of entries in  $P[n]$  must equal the number in external array  $A[n]$ , and like  $A[n]$ ,  $P[n]$  is indexed by individual number. But the number of time bins  $Q[h]$  may be smaller or larger than the number of individuals. The size of  $Q[n]$  is a matter of optimization. It is typical to have one time bin for each event that could be scheduled, meaning each bin will represent a single event on average. A space–time tradeoff occurs because optimal allocation leaves about one-third of the bins empty.<sup>4</sup>

Each bin  $Q[h]$  represents many related times, all equal modulo the width of the series of time bins,  $Qw$ . The width  $Qw$  of all bins combined is also a matter of optimization. If it is much too large, events will tend to cluster near the bin being dispatched. If it is much too small, events will tend to spread out, with most bins containing events that are for the more distant future. A suitable value for  $Qw$  can be found by knowledge of the system being analysed, or by experimental trials to find a good speed of operation.

For speed of addition, the lists of events in  $P[i]$  are not maintained in any particular order, but each bin is sorted chronologically before it is dispatched. Any sorting algorithm used should have (1) best performance when the list is already partially sorted, e.g. Order  $N$ , important because lists will remain partially sorted from earlier passes, (2) high-speed when sorting only 1 and 2 entries, which are the most common, and (3) good worst-case performance, e.g. Order  $N \log_2 N$ . The sorting routine presented as Algorithm 5 in the appendix has these properties.

## 7. Conclusions

The algorithms presented here can be incorporated into any individual-based or other microscale model, where they can speed simulations many orders of magnitude over alternative methods that are not Order-1.

They are part of a large-scale simulation model developed by one of us (A.K.) for tuberculosis in the UK. Sixty million individuals thus can be handled by allocating less than a gigabyte of random access memory—within the reach even of portable computers. In practice, these algorithms should be able to schedule, cancel, and dispatch up to  $10^7$  or more events per second with 60 million or more pending events

<sup>4</sup>Under random distribution,  $1/e = 37\%$  will be empty. That can be shown to be optimal for overall speed if all bin operations are equally fast.

maintained in the queue. Therefore, they should not become a bottleneck in the simulation as a whole.

Compilable copies of the code described here and related simulation algorithms are available free from the authors upon request.

## 8. Acknowledgements

We thank Fred Lehman, Holly MacCormick, Peter Hawthorne, Ben Kerr, Celia Hemmerich, and Shelby Williams for discussions and encouragement. The project was supported in part by a resident fellowship grant to C.L. from the UMN Institute on the Environment, by grants of computer time from the Minnesota Supercomputer Institute, and by doctoral research funding to A.K. from the Modelling and Economics Unit at the Health Protection Agency, London.

## 9. Contributions

C.L. considered the notion of stochastically prescheduling the earliest future event for each individual and developed an initial application. A.K. expanded the approach for her large-scale tuberculosis model, leading to refinements in the algorithms. A.K. conceived a grouping method to work in concert and make these algorithms practical for multi-compartment simulation models [8]. R.B. participated in the evaluation, applications to other areas, and the literature review. C.L. coded the algorithms and A.K. tested them in large-scale operation. All authors contributed to the manuscript.

## References

- [1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes: The art of scientific computing, third edition," *Cambridge University Press, New York*, 2007.
- [2] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *Journal of Physical Chemistry*, vol. 81, pp. 2340–2361, 1977.
- [3] R. Brown, "Calendar queues: a fast  $O(1)$  priority queue implementation for the simulation event set problem," *Commun. ACM*, vol. 31, no. 10, pp. 1220–1227, Oct. 1988.
- [4] R. Rönngren, J. Riboe, and R. Ayani, "Lazy queue: an efficient implementation of the pending-event set," *SIGSIM Simul. Dig.*, vol. 21, no. 3, pp. 194–204, Apr. 1991.
- [5] G. A. Davidson, "Calendar p's and q's," *Communications of the ACM*, vol. 32, pp. 1241–1242, 1989.
- [6] T. Hui and I. Thng, "Felt: A far future event list structure optimized for calendar queues," *Simulation*, vol. 78, no. 6, pp. 343–361, 2002.
- [7] G. Yan and S. Eidenbenz, "Sluggish calendar queues for network simulation," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on*, 2006, pp. 127–136.
- [8] A. Keen and C. Lehman, "Trading space for time: Constant-speed algorithms for grouping objects in scientific simulations," *Proceedings, International Conference on Scientific Computing.*, pp. 146–151, 2012.
- [9] A. I. Dumey, "Indexing for rapid random access memory systems," *Computers and Automation*, vol. 6, pp. 6–9, 1956.
- [10] D. E. Knuth, "The art of computer programming, volume 3: Sorting and searching, second edition," *Addison-Wesley, Reading, MA*, 1998.
- [11] R. Goh and I. Thng, "Mlist: An efficient pending event set structure for discrete event simulation," *International Journal of Simulation-Systems, Science & Technology*, vol. 4, no. 5-6, pp. 66–77, 2003.

## 10. Appendix

To use the algorithms described in this paper, it is only necessary to understand the entry and exit conditions that appear at the beginning of each, not the code itself. Nonetheless, to allow complete evaluation of the algorithms, and to encourage further development of them, we present them as pseudo-code inspired by and simplified from the programming languages C, Python, and R. The algorithms are defined with sufficient precision that they can be run, tested, timed,

modified, or translated to other languages. Familiarity with a relatively few operators\* and with the syntax of flow control (if, for, while, etc.), is sufficient to follow the algorithms. *WarnMsg* and *ExitMsg* display error messages and the latter terminates the program. Not all functions return values. Text copies of this pseudo-code translated into operational C are available from the authors upon request, or from the associated website [www.cbs.umn.edu/modeling](http://www.cbs.umn.edu/modeling).

---

### PROGRAM PARAMETERS

$TN \equiv (100000000)$	Example, maximum number of time bins.
$PN \equiv (100000003)$	Example, maximum number of forward indexes to time bins.
$TW \equiv 20$	Example, time width of all bins combined (for optimization).

---

### INTERNAL DATA STRUCTURES

$PZ \equiv -1$	Marker for empty bins.
<b>real</b> $T[PN] \leftarrow 0;$	Time for each scheduled event.
<b>integer</b> $P[PN] \leftarrow PZ;$	Forward indexes within bins, ending with zero.
<b>integer</b> $Q[TN] \leftarrow 0;$	First index for the bin, with zero for empty bins, negative for unsorted bins.
<b>real</b> $Qw \leftarrow TW;$	Interval of time represented for each cycle in $Q$ .
<b>integer</b> $Qn \leftarrow TN;$	Number of elements in $Q$ .
<b>integer</b> $Qi \leftarrow 0;$	Index of the immediate time bin.
<b>integer</b> $Qe \leftarrow 0;$	Number of events in all bins.
<b>real</b> $Qt0 \leftarrow 0;$	Earliest time representable this cycle in $Q$ .
<b>real</b> $Qt1 \leftarrow TW;$	Earliest time beyond this cycle in $Q$ .
<b>real</b> $t \leftarrow 0;$	Current time, last dispatched event.

---

### Algorithm 1. SCHEDULE A NEW EVENT

**Upon entry to the algorithm,** (1)  $n$  contains the number (starting with 1) of a new event. (2)  $te$  contains the time at which the new event will occur. (3)  $P[n]$  indicates that the event is unscheduled (equal to  $PZ$ ). (4) The scheduling data structures are prepared as described above. **At exit,** (1) the event has been scheduled, to occur when the proper time arrives. (2)  $T[n]$  records the time  $te$  of the event. (3)  $P[n]$  links the event with others in its time bin.

*EventSchedule*( $n, te$ ) **integer**  $n$ , **real**  $te$ ; **integer**  $i$ ; **real**  $tr$ ;

**if**  $n < 1$  **or**  $n \geq PN$ : *ExitMsg*(3);

**if**  $P[n] \neq PZ$ : *ExitMsg*(4);

**if**  $te < t$ : *ExitMsg*(5);

$te \rightarrow T[n];$

$(te - Qt0)/Qw \rightarrow tr; tr - (int)tr \rightarrow tr;$

$tr * Qn \rightarrow i;$

$abs(Q[i]) \rightarrow P[n], -n \rightarrow Q[i], \uparrow Qe;$

1. Check the index and make sure an event is not already scheduled and is not in the past.
2. Record the time of the new event.
3. Convert the time to a bin number.
4. Add the event to the list for that bin and increment the number of events.

\* The pseudo-code given here is two-dimensional, as in the language Python, so that indentation completely defines the nested structure, with no need for bracketing characters such as '{' and '}'. Variables and function names are italicized and flow control and reserved words are bolded.

The assignment operator is represented either as ' $\leftarrow$ ' or ' $\rightarrow$ ', similar to assignments in R. The compound assignments ' $a + 1 \rightarrow a \rightarrow b \rightarrow W[i][j]$ ' and ' $W[i][j] \leftarrow b \leftarrow a \leftarrow a + 1$ ' are equivalent, first incrementing  $a$  and placing the results back in  $a$ , then in  $b$ , and then in the  $i, j$ th element of the array  $W$ .

The expression structure ' $c?u:v$ ', where  $c$  is a condition,  $u$  is

an if-expression, and  $v$  is an else-expression, follows that of C. Using up-tick and down-tick operators to write ' $\uparrow a$ ', ' $\downarrow a$ ', ' $a\uparrow$ ', and ' $a\downarrow$ ' form pre- and post-increments by one, as in ' $++a$ ', ' $--a$ ', ' $a++$ ', and ' $a--$ ' of C.

Arrays are indexed as in the language C, starting with 0. Data types are '**integer**' and '**real**', with the latter specifying floating point. Operator precedence is that of C, with assignments having lowest precedence. Logical operators such as '**and**' and '**or**' are preemptive, terminating a chain of logical operations as soon as the result is known. Permanent global assignments, as would be represented '#define  $\alpha \beta$ ' in C, are rendered as ' $\alpha \equiv \beta$ '.

**Algorithm 2. CANCEL AN EXISTING EVENT**

**Upon entry to the algorithm, (1)**  $n$  contains the number (starting with 1) of the event to be cancelled. **(2)**  $T[n]$  contains the scheduled time of the event. **(3)** the scheduling data structures are prepared as described above. **At exit,** the event has been removed from the list.

*EventCancel*( $n$ ) **integer**  $n$ ; **integer**  $i, j, jp$ ; **real**  $tr$ ;

**if**  $n < 1$  **or**  $n \geq PN$ : *ExitMsg*(6);

**if**  $P[n] = PZ$ : *ExitMsg*(7);

$(T[n] - Qt0)/Qw \rightarrow tr$ ;  $tr - (int)tr \rightarrow tr$ ;  
 $tr * Qn \rightarrow i$ ;

**if** *subcancel*( $n, i$ ): **return**;

$(i - 1 + Qn) \bmod Qn \rightarrow i$ ; **if** *subcancel*( $n, i$ ): **return**;

$(i + 2 + Qn) \bmod Qn \rightarrow i$ ; **if** *subcancel*( $n, i$ ): **return**;

*ExitMsg*(8);

**integer** *subcancel*( $n, i$ ) **integer**  $n, i$ ; **integer**  $j, jp$ ;

$0 \rightarrow jp$ ,  $abs(Q[i]) \rightarrow j$ ;

**loop while**  $j > 0$ :

**if**  $j = n$ :

**if**  $jp > 0$ :  $P[j] \rightarrow P[jp]$ ;

**else**  $Q[i] > 0 ? P[j] : -P[j] \rightarrow Q[i]$ ;

$PZ \rightarrow P[j]$ ; **if**  $\downarrow Qe < 0$ : *ExitMsg*(9);

**return** 1;

$j \rightarrow jp$ ,  $P[j] \rightarrow j$ ;

**return** 0;

1. Check the index and make sure an event is scheduled.

2. Convert the time to a bin number, modulo the duration of the cycle.

3. Remove it from its normal bin or from an adjacent bin above or below (due to rounding error).

4. If the specified event was not in the list, signal an error.

1. Scan the list of pending events in this bin and remove the specified event. (The average number of events in non-empty bins is about 1.5)

**Algorithm 3. DISPATCH THE NEXT EVENT**

**Upon entry to the algorithm, (1)**  $T$  contains the time for each scheduled event. **(2)** The scheduling data structures are prepared as described above. **At exit, (1)** *EventNext* contains the number of the next event. If zero, no events are scheduled. **(2)**  $t$  contains the time of the next event, if *NextEvent* is not zero.

**integer** *EventNext*() **integer**  $j, n$ ;

**loop while**  $Qe > 0$ :

**loop while**  $Qi < Qn$ :

$Q[Qi] \rightarrow j$ ; **if**  $j = 0$ :  $\uparrow Qi$ ; **repeat loop**;

**if**  $j < 0$ :

$sort(P, -j, 0, order) \rightarrow Q[Qi] \rightarrow j$ ;

**if**  $T[j] < Qt1$ :

**if**  $P[j] = PZ$ : *ExitMsg*(2);

$P[j] \rightarrow Q[Qi]$ ,  $PZ \rightarrow P[j]$ ,  $\downarrow Qe$ ;

$T[j] \rightarrow t$ ; **return**  $j$ ;

$\uparrow Qi$ ;

$0 \rightarrow Qi$ ,  $Qt0 + Qw \rightarrow Qt0$ ,  $Qt0 + Qw \rightarrow Qt1$ ;

**return** 0;

1. Advance to the next non-empty bin.

2. Sort the bin if it may be necessary (usually sorts 1 or 2).

3. If the event belongs to this pass, remove it, decrement the number of events, advance the time, and return its index.

4. Advance to the next bin and repeat.

5. Circle back to the first bin.

6. Signal completion of all events.

**Algorithm 4. RENUMBER AN EVENT**

**Upon entry to the algorithm, (1)**  $n$  contains the new index number, which has no event scheduled. **(2)**  $m$  contains the current index number of the event. **At exit, (1)**  $n$  is the new index number. **(2)** The event originally scheduled as  $m$  is re-scheduled as  $n$ . Event  $m$  no longer has an event scheduled and the index is free to be reused.

*EventRenumber*( $n, m$ ) **integer**  $n, m$ ;

**if**  $n < 1$  **or**  $n \geq PN$ : *ExitMsg*(10);

**if**  $m < 1$  **or**  $m \geq PN$ : *ExitMsg*(11);

**if**  $n \neq m$ :

$T[m] \rightarrow T[n]$ ;

*EventCancel*( $m$ );

*EventSchedule*( $n, T[n]$ );

1. Check the indexes and make sure they are in range.

2. Transfer the time.

3. Cancel the old number.

4. Reschedule as the new number.

**Algorithm 5. SORTING**

**Upon entry to the algorithm, (1)**  $list$  points to an array of forward indexes.  $list[0]$  is unused. **(2)**  $p$  indexes the first element of the list, which ends with a zero. **(3)**  $n$  contains the number of items in the list, if known. If zero, the number of items is not known and  $sort$  should count. **(4)**  $c$  compares two list elements  $u$  and  $v$ . It returns negative, zero, or positive when  $u < v$ ,  $u = v$ , and  $u > v$ , respectively. **At exit**,  $sort$  indexes the first element in the sorted list, which ends with a zero. The original ordering is preserved for entries that are equal.

**integer**  $*P, pc, pr, m, (*order)(int, int);$

**integer**  $sort(list, p, n, c)$  **integer**  $list[], p, n, (*c)(int, int);$  **integer**  $i;$

$c \rightarrow order, list \rightarrow P;$

**if**  $n = 0: p \rightarrow i; \text{loop while } i > 0: P[i] \rightarrow i, n \uparrow;$

**if**  $n = 0$  **or**  $p = 0: \text{return } 0;$

**if**  $n = 1: \text{return } p;$

**if**  $n = 2:$

**if**  $order(p, P[p]) \leq 0: \text{return } p;$

$P[p] \rightarrow i, p \rightarrow P[i], 0 \rightarrow P[p];$

**return**  $i;$

$p \rightarrow pc; \text{return } isort(n);$

1. Record calling parameters.

2. Count the number of elements and return empty and single-element lists immediately.

3. If the list contains only two elements, sort it by inspection.

4. Otherwise sort the full list.

**Partition into sorted sublists. Upon entry, (1)**  $n$  defines the minimum number of elements to be sorted. **(2)**  $P$  is the list of forward indexes. **(3)**  $pc$  indexes the first element of the list. **(4)**  $order$  compares two list elements. **At exit, (1)**  $isort$  indexes the first element in the sorted list, which ends with a zero index. **(2)**  $m$  defines the number of elements which were actually sorted, greater than or equal to its value on entry. **(3)**  $pc$  indexes the element following the last element sorted. If the entire list has been sorted,  $pc$  is null.

**integer**  $isort(n)$  **integer**  $n; \text{integer } wp1, wp2, m1;$

**if**  $n \leq 1:$

**if**  $pc = 0: \text{return } 0;$

$pc \rightarrow wp1, 0 \rightarrow m;$

**loop** :  $pc \rightarrow pr, P[pc] \rightarrow pc, m + 1 \rightarrow m;$

**if**  $pc = 0: \text{return } wp1;$

**if**  $order(pr, pc) > 0: \text{exit loop};$

$0 \rightarrow P[pr]; \text{return } wp1;$

$isort(n/2) \rightarrow wp1;$

**if**  $n \leq m: \text{return } wp1;$

$m \rightarrow m1, isort(n - m) \rightarrow wp2, m + m1 \rightarrow m;$

**return**  $imerge(wp1, wp2);$

1. If a single element is requested, initialize variables and check for error in count.

2. Then scan forward in the list to find the longest list that is already in order and return that list.

3. If multiple elements are requested, sort the first part of the list and return if enough was sorted.

4. If it was not, then sort what remains and merge the two sublists.

**Merge sublists. Upon entry, (1)**  $P$  is the list of forward indexes. **(2)**  $p$  and  $q$  index the first element of a sorted primary and secondary list, respectively. **(3)**  $order$  compares two list elements. **At exit**,  $imerge$  indexes the first element of the list merged in order. In case of equal entries, those from the primary list appear first.

**integer**  $imerge(p, q)$  **integer**  $p, q; \text{integer } pb, v;$

**if**  $p = 0: \text{return } q; \text{if } q = 0: \text{return } p;$

**if**  $order(p, q) > 0: q \rightarrow pb, 1 \rightarrow v;$

**else**  $p \rightarrow pb, 3 \rightarrow v;$

**loop while**  $v > 0:$

**loop while**  $v = 1: q \rightarrow pr, P[q] \rightarrow q;$

**if**  $q = 0: -2 \rightarrow v;$

**else if**  $order(p, q) \leq 0: 2 \rightarrow v;$

**if**  $v = 2: p \rightarrow P[pr];$

**loop while**  $v \geq 2: p \rightarrow pr, P[p] \rightarrow p;$

**if**  $p = 0: -1 \rightarrow v;$

**else if**  $order(p, q) > 0: 1 \rightarrow v;$

**if**  $v = 1: q \rightarrow P[pr];$

**if**  $v < -1: p \rightarrow P[pr]; \text{else } q \rightarrow P[pr];$

**return**  $pb;$

1. Handle empty lists.

2. Save the beginning of the list and select the proper routine.

3. Scan for a secondary element greater than or equal to the current primary element and mend the secondary list.

4. Scan for a primary element greater than the current secondary element, mend the primary list, and repeat.

5. Attach any remaining elements and return the merged list.

# The Calculus of Semiautomatic Differentiation

Abdulwahab A. Abokhodair

Department of Earthsciences  
King Fahd University of Petroleum and Minerals  
Dhahran, KSA

**Abstract** - The need for approximation of derivatives of explicit functions arises in a wide range of geosciences applications. In this paper I describe a technique for derivative approximation based on complex variable theory which was shown to be superior to ordinary finite difference (FD) schemes in accuracy, robustness and ease of implementation. The method is in fact a close relative of true automatic differentiation (AD), hence, the name: semiautomatic differentiation (SD). The method was generalized to multi-parameters functions and extended to second order derivatives so that it is capable of approximating first and second order derivatives and derivative structures such as directional derivatives, gradients, Jacobians and Hessians. Because of its AD-like features, it has a high potential as an effective teaching and research tool in the geosciences. The goals of this paper, in addition to bringing the technique to the attention of the wider geoscience community, are to provide a guide for computation with SD and illustrate its potential through application to common geoscience problems.

**Keywords:** Differentiation, Modeling, Inversion, Imaging

## 1 Introduction

Numerical differentiation plays a key role in modern computational geoscience. Advances in computational resources, geoscience instrumentation and numerical algorithms have completely altered the approaches to data analysis and interpretation. Today, any geoscientific data interpretation approach entails solution of complex optimization problems such as data inversion, parameters estimation, sensitivity analysis, design optimization, optimal control, system modeling and process simulation. State-of-the-art numerical algorithms developed to solve these types of optimization problems are derivative-based, requiring users to supply derivatives of model functions and associated constraint equations in the form of gradients, Jacobians and Hessians. The accuracy with which these derivative structures are computed is pivotal to the successful performance of these algorithms. Therefore, a need exists for development of efficient and accurate methods to approximate derivatives. Key issues in assessing a differentiation technique are accuracy, robustness, computational cost and ease of implementation.

The classic finite difference (FD) is the most widely used method for derivative approximation. The method, however, is neither efficient nor accurate; it owes its universal

popularity primarily to its ease of implementation. On one hand, the accuracy of the method depends critically on the size of the differencing interval (step size). For maximum accuracy, an optimum step size must be sought often by trial and error, a process that significantly impedes the efficiency of the method (Mark and Workman Jr., 2003; Burge and Newman, 2003). On the other hand, from filtering point of view, generic FD differentiation filters have the undesirable property of noise amplification by factors exceeding 10.0 (Orfanidis, 1996; Abokhodair, Submitted to GEM); noise amplification factor is proportional to the ratio of the noise variances on output and input into the filter.

An alternative to FD with superior qualities exists in a method, little known in the geoscience literature, based on the theory of complex variable and called the complex-step derivative (CSD). The CSD method for first-derivative approximation of analytic scalar functions was first reported by Squire and Trapp (1998) who showed it to be highly accurate, extremely robust and very easy to implement. It has since been gaining recognition and successfully applied in several large scale studies including global and local sensitivity analyses, aerodynamic design optimization and pseudospectral algorithms (e.g. Cerviño and Bewley, 2003; Martin et al., 2001; Burge and Newman, 2003; Vatsa, 2000; Wang, 2004; De Pauw and Vanrolleghem, 2006).

As shown by Martins et al. (2001) and demonstrated in the next section, CSD is a close relative of automatic differentiation (AD) when the latter is implemented via object oriented programming; the complex arithmetic provides the necessary data typing and operator overloading. Hence, in terms of accuracy, CSD is comparable to AD and is as simple to implement as FD but without the inherent step-size limitation of FD. Moreover, it is easily accessible, without any prior programming, in any computational environment that supports complex arithmetic. Details of the method, its generalization to multi-parameter functions, and its extension to second derivatives were reported by the present author (Abokhodair, 2007). The method has also been automated into a Matlab-based toolkit referred to "Semiautomatic Differentiation Tool (SD)" that allows computation of derivative structures including gradients, Jacobians and Hessians (Abokhodair, 2009), hence the title of this paper.

Because of its accuracy, simplicity and general accessibility, the CSD method (hereafter will be referred to as SD) has a high potential as an effective teaching and research tool in the geosciences. The goals of this paper, in addition to

bringing the technique to the attention of the wider geoscience community, are to provide a guide for computation with SD and illustrate its potential through application to common geoscience problems. The intention is not to document the SD toolkit, but rather to show by example that the technique is simple, powerful, and widely applicable; hence, it easily challenges FD in performance qualities. The aim is to make prospective users, students and researchers, feel comfortable using the method and confident in its results as they discover its key advantages over FD schemes.

## 2 A heuristic example

To understand why the SD method works, we examine a simple function, say,  $f(x) = e^{\sin x}$ . The objective is to estimate its derivative  $f'(x)$ ; of course, the exact derivative is  $f'(x) = e^x \cos x$ . The key idea of SD is to perturb the target variable  $x$  with a pure imaginary step  $h \ll 1$  and construct the complex argument:  $z = x + ih$  where  $i = \sqrt{-1}$ . Thus, the 'complexified' version of our original real-valued function becomes:

$$f(z) = e^{\sin z}.$$

Expanding  $f(z) = f(x + ih)$  in Taylor series and neglecting terms in  $h^2$  and higher yields:

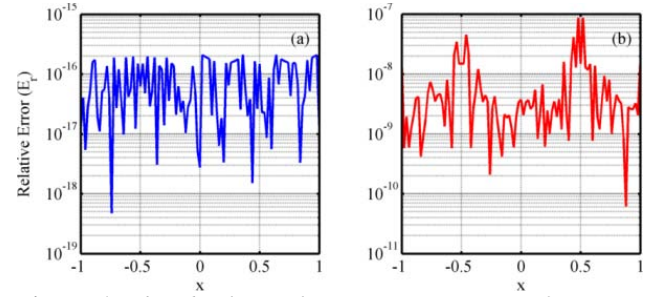
$$f(z) \approx e^{\sin x} + ih e^{\sin x} \cos x \\ f(x) + ih f'(x)$$

This is the original formula reported by Squire and Trapp (1998). It provides  $O(h)$  approximations of  $f(x)$  and its first derivative  $f'(x)$  in the real and the imaginary parts of  $f(z)$  respectively. Unlike FD, the SD approximation formula for  $f'(x)$  involves no differencing operation; hence no subtractive cancellation errors or step-size adverse effects are to be expected. This is the essence of the SD method. The accuracy of the derivative approximation of the example above is assessed by examining the absolute relative error defined as:

$$E_r(x) = \left| \frac{f_{ED}^k(x) - f_{MD}^k(x)}{f_{MD}^k(x)} \right|,$$

where  $f_{ED}^k(x)$  and  $f_{MD}^k(x)$  are respectively the estimated and exact (manual) derivatives. Figure 1 compares the absolute relative errors  $E_r(x)$  of the SD (a) and centered finite difference (CFD) approximations (b) of  $f'(x)$  for respective step sizes of  $h = 10^{-25}$  and  $10^{-8}$ . Whereas the

SD relative error fluctuates randomly about a mean value of  $10^{-16}$ , the CFD error is several orders of magnitude larger with a mean value of about  $10^{-9}$ .



**Figure 1:** The absolute relative errors  $E_r(x)$  of SD (a) and CFD (b) approximations of the derivative in the heuristic example using step sizes  $h = 10^{-25}$  and  $10^{-8}$  respectively.

The SD derivative of this example is computed by the Matlab script in the box below.

```
x=pi*(-1:.02:1)';
h=1.0e-25; % Step
f=@(x) exp(sin(x));
z=complex(x,h); % Complex
argument z = x + ih
fc=f(z); % Complexified
function f(z)
sdfp=imag(fc)/h; % Sd derivative
Approximation
```

## 3 First Order Derivatives

The generalized Squire-Trapp formula for vector-valued functions of several variables is:

$$\mathbf{F}(\mathbf{z}) = \mathbf{F}(\mathbf{x} + i\mathbf{h}\mathbf{e}) = \mathbf{F}(\mathbf{x}) + ih\mathbf{e}^T \mathbf{J}(\mathbf{x}) + O(h^2), \quad (1)$$

where the column vectors  $\mathbf{F}(\mathbf{x}) = [f_k(\mathbf{x})]_{k=1}^N$ ,  $\mathbf{x} = [x_k]_{k=1}^M$  and the  $N \times M$  matrix  $\mathbf{J}(\mathbf{x})$  is the Jacobian with respect to  $\mathbf{x}$ , i.e.  $\mathbf{J}(\mathbf{x}) = \nabla_{\mathbf{x}} \mathbf{F}(\mathbf{x})$ . (Abokhodair, 2007).

The imaginary part of equation (1) provides a first derivative approximation formula as:

$$\mathbf{e}^T \mathbf{J}(\mathbf{x}) = \frac{1}{h} \text{Im}[\mathbf{F}(\mathbf{z})] + O(h^2). \quad (2)$$

This formula can be used to compute high-accuracy numerical approximations of partial derivatives, gradients, directional derivatives and Jacobians.



### 1.1. Gradients and directional derivatives

Table 1 summarizes the SD differentiation rules for first partial derivatives. The pair of variables  $(x_k, x_l)$  are perturbed singly or in combination by a pure imaginary step  $ih$  to form the corresponding complex variable  $(z_k, z_l)$ . The imaginary part of the complexified function  $f(z_k, z_l)$  scaled by  $h$  yields the derivative estimate listed in column 6 of the table. To illustrate these rules, we use the Rosenbrock function of two variables defined as:

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 + x_1)^2. \quad (3)$$

This is a scalar-valued function for which the Jacobian matrix ( $\mathbf{J}$ ) in equation (2) reduces to a gradient vector  $\mathbf{g}$  given by:

$$\mathbf{g}(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{bmatrix}, \quad (4)$$

and the directional derivative in the direction of the unit vector  $\mathbf{e}(\theta) = [e_{x_1}(\theta) \ e_{x_2}(\theta)]^T$ , by definition, is:

$$\mathbf{D}_{\mathbf{e}}[f(\mathbf{x})] = \begin{bmatrix} e_{x_1} & e_{x_2} \end{bmatrix} \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{bmatrix}. \quad (5)$$

First Derivative Rules	Complex Parameters		$U = \text{Re}[f(z_k, z_l)]$		$V = \text{Im}[f(z_k, z_l)]$	
	$z_k$	$z_l$	$U/\Delta$	$\Delta$	$V/\Delta$	$\Delta$
	$x_k + ih$	$x_l$	$f(\mathbf{x})$	1	$\partial f / \partial x_k$	$h$
	$x_k$	$x_l + ih$	$f(\mathbf{x})$	1	$\partial f / \partial x_l$	$h$
	$x_k + ih$	$x_l + ih$	$f(\mathbf{x})$	1	$e^T \nabla_{\mathbf{x}} f$	$h$

Table 1: SD differentiation rules for first order partial derivatives.  $\Delta$  in columns 5 and 7 is a scaling factor.

The Matlab function `cFPD.m` of the Appendix returns  $\mathbf{D}_{\mathbf{e}}[f(\mathbf{x})]$  given the direction  $\theta$ . It should be noted that the partial derivatives  $[\partial f / \partial x_1, \partial f / \partial x_2]$ , are obtained by setting  $\theta = [0, 90]$  respectively. Figure 2 shows the output of the function for different values of  $\theta$ .

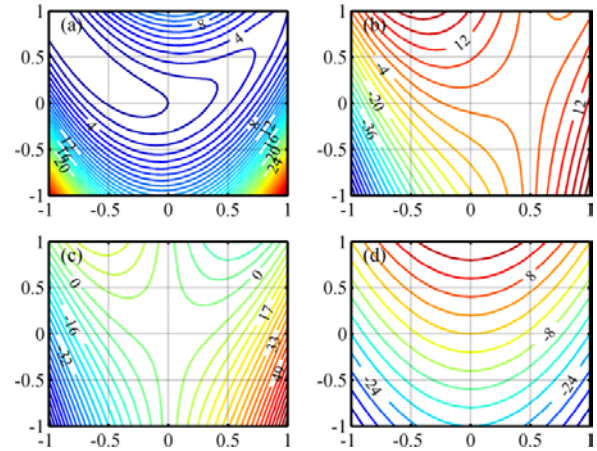


Figure 2: Output of function `fpd(theta)` showing (a) contours of  $f(\mathbf{x})$ , (b) the directional derivative  $\mathbf{D}_{\mathbf{e}}(\mathbf{x})$  ( $\theta = 45$ ), (c)  $\partial f / \partial x_1$  ( $\theta = 0$ ), and (d)  $\partial f / \partial x_2$  ( $\theta = 90$ )

### 1.2. The Jacobian

The Jacobian, we recall, is a matrix of first-order partial derivative defined for vector-valued functions as:

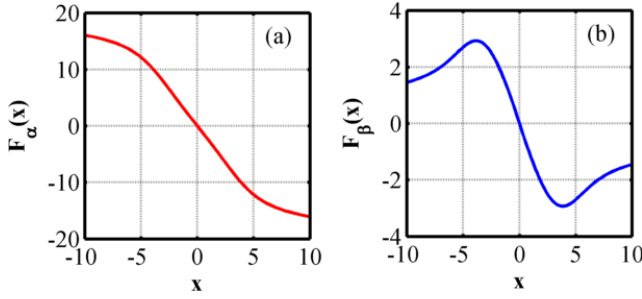
$$\mathbf{J}^T(\mathbf{x}) = \nabla_{\mathbf{x}} \mathbf{F}^T(\mathbf{x}) = \begin{bmatrix} \partial / \partial x_1 \\ \partial / \partial x_2 \\ \vdots \\ \partial / \partial x_M \end{bmatrix} \begin{bmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) & \cdots & f_N(\mathbf{x}) \end{bmatrix}, \quad (6)$$

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \partial f_1 / \partial x_1 & \partial f_1 / \partial x_2 & \cdots & \partial f_1 / \partial x_M \\ \partial f_2 / \partial x_1 & \partial f_2 / \partial x_2 & \cdots & \partial f_2 / \partial x_M \\ \vdots & \vdots & \vdots & \vdots \\ \partial f_N / \partial x_1 & \partial f_N / \partial x_2 & \cdots & \partial f_N / \partial x_M \end{bmatrix}$$

Note that, by convention, the rows of  $\mathbf{J}(\mathbf{x})$  are the transposed gradients of the component functions  $[f_k(\mathbf{x})]_{k=1}^N$ . Since the Jacobian is a first-order derivative structure, its computation follows the same rules of Table 1. Consider, with some change in notation, the function:

$$f(\mathbf{p}, x) = \frac{\alpha \tan^{-1}(x / \beta)}{\sqrt{\alpha^2 + e^{-(x/\beta)^2}}} \quad (7)$$

where  $\mathbf{p} = [\alpha \ \beta]$  is the vector of parameters and  $x$  is a spatial variable. The gradient  $\nabla_{\mathbf{p}} f(\mathbf{p}, x)$  over a range of  $x$ , say,  $a \leq x \leq b$  is the Jacobian  $\mathbf{J}(\mathbf{p})$  of  $f(\mathbf{p}, x)$  with respect to its parameters. The Jacobian of equation 7 shown in Figure 3 is computed by the Matlab code `cJacob.m` in the Appendix.



**Figure 3:** The Jacobian of the 2-parameters function in equation 7 computed by function *sdJacob.m* of the Appendix.

Note that here we use the column vectors of an identity matrix instead of  $\mathbf{e}(\theta)$  to impulse individual components of  $\mathbf{J}$  since these column vectors form the bases of the parameter space.

## 4. Second Order Derivatives

The approximation formula for first order derivative (equation 1) is the only subtraction-free formula that can be obtained from complex variable theory. However, at the expense of a small lose in accuracy this formula can be extended to second-order derivatives by any ordinary finite difference scheme. This may be accomplished by perturbing the target function twice, once along the imaginary axis by a step  $ih$  and a second along the real axis by a step of size  $\delta x$ , expanding each time in a Taylor series. The approximation formula so obtained using a centered difference (CFD) scheme is (Abokhodair, 2007):

$$\mathbf{e}^T \mathbf{H} \mathbf{e} = \frac{1}{2h\delta x} \text{Im} [\Delta f(\mathbf{z})] + O(h^2), \quad (8)$$

where  $\mathbf{H}$  is the hessian matrix. The differentiation rules for computing second-order partial derivatives from this formula are given in Table 2. As shown in the table, equation 2 provides approximations for both first and second order derivatives in its real and imaginary parts respectively. Because of the differencing operation involved in the derivation of the formula, however, the approximation of first-order derivatives is less accurate than that of equation 1.

	Complex Parameters		$U = \text{Re}[f(z_k, z_l)]$		$V = \text{Im}[f(z_k, z_l)]$	
	$z_k$	$z_l$	$U/\Delta$	$\Delta$	$V/\Delta$	$\Delta$
Second Derivative Rules	$(x_k \pm \delta x) + ih$	$x_l$	$\partial f / \partial x_k$	$2\delta x$	$\partial^2 f / \partial x_k^2$	$2h\delta x$
	$x_k$	$(x_l \pm \delta x) + ih$	$\partial f / \partial x_l$	$2\delta x$	$\partial^2 f / \partial x_l^2$	$2h\delta x$
	$x_k \pm \delta x$	$x_l + ih$	$\partial f / \partial x_k$	$2\delta x$	$\partial^2 f / \partial x_k \partial x_l$	$2h\delta x$
	$x_k + ih$	$x_l \pm \delta x$	$\partial f / \partial x_l$	$2\delta x$	$\partial^2 f / \partial x_l \partial x_k$	$2h\delta x$
	$(x_k \pm \delta x) + ih$	$(x_l \pm \delta x) + ih$	$\mathbf{e}^T \nabla_x f$	$2\delta x$	$\mathbf{e}^T \mathbf{H} \mathbf{e}$	$2h\delta x$

**Table 2:** SD differentiation rules for second order partial derivatives.  $\Delta$  in columns 5 and 7 is a scaling factor.

### 1.1. Second partial and cross partial derivatives

To demonstrate application of the rules in Table 2, we use the following test function:

$$F(x, y) = \frac{(x^2 - y^2 - xy)}{(x^2 + y^2 + 1)^{3/2}}. \quad (9)$$

The code fragment below implements the third rule for the following two partial derivatives  $[\partial F / \partial x, \partial F / \partial x \partial y]$  at any point  $(x_o, y_o)$  with output  $[-0.0163, -0.0526]$  at  $(3.5, -1.76)$ .

```
F=@(x,y) (x.^2-y.^2-x.*y)./(x.^2+y.^2+1).^(3/2);
dx=0.05e-5; % Real step
h=1.0e-25; % Imaginary step
s=2*dx*h;
xo=3.5; yo=-1.76;
z2=complex(yo,h); % Imaginary step
% Forward
z1= xo+dx; % Real perturbation
Fp=F(z1,z2); % F(xo+dx,yo+ih)
% Backward
z1= xo-dx;
Fn=F(z1,z2); % F(xo-dx,yo+ih)
DF=Fp-Fn; % Centered
Difference
Fx=real(DF)/(2*dx); % FPD in real part
Fxy=imag(DF)/s; % XPD in imaginary
part
out=[Fx Fxy];
```

### 1.1. The hessian matrix

The hessian matrix is a second-order derivative structure defined for scalar-valued functions as:

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_M} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_M \partial x_1} & \frac{\partial^2 f}{\partial x_M \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_M^2} \end{bmatrix}. \quad (10)$$

Since it contains second partial and cross partial derivatives, its computation requires double loops over the parameters of the target function using the first and third rules of Table 2. The accuracy of the scheme in equation 8 for the hessian may be evaluated from the relative error with respect to the exact (manual) derivatives. Using the Matlab

code *cHess.m* of the Appendix, we compute the full hessian and its relative errors for the test function (equation 9). The results (Table 3) indicate that the relative error in  $\mathbf{H}$  computed at a step size  $h = 10^{-25}$ , is of the order of  $10^{-11}$ , much smaller than any FD scheme can deliver.

SD	1.5172e-14	1.2545e-14
	1.1911e-14	4.7402e-13
CFD	7.1972e-06	4.1858e-06
	4.1858e-06	1.6096e-05

**Table 3:** Absolute relative errors in hessian approximation by SD and CFD.

Examination of Table 3 reveals a difference in accuracy of about 8 orders of magnitude between SD and CFD results.

The size of  $\delta\mathbf{H}$ , in fact, provides a meaningful measure of the quality of an approximation of  $\mathbf{H}$ . To explain, suppose that  $\mathbf{H}$  is entered into a computation of the type:  $\mathbf{y} = \mathbf{H}\mathbf{x}$ , so that  $\mathbf{x} = \mathbf{H}^{-1}\mathbf{y}$ . Suppose further that  $\mathbf{H}$  is in error by an amount  $\delta\mathbf{H}$ . Then, as shown by Forsythe and Moler (1967), the relative errors introduced in  $\mathbf{x}$  due to errors  $\delta\mathbf{H}$  are bounded by  $q$  such that for any norm  $\|\cdot\|$ :

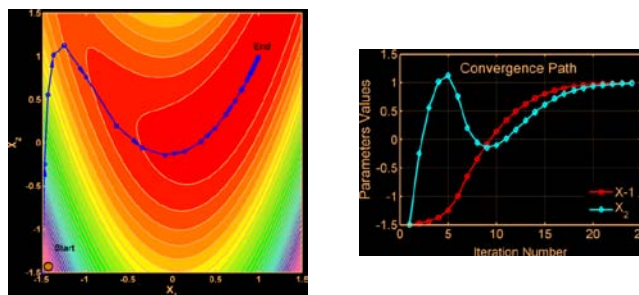
$$q = \frac{\|\delta\mathbf{H}\|}{\|\mathbf{H}\|} \kappa \geq \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|}, \quad (11)$$

where  $\kappa$ , is the condition number of  $\mathbf{H}$ . For the test case considered here,  $q$  was computed for all four commonly used norms: one, Euclidian, infinity and Frobenius (L1, L2, Linf, and Lfro). The upper bound for the CSD and CFD approximations was found to be  $q \leq 1.1416 \times 10^{-12}$  and  $q \leq 1.540 \times 10^{-6}$  respectively; and is largest for the L1 norm, and smallest for the Lfro norm ( $1.0146 \times 10^{-10}$ ,  $1.2546 \times 10^{-6}$ ). These results clearly indicate that, in addition to its stability, the SD approximation of second order derivatives in equations 8 is decisively of higher accuracy than ordinary FD approximation by several orders of magnitude.

## 5. Application in inversion

Utility of the SD method in optimization is demonstrated here by the classic problem of minimizing the Rosenbroke function (equation 3) which is a popular test function of optimization algorithms. The algorithm used in this example is a variant of Newton method with a Levenberg–Marquardt-type damping. The iteration step  $\Delta\mathbf{x}$  is determined by solution of the damped linear system:  $(\mathbf{H}(\mathbf{x}) + \lambda\mathbf{I})\Delta\mathbf{x} = -\mathbf{g}(\mathbf{x})$ , where  $\lambda$  is the damping factor; hence, the algorithm requires both the Hessian ( $\mathbf{H}$ ), and the gradient ( $\mathbf{g}$ ) of the objective function. The optimization results are shown in Fig. 6. It is worth noting that the routine

used internally checks the accuracy of the user's implementation of  $\mathbf{g}$  and  $\mathbf{H}$  and terminates with error flags if either derivative object fails to pass the preset accuracy level or if  $\mathbf{H}$  exhibits any asymmetry.



**Figure 4:** Results of optimization of the Rosenbroke test function using a variant of Newton's method with Levenberg–Marquardt-type damping requiring the gradient and hessian of the objective.

## 6. Summary

In this paper, the semiautomatic differentiation system was presented as alternative to FD schemes. The accuracy and simplicity of the method was illustrated by computation examples of important derivative structures including gradients, directional derivatives, Jacobians and Hessians. The SD technique, generalized to multi-parameter functions and extended to second-order derivatives, provides a complete system for the differentiation of real-valued functions. Compared to FD, the method is superior in performance with the added advantage of being step-size insensitive. It is also competitive with AD in performance, with the added advantage of being more easily accessible to students and researchers. With these features, the SD has a high potential as a pedagogical and research tool in the geosciences.

## 7. Acknowledgements

This research work was conducted at the Department of Earth Science, King Fahd University of Petroleum and Minerals, (KFUPM), Dhahran, Saudi Arabia. I thank Professor G. Korvin and Dr. A. Qahwash for their diligence while reviewing the manuscript and for their very valuable suggestions.

## 8. References

- Abokhodair, A. A.: Numerical tools for geoscience computations: Semiautomatic differentiation—SD. *Comput. Geosci.* **11**, 283-296 (2007).
- Abokhodair, A. A.: Complex differentiation tools for geophysical inversion. *Geophysics* 74(2), H1-H11 (2009).
- Abokhodair, A. A.: Filters on the fly: Least Squares Differentiation filters. (submitted to GEM).

- Burg, C. O. E, and Newman, J.C.: Computationally efficient, numerically exact design space derivatives via the complex Taylor's series expansion method. *Computers & Fluids* 32, 373-383 (2003).
- Cerviño, L. I. and Bewley, T. R.: On the extension of the complex-step derivative technique to pseudospectral algorithms, *J. Comp. Phys.*, 187, 544-549 (2003).
- Mark, H. and Workman Jr., J.: Derivatives in Spectroscopy: Part I – The Behavior of the derivative, *Spectroscopy* 18(4), 32-37 (2003).
- Martin, J. R. R. A., Kroo, I. M. and Alonso, J. j.: An Automated method for sensitivity analysis using complex variables, Paper AIAA 2001-689 (2001).
- Squire, W. and Trapp, G. Using complex variables to estimate derivatives of real functions, *SIAM Rev.*, 40(1), 110-112 (1998).
- Vatsa, V.: Computation of sensitivity derivatives of Navier-Stokes equations using complex variables. *Advances in Engineering Software* 31, 655–659 (2000).
- Wang, B. P.: Complex variable method for sensitivity analysis in structural optimization. 4th World Congress on Computational Mechanics, in conjunction with APCOM (2004).
- DePauw, D. J. W. and Vanrolleghem, P. A.: Avoiding the finite difference sensitivity analysis deathtrap by using the complex-step derivative approximation technique. 3rd biennial meeting of the International Environmental Modeling and Software Society (iEMSs), Burlington (2006).
- Orfanidis, S.J.: *Introduction to Signal Processing*. Prentice Hall, Englewood Cliffs, NJ, (1996).

# Formulation of the Stress Distribution Due to a Concentrated Force Acting on the Boundary of Viscoelastic Half-Space

Yun Peng<sup>1</sup> and Debao Zhou<sup>2</sup>

Department of Mechanical and Industrial Engineering  
University of Minnesota, Duluth MN 55812 USA  
<sup>1</sup>peng0234@d.umn.edu and <sup>2</sup>dzhou@d.umn.edu.

**Abstract** - This paper presents the derivative of the stress distribution in the condition that a point (or concentrated) force acts on the surface of a viscoelastic half-space. This solution is obtained through the combination of the solution to the stress distribution from the forces in both normal and tangential directions. The tangential-force viscoelastic problem is defined as the application of a tangential point force acting on the surface of a viscoelastic half-space and the normal-force viscoelastic problem is related to a normal force in the same situation. The elastic-viscoelastic correspondence principle is used in the derivation of the stress distribution. Two time-dependent functions, also called determining functions, are utilized to express the multiplication of the external force and the time-dependent viscoelastic material properties, i.e. the volumetric and deviatoric relaxation functions and Poisson's ratio. Using the boundary conditions and the equilibrium conditions of a half-space, the mathematical expressions of the determining functions can be explicitly obtained. The stress distribution in a viscoelastic material under general point force (with both normal and tangential components) is then obtained. These solutions can be used to formulate the stress distribution in the crushing or cutting of linear viscoelastic materials.

**Key words:** *Stress distribution; linear viscoelastic; tangential point force; half-space.*

## I INTRODUCTION

Cutting operations are involved in the process of many viscoelastic materials. Modeling the relationship between the cutting force and stress distribution can help predict the fracture due to cutting and identify different materials along the cutting path. Cutting problem can be modeled as a belt-shaped area force acting on the surface of a half-space. The study of the stress distribution due to a point force will be the first step in modeling the cutting stress distribution. A cutting force can be considered as the resultant of a normal and a tangential component acting on the contact surface between the tool and the material. The stress distribution in an *elastic* half-space due to a normal point force and that due to a tangential point force has been modeled by Boussinesq and Cerruti, respectively. Correspondingly, they are called Boussinesq's problem and Cerruti's problem. For viscoelastic materials, Talybly (2010) raised the method by substituting the multiplication of external force and viscoelastic material functions with time-dependent functions. This paper will concentrate on the formulation

of stress distribution in a viscoelastic under a tangential point force. The simplified model is shown in Fig. 1. Due to the asymmetry resulting from the tangential force, only Cartesian coordinates could be used. This makes our formulation much more complicated.

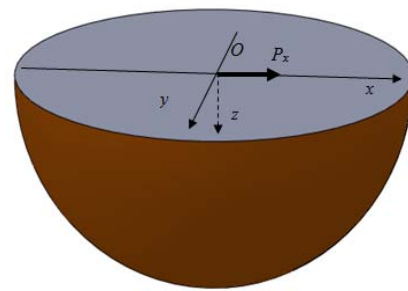


Fig. 1: A tangential force ( $P_x$ ) acts on the surface of a half-space

Furthermore, using the solution formulated in this paper, a solution to the problem in viscoelastic materials under both normal and tangential point forces can be obtained. This generalized solution can be used in many further problems, for example, the food slicing cut problems formulated by Zhou and McMurray (2011). The remainder of this paper is as follows: tangential-force viscoelastic problem is explained in Section II and the solution to this problem is derived in Section III. The general solution is shown in Section IV. The conclusions are drawn in Section V.

## II STATEMENT OF A VISCOELASTIC HALF-SPACE UNDER TANGENTIAL-FORCE

### A. Tangential Point Force

In this problem, a reference frame  $O-xyz$  is defined on the half-space as shown in Fig. 1, where the boundary of the half-space is at  $z = 0$ , and a tangential point force  $P_x$ , which could be a function of time, is applied at the origin along the  $x$ -axis, and the positive  $z$ -axis points towards the interior of the half-space. The stress solution to this problem for elastic materials was given by Cerruti in 1882 through the use of singularities from potential theory. The results were also presented by Love (1927). The displacement distributions at point  $(x, y, z)$  inside the half-space body are:

$$\begin{bmatrix} u^e \\ v^e \\ w^e \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} \frac{1}{R} + \frac{1}{R+z} + \frac{x^2}{R^3} - \frac{x^2}{R(R+z)^2} & -\frac{2}{R+z} + \frac{2x^2}{R(R+z)^2} \\ \frac{xy}{R^3} - \frac{xy}{R(R+z)^2} & \frac{2xy}{R(R+z)^2} \\ \frac{xz}{R^3} + \frac{x}{R(R+z)} & \frac{-2x}{R(R+z)} \end{bmatrix} \begin{bmatrix} \frac{P_x}{G} \\ \frac{P_y}{G} \\ \frac{P_z}{G} \end{bmatrix}$$

where  $u^e$ ,  $v^e$  and  $w^e$  denote the displacement in the positive  $x$ -,  $y$ - and  $z$ -axis directions in elastic case,  $R^2 = x^2 + y^2 + z^2$ ,  $P_x$  is the external tangential force,  $G$  is the elastic shear modulus and  $\nu$  is the Poisson's ratio. The stress distributions can then be obtained using the kinematic equations and constitutive equations for elastic materials.

The stresses at point  $(x, y, z)$  satisfy the following boundary conditions for this problem:

$$\begin{cases} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \sigma_{xx} dx dy + P_x = 0; & \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (y\sigma_{xz} - z\sigma_{xy}) dx dy = 0; \\ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \sigma_{yy} dx dy = 0; & \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x\sigma_{yz} - z\sigma_{yx}) dx dy = 0; \\ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \sigma_{zz} dx dy = 0; & \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (y\sigma_{zx} - x\sigma_{zy}) dx dy = 0, \end{cases}$$

where the three equations in left column represent the force balance, and the three equations in right column represent the torque balance.

### B. Viscoelastic Half-Space under Tangential Point Force

In our derivation, the viscoelastic effect is taken into consideration based on the original tangential-force elastic problem. The difference between elastic and viscoelastic problems lies in the constitutive equations. In an elastic material, the relationship between stress and strain can be described by Hooke's Law. However, in a viscoelastic problem, the material will show elastic behaviors like solids and also show viscous behaviors like fluids. This changes the relationship between stress and strain. During the calculation of the stress, instead of multiplying strain with material properties, such as Young's Modulus and Poisson's ratio, the convolutions of strain with relaxation functions are used. These relationships are shown as:

$$\begin{cases} \sigma_x = \int_0^t G_2(t-\tau) \frac{\partial \varepsilon}{\partial \tau} d\tau + \int_0^t G_1(t-\tau) \frac{\partial (\varepsilon_x - \varepsilon)}{\partial \tau} d\tau; & \sigma_{yz} = \sigma_{zy} = \int_0^t G_1(t-\tau) \frac{\partial (\varepsilon_{yz} - \varepsilon)}{\partial \tau} d\tau; \\ \sigma_y = \int_0^t G_2(t-\tau) \frac{\partial \varepsilon}{\partial \tau} d\tau + \int_0^t G_1(t-\tau) \frac{\partial (\varepsilon_y - \varepsilon)}{\partial \tau} d\tau; & \sigma_{zx} = \sigma_{xz} = \int_0^t G_1(t-\tau) \frac{\partial (\varepsilon_{zx} - \varepsilon)}{\partial \tau} d\tau; \\ \sigma_z = \int_0^t G_2(t-\tau) \frac{\partial \varepsilon}{\partial \tau} d\tau + \int_0^t G_1(t-\tau) \frac{\partial (\varepsilon_z - \varepsilon)}{\partial \tau} d\tau; & \sigma_{xy} = \sigma_{yx} = \int_0^t G_1(t-\tau) \frac{\partial (\varepsilon_{xy} - \varepsilon)}{\partial \tau} d\tau, \end{cases}$$

where  $G_1$  and  $G_2$  are the deviatoric and volumetric relaxation functions, and  $\varepsilon = (\varepsilon_x + \varepsilon_y + \varepsilon_z)/3$  is the mean strain.

Therefore, we state the tangential-force viscoelastic problem as follows: finding out the solutions to the stress distribution of a half-space body under a point tangential force applied to the surface, with boundary conditions and stress-strain relationships satisfied.

## III SOLUTIONS TO TANGENTIAL-FORCE VISCOELASTIC PROBLEM

### A. Displacements

For the linear viscoelastic case, by applying the elastic-viscoelastic correspondence principle and replacing  $\frac{P_x}{G}$  and  $\frac{P_y}{G}$  with  $\varphi(t)$  and  $\psi(t)$ , The following expressions can be obtained:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} u_\varphi & 2u_\psi \\ v_\varphi & 2v_\psi \\ w_\varphi & 2w_\psi \end{bmatrix} \begin{bmatrix} \varphi(t) \\ \psi(t) \end{bmatrix}$$

where  $u$ ,  $v$  and  $w$  denote the displacements in the positive directions of the  $x$ -,  $y$ - and  $z$ -axis with:

$$\begin{cases} u_\varphi = \frac{1}{R} + \frac{1}{R+z} + \frac{x^2}{R^3} - \frac{x^2}{R(R+z)^2}; & u_\psi = -\frac{1}{R+z} + \frac{x^2}{R(R+z)^2}; \\ v_\varphi = \frac{xy}{R^3} - \frac{xy}{R(R+z)^2}; & v_\psi = \frac{xy}{R(R+z)^2}; \\ w_\varphi = \frac{xz}{R^3} + \frac{x}{R(R+z)}; & w_\psi = -\frac{x}{R(R+z)}. \end{cases}$$

In following sections, all derivations will be conducted with the terms of  $\varphi$  and  $\psi$  carried out individually.

### B. Normal Strains

The normal strains are obtained via the derivative of displacements with coordinates. The obtained normal strains are:

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} \frac{\partial u_\varphi}{\partial x} & 2 \frac{\partial u_\psi}{\partial x} \\ \frac{\partial v_\varphi}{\partial x} & 2 \frac{\partial v_\psi}{\partial x} \\ \frac{\partial w_\varphi}{\partial x} & 2 \frac{\partial w_\psi}{\partial x} \end{bmatrix} \begin{bmatrix} \varphi(t) \\ \psi(t) \end{bmatrix}$$

where:

$$\begin{cases} \frac{\partial u_\varphi}{\partial x} = -\frac{3x}{R(R+z)^2} + \frac{x^3}{R^3(R+z)^2} + \frac{2x^3}{R^2(R+z)^3} + \frac{x}{R^3} - \frac{3x^3}{R^5} \\ \frac{\partial u_\psi}{\partial x} = \frac{3x}{R(R+z)^2} - \frac{x^3}{R^3(R+z)^2} - \frac{2x^3}{R^2(R+z)^3} \\ \frac{\partial v_\varphi}{\partial y} = -\frac{x}{R(R+z)^2} + \frac{xy^2}{R^3(R+z)^2} + \frac{2xy^2}{R^2(R+z)^3} + \frac{x}{R^3} - \frac{3xy^2}{R^5} \\ \frac{\partial v_\psi}{\partial y} = \frac{x}{R(R+z)^2} - \frac{xy^2}{R^3(R+z)^2} - \frac{2xy^2}{R^2(R+z)^3} \\ \frac{\partial w_\varphi}{\partial z} = -\frac{xz}{R^3(R+z)} - \frac{x}{R^2(R+z)} + \frac{x}{R^3} - \frac{3xz^2}{R^5} = -\frac{3xz^2}{R^5} \\ \frac{\partial w_\psi}{\partial z} = \frac{xz}{R^3(R+z)} + \frac{x}{R^2(R+z)} = \frac{x(R+z)}{R^3(R+z)} = \frac{x}{R^3} \end{cases}$$

Thus the mean strain is:

$$\varepsilon = \frac{1}{3}(\varepsilon_x + \varepsilon_y + \varepsilon_z) = \frac{1}{4\pi} \left( -\frac{2x}{3R^3} \right) \varphi(t) + \frac{1}{2\pi} \left( \frac{2x}{3R^3} \right) \psi(t)$$

and the mean stress is

$$\sigma = G_2 * d\varepsilon = \frac{1}{4\pi} \left( -\frac{2x}{3R^3} \right) G_2 * d\varphi(t) + \frac{1}{2\pi} \left( \frac{2x}{3R^3} \right) G_2 * d\psi(t)$$



C. Shear Strains

The shear strains  $\epsilon_{yz}$ ,  $\epsilon_{xz}$  and  $\epsilon_{xy}$  can be obtained as:

$$\begin{bmatrix} \epsilon_{yz} \\ \epsilon_{xz} \\ \epsilon_{xy} \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} \frac{1}{2} \left( \frac{\partial v_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial y} \right) \\ \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial y} \right) \\ \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial v_\varphi}{\partial y} \right) \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} \frac{1}{2} \left( \frac{\partial v_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial y} \right) & \frac{1}{2} \left( \frac{\partial v_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial y} \right) \\ \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial y} \right) & \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial y} \right) \\ \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial v_\varphi}{\partial y} \right) & \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial v_\varphi}{\partial y} \right) \end{bmatrix} \begin{bmatrix} \varphi(t) \\ \psi(t) \end{bmatrix}$$

where

$$\begin{cases} \frac{1}{2} \left( \frac{\partial v_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial y} \right) = -\frac{3xyz}{R^5}; \\ \frac{1}{2} \left( \frac{\partial v_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial y} \right) = 0; \\ \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial x} \right) = -\frac{3x^2z}{R^5}; \\ \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial x} \right) = 0; \\ \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial y} + \frac{\partial v_\varphi}{\partial x} \right) = -\frac{y}{R(R+z)^2} + \frac{x^2y}{R^3(R+z)^2} + \frac{2x^2y}{R^2(R+z)^3} - \frac{3x^2y}{R^5}; \\ \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial y} + \frac{\partial v_\varphi}{\partial x} \right) = \frac{y}{R(R+z)^2} - \frac{x^2y}{R^3(R+z)^2} - \frac{2x^2y}{R^2(R+z)^3}. \end{cases}$$

Noted is that the coefficients of  $\psi(t)$  in  $\epsilon_{yz}$  and  $\epsilon_{xz}$  are zero, then the strains can be written as

$$\begin{bmatrix} \epsilon_{yz} \\ \epsilon_{xz} \\ \epsilon_{xy} \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} -\frac{3x^2z}{R^5} & 0 \\ -\frac{3xyz}{R^5} & 0 \\ \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial y} + \frac{\partial v_\varphi}{\partial x} \right) & \frac{1}{2} \left( \frac{\partial u_\varphi}{\partial y} + \frac{\partial v_\varphi}{\partial x} \right) \end{bmatrix} \begin{bmatrix} \varphi(t) \\ \psi(t) \end{bmatrix}$$

D. Stresses

With strains known, the stresses can be obtained from viscoelastic constitutive equations (Zhang 1994, p. 63) as:

$$\begin{aligned} \sigma_x &= \frac{1}{4\pi} \left( -\frac{2x}{3R^3} \right) G_2 * d\varphi(t) + \frac{1}{2\pi} \left( \frac{2x}{3R^3} \right) G_2 * d\psi(t) \\ &+ \frac{1}{4\pi} \left( -\frac{3x}{R(R+z)^2} + \frac{x^3}{R^3(R+z)^2} + \frac{2x^3}{R^2(R+z)^3} + \frac{x}{R^3} - \frac{3x^3}{R^5} + \frac{2x}{3R^3} \right) G_1 * d\varphi(t) \\ &+ \frac{1}{2\pi} \left( \frac{3x}{R(R+z)^2} - \frac{x^3}{R^3(R+z)^2} - \frac{2x^3}{R^2(R+z)^3} - \frac{2x}{3R^3} \right) G_1 * d\psi(t); \\ \sigma_y &= \frac{1}{4\pi} \left( -\frac{2x}{3R^3} \right) G_2 * d\varphi(t) + \frac{1}{2\pi} \left( \frac{2x}{3R^3} \right) G_2 * d\psi(t) \\ &+ \frac{1}{4\pi} \left( -\frac{x}{R(R+z)^2} + \frac{xy^2}{R^3(R+z)^2} + \frac{2xy^2}{R^2(R+z)^3} + \frac{x}{R^3} - \frac{3xy^2}{R^5} + \frac{2x}{3R^3} \right) G_1 * d\varphi(t) \\ &+ \frac{1}{2\pi} \left( \frac{x}{R(R+z)^2} - \frac{xy^2}{R^3(R+z)^2} - \frac{2xy^2}{R^2(R+z)^3} - \frac{2x}{3R^3} \right) G_1 * d\psi(t); \\ \sigma_z &= \frac{1}{4\pi} \left( -\frac{2x}{3R^3} \right) G_2 * d\varphi(t) + \frac{1}{2\pi} \left( \frac{2x}{3R^3} \right) G_2 * d\psi(t) \\ &+ \frac{1}{4\pi} \left( -\frac{3xz^2}{R^5} + \frac{2x}{3R^3} \right) G_1 * d\varphi(t) \\ &+ \frac{1}{2\pi} \left( \frac{x}{R^3} - \frac{2x}{3R^3} \right) G_1 * d\psi(t); \end{aligned}$$

$$\begin{aligned} \sigma_{yz} &= G_1 * d\epsilon_{yz} = \frac{1}{4\pi} \left( -\frac{3xyz}{R^5} \right) G_1 * d\varphi(t); \\ \sigma_{xz} &= G_1 * d\epsilon_{xz} = \frac{1}{4\pi} \left( -\frac{3x^2z}{R^5} \right) G_1 * d\varphi(t); \\ \sigma_{xy} &= G_1 * d\epsilon_{xy} = \frac{1}{4\pi} \left( -\frac{y}{R(R+z)^2} + \frac{x^2y}{R^3(R+z)^2} + \frac{2x^2y}{R^2(R+z)^3} - \frac{3x^2y}{R^5} \right) G_1 * d\varphi(t) \\ &+ \frac{1}{2\pi} \left( \frac{y}{R(R+z)^2} - \frac{x^2y}{R^3(R+z)^2} - \frac{2x^2y}{R^2(R+z)^3} \right) G_1 * d\psi(t). \end{aligned}$$

E. Boundary Conditions

Now consider the boundary conditions. Integrating the stress component and noticing that convolution will not take part in spatial integrals, the first boundary condition becomes:

$$\frac{1}{4\pi} G_1 * d\varphi(t) \int \int \left( -\frac{3x^2z}{R^5} \right) dx dy + P_x(t) = -\frac{1}{2} G_1 * d\varphi(t) + P_x(t) = 0$$

The above equation means that the resultant force in x direction on the O-x-y plane cancels the external force. It is worthy to mention that the integrals of the other stress components are all zeros because all of them are odd functions of x with the integrations over  $x \in (-\infty, \infty)$ .

The value of the above integral should be zero as required by the balance of force. Then the following expression can be obtained:

$$G_1 * d\varphi(t) = 2P_x(t) \tag{1}$$

F. Equilibrium Equation

Bringing all the stresses into the first equilibrium

$$\text{equation } E = \frac{\partial \sigma_x}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \sigma_z}{\partial z} = 0 \text{ will yield:}$$

$$\begin{aligned} E &= \frac{1}{4\pi} \frac{\partial}{\partial x} \left( \frac{2x}{3R^3} \right) G_2 * d(-\varphi(t) + 2\psi(t)) \\ &+ \frac{1}{4\pi} \left[ \frac{\partial}{\partial x} \left( \frac{\partial u_\varphi}{\partial x} + \frac{2x}{3R^3} \right) + \frac{1}{2} \frac{\partial}{\partial y} \left( \frac{\partial u_\varphi}{\partial y} + \frac{\partial v_\varphi}{\partial x} \right) + \frac{1}{2} \frac{\partial}{\partial z} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial x} \right) \right] G_1 * d\varphi(t) \\ &+ \frac{1}{2\pi} \left[ \frac{\partial}{\partial x} \left( \frac{\partial u_\psi}{\partial x} - \frac{2x}{3R^3} \right) + \frac{1}{2} \frac{\partial}{\partial y} \left( \frac{\partial u_\psi}{\partial y} + \frac{\partial v_\psi}{\partial x} \right) \right] G_1 * \psi(t) = 0, \end{aligned}$$

where the coefficients of  $G_1 * d\varphi$  and  $G_1 * \psi(t)$  are:

$$\begin{cases} \frac{\partial}{\partial x} \left( \frac{\partial u_\varphi}{\partial x} + \frac{2x}{3R^3} \right) + \frac{1}{2} \frac{\partial}{\partial y} \left( \frac{\partial u_\varphi}{\partial y} + \frac{\partial v_\varphi}{\partial x} \right) + \frac{1}{2} \frac{\partial}{\partial z} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial x} \right) \\ = \frac{\partial}{\partial x} \left( \frac{\partial u_\varphi}{\partial x} \right) + \frac{1}{2} \frac{\partial}{\partial y} \left( \frac{\partial u_\varphi}{\partial y} + \frac{\partial v_\varphi}{\partial x} \right) + \frac{1}{2} \frac{\partial}{\partial z} \left( \frac{\partial u_\varphi}{\partial z} + \frac{\partial w_\varphi}{\partial x} \right) + \frac{\partial}{\partial x} \frac{2x}{3R^3} \\ = \frac{\partial}{\partial x} \frac{2x}{3R^3}; \\ \frac{\partial}{\partial x} \left( \frac{\partial u_\psi}{\partial x} - \frac{2x}{3R^3} \right) + \frac{1}{2} \frac{\partial}{\partial y} \left( \frac{\partial u_\psi}{\partial y} + \frac{\partial v_\psi}{\partial x} \right) \\ = \frac{\partial}{\partial x} \left( \frac{\partial u_\psi}{\partial x} \right) + \frac{1}{2} \frac{\partial}{\partial y} \left( \frac{\partial u_\psi}{\partial y} + \frac{\partial v_\psi}{\partial x} \right) - \frac{\partial}{\partial x} \frac{2x}{3R^3} = \frac{\partial}{\partial x} \frac{x}{R^3} - \frac{\partial}{\partial x} \frac{2x}{3R^3} \\ = \frac{\partial}{\partial x} \frac{x}{3R^3}. \end{cases}$$

Thus there is:

$$E = \frac{1}{4\pi} \frac{\partial}{\partial x} \left( \frac{2x}{3R^3} \right) G_2 * d(-\varphi(t) + 2\psi(t)) + \frac{1}{4\pi} \frac{\partial}{\partial x} \left( \frac{2x}{3R^3} \right) G_1 * d(\varphi(t) + \psi(t)) = 0$$

By eliminating the same coefficients containing the coordinates, there is:

$$G_2 * d(\varphi(t) - 2\psi(t)) - G_1 * d(\varphi(t) + \psi(t)) = 0 \quad (2)$$

The second equilibrium equation will yield the same equation relationship as given in (1). The third equilibrium equation  $\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} = 0$  has been satisfied automatically.

#### G. Solution to Tangential-Force Viscoelastic Problem

$\varphi(t)$  and  $\psi(t)$  are linearly independent in Equations (1) and (2). Thus,  $\varphi(t)$  and  $\psi(t)$  can be uniquely solved by solving the Volterra integral equations of the second kind (Zhang 1994, p.150-161).

Given the initial condition  $P_x(0) = 0$ , all stress and strain components are zero at  $t = 0$  and  $\varphi(0) = 0$ ,  $\psi(0) = 0$ , the solution to  $\varphi(t)$  can be obtained. Noted is that the detailed derivations are not included in this part for the purpose of concision since the detailed derivations and the proof of the uniqueness can be found in the paper by Talybly (2010). Manipulating equations (1) and (2), there is:

$$(G_1 + 2G_2) * d(\varphi(t) - 2\psi(t)) = 6P_x(t)$$

Viewing  $\varphi(t) - 2\psi(t)$  in whole and solving this Volterra integral equation,  $\psi(t)$  can be obtained with  $\varphi(t)$  previously solved.

Taking advantage of (1) and (2), the term of  $G_2$  can be eliminated and the stress distributions can be written as follows:

$$\begin{cases} \sigma_x = \frac{1}{2\pi} \left[ \left( -\frac{3x}{R(R+z)^2} + \frac{x^3}{R^3(R+z)^2} + \frac{2x^3}{R^2(R+z)^3} + \frac{x}{R^3} \right) (P_x(t) - G_1 * d\psi(t)) - \frac{3x^3}{R^5} P_x(t) \right]; \\ \sigma_y = \frac{1}{2\pi} \left[ \left( -\frac{x}{R(R+z)^2} + \frac{xy^2}{R^3(R+z)^2} + \frac{2xy^2}{R^2(R+z)^3} + \frac{x}{R^3} \right) (P_x(t) - G_1 * d\psi(t)) - \frac{3xy^2}{R^5} P_x(t) \right]; \\ \sigma_z = -\frac{3xz^2}{2\pi R^5} P_x(t); \\ \sigma_{yz} = -\frac{3xyz}{2\pi R^5} P_x(t); \\ \sigma_{xz} = -\frac{3x^2z}{2\pi R^5} P_x(t); \\ \sigma_{xy} = \frac{1}{2\pi} \left[ \left( -\frac{y}{R(R+z)^2} + \frac{x^2y}{R^3(R+z)^2} + \frac{2x^2y}{R^2(R+z)^3} \right) (P_x(t) - G_1 * d\psi(t)) - \frac{3x^2y}{R^5} P_x(t) \right]. \end{cases} \quad (3)$$

Comparing with classic Cerruti's solutions (Johnson, 1985 p.69-70), we found that the term  $P_x(t) - G_1 * d\psi$  in viscoelastic solutions plays the same role as the term  $(1-2\mu)P$  in elastic solutions.

#### IV STRESS DISTRIBUTION IN A VISCOELASTIC HALF-SPACE UNDER GENERAL POINT FORCES

We now consider a general force with both normal and tangential components, in which two tangential forces  $P_1(t)$  and  $P_2(t)$ , along  $x$ - and  $y$ - axis respectively and one normal force  $P_3(t)$  along  $z$ -axis are applied at the origin. The displacement and stress distributions are obtained using the superposition of the displacement and stress distribution by  $P_1(t)$ ,  $P_2(t)$  and  $P_3(t)$ . The stress and displacement distributions for the problem when only one tangential force  $P_1(t) = P_x(t)$  along  $x$ -axis has been discussed in previous discussions. For disambiguation, we rewrite the solution with proper superscript ( $i$ ) ( $i = 1, 2$  or  $3$ ) to denote the applied forces. Therefore we have:

$$\sigma_x^{(i)} = \sigma_x, \quad \sigma_y^{(i)} = \sigma_y, \quad \sigma_z^{(i)} = \sigma_z,$$

$$\sigma_{yz}^{(i)} = \sigma_{yz}, \quad \sigma_{xz}^{(i)} = \sigma_{xz}, \quad \sigma_{xy}^{(i)} = \sigma_{xy}.$$

where the expressions of the right hand side are the same as in (3) only with  $P_x(t)$  replaced by  $P_i(t)$ .

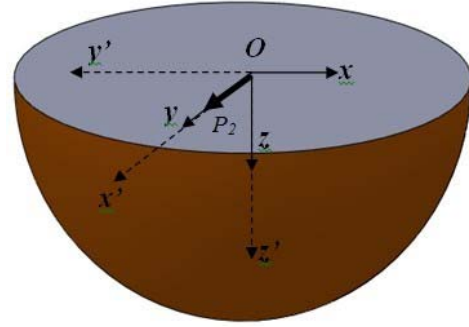


Fig. 2: Model for coordinate's transformation

When there is a single tangential force  $P_2(t)$  applied on the  $O$ - $x$ - $y$  plane at point  $O$  along  $y$ -axis, the stress distribution is obtained through the calculation of the frame rotation. In this method, as shown in Fig. 2, we first rotate the frames around  $z$ -axis by  $90^\circ$  and denote the new coordinate system as  $O$ - $x'$ - $y'$ - $z'$ .

The corresponding Jacob matrix is:

$$R = \begin{bmatrix} \cos(90^\circ) & -\sin(90^\circ) & 0 \\ \sin(90^\circ) & \cos(90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The coordinates are transformed as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} -y' \\ x' \\ z' \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} y \\ -x \\ z \end{bmatrix}$$

The corresponding stress tensor is transformed as

$$\begin{pmatrix} \sigma_x & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_y & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z \end{pmatrix} = R \begin{pmatrix} \sigma_{x'} & \sigma_{x'y'} & \sigma_{x'z'} \\ \sigma_{y'x'} & \sigma_{y'} & \sigma_{y'z'} \\ \sigma_{z'x'} & \sigma_{z'y'} & \sigma_{z'} \end{pmatrix} R^T = \begin{pmatrix} \sigma_{y'} & -\sigma_{y'x'} & -\sigma_{y'z'} \\ -\sigma_{x'y'} & \sigma_x & \sigma_{xz} \\ -\sigma_{z'y'} & \sigma_{z'x'} & \sigma_z \end{pmatrix}$$

Correspondingly, the determining functions can be rewritten as (Talybly, 2010):



$$\varphi^{(i)}(t) = \frac{2}{G_1(0)} \left( P_i(t) + \int_0^t \Gamma(t-\tau) P_i(\tau) d\tau \right), i=1,2,3,$$

$$\psi^{(i)}(t) = \frac{1}{2} \varphi^{(i)}(t) - \frac{3}{G_1(0) + 2G_2(0)} \left( P_i(t) + \int_0^t U(t-\tau) P_i(\tau) d\tau \right), i=1,2,3.$$

In coordinates  $O-x'y'z'$ ,  $P_2(t)$  is applied at the origin along  $x'$ -axis. Based on the solution we obtained in (3), the stress in  $O-x'y'z'$  system can be obtained as:

$$\begin{cases} \sigma_x^{(2)} = \frac{1}{2\pi} \left[ -\frac{3x'}{R(R'+z)^2} + \frac{x'^3}{R^3(R'+z)^2} + \frac{2x'^3}{R^2(R'+z)^3} + \frac{x'}{R^3} \right] (P_2(t) - G_1 * d\psi^{(2)}(t)) - \frac{3x'^3}{R^5} P_2(t); \\ \sigma_y^{(2)} = \frac{1}{2\pi} \left[ -\frac{x'}{R(R'+z)^2} + \frac{x'y'^2}{R^3(R'+z)^2} + \frac{2x'y'^2}{R^2(R'+z)^3} + \frac{x'}{R^3} \right] (P_2(t) - G_1 * d\psi^{(2)}(t)) - \frac{3x'y'^2}{R^5} P_2(t); \\ \sigma_z^{(2)} = \frac{3x'^2 z'}{2\pi R^5} P_2(t); \\ \sigma_{y'z'}^{(2)} = \frac{3x'y'z'}{2\pi R^5} P_2(t); \\ \sigma_{x'z'}^{(2)} = -\frac{3x'^2 z'}{2\pi R^5} P_2(t); \\ \sigma_{x'y'}^{(2)} = \frac{1}{2\pi} \left[ -\frac{y'}{R(R'+z)^2} + \frac{x^2 y'}{R^3(R'+z)^2} + \frac{2x^2 y'}{R^2(R'+z)^3} \right] (P_2(t) - G_1 * d\psi^{(2)}(t)) - \frac{3x^2 y'}{R^5} P_2(t). \end{cases} \quad (4)$$

where  $R'^2 = x'^2 + y'^2 + z'^2$

After transformation to  $O-xyz$  coordinates, stresses in (4) can be written as:

$$\begin{cases} \sigma_x^{(2)} = \frac{1}{2\pi} \left[ -\frac{y}{R(R+z)^2} + \frac{x^2 y}{R^3(R+z)^2} + \frac{2x^2 y}{R^2(R+z)^3} + \frac{y}{R^3} \right] (P_2(t) - G_1 * d\psi^{(2)}(t)) - \frac{3x^2 y}{R^5} P_2(t); \\ \sigma_y^{(2)} = \frac{1}{2\pi} \left[ -\frac{3y}{R(R+z)^2} + \frac{y^3}{R^3(R+z)^2} + \frac{2y^3}{R^2(R+z)^3} + \frac{y}{R^3} \right] (P_2(t) - G_1 * d\psi^{(2)}(t)) - \frac{3y^3}{R^5} P_2(t); \\ \sigma_z^{(2)} = -\frac{3yz^2}{2\pi R^5} P_2(t); \\ \sigma_{yz}^{(2)} = -\frac{3yz^2}{2\pi R^5} P_2(t); \\ \sigma_{xz}^{(2)} = -\frac{3xyz}{2\pi R^5} P_2(t); \\ \sigma_{xy}^{(2)} = \frac{1}{2\pi} \left[ -\frac{x}{R(R+z)^2} + \frac{xy^2}{R^3(R+z)^2} + \frac{2xy^2}{R^2(R+z)^3} \right] (P_2(t) - G_1 * d\psi^{(2)}(t)) - \frac{3xy^2}{R^5} P_2(t). \end{cases} \quad (5)$$

When a single normal force  $P_3(t)$  is applied at point  $O$  and along the positive  $z$ -axis, the stress distribution is shown in (6). Noted is that this expression has been given by Talybly (2010).

$$\begin{cases} \sigma_x^{(3)} = \frac{1}{2\pi} \left[ \frac{1}{R(R+z)} (P_3(t) - G_1 * d\psi^{(3)}(t)) + \left( -\frac{3z}{R^3} + \frac{3z^2}{R^4} \right) P_3(t) \right]; \\ \sigma_y^{(3)} = \frac{1}{2\pi} \left( \frac{z}{R^3} - \frac{1}{R(R+z)} \right) (P_3(t) - G_1 * d\psi^{(3)}(t)); \\ \sigma_z^{(3)} = -\frac{3z^3}{2\pi R^5} P_3(t); \\ \sigma_{yz}^{(3)} = 0; \\ \sigma_{xz}^{(3)} = -\frac{3xz^2}{2\pi R^5} P_3(t); \\ \sigma_{xy}^{(3)} = 0. \end{cases} \quad (6)$$

Based on (4), (5) and (6), we provide the solution of stress distributions to the general problem as follows:

$$\begin{cases} \bar{\sigma}_x^{(3)}(t) = \sum_{i=1}^3 \sigma_x^{(i)}(t), \quad \bar{\sigma}_{yz}^{(3)}(t) = \sum_{i=1}^3 \sigma_{yz}^{(i)}(t), \\ \bar{\sigma}_y^{(3)}(t) = \sum_{i=1}^3 \sigma_y^{(i)}(t), \quad \bar{\sigma}_{xz}^{(3)}(t) = \sum_{i=1}^3 \sigma_{xz}^{(i)}(t), \\ \bar{\sigma}_z^{(3)}(t) = \sum_{i=1}^3 \sigma_z^{(i)}(t), \quad \bar{\sigma}_{yz}^{(3)}(t) = \sum_{i=1}^3 \sigma_{yz}^{(i)}(t). \end{cases}$$

Similarly, the displacement distributions to viscoelastic problem ( $\bar{u}$ ,  $\bar{v}$  and  $\bar{w}$ ) as follows:

$$\begin{cases} \bar{u} = \sum_{i=1}^3 u^{(i)}, \\ \bar{v} = \sum_{i=1}^3 v^{(i)}, \\ \bar{w} = \sum_{i=1}^3 w^{(i)}, \end{cases}$$

where the components are:

$$\begin{cases} \begin{bmatrix} u^{(1)} \\ v^{(1)} \\ w^{(1)} \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} \frac{1}{R} + \frac{1}{R+z} + \frac{x^2}{R^3} - \frac{x^2}{R(R+z)^2} & -\frac{2}{R+z} + \frac{2x^2}{R(R+z)^2} \\ \frac{xy}{R^3} - \frac{xy}{R(R+z)^2} & \frac{2xy}{R(R+z)^2} \\ \frac{xz}{R^3} + \frac{x}{R(R+z)} & \frac{-2x}{R(R+z)} \end{bmatrix} \begin{bmatrix} \varphi^{(1)}(t) \\ \psi^{(1)}(t) \end{bmatrix} \\ \begin{bmatrix} u^{(2)} \\ v^{(2)} \\ w^{(2)} \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} \frac{1}{R} + \frac{1}{R+z} + \frac{y^2}{R^3} - \frac{y^2}{R(R+z)^2} & \frac{-2}{R+z} + \frac{2y^2}{R(R+z)^2} \\ \frac{xy}{R^3} - \frac{xy}{R(R+z)^2} & \frac{2xy}{R(R+z)^2} \\ \frac{yz}{R^3} + \frac{y^2}{R(R+z)} & \frac{-2y}{R(R+z)} \end{bmatrix} \begin{bmatrix} \varphi^{(2)}(t) \\ \psi^{(2)}(t) \end{bmatrix} \\ \begin{bmatrix} u^{(3)} \\ v^{(3)} \\ w^{(3)} \end{bmatrix} = \frac{1}{4\pi} \begin{bmatrix} \frac{xz}{R^3} - \frac{x}{R(R+z)} & \frac{2x}{R(R+z)} \\ \frac{yz}{R^3} - \frac{y}{R(R+z)} & \frac{2y}{R(R+z)} \\ \frac{z}{R^3} + \frac{2}{R} & \frac{-2}{R} \end{bmatrix} \begin{bmatrix} \varphi^{(3)}(t) \\ \psi^{(3)}(t) \end{bmatrix} \end{cases}$$

The results will be used in our future research about the formulation of the cuttings for linear viscoelastic materials, in which a distributive force is used instead of a point force. We could obtain the stress response  $\sigma_{ij}(\vec{r}, t)$  to a certain distributive force by considering the point force  $\vec{P}$  as a function of  $\xi$  and  $\zeta$ , replacing  $x$  by  $x - \xi$ ,  $y$  by  $y - \zeta$  and  $R$  by  $\bar{R}^2 = (x - \xi)^2 + (y - \zeta)^2 + z^2$ , and then integrating the corresponding stress or displacement components for  $\xi$  and  $\zeta$  in  $x$ - $y$  plane.

## V CONCLUSION

The solutions to the stress and displacement distributions in a viscoelastic half-space under a point force are presented in this paper. The method is based on the elastic-viscoelastic corresponding principle. The solution to the displacement of an elastic half-space under a tangential point force was used as the displacement solution to the viscoelastic problem. Based on the equilibrium equations and boundary conditions of a viscoelastic half-space, we obtained the solution to the two time-dependent determining functions via the Volterra integral equations of the second kind. Then, the stress distribution due to a tangential force in  $y$ -axis direction is solved based on a frame rotation method.

Finally, by combining the solutions when there is only a normal force component, we get the results for the generalized case, where an arbitrary force with three non-zero components in  $x$ ,  $y$  and  $z$  directions, is applied. These results could be further used to solve the stress distribution in a viscoelastic half-space under a distributive force, by taking the integral over the area where the distributive force is applied.

## VI REFERENCES

Boussinesq, J., *Application des Potentiels a l'Etude de l'Equilibre et du Mouvement des Solides Elastiques*, Gauthier-Villars, Paris, (1885).

Love, A. E. H., *A Treatise on the Mathematical Theory of Elasticity*, 4th Edition. Cambridge University Press (1927).

Talybly, L. K., 'Boussinesq's viscoelastic problem on normal concentrated force on a half-space surface', *Mechanics of Time-Dependent Materials*. 3:253-259, (2010).

Xu, Z., *Elasticity Mechanics (Tan Xing Li Xue) 5<sup>th</sup> Edition*, Higher-level Education Publication, in Chinese, (1996).

Zhang, C. Y., *Viscoelastic Fracture Mechanics (Nian Tan Xing Duan Lie Li Xue)*, Huazhong University of Science and Technology Press, in Chinese, (1994).

Zhou, D. and McMurray, G., "Slicing Cuts on Food Materials Using Robotic Controlled Razor Blade", *Modelling and Simulation in Engineering*, Accepted and to be published in Nov 2011, Hindawi Publishing Corporation, (2011).

Johnson, K. L., *Contact Mechanics*, Cambridge University Press, (1985).

---

## Conjugate Gradient Type Algorithms for Indefinite Linear Systems

Marek Szularz

30 July 2012

**Abstract** This paper presents the new versions of the well established iterative Krylov-subspace solvers for large, sparse and symmetric linear systems. Notably, the classical Hestenes's and Stiefel's Conjugate Gradient (CG) algorithm is generalized into indefinite case, resulting in a breakdown-free method.

Also, the indefinite variant of the well established Conjugate Residual (CR) algorithm, is derived here, merely as a simple correction to the solution of the generalized CG method. Such a new CR algorithm has the second advantage over the classical CR algorithm because of an 'easy' computation of the residual norms.

**Keywords** GMRES Algorithm · Conjugate Gradient Algorithm · Conjugate Residual Algorithm · Arnoldi method · Lanczos method

**Mathematics Subject Classification (2000)** 15A06 · 15A23 · 15A60 · 15B99 · 65F99 · 93E24

### 1 Introduction

The Krylov subspace projection methods for solving the linear systems  $Ax = b$ , where  $A \in \mathbb{R}^{n \times n}$  is an arbitrary, nonsingular matrix, are the methods in which the approximate solution is obtained as

$$x_m = x_0 + V_m y_m, \quad y_m \in \mathbb{R}^m, \quad (1)$$

where  $V_m = [v_1, \dots, v_m]$  (with  $v_i \in \mathbb{R}^n$ ) is the orthonormal basis for the  $m$ -dimensional Krylov subspace

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \dots, A^{m-1} r_0\}, \quad (2)$$

---

This work was supported by the Engineering and Physical Sciences Research Council (UK) under grant GR/L75139

---

M. Szularz  
School of Computing & Information Engineering, University of Ulster, Coleraine BT52 1SA,  
Northern Ireland

and  $x_0$  and  $r_0$  are the initial guess and the corresponding residual  $b - Ax_0$  respectively. Thus the  $m$ -th step residual will take the form:

$$r_m = b - Ax_m$$

The 'optimal mix' of Krylov vectors in (1) depends on the choice of the inner product norm in which the error is minimized at each step. The optimization procedure applied here may result in a solution of a problem involving matrix  $H_m \in \mathbb{R}^{m \times m}$ , the orthogonal projection of  $A$  onto the Krylov subspace. When  $A$  is unsymmetric, Arnoldi's algorithm can be used to generate  $V_m$  and  $H_m = V_m^T A V_m$ , that is upper Hessenberg. In particular, with some inner product norms, this approach may lead to the solution of the linear system

$$H_m y_m = u_m, \quad (3)$$

where the right-hand side  $u_m$  depends precisely on the choice of such a norm.

A Krylov subspace method for symmetric linear systems that applies the Lanczos algorithm (that can be viewed as a symmetric variant of Arnoldi method) to generate  $V_m$  ( $H_m$ ) and explicitly uses system (3) to compute  $y_m$  at each step is known (see [5]) as a Direct-Lanczos method for linear systems. In such a case  $H_m = T_m$  is symmetric tridiagonal. Because of this simplification the approximate solution  $x_m$  does not have to be computed as explicitly as (1) and (2) suggest; instead  $x_m$  can be computed as a simple update of  $x_{m-1}$  in the form of some linear combination of  $x_{m-1}$  and  $p_m$ , where  $p_m$  is usually known as the  $m$ -th step search direction. In turn,  $p_m$  can be computed as some linear combination of the previous search direction  $p_{m-1}$  and the current Lanczos vector  $v_m$ . Consequently  $V_m$  need not to be stored entirely, which is by far the most important issue with such methods. Thus any Direct-Lanczos (D-Lanczos) method will take advantage of this i.e. will compute  $y_m$  ( $x_m$ ) in a progressive manner. In particular, in a class of Direct-Lanczos methods for linear systems that minimize  $\|r_m^T A r_m\|_2$  at each step,  $u_m$  in equation (3) turns out to be  $\beta e_1$ , where  $\beta = \|r_0\|_2$  (see [5] pp. 152-158). D-Lanczos method may be breakdown prone if  $A$  is not positive definite, and this depends on the factorization applied to  $T_m$ .

A Direct-Lanczos method defined above may have its mathematical equivalent<sup>1</sup> in Conjugate Gradient (CG) method. CG method can be derived from any equivalent D-Lanczos method by observing, and imposing the  $A$ -conjugacy condition of the two subsequent search directions. Quoting Ashby et al. [1], a Conjugate Gradient (CG) method is defined as 'a gradient method in which the iterates are chosen from a nested sequence of translated Krylov subspaces in such a way that the error is minimal in the given inner product norm at each step'. When  $A$  is not positive definite then CG algorithm is not guaranteed to be breakdown-free.

A D-Lanczos method that minimizes  $\|r_m\|_2$  at each step (Minimum Residual or MR method) is a symmetric equivalent of the GMRES (Generalized Minimum Residual) algorithm for nonsymmetric linear systems. In particular, one may derive the well-known Conjugate Residual method from MR algorithm by observing and imposing the  $A$ -conjugacy condition of the two subsequent residuals. Again, as with CG, the algorithm may suffer from a breakdown when  $A$  is indefinite.

Somewhat ironically, in the CG and the CR algorithms the minimized inner product norms come naturally as the byproducts of the algorithms, but unfortunately the other

---

<sup>1</sup> Any two algorithms will be called mathematically equivalent, or simply equivalent, if both provide an identical solution  $x_m$  for  $1 \leq m \leq n$  (in exact arithmetic)

way round. Particularly in the case of the CR algorithm this can be a nuisance since  $\|r_m\|_2$  has to be computed directly in order to monitor the convergence. This paper deals with a case when  $A$  is a symmetric and indefinite (ideally sparse) matrix, hence in the algorithms considered here  $H_m = T_m$  in (2) will be symmetric, tridiagonal and in general indefinite matrix ( $T_m$ ) as a result of the Lanczos algorithm. The problem of deriving the breakdown-free D-Lanczos algorithms (or simply D-algorithms) is the case a stable factorization of  $T_m$ ; this rules out, for example, the LU decomposition without pivoting (see [5] p.177), but leaves, for example, the LQ factorization as an option. Indeed, the latter approach gives rise to the Paige and Saunders algorithm known as SYMMLQ [4] in the case when  $u_m = \beta e_1$  in (3). The D-algorithms (not presented here due to space limits) are derived from the rank-one and rank-two tearing of  $T_m$ , yielding a recursive formula(e) for its inverse  $T_m^{-1}$ . The two resulting CG-type algorithms for indefinite case called here Indefinite Conjugate Gradient (ICG) and MR respectively, are in turn derived from these D-algorithms by observing and imposing the new  $A$ -conjugacy conditions applicable to the cases when  $T_m$  is singular. The two algorithms may be implemented as a single method where the MR solution comes as an option, in addition to the CG 'equivalent' solution. In the latter case, unlike in the CR algorithm, the residual norms that give an indication of the convergence are computed effortlessly.

## 2 The Inverses of Tridiagonal Matrices

### 2.1 The Sherman-Morrison-Woodbury Approach

The results in this section can be derived using the Sherman-Morrison-Woodbury formula for the inverse of the rank-one and rank-two update of a tridiagonal matrix, see for example [2] p.51. These results, presented here in the form of two technical lemmas, combined with the results recalled from [6] should hopefully allow the decomposition of  $y_m$  into  $y_{m-1}$  and some easily available  $m$ -th step update. This in turn would allow to express the current solution  $x_m$  as an update of the previous solution  $x_{m-1}$ . First, assume here that matrix  $T_m$  takes the form

$$T_m = \begin{bmatrix} \alpha_1 & \beta_1 & & \dots & & \\ & \beta_1 & \alpha_2 & \beta_2 & & \\ & & \vdots & \vdots & & \\ & & & \beta_{m-2} & \alpha_{m-1} & \beta_{m-1} \\ & & & & \beta_{m-1} & \alpha_m \end{bmatrix} \tag{4}$$

Also, let

$$\theta_i = \alpha_i - \beta_{i-1}^2 e_{i-1}^T T_{i-1}^{-1} e_{i-1} \tag{5}$$

Assume throughout that  $\beta_m \neq 0$  (unless stated otherwise), so that no 'lucky' breakdown takes place.

**Lemma 1** Assume that  $T_{m-1}$  is nonsingular and that  $\theta_m \neq 0$ . Then, the inverse of  $T_m$  takes the following form:

$$\begin{aligned}
 T_m^{-1} = & \begin{bmatrix} T_{m-1}^{-1} & 0 \\ 0^T & \theta_m^{-1} \end{bmatrix} - \frac{\beta_{m-1}}{\theta_m} e_m \begin{bmatrix} T_{m-1}^{-1} e_{m-1} \\ 0 \end{bmatrix}^T + \\
 & + \frac{\beta_{m-1}^2}{\theta_m} \begin{bmatrix} T_{m-1}^{-1} e_{m-1} \\ 0 \end{bmatrix} \begin{bmatrix} T_{m-1}^{-1} e_{m-1} \\ 0 \end{bmatrix}^T - \\
 & - \frac{\beta_{m-1}}{\theta_m} \begin{bmatrix} T_{m-1}^{-1} e_{m-1} \\ 0 \end{bmatrix} e_m^T
 \end{aligned} \tag{6}$$

Moreover,

$$\theta_m = \frac{1}{e_m^T T_{m-1}^{-1} e_m} = \frac{|T_m|}{|T_{m-1}|},$$

and consequently  $\theta_m = 0$  implies the singularity of  $T_m$  (so when  $T_m$  is nonsingular  $\theta_m = \frac{1}{e_m^T T_{m-1}^{-1} e_m}$ ).

The second lemma deals with a slightly more general case, that will be necessary when matrix  $T_{m-1}$  is singular.

**Lemma 2** Assume that  $T_{m-2}$  is nonsingular and that

$$\psi_m = \alpha_m \theta_{m-1} - \beta_{m-1}^2 \neq 0.$$

Then, the inverse of  $T_m$  takes the following form:

$$\begin{aligned}
 T_m^{-1} = & \begin{bmatrix} T_{m-2}^{-1} & & 0 \\ 0 & \frac{1}{\psi_m} \begin{pmatrix} \alpha_m & -\beta_{m-1} \\ -\beta_{m-1} & \theta_{m-1} \end{pmatrix} \end{bmatrix} - \\
 & - \frac{\beta_{m-2}}{\psi_m} \begin{bmatrix} 0 \\ \alpha_m \\ -\beta_{m-1} \end{bmatrix} \begin{bmatrix} T_{m-2}^{-1} e_{m-2} \\ 0 \\ 0 \end{bmatrix}^T + \\
 & + \frac{\alpha_m \beta_{m-2}^2}{\psi_m} \begin{bmatrix} T_{m-2}^{-1} e_{m-2} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} T_{m-2}^{-1} e_{m-2} \\ 0 \\ 0 \end{bmatrix}^T - \\
 & - \frac{\beta_{m-2}}{\psi_m} \begin{bmatrix} T_{m-2}^{-1} e_{m-2} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ \alpha_m \\ -\beta_{m-1} \end{bmatrix}^T
 \end{aligned} \tag{7}$$

Moreover,

$$\psi_m = \frac{|T_m|}{|T_{m-2}|},$$

(and  $\psi_m = \theta_m \theta_{m-1}$  when  $T_{m-1}$  is nonsingular), implying that  $T_m$  is singular when  $\psi_m = 0$ .

### 2.2 The Last Column of the Inverse of the Hessenberg Matrix

In theory,  $T_m^{-1}$  given by equations (7) or (10) can be now substituted in equation (3) in hope of obtaining  $y_m$  as a simple update of  $y_{m-1}$ . However, in such a case one has to form  $T_m^{-1}$  explicitly in order to extract its last column. Fortunately, this will not be necessary since, as shown in [6], one may use a simple recursive formula for  $T_m^{-1}e_m$  alone. One difficulty arising here is a possibility of an intermediate matrix, say  $T_{m-1}$ , being singular. It turns out that this problem can be resolved in a robust, yet simple way. Introducing the vector  $s_m = [\sigma_1, \dots, \sigma_m]^T \in \mathbb{R}^m$  with  $\sigma_1 = 1$  such that:

$$T_m s_m = \xi_m e_m \tag{8}$$

where, if  $T_m$  is nonsingular,

$$\frac{1}{\xi_m} = e_1^T T_m^{-1} e_m, \tag{9}$$

and  $\xi_m = 0$  otherwise. Thus the vector  $s_m$  always exists while the ‘corresponding’  $T_m^{-1}e_m$  may not. Consequently, if  $T_m$  is nonsingular then

$$T_m^{-1} e_m = \frac{1}{\xi_m} s_m \tag{10}$$

The quantities introduced above can be now generated by the following algorithm:

**Algorithm 1.** Set  $\sigma_1 = 1$  and  $s_1 = [\sigma_1]$ .

Then, in the  $m$ -th step of the Lanczos factorization compute:

$$\xi_m = s_m^T (T_m e_m) \tag{11}$$

followed by

$$\sigma_{m+1} = -\frac{\xi_m}{\beta_m}, \tag{12}$$

and then update  $s_m$  as

$$s_{m+1} = [s_m^T, \sigma_{m+1}]^T$$

. Since  $T_m$  is symmetric tridiagonal, the computation in (14) reduces to

$$\xi_m = \sigma_{m-1} \beta_{m-1} + \sigma_m \alpha_m \tag{13}$$

Observe that  $\xi_m = \sigma_{m+1} = 0$  implies the singularity of  $T_m$ .

### 2.3 The Inverses of Tridiagonal Matrices

Ultimately, it is now necessary to ‘translate’ equations (7) and (10) into the ‘language’ of §2.2, under the assumptions of Lemma 1 and Lemma 2 respectively, or namely, to substitute  $T_i^{-1}e_i$  in those equations by the easily available quantities  $s_i$  and  $\xi_i$  introduced in §2.1.

Thus, it follows from equations (13) and (14) that equation (7) will now take the form

$$T_m^{-1} = \begin{bmatrix} T_{m-1}^{-1} & 0 \\ 0^T & \theta_m^{-1} \end{bmatrix} + \frac{1}{\xi_m} e_m \begin{bmatrix} s_{m-1} \\ 0 \end{bmatrix}^T + \omega_m \begin{bmatrix} s_{m-1} \\ 0 \end{bmatrix} \begin{bmatrix} s_{m-1} \\ 0 \end{bmatrix}^T + \frac{1}{\xi_m} \begin{bmatrix} s_{m-1} \\ 0 \end{bmatrix} e_m^T \quad (14)$$

where

$$\omega_m = \frac{1}{\xi_m \sigma_m} \quad (15)$$

Ultimately, using equations (13), (14) again, equation (10) now takes the form

$$T_m^{-1} = \begin{bmatrix} T_{m-2}^{-1} & 0 \\ 0 & \frac{1}{\psi_m} \begin{pmatrix} \alpha_m & -\beta_{m-1} \\ -\beta_{m-1} & \theta_{m-1} \end{pmatrix} \end{bmatrix} - \frac{1}{\beta_{m-1} \xi_m} \begin{bmatrix} 0 \\ \alpha_m \\ -\beta_{m-1} \end{bmatrix} \begin{bmatrix} s_{m-2} \\ 0 \\ 0 \end{bmatrix}^T - \bar{\omega}_m \begin{bmatrix} s_{m-2} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} s_{m-2} \\ 0 \\ 0 \end{bmatrix}^T - \frac{1}{\beta_{m-1} \xi_m} \begin{bmatrix} s_{m-2} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ \alpha_m \\ -\beta_{m-1} \end{bmatrix}^T \quad (16)$$

where

$$\bar{\omega}_m = \frac{\alpha_m}{\beta_{m-1} \sigma_{m-1} \xi_m} \quad (17)$$

Observe that no assumption of the singularity of  $T_{m-1}$  ( $\theta_{m-1} = 0$ ) was used here, despite the fact that equation (21) has been derived specifically for such an occasion. It is shown in [6] that the singularity of  $T_{m-2}$  and  $T_{m-1}$  implies the singularity of  $A$ , so consequently (under the sweeping assumption of  $\beta_m \neq 0$ ) at least one of the two matrices considered here must be invertible.

### 3 The Combined Indefinite Conjugate Gradient and Minimum Residual Algorithm

Relations (17) and (20) will allow the formulation of the D-Lanczos methods for indefinite linear systems. From here, using the appropriate conjugacy conditions, it is now possible to derive the equivalent Generalized Conjugate Gradient (ICG) and Minimum Residual algorithms embodied in a single routine shown below.

All operations marked with '(o)' in the algorithm above are only executed if the Minimum Residual solution  $\overset{\circ}{x}_m$  is required (lines 1, 2, 7, 11 and 18). Lines 5,...,9 constitute the body of the CG algorithm, whereas the remaining lines deal with its breakdown case. Also again, the stopping criterion for both algorithms (line 2) is based on their residual norms  $\|r_m\|_2, \|\overset{\circ}{r}_m\|_2$  respectively.

Observe that the 'breakdown' step in Algorithm 5 requires two matrix-vector products ( $A\mathbf{p}_j$  in line 5 and  $A\bar{r}_{j+1}$  in line 12), but in compensation no such product is needed in the follow-up step. Also, notice that like in the case of the CR algorithm the



MR solution in Algorithm 5 requires one more vector update per step then the GCG solution.

**Algorithm 2.** GCG/MR

1. compute  $r_0 = b - Ax_0, \mathbf{p}_0 = r_0$       (o) (set  $\overset{\circ}{x}_0 = 0$ )
2. do  $j = 0, 1, 2, \dots$  until  $\|r_j\|_2$       (o) ( $\|r_j\|_2 = \left(\sum_{i=0}^j \|r_i\|_2^{-2}\right)^{-\frac{1}{2}}$ ) is small enough
3. if ( $j > 0$  and  $\mathbf{p}_{j-1}^T A \mathbf{p}_{j-1} \neq 0$ ) then
4.    if  $\mathbf{p}_j^T A \mathbf{p}_j \neq 0$  then
5.     compute  $\lambda_j = \frac{\|r_j\|_2^2}{\mathbf{p}_j^T A \mathbf{p}_j}$
6.     compute  $x_{j+1} = x_j + \lambda_j \mathbf{p}_j$
7.     compute  $r_{j+1} = r_j - \lambda_j A \mathbf{p}_j$       (o) ( $\overset{\circ}{x}_{j+1} = \frac{\sum_{i=0}^j \|r_i\|_2^{-2}}{\sum_{i=0}^{j+1} \|r_i\|_2^{-2}} (\overset{\circ}{x}_j - \lambda_j \mathbf{p}_j)$ )
8.     compute  $\mu_j = \frac{\|r_{j+1}\|_2^2}{\|r_j\|_2^2}$
9.     compute  $\mathbf{p}_{j+1} = r_{j+1} + \mu_j \mathbf{p}_j$
10.    else
11.     set  $\bar{r}_{j+1} = A \mathbf{p}_j$       (o) ( $\|r_{j+1}\|_2^{-2} = 0$ )
12.     compute  $\bar{\mu}_j = -\frac{\bar{r}_{j+1}^T A \bar{r}_{j+1}}{\|\bar{r}_{j+1}\|_2^2}$
13.     compute  $\bar{\mathbf{p}}_{j+1} = \bar{r}_{j+1} + \bar{\mu}_j \mathbf{p}_j$
14.     end if
15.    else
16.     compute  $\bar{\lambda}_j = \frac{\|r_{j-1}\|_2^2}{\|\bar{r}_j\|_2^2}$
17.     compute  $x_{j+1} = x_{j-1} + \bar{\lambda}_j \bar{\mathbf{p}}_j$
18.     compute  $r_{j+1} = r_{j-1} - \bar{\lambda}_j (A \bar{r}_j + \bar{\mu}_{j-1} \bar{r}_j)$       (o) ( $\overset{\circ}{x}_{j+1} = \frac{\sum_{i=0}^{j-1} \|r_i\|_2^{-2}}{\sum_{i=0}^{j+1} \|r_i\|_2^{-2}} (\overset{\circ}{x}_{j-1} - \bar{\lambda}_j \bar{\mathbf{p}}_j)$ )
19.     compute  $\bar{\mu}_j = \frac{\|r_{j+1}\|_2^2}{\|r_{j-1}\|_2^2}$
20.     compute  $\mathbf{p}_{j+1} = r_{j+1} + \bar{\mu}_j \mathbf{p}_{j-1}$
21.     end if
22. end do
23. (o) (compute  $x_m = x_m + \overset{\circ}{x}_m$ )

**References**

1. Ashby S.F., Manteuffel T.A., and Saylor P.E., 'A Taxonomy for Conjugate Gradient Methods', *SIAM J. Numer. Anal.*, **27**, No 6, 1990, pp. 1542-1568.
2. Golub G. and van Loan C.F., *Matrix Computations*, John Hopkins University Press, London, 1989.
3. Hestenes M.R., and Stiefel E., 'Methods of Conjugate Gradients for Solving Linear Systems', *J. Res. Nat. Bur. Standards*, **49**, (1952), 409-435.
4. Paige C.C., and Saunders M.A. 'Solution of Sparse Indefinite Systems of Linear Equations', *SIAM Journal on Numerical Analysis*, **12**, 1975: 617-624.
5. Saad Y., *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.
6. Szularz M., 'The Least-Squares Problem in the GMRES algorithm', submitted to *Numerical Linear Algebra with Applications*, 2012.

# Parallelization of Dependent Iterations in Scientific Computing by the Parareal-in-Time Algorithm

Toshiya Takami<sup>1,2</sup>, Keiichiro Fukazawa<sup>1</sup>, Hiroaki Honda<sup>1</sup>, Yuichi Inadomi<sup>1</sup>  
Ryutaro Susukita<sup>2</sup>, Taizo Kobayashi<sup>1</sup>, and Takeshi Nanri<sup>1,2</sup>

<sup>1</sup>Research Institute for Information Technology, Kyushu University,  
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan

<sup>2</sup>Institute of Systems, Information Technologies and Nanotechnologies (ISIT),  
System LSI Center bld. 3F, 3-8-33 Momochihama, Sawara-ku, Fukuoka 814-0001, Japan

**Abstract**—A new field of application by a prominent time-parallel technique, the ‘parareal-in-time’ algorithm, is explored with the perturbative description. The parareal algorithm introduced by Lions, et al. [J. Lions, Y. Maday, and G. Turinici, C. R. Acad. Sci., Ser. I 332, 661–668 (2001)] has been applied to parallelization of time evolutions and is usually implemented by coarsening the width of time-steps. In this contribution, another description based on the concept of perturbation is introduced for this algorithm. It turns out that the new description is more general and suitable for scientific applications, where all kinds of dependent iterations are within the range of applicable calculations. Several examples of the parallel implementations are shown to be feasible for the forthcoming post-peta era.

**Keywords:** parareal-in-time algorithm, perturbative description, post-peta computing, speed-up ratio, parallel efficiency

## 1. Introduction

As it is clearly shown in the Top500 list [1], massively parallel computers have been constantly developed for decades, and the top performance of supercomputers is expected to grow further toward post-peta or exa scales. The significant problem for scientists who write parallel applications on those computers is that an infinitely large number of parallel nodes and cores are available, and that strict requirements for an effective use of those resources may narrow down applicable areas in computational sciences.

High performance computing techniques to configure efficient parallel applications have been widely developed and applied to scientific simulations. However, the parallel speed-up is limited by the number of independent calculations that can be executed concurrently. For the moderate sizes of applications, efficient parallel executions are not expected on massively parallel computers. Thus, a new strategy is desired to parallelize even dependent calculations.

When the number of independent components is limited in spatial dimensions, the time axis usually remains not parallelized because of the causal dependency. One of the strategies for time-domain decomposition is the ‘parareal-in-time’ algorithm introduced by Lions, et al. [2], which

is considered as space-time multigrid or multiple shooting methods [3], and it is also known that higher efficiency is guaranteed when it is used with a coarser discretization in space. Many works have been published on convergence properties and applicability of this algorithm for time integrations in various scientific fields [4], [5], [6], [7], [8], [9], [10]. The most important step is to configure a coarse solver that approximates the exact time evolution. While the solver is easily defined by coarsening the width of a time-step, other definitions have also been studied in several literature.

It is already known that the parareal algorithm is not limited to time-evolving simulations, and simple matrix-vector multiplications have been parallelized [11] by introducing approximated operators. The main purpose of this contribution is to extend applicable ranges of the algorithm, and to investigate new possibility to avoid limitations by spatial parallelization. In order to achieve the purpose, the perturbative description of the problem is introduced, by which we can naturally apply the algorithm to all kinds of iterative methods with dependency. In addition to the theoretical formulation in scientific computing, actual implementation of programs is also a matter of concern. In this paper, we present a template class and interface classes to describe dependent sequences to assist the parallel programming of the parareal acceleration.

This contribution is organized as follows. Section 2 describes the usual introduction of the ‘parareal-in-time’ algorithm and the speed-up properties. In Section 3, the perturbative description of the algorithm is introduced. Several applicable problems for dependent sequences are implemented in Section 4. The actual implementation of the parareal accelerated codes is shown in Section 5, and summary and discussions are given in Section 6.

## 2. Parareal Algorithm and Efficiency

After a certain discretization process, we obtain time evolutions of scientific problems defined by

$$x(t + \delta t) = f_{\delta t}(x(t)), \quad (1)$$

where  $x(t)$  represents a state vector at a time  $t$ , and  $f_{\delta t}(\cdot)$  is a time integrator with an interval  $\delta t$ . Because of the

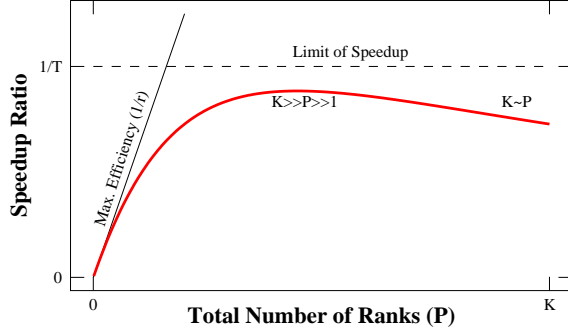


Figure 1: Speed-up ratio by the parareal algorithm.

strict dependency to the previous state  $x(t)$ , usual domain-decomposition techniques cannot be applied to the time direction.

The ‘parareal-in-time’ algorithm, however, is introduced when a coarse solver  $G(\cdot)$  and the fine solver  $F(\cdot)$  are defined as approximations of the original operator  $f(\cdot)$ . If we write  $x(n\Delta t) = x_n$  and introduce  $x_n^{(r)}$  as  $r$ -th approximation of the time evolution, the parareal iteration is defined by the formula

$$x_{n+1}^{(r+1)} = G(x_n^{(r+1)}) + F(x_n^{(r)}) - G(x_n^{(r)}), \quad (2)$$

which will be converged to the exact evolution  $x_n$  after the sufficient number of iterations in  $r$ .

For example, the coarse solver  $G$  and the fine solver  $F$  are often defined by the same operator  $f(\cdot)$  with a coarse and fine time-steps  $\Delta t = T\delta t$ ,

$$G(x(t)) = f_{\Delta t}(x(t)), \quad F(x(t)) = [f_{\delta t}]^T(x(t)). \quad (3)$$

These definitions of  $G$  and  $F$  can be applied generally to time-evolutions in every scientific simulation, and it is also useful in determining computational costs or analyzing convergence property of  $G$  and  $F$ .

For simplicity, suppose that the computational time of the evolving function  $f(\cdot)$  is independent from the state of the system and represented by  $t_f$ . If we introduce  $T_f$  and  $T_g$  to represent computational times for  $F$  and  $G$ , respectively,  $T_f$  is  $T$  times larger than  $T_g$ , i.e.,  $T = T_f/T_g$ . If we define  $T_c$  as a communication time to send a vector to the next resource and  $\tau$  as a rate of  $T_c$  compared to  $T_f$ , i.e.,  $\tau = T_c/T_f$ , the optimal speed-up ratio  $S$  by this scheme is represented by

$$S = \frac{P}{R + TP + \frac{P(P-1)}{K}\tau}, \quad (4)$$

where  $P$  is available parallel resources and  $K$  is the number of time-steps [11]. The schematic properties of this speed-up ratio is shown in Fig. 1.

### 3. Parareal as Perturbation

In physics, we often describe complex interactions as perturbation to simple unperturbed representations. In the usual situation, the perturbation is introduced to correct simple analytic results by the unperturbed description. For numerical analysis, we can introduce simple and easily calculated representations as the unperturbed description, and the perturbation will be considered as complex and computationally expensive interactions.

Perturbative description can be introduced to all kinds of iterative calculations to obtain sequence of vectors  $\{x_n\}$ , where each state  $x_{n+1}$  is dependent on the previous state  $x_n$  through a relation

$$x_{n+1} = f(x_n). \quad (5)$$

An unperturbed evolution operator  $f^{(0)}$  defines the unperturbed sequence  $\{x_n^{(0)}\}$  ( $x_0^{(0)} = x_0$ ) by,

$$x_{n+1}^{(0)} = f^{(0)}(x_n^{(0)}). \quad (6)$$

If we introduce the first order perturbation  $\varepsilon\tilde{f}$ , the perturbed sequence  $\{\tilde{x}_n\}$  ( $\tilde{x}_0 = x_0$ ) is given by

$$\tilde{x}_{n+1} = [f^{(0)} + \varepsilon\tilde{f}](\tilde{x}_n), \quad (7)$$

where  $\varepsilon$  is assumed small.

In the usual perturbative representations, the magnitude of correction by the perturbation  $\varepsilon\tilde{f}$  is small, but the computational cost is relatively higher than that for the unperturbed part. If we assign a coarse and fine operators by

$$G(x_n) = f^{(0)}(x_n), \quad F(x_n) = f^{(0)}(x_n) + \varepsilon\tilde{f}(x_n), \quad (8)$$

the  $r$ -th approximation  $\{x_n^{(r)}\}$  of the accurate sequence  $\{\tilde{x}_n\}$  is calculated by the parareal iteration,

$$x_{n+1}^{(r+1)} = f^{(0)}(x_n^{(r+1)}) + \varepsilon\tilde{f}(x_n^{(r)}). \quad (9)$$

This is the parareal procedure to obtain  $r$ -th approximation of the perturbed sequence (7). If we have sufficiently large parallel resources, those terms with different  $n$  and the same  $r$ ,  $\varepsilon\tilde{f}(x_n^{(r)})$ , are calculated in parallel.

The perturbation representation (9) is introduced independently from coarsening the width of time-steps. In other words, the applicable area of the parareal algorithm (2) is not limited to time evolutions, but can be extended to all kinds of iterative calculations. In the next section, we give several examples from scientific calculations.

### 4. Design of Dependent Calculations

In scientific applications, we often encounter iterative calculations, in which a set of calculations are repeated to obtain a sequence of vectors with a certain length or until convergence. The problem is that such calculations should be executed sequentially since each calculation is closely dependent on the previous result. The time evolution is an

example of such dependent calculations, but there are various types of iterative calculations and the applicable area is not limited to the time evolution.

There are various dependent iterations in physics and chemistry, i.e., self-consistent field calculations in quantum chemistry [12], Poisson solvers used in fluid mechanics calculations [13], various iterative methods implemented in libraries for linear systems [14]. In usual implementations, these calculations have not been parallelized in the direction of iterations because of the dependency. In this section, we show that some of those sequential calculations can be parallelized by the parareal-in-time algorithm even if each calculation depends on the previous result.

### 4.1 Molecular Dynamics

Baffico, et al. [4], applied the parareal algorithm to *ab initio* molecular dynamics calculations, where quantum propagators are parallelized. Their approach can be extended to every types of molecular dynamics simulations. We can naturally introduce an unperturbed trajectory defined by cheap and less accurate integrators as well as a perturbed trajectory defined by expensive and accurate integrators.

Among many combinations of approximated and accurate descriptions, two examples are shown here. In the first one, integrators are defined by changing an accuracy parameter. If we assign an unperturbed integrator configured by a smaller cut-off length for short-range interactions, a perturbation is defined as corrections by an accurate integrator configured by a longer cut-off length. When we do not adopt description from first principles, it is easy to introduce various accuracy parameters in molecular dynamics calculations.

In the next example, we assign integrators with respect to the long and short range interactions. The unperturbed integrator is configured by the short-range interactions only, while the perturbation is introduced as corrections by the long-range interactions. Since long-range parts of Coulomb potentials still contain complex and expensive calculations even in the approximation of Particle Mesh Ewald, Fast Multipole Method, etc., this implementation is naturally introduced in package software of the molecular dynamics calculations.

### 4.2 Symplectic Integrator

In order to discretize the time integration of Hamiltonian systems including molecular dynamics, celestial mechanics, quantum mechanics, etc., symplectic integrators [15] are widely used in energy-conserved systems. In many applications, the second order symplectic integrator,

$$\hat{S}_2(\delta t) \equiv \exp \left( \frac{\hat{D}_U \delta t}{2} \right) \exp \hat{D}_T \delta t \exp \left( \frac{\hat{D}_U \delta t}{2} \right), \quad (10)$$

is sufficient stability and performance.  $\hat{D}_T$  and  $\hat{D}_U$  represent evolving operators with respect to momentum and coordinates, respectively, and the Hamiltonian  $H$  is assumed to

have a separated form  $H(\{q_j\}, \{p_j\}) = T(\{p_j\}) + U(\{q_j\})$ . On the other hand, applications that require higher accuracy for energy preservation are often implemented by the 8-th symplectic integrator. By the use of Yoshida's coefficient  $\{c_j\}$  [15], we can construct 8-th integrator by 15 iterations of the second order integrator (10),

$$\hat{S}_8(\delta t) \equiv \prod_{j=1}^{15} \hat{S}_2(c_j \delta t). \quad (11)$$

Thus, we can assign the unperturbed iterations as the second order integrator, and the perturbation as the correction by the 8-th integrator,

$$G = \hat{S}_2(\delta t), \quad F = \hat{S}_8(\delta t). \quad (12)$$

The computational cost for  $G$  is 15 times smaller than that for  $F$ , i.e.,  $T \approx 15$  in eq. (4). In the actual computations, convergent property may be improved by Interpolated Predictor Corrector scheme introduced by Bal, et al. [16].

### 4.3 Quantum Mechanics

For the optimal control problem in quantum mechanics, the parareal algorithm was already applied and analyzed [5], [7]. Since the optimally controlled dynamics by weak external fields can be studied by almost analytic approach [17], we can define unperturbed evolutions by isolated dynamics without external fields, and transitions by the external field are considered as perturbation.

Here, we present more general expressions for quantum mechanics, i.e., parareal accelerated infinitesimal unitary transformations based on the approximation of the unitary operator. A unitary transformation is written by

$$\psi_\varepsilon = \exp i\varepsilon \hat{H} \psi. \quad (13)$$

with the Hermite operator  $\hat{H}$ . When the parameter  $\varepsilon$  is small enough, we can assign an identity operator  $\hat{I}$  for an unperturbed system since we can write

$$\exp i\varepsilon \hat{H} = \hat{I} + \left[ \exp i\varepsilon \hat{H} - \hat{I} \right], \quad (14)$$

and the second term in the right-hand-side is assumed as a perturbation.

When we calculate a sequence of wavefunctions defined by repeated operations of eq. (13), the parareal acceleration is applied with the coarse and fine solvers by

$$G = \hat{I}, \quad F = \exp i\varepsilon \hat{H} - \hat{I}. \quad (15)$$

Note that computational costs of  $G$  is negligible compared to that of  $F$ , and that the spectral radius of  $F$  is linear to  $\varepsilon$ . When  $\hat{H}$  is a Hamiltonian and  $\varepsilon$  is a step size for time evolutions, the transformation (13) is nothing but a time-evolution by Schrödinger's equation. Actual calculations for  $F$  is implemented by a symplectic integrator.

The convergence and speed-up by this representation have already been analyzed as matrix multiplications [11], and it is realized that the series of unitary operations are efficiently parallelized by the parareal algorithm even when the spatial parallelism does not work effectively.

### 4.4 Iterative Linear Methods

Iterative linear calculations are generally used in scientific applications. In the usual implementation of those iterative methods, only spatial parallelism is adopted, i.e., calculations for independent elements in a matrix or a vector are executed in parallel. In massively parallel machines, however, the number of independent elements can be smaller than the number of parallel resources. Then, another axis for parallelism is required if we achieve effective parallelization in those machines.

The parareal implementation for an iterative linear sequence is quite simple. If we introduce an approximate linear operator  $G$  to the accurate transformation  $F$  which defines the exact sequence of vectors  $\{x_n\}$  by

$$x_{n+1} = Fx_n, \tag{16}$$

the parareal iteration for  $r$ -th approximated sequence  $\{x_n^{(r)}\}$  is represented by

$$x_{n+1}^{(r)} = Gx_n^{(r+1)} + (F - G)x_n^{(r)}. \tag{17}$$

Since the approximate and accurate transformations  $G$  and  $F$  are linear by definition, convergence properties of the sequence are analyzed by spectral radii of  $G$  and  $F$  [11].

As an example, we present parallel implementation of iterative solvers for a linear equation  $Ax = b$ . If the symmetric matrix  $A$  is diagonally dominant or positive definite, iterative procedure by the successive over-relaxation (SOR),

$$x_{n+1} = (1 - w)x_n + w(D + U)^{-1}[b - Lx_n], \tag{18}$$

converges to the exact solution  $x$ , where  $w$  is a constant chosen as  $0 < w < 2$ , and  $D, U$ , and  $L$  are diagonal, upper-triangle, and lower-triangle parts of  $A$ , respectively.

In order to parallelize the iteration in  $n$ , we analyze the following definitions of the approximate transformation  $G$ , **Model 1:**  $Gx \equiv x$  ( $G$  is an identity operation.)

$$x_{k+1}^{(r+1)} = x_k^{(r+1)} + w \left[ (D + \varepsilon U)^{-1}(b - \varepsilon Lx_k^{(r)}) - x_k^{(r)} \right] \tag{19}$$

**Model 2:**  $Gx \equiv (1 - w)x + wD^{-1}b$  ( $G$  as a diagonal approximation,  $A \approx D$ .)

$$x_{k+1}^{(r+1)} = (1 - w)x_k^{(r+1)} + wD^{-1}b + w \left[ (D + \varepsilon U)^{-1}(b - \varepsilon Lx_k^{(r)}) - D^{-1}b \right] \tag{20}$$

In both cases,  $F$  is defined as the original SOR iteration (18), and  $F - G$  is considered as a perturbation to the unperturbed transformation by  $G$ .

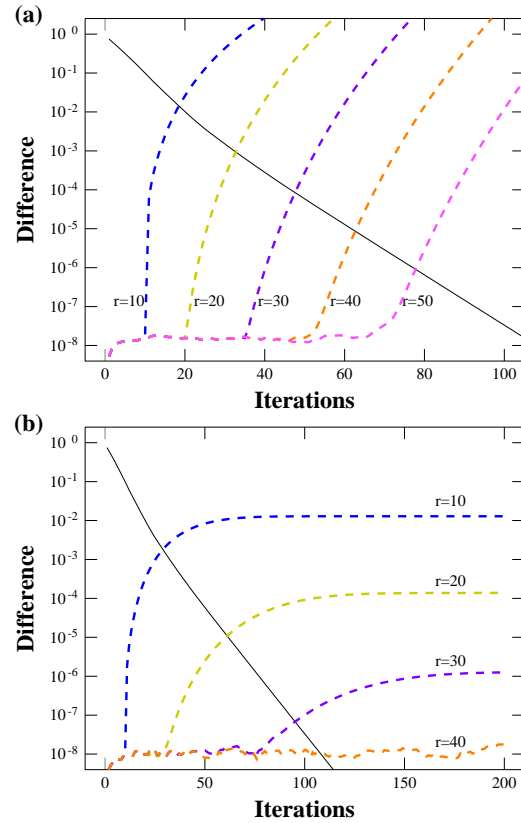


Figure 2: The solid curve represents residue  $|x_k - x_{k-1}|$  of the original SOR method, and dashed curves show the remaining errors of  $r$ -th parareal sequences: (a) Model 1 and (b) Model 2.

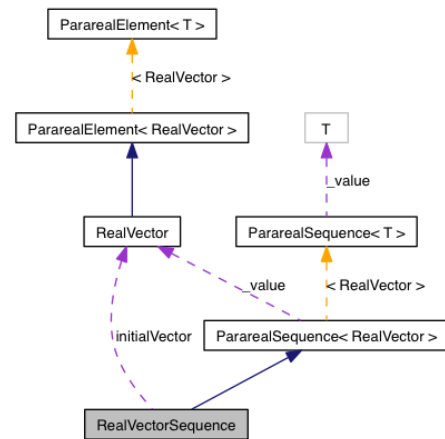


Figure 3: Related classes of Parareal<T>.

```

#include "PararealSequence.h"
#include "RealVector.h"
class RealVectorSequence
: public PararealSequence<RealVector> {
private:
    RealVector initialVector;
public:
    RealVectorSequence(int length, const RealVector& initValue)
: PararealSequence<RealVector>(length, initValue) {
    initialVector = initValue;
}
~RealVectorSequence(void) {}
void coarseEvolution(const RealVector& a, RealVector& b) {
/* actual calculations for the coarse evolution: b=G(a) */
}
void fineEvolution(const RealVector& a, RealVector& b) {
/* actual calculations for the fine evolution: b=F(a) */
}
};

```

Figure 4: RealVectorSequence is a derived class to implement the abstract class PararealSequence<T>.

In Figure 2, convergence properties of the parareal accelerated SOR are shown, where the matrix  $A$  and the right-hand-side vector  $b$  are defined by random elements. While the result by the model 1 in Fig. 2(a) shows slow convergence, the model 2 in Fig. 2(b) is feasible. Further works should be done to clarify convergence properties of the parareal accelerated SOR method.

## 5. Parallel Implementation

Unlike the usual parallel programming in scientific computing, the parallel implementation of the parareal-in-time algorithm cannot be complete within the level of numerical libraries. Instead, we must configure a new layer for the sequence calculation, which requires a certain skill of programming. In this section, we present a set of template classes to assist implementation of the parareal algorithm.

Our template classes for implementation of the parareal algorithm consist of the main class, Parareal<T>, and two interface classes, PararealSequence<T> for the sequence  $\{x_k^{(r)}\}$  and PararealElement<T> for the state  $x_k^{(r)}$ , where T is a user-provided class to represent  $x_k^{(r)}$ . The interface PararealSequence<T> requires implementation of two methods coarseEvolution(const T& x0, T& x1) and fineEvolution(const T& x0, T& x1) in user-defined derived classes. These methods describe the approximate iteration,

$$x_{k+1} = \mathcal{G}(x_k), \quad (21)$$

and the exact iteration,

$$x_{k+1} = \mathcal{F}(x_k). \quad (22)$$

Another interface class PararealElement<T> requires implementation of basic operations, i.e., substitution '=', add-and-store '+=', and subtract-and-store '-=', to the state vector  $x_k$ .

In Fig. 3, we show class relations when we implement these interface classes to calculate a vector sequence. A sample implementation of the abstract class

```

#include "PararealElement.h"
class RealVector
: public PararealElement<RealVector> {
public:
    RealVector& operator=(const RealVector& x) {
/* actual instruction for copy */
}
    RealVector& operator+=(const RealVector& x) {
/* actual instruction for addition and store */
}
    RealVector& operator-=(const RealVector& x) {
/* actual instruction for subtraction and store */
}
/* other methods */
...
};

```

Figure 5: RealVector is a derived class of the abstract class PararealElement<T>.

```

#include "Parareal.h"
#include "RealVectorSequence.h"

int main(int argc, char **argv) {
...

    RealVector initVec = RealVector(Nelem);
    RealVectorSequence RVSequence(Nstep, initVec);

    Parareal<RealVector> PRV(Nstep);
    RealVector *exactResult
= PRV.exactSequence(RVSequence);
    RealVector *paraResult
= PRV.pararealSequence(RVSequence, Niter);

...
}

```

Figure 6: The main program to calculate a vector sequence.

PararealSequence is shown in Fig. 4. In this sample, T is a class for a real vector, RealVector, which is also an implementation of the abstract class PararealElement. By the use of these classes, the main program to calculate a vector sequence becomes very simple shown in Fig. 6.

The main loop for the parareal calculation is shown in Fig. 7, where the non-parallel version of the method T\* pararealSequence() is shown here for simplicity. By the use of MPI\_Send and MPI\_Recv or collective communications with partial communicators, these loops are parallelized. The final communication pattern of this calculation becomes 'Pipeline-type' configuration.

In the parallel implementation, communication time in the dependent loop dominates the parallel efficiency. The use of non-blocking collective communications or pipeline communication libraries may give efficient parallel implementations of the Parareal-in-Time calculations. Further investigation will be done in this direction.

## 6. Summary and Discussions

In this contribution, we presented a new description of the parareal-in-time algorithm based on the perturbation of the sequence. Since the perturbation is one of the common concept in scientific description, most problems in science

```

T* pararealSequence(PararealSequence<T>& seq, int Niter) {
  coarse = new T[Nstep + 1];
  modify = new T[Nstep + 1];
  for (int k = 1; k <= Nstep - Niter + 1; k++) {
    seq.coarseEvolution(seq[k-1], coarse[k]);
    seq[k] = coarse[k];
  }
  for (int r = 1; r <= Niter; r++) {
    for (int k = r; k <= Nstep - Niter + r; k++) {
      seq.fineEvolution(seq[k-1], modify[k]);
      modify[k] -= coarse[k];
    }
    for (int k = r; k <= Nstep - Niter + r + 1; k++) {
      if (k > Nstep) break;
      seq.coarseEvolution(seq[k-1], coarse[k]);
      seq[k] = coarse[k];
      if (k <= Nstep - Niter + r)
        seq[k] += modify[k];
    }
  }
  return &seq[0];
}

```

Figure 7: The main loop of the parareal calculation is given in the method `pararealSequence()` in the class `Parareal<T>`.

will be parallelized by this approach. Indeed, we showed several implementations of parallel calculations for dependent sequences taken from scientific applications.

Researchers in the field of high performance computing have tried to parallelize various problems with a large number of independent components, so far. Since most independent components have already been parallelized, we must try to analyze heavily dependent parts in scientific problems. As it was shown in this contribution, one of the strategies to parallelize such dependent calculations is the parareal-in-time algorithm. Further investigations of applicable areas are required if scientists will use the next-generation massively parallel resources in the post-peta era.

Recently, many related works on the Parareal-in-Time method have been published [18], [19]. This shows that the time-parallel method has already been one of the standard strategies to accelerate scientific calculations. Hereafter, we expect that this algorithm will be implemented in various package programs for scientific computing.

Finally, we emphasize the necessity for collaborations between computer scientists and computational scientists. Analyticity of scientific representation is one of the key properties assumed in large-scale numerical simulations. Because of this property, we can introduce approximate representations. The parareal-in-time algorithm also utilizes this property to configure convergent iterations. However, the definitions of the accurate and approximate solvers are not given only from computer algorithm since the approximation must be done based on a scientific insight for each problem. In a sense, collaborations between computer and computational scientists will be more and more important to configure large-scale scientific simulations. We expect such collaborations over different fields of sciences.

## Acknowledgements

This research was supported by JST, CREST.

## References

- [1] Top500 supercomputer sites, <http://top500.org>
- [2] J. Lions, Y. Maday, and G. Turinici, *C. R. Acad. Sci., Ser. I* 332, 661–668 (2001).
- [3] M. J. Gander and S. Vandewalle, *LCSE* 55, 291–298 (Springer, 2007); *ibid.*, *SIAM J. Sci. Comput.* 29, 556–578 (2007).
- [4] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah, *Phys. Rev. E* 66, 057701 (2002).
- [5] Y. Maday and G. Turinici, *C. R. Acad. Sci. Paris, Ser. I* 335, 387–392 (2002).
- [6] C. Farhat and M. Chandesris, *Int. J. Numer. Meth. Engng.* 58, 1397–1434 (2003).
- [7] Y. Maday and G. Turinici, *Int. J. Quant. Chem.* 93, 223–228 (2003).
- [8] P. F. Fischer, F. Hecht, and Y. Maday, *LCSE* 40, 433–440 (Springer, 2005).
- [9] G. A. Staff and E. M. Rønquist, *LCSE* 40, 449–456 (Springer, 2005).
- [10] A. Srinivasan and N. Chandra, *Parallel Computing* 31, 777–796 (2005).
- [11] T. Takami and A. Nishida, *Adv. Par. Comp.* 22, 437–444 (2012).
- [12] Y. Inadomi, T. Takami, J. Maki, T. Kobayashi, and M. Aoyagi, *Adv. Par. Comp.* 19, 220–227 (2009).
- [13] Y. Ito, T. Kobayashi, K. Takahashi, T. Takami, A. Nishida, and M. Aoyagi, *Proc. OSCIC* (2011).
- [14] A. Nishida, *Proc. ICCSA 2010, Part II, LNCS* 6017, 448–462 (2010).
- [15] H. Yoshida, *Phys. Lett. A* 150, 262–268 (1990).
- [16] G. Bal and Q. Wu, *LCSE* 60, 401–408 (Springer, 2008).
- [17] T. Takami and H. Fujisaki, *Phys. Rev. E* 75, 036219 (2007).
- [18] E. Aubanel, *Parallel Computing* 37, 172–182 (2011).
- [19] W. R. Elwasif, S. S. Foley, D. E. Bernholdt, L. A. Berry, D. Samaddar, D. E. Newman, and R. Sanchez, in *Proc. MTAGS2011* (2011).



# Optimizing a cricket edge detection system using feature extraction from wavelets over the time domain.

R. Rock (rodrick.rock@mycavehill.uwi.edu), A. Als (adrian.als@cavehill.uwi.edu), P. Gibbs (peter.gibbs@cavehill.uwi.edu),  
Department of Computer Science, Mathematics and Physics

University of the West Indies (Cave Hill Campus) Barbados

Cavehill, St Michael. Barbados West Indies

CSC'2012

**Abstract-** *In the game of cricket an edge is defined as an inadvertent connection between ball and bat. Edges, which result in the ball carrying directly to a fielder and being caught, warrant the batsman being adjudged out. Unfortunately there are various sounds that on-field umpires may confuse with genuine edges thereby causing the batsman to be erroneously adjudged as being out. In this paper the edge detection system proposed by Rock et al [1] is modified and extended to handle scenarios where there are edges and both the bat and pad are involved. Live audio samples of ball-on-bat, ball-on-pad and cases where both events occur within a narrow time window (<1 sec) will be recorded. Wavelet analysis, feature extraction and neural network classification will then be employed on these samples. Results will show the ability to differentiate amongst the three types of events, which is crucial to the development of a fully automated edge detection system.*

**Keywords:** Cricket, Wavelets, Neural Networks, Edge-detection, feature classification

## I. Introduction

The limited overs version of the game of cricket is played between two teams with 11 players on each side. On winning the coin toss, the captain decides whether his team bats or fields first. The conclusion of the allotted number of overs, or when all the batsmen are given out, marks the close of an innings. Each team is allowed one (1) innings. The aim of the team batting first is to score as many runs as possible during their innings. In order to win, they must limit the other team to fewer runs [2]. The test match version of the game is played over multiple ( $\leq 5$ ) days and is more complex as each team may accumulate runs over multiple ( $\leq 2$ ) innings. Cricket is the second most popular sport, and the Indian Premiere League's (IPL) 20/20 format



boasts of being the second highest paid sport ahead of the football's English Premier League (EPL) [3]. In 2009, the Indian Premier League (IPL) offered paychecks as high as US\$1.55 million to top class cricketers for a five-week contract [4]. This figure was eclipsed in 2011 when Gautam Gambhir of the Kolkata Knight Riders was awarded a contract for US\$2.4 million [5].

On the field of play, there are two umpires officiating a match. One umpire stands behind the stumps at the bowler's end of the pitch, while the other umpire stands at square leg. At international level there is also a third umpire on the sidelines and a match referee. The umpire at the bowler's end makes decisions on lbw and bat-pad appeals, no balls, wides and leg byes [6]. The square leg umpire will judge stumpings and run outs [6]. At the end of each over, the umpires change position [2]. The third umpire uses TV replays to rule on run outs, stumpings, boundary infringements and close catches. However, the third umpire can only make a decision if requested by the on field umpires. Their involvement in the game has become increasingly influential, with fans and commentators alike calling for technology to be used for every contentious appeal [2]. In the last few years, the ICC has trialed a review system which allowed players to challenge the on-field umpires and have their decisions referred to the third umpire - in Test cricket. The dismissed batsman or the fielding captain could appeal by making a "T" sign with both forearms at shoulder height, each team having a maximum of two unsuccessful challenges per innings [2].

The use of technology serves to protect both players' careers by avoiding incorrect decisions and the reputation of the game. The snickometer and hotspot are two devices which have been used and to some

extent are still being used in cricket mainly for bat pad decisions. These two devices have come under some criticism for their accuracy and the use of them in cricket has not been fully embraced by cricketing administrators [1].

The aim of this paper is to employ wavelet analysis, unique feature extraction methods and artificial neural networks to implement a fully automated decision making system for bat on pad edge and bat/pad decisions, thereby extending and improving the work done by Rock et al in [1]. This will greatly minimize the number of errors currently seen in the game.

## II. Background

The use of wavelets to analyze real-world signals (i.e. the constituent frequencies change over time, or have pulses, anomalies or other transient events) is well documented. Although these non-stationary signals may be intermittent and noisy, wavelet analysis can be employed to simultaneously monitor events in both time and frequency [7]. This special attribute makes the use of wavelet analysis more convenient than that of Fourier analysis as tradeoffs between knowing the time occurrence of an event and the constituent frequencies are avoided. The analysis of sound signals using the continuous wavelet transform (CWT) is well documented in the literature [8-10]. In the CWT approach the target signal to be analyzed is correlated with an analysis wavelet thereby producing a set correlation values along a time axis. The analysis wavelet is repeatedly stretched and used in other correlations with the target signal. Each stretching instance results in what is referred to as a scale value (y-axis). Thus this method provides another view of temporal signals as it transforms the regular

time vs. amplitude signal to time vs. scale, where scale can be converted to a pseudo-frequency. Therefore one can examine the temporal nature of audio events and the corresponding frequencies involved simultaneously. The correlation values produced during the transformation process provide critical information on the characteristics of the signal. Therefore by extracting and analyzing these correlation values a distinction can be made between different audio events. In this work, four main features were extracted across time from the CWT. These included the average **pseudo-frequency** for the CWT time range, along with the **standard deviation**, **kurtosis** and **skewness** of the said frequencies. These features were fed into an Artificial Neural Network (ANN) to produce the final result. ANNs are information processing systems that have performance characteristics common to biological neural networks. They consist of a number of interconnected neurons, each with an associated weight. These neurons work together to help solve various problems [11]. One of the main features of the ANN is its ability to take a set of features it has not encountered before and accurately output the desired classification result.

There are many instances where the CWT and neural network classification has been used. Kaewkongka [12] obtained a recognition rate of 90% success of rotodynamic machine conditions for four machine operating conditions using features extracted from the continuous wavelet transform and fed into a neural network. Kilby used the wavelet transform to enhance features extracted from the surface electromyography (SEMG) [13]. These features were taken from the time-based information as well the scale (i.e. pseudo frequency) axes. Using the extracted features of the dominant (pseudo) frequencies from the wavelet transform and

the related scales, they were able to train and validate an artificial neural network for SEMG classification. Kotani [14] used the wavelet transform and neural network classification to perfectly detect the surging sound (non-stationary signal) which leads to the destruction of a dryer machine.

Other than the work by Rock et. al in [1] there are no known instances where CWT and Neural Network classification has been applied in the area of cricket were uncovered in the literature. The Neural Networks are utilized to classify the features that are extracted from the sound files using the CWT. This classification can then be used to accurately determine both the order and the source of the events in a cricket match. The automated sound detection technique can greatly decrease the number of incorrect decisions being made in the game, which may ultimately protect a player's career.

### III. Methodology

The equipment setup, shown in Figure 1, is identical to that used for international matches and was configured at various hardball cricket grounds throughout Barbados. The microphone transmitter is covered in a small hole directly behind the stumps. The receiver and the laptop are assembled inside the players' pavilion. The recordings, made using the laptop's sound recorder program, are stored as a 16-bit pulse coded modulation (PCM) .WAV file, sampled at 44,100 kHz (stereo) for later processing.

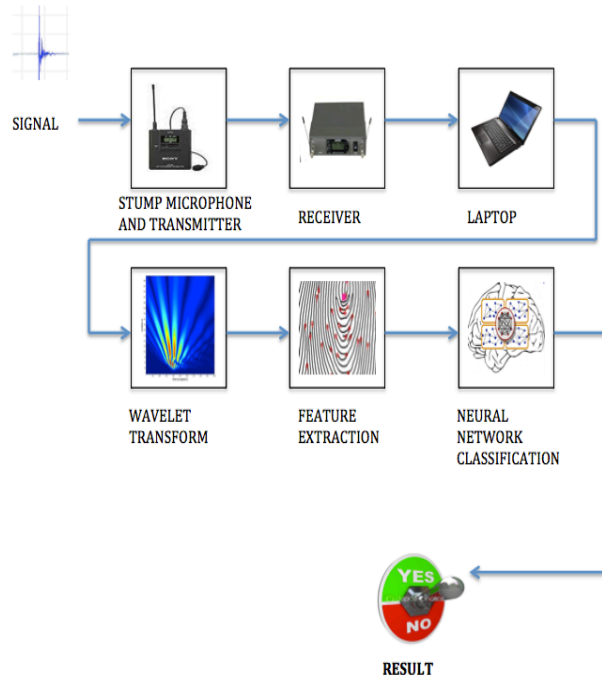


Figure 1: Schematic of experimental setup.  
 The key specifications for equipment used in recording the audio data are listed in Table 1.

TABLE I. EQUIPMENT PARAMETERS	
EQUIPMENT	KEY PARAMETERS
Shure SLX14/84 Wireless Lavalier Microphone System	<b>WL184 Supercardioid Lavalier Condenser Mic:</b>  Supercardioid pickup pattern for high noise rejection and narrow pickup angle
	<b>SLX1 Body pack Transmitter:</b>  518 - 782 MHz operating range
	<b>SLX4 Wireless Receiver:</b>  960 Selectable frequencies across 24MHz bandwidth
Mobile Precision M6400 Notebook Computer	Precision M6400, Intel Core 2 Quad Extreme Edition QX9300 2.53GHz, 1067MHZ

MATLAB programs were written to perform the CWT analysis and extract the following features: the average pseudo-frequency (Pfreq) from the selected CWT time range along with the standard deviation ( $\sigma$ ), kurtosis (k) and skewness (skn) of the said frequencies. The average pseudo-frequency was obtained from using the frequencies corresponding to the highest correlation values of the wavelet transform in each time interval. These features were used as input to the fully connected 4-input Multi-Layer Perceptron neural network depicted in Fig. 2. The network consists of a single hidden layer with three neurons each of which employed the tanh transfer function.

The network was trained with 232 data samples using a backpropagation algorithm.

The data set was divided into 116 incidences of bat-on-ball signals and 116 of ball-on-pad. Testing was done on 44 previously unknown signals. The output from the network was a decision on whether the ball hit the bat or the pad. Note that some of the audio samples included both bat-on-ball and ball-on-pad noises occurring in quick succession (<1 sec). These were analyzed as two separate events. The chronological nature of the decisions output from the ANN could then be used to determine the outcome of an appeal for LBW or bat/pad.

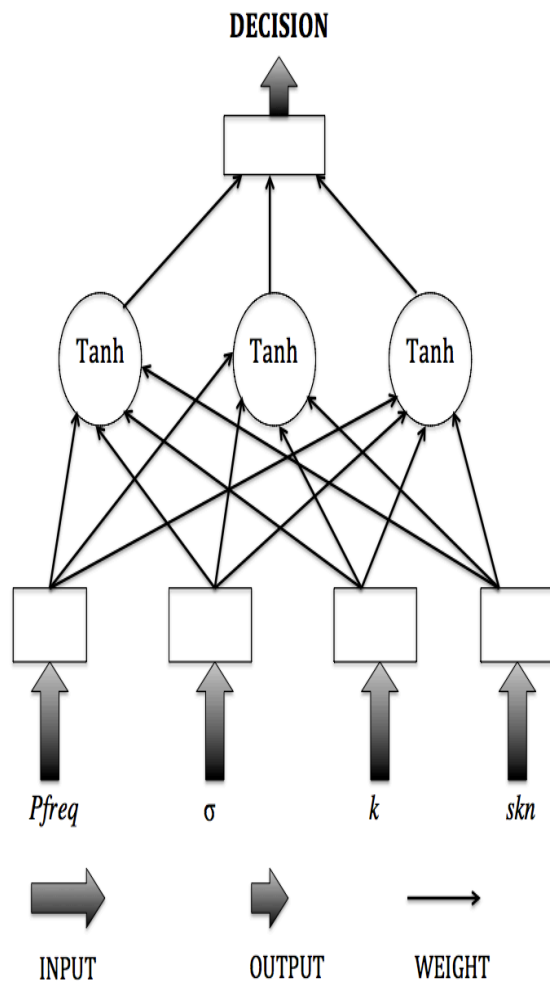


Figure 2: 4-Input Multi-Layered Perceptron

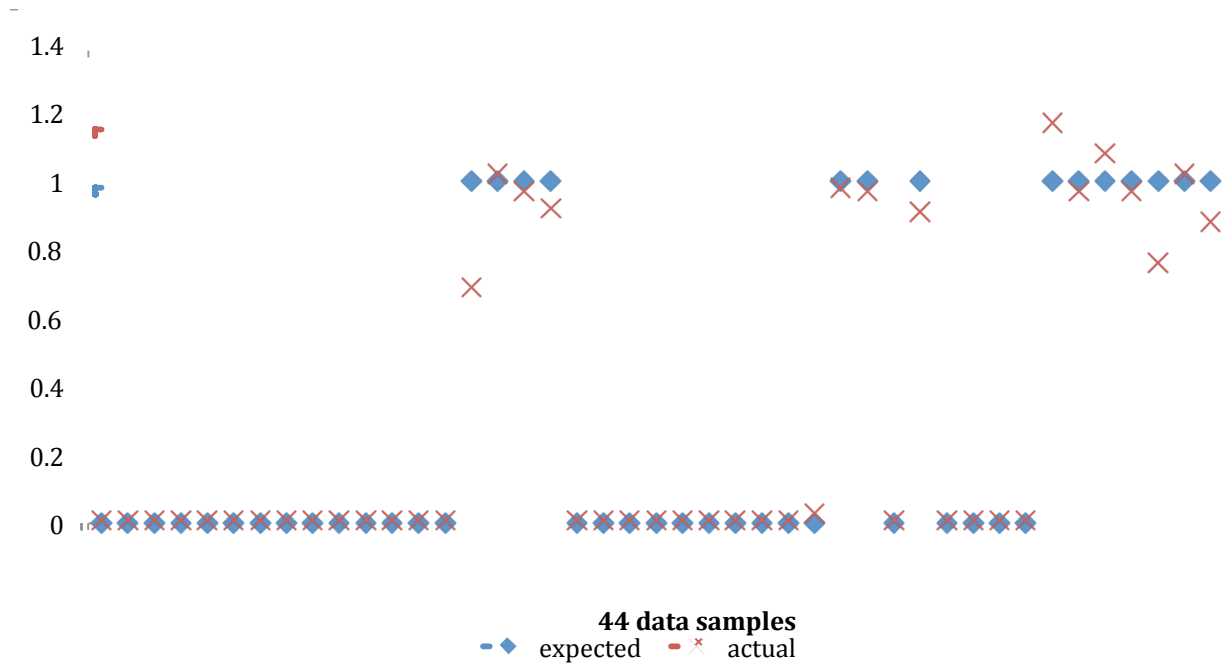
#### IV. Results

Three types of recordings were successfully compiled and analyzed. These included the impact of ball hitting bat, ball hitting pad and cases where both events occur within a narrow time window (<1 sec). Figure 3 shows the plot of desired output and actual output versus number of samples used for testing. In this work, the average pseudo-frequency along with the standard deviation, kurtosis and skewness were extracted across time instead of scale as was done in [REF]. There are two rationales for this modification. Firstly, in the case where both events occur within a narrow time window, there has to be a clear differentiation as to when in time these two events occurred. Secondly the observation was made that as the ball passes the bat or pad, the signal properties change. The extracted features from across the time domain best represented these changes and provided the better results. The Neural Network then successfully classified these features with a one (1) and zero (0) representing ball-on-bat and ball-on-pad, respectively. A threshold value of 0.5 was used in separating data, as anything above 0.5 was considered 1 and anything below 0.5 was considered 0. The line with the diamond markers represents the expected results whereas the line with x's is the actual results output from the neural network. The neural network performed exceptionally well. Observe that in Figure 3 the neural network output for some of the ball-on-bat cases deviate from the expected results. However, the threshold value ensures that all the deviations were still correctly classified. This resulted in a 100% correct classification for data not previously encountered.

**V. Conclusion**

Results show the neural network performed exceptionally well rendering a correct classification of 100% for data not previously encountered. It is believed that better results may be obtained from training the neural network with more sound files.

The methods used in this paper to completely remove the human factor from the data gathering and information-processing portion of the adjudication process will provide an edge detection system, which is a lot more accurate than the ones currently being used in the game of cricket.



- [et/rules\\_and\\_equipment/4183398.stm](#).
3. *IPL 2nd highest-paid league, edges out EPL*. 2010 [cited 2011 17/08]; Available from: <http://timesofindia.indiatimes.com/ip-larticleshow/5736736.cms>.
  4. Peter J. Schwartz, C.S. *The World's Top-Earning Cricketers*. [cited 2011 January, 1st]; Available from: <http://www.forbes.com/2009/08/27/cricket-ganguly-flintoffl-business-sports-cricket-players.html>.
  5. *Income of IPL 2011 Players* [cited 2011 17/08]; Available from: [http://www.paycheck.in/main/salarycheckers/copy\\_of\\_income-of-ipl-2011-players](http://www.paycheck.in/main/salarycheckers/copy_of_income-of-ipl-2011-players).
  6. [cited 2012 22nd May]; Available from: <http://www.lords.org/laws-and-spirit/laws-of-cricket/laws/law-24-no-ball,50,ar.html>.
  7. Fugal, D., *Conceptual Wavelets in Digital Signal Processing - An In Depth, Practical Approach for the Non-Mathematician*. 2009: CA Space and Signals Technical Publishing.
  8. Lambrou, T., et al., *Classification of audio signals using statistical features on time and wavelet transform domains*, in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. 1998.
  9. Schclar, A., et al., *A diffusion framework for detection of moving vehicles*. *Digital Signal Processing*. **20**(1): p. 111-122.
  10. D. Kumar, P.C., M. Henriques, M. Maldonado, R. Schmidt, J. Habetha, *Wavelet transform and simplicity based heart murmur segmentation*. *Computers in Cardiology*, 2006: p. 173-176.
  11. Fausett, L., *Fundamentals of Neural Networks Architectures, Algorithms and Applications*. 1994: Prentice Hall
  12. Kaewkongka, T., et al. *Continuous wavelet transform and neural network for condition monitoring of rotodynamic machinery*. in *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*. 2001.
  13. Kilby, J. and H.G. Hosseini. *Extracting Effective Features of SEMG Using Continuous Wavelet Transform*. in *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*. 2006.
  14. Kotani, M., et al. *Acoustic diagnosis for blower with wavelet transform and neural networks*. in *Neural Networks, 1995. Proceedings., IEEE International Conference on*. 1995.

# Generating Optional Number of Random Polygons Using a Point Set

Ali Nourollah<sup>1</sup>, Sara Maleki<sup>2</sup>

<sup>1</sup>Department of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran

<sup>2</sup>Department of Computer Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

**Abstract** - Generating polygons from random input data is one of the attractive problems in Computational Geometry. In this paper, we address generating optional number of simple convex or non-convex polygons over a finite points set of plane that generated randomly. composition of this optional number of polygons is possible by using random lines and plane zoning. the algorithm is usable in problems like generating random barriers and simulation robots motion between them. This problem is new problem and there is no report on it until now.

**Keywords:** robots motion planning, simple polygons, partitioning lines

## 1 Introduction

Computational Geometry is one of the most important research fields in computer science in which calculations are done on geometric objects like polygons. Polygons are plain and suitable present for real world objects[1]. In computational geometry, There are different methods for generating random convex and non-convex plain polygons in plane. Almost all of these methods have time complexity of  $O(n \log n)$  order.

There are variety of methods to generate optional number of polygons having a point set. Our method is drawing random lines, creating areas in plane and then generating points with random coordinates in them. To

generate polygons, we can use each of the known methods in computational geometry and separating lines can be assume as paths between barriers.

The rest of the paper is organized as follows: some primary concepts are explaining in next section. In the section 3, some of the different methods to draw polygons are explained. Section 4 defines the problem formally and present the suggested method. Finally in the paper is concluded in section 5.

## 2 Primary Concepts

In geometry a polygon is a flat shape that is surrounded by finite trail of straight edges. This finite trail of edges are called polygon and the cross points of each two edge are polygon's vertex. If  $S$  is the set of random vertex, there can be  $T$  plain polygons on  $S$ , each polygon can be generated with the probability of  $\frac{1}{T}$ . polygon with edges crossing point only on  $S$  vertex is called plain polygon.

A convex polygon is a plain polygon that for each two vertices  $x$  and  $y$  of polygon vertices, the  $\overline{xy}$  line is in polygon or on it's edges, it means that  $\overline{xy} \subseteq P$ . in other words the polygon is convex if all it's vertices be less than 180 degree and the plain polygon having a vertex more than 180 degree is called non-convex.

## 3 Polygon Generation

Random generation of geometric objects always has been mentioned. An algorithm with worst case of  $O(n^2)$  order was presented to composite x-monotone polygons[4] and different innovated methods was described by Held and Auer to generate a polygon with having a point set on a flat plane:

- Steady Growth : An incremental algorithm that adds each points and it's corresponding edge to polygon in each stage. It's time complexity is  $O(n \log n)$ [5].
- Space Partitioning : It is working according to divide and conquer method and generate subsets of primarily set. It's time complexity is  $O(n^2)$ [5].
- 2-opt Moves : This algorithm starts with a random polygon and achieved a simple polygon by changing edges. It's time complexity is  $O(n^4)$  [5].

There are other methods like two-peasants and so on. Each of explained methods above can be used in this paper for the mean of generating polygons. We use the method with the best order.

## 4 Proposed Algorithm

In this section suggested algorithm for generating optional number of plain polygon with creating  $n$  random number as polygons vertex will be described. The goal is creating  $k$  polygons on plane, in the way that polygon's vertex cover  $n$  random points. polygon's edges should not be crossed and polygons should not be inner each other. Algorithm that is



suggested in this paper, supply the goal of problem in 5 phases :

- A. region creation ,B. calculating the area of each one, C. calculating number of points in each area, D. finding point's coordination in each region, E. drawing polygons.

### 4.1 Region Creation

To find polygons with features said in previous section, first we should divide plane in to separated regions. For generating regions, we use the method of drawing lines with random slop and intercept and using them create regions in plane in the number of polygons. number of polygons that can be created is at most  $n/3$  and at least number of lines needed is calculating using  $k = \frac{d^2+d+2}{2}$  formula in which  $d$  is number of lines should be drawn[6]. This number will be achieve when all the lines have cross points in the plane. So in drawing lines problem of parallel lines will always check. If  $R(l)$  is number of areas generating by drawing  $l$  Lines , then after drawing a new line, number of areas will increase to  $R(l) + l + 1$ . We assume that cross point of each two lines is located in our plane. If the number of required areas is between  $R(l)$  and  $R(l + 1)$ ,the number of required lines will be  $l + 1$ . We should select  $k$ regions of generated regions to locate polygons in them. We use two random number  $m$  and  $a$  as line 's slop and intercept To draw them.

If  $d$  is number of lines, the time complexity order of this phase is  $O(d^2)$ .

### 4.2 Calculating The Area of Each Region

There are different ways to calculate area of each region. All of regions are convex because of they are built by crossing lines. To find areas we can do like bellow:

Suppose that the tail  $(x_1, y_1), \dots, (x_n, y_n)$  be the vertex of one region. The area of a polygon that non ofit's edges cross each other can be calculated by formula (1):

$$A = \frac{1}{2} \left( \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix} + \dots + \begin{vmatrix} x_n & x_1 \\ y_n & y_1 \end{vmatrix} \right), \tag{1}$$

If points are in counter clockwise order, calculated area is positive otherwise absolute of calculated number is area[2]. The need to these calculations will be clear in the next part of algorithm.

Having  $k$  regions, the time complexity of this phase will be  $O(k)$ .

### 4.3 Calculating Number of Points in Each Area

Points should be scattered in regions in a way that their density in all places be balanced. To solve this part of problem, we use ratio of areas mean numbers that have been calculated in previous part of algorithm. Number of points in each region can be found using formula (2):

$$N_j = \frac{S_j}{\text{Total area of selected regions}} \times n \tag{2}$$

In which,  $n$  is total number of points,  $S_j$  is the area of  $j$ th region and  $N_j$  is number of points in  $j$ th region.

It is clear that the time complexity of this phase will be  $O(k)$ .

### 4.4 Finding Point 'sCoordinates in Each Region

After finding number of points in each region, we should specify their coordinates. There are different methods like points on longest diagonal or triangulation the regions and finding points in them and so on .

In implementation of this paper's algorithm, we use the method of triangulation and finding points in triangles. Triangulation of these areas is plain because of they are convex regions[6].

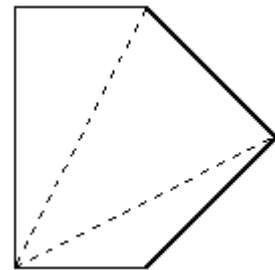


fig.1. convex polygon triangulation

According to figure 1, select one of vertices as main, with drawing those diagonals of polygon that one of their vertices is the main vertices, the region will be triangulated.

Having three vertex of each triangle, finding points in it is so easy. To catch this goal, we use formula 3:

$$\text{Point} = a_1 v_1 + a_2 v_2 \tag{3}$$

In which,  $a_1$  and  $a_2$  are monotonous identifier in  $[0,1]$ .

For finding  $n$  points, the time complexity of this phase will be  $O(n)$ .

### 4.5 Drawing Polygons

After finding points in each region, for drawing convex or non-convex polygons in each region we can use any of existence methods in computational geometry. In implementation of this paper 's algorithm, we used Two Peasants method.

The total time complexity of algorithm will be  $O(n \log n + k + d^2)$ .

Figure 2, shows a sample of suggested algorithm output. Total number of random points is 72 points that are scattered in total parts proportional to each part's area. Random number of polygons , selected byuser is 12. So we use 5 lines to partition plane and select 12 regions out of 16 generated regions to show polygons in them.



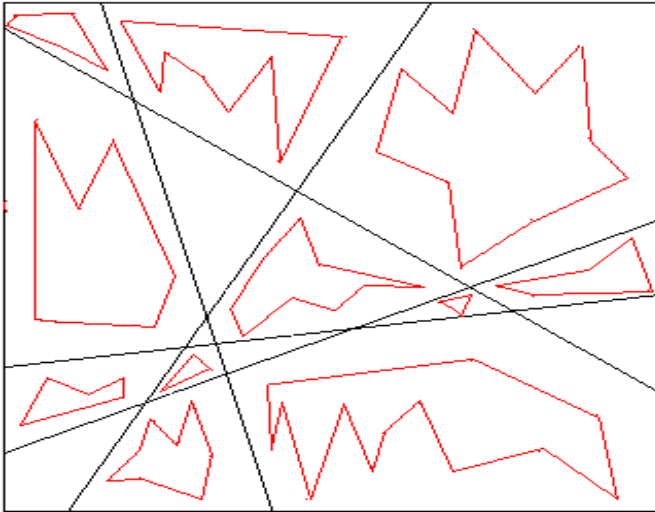


fig. 2. algorithm output

Algorithm1 :Generating k Random Polygons

```

Inputs :
n: total number of point,
k: number of polygons.
Outputs :
k random polygons,

Calculate required number of lines as
line_no with having k

For i← 1 to line_no do
  Select m and a as random numbers
  Draw the line L:y=mx+a
EndFor

Select k areas of R(L) areas

For j← 1 to k do
  Calculate area of jth region as Sj
  Generate  $\frac{S_j}{\text{total area of k regions}} \times n$ 
  random points in the region
EndFor

For f← 1 to k do
  Two Peasants(points in fth region )
End For

```

## 5 Conclusions

In this paper a new method to Generate Optional Number of Random Polygons Using a Point Set in a 2d plane was presented. In order to composition of regions in the plane random lines have been used.to find Random points, we triangulated each region and used a simple formula and to determine number of points in each region, areas ratio was useful. for composition polygons any of methods in

computational geometry is usable. Random Lines drawn between polygons can be used as paths for robots motion between barriers.

## 6 References

- [1] Berg M. D., *Computational Geometry: Algorithms and Applications*, 3rd edition, published by Springer-Verlag, 2008.
- [2] Beyer, W. H. (Ed.). *CRC Standard Mathematical Tables*, 28th ed. Boca Raton, FL: CRC Press, pp.123-124, 1987.
- [3] Ana Paula Tomas and Antonia Leslie Bajuelos, "Generating Random Orthogonal Polygons", 1996.
- [4] C.Zhu, G.Sundaram, J.Snoeyink and J.S.B Mitchell, "Generating random polygon with given vertices", *Compute.Geom.:Theory Applic*, 1996.
- [5] [http://www.geometrylab.de/polygon/Random Polygon.html.en](http://www.geometrylab.de/polygon/RandomPolygon.html.en)
- [6] Dailey D., Whitfield D., "Constructing Random Polygons " *SIGITE'08, USA*, pp. 119-124, 2008.

# Numerical Simulation and Modeling of Supersonic Combustion Flow

Tsung Leo Jiang and Jun Yuan Chen

Department of Aeronautics and Astronautics, National Cheng Kung University, Tainan, Taiwan, ROC

**Abstract** - It is of importance for the development of a numerical method for the supersonic combustion simulation and modeling, since they are characterized by the high-temperature and high-speed combustion which is very hard to measure. In the present study, a computational model, employing the software Fluent adopting the SST- $k-\omega$  turbulent model and the flamelet turbulent-combustion model, is developed for the supersonic turbulent combustion flow. The results obtained from the present study show that although the predicted pressure distributions for the non-reactive flow are in agreement with the experimental data and the published numerical results, those for the reaction flow are only qualitatively in agreement with the experimental data. The pressure is under-estimated in the downstream expansion region. This is due to the oversimplification of the flamelet model for the complex detailed chemistry. An improvement on the turbulent combustion model for supersonic combustion is addressed.

**Keywords:** Supersonic Combustion, Numerical Simulation

## 1 Introduction

Supersonic combustion is extremely complex, since the fuel has to be injected, mixed, ignited, and burned with the air in a supersonic stream within a millisecond. The numerical simulation is thus a vital approach for the understanding of supersonic combustion which is characterized by the high-temperature and high-speed combustion and very hard to measure. The use of the numerical simulation for the study of supersonic combustion is able to reduce the cost and risk of the experimental work. Nevertheless, the numerical simulation has to be validated by the experimental data before it can be applied to the detailed analysis of supersonic combustion.

In the present study, the software Fluent [1] adopting the SST- $k-\omega$  turbulent model and the flamelet turbulent-combustion model is employed for the simulation of the supersonic combustion flow of the LAERTE scramjet engine combustor, developed by the French Aerospace Lab (ONERA) [2], as shown in Fig. 1. The length of the combustion chamber is 870mm. It consists of two segments. One is the constant cross-section area duct of 370 mm long, and the other is a divergent duct of 500 mm long with 1.15

degree expansion along the top and bottom walls. A fuel injector is installed in the center of the combustion chamber with the outer diameter being 10mm, and the inner one 6mm. The inlet flow conditions are shown in Table 1.

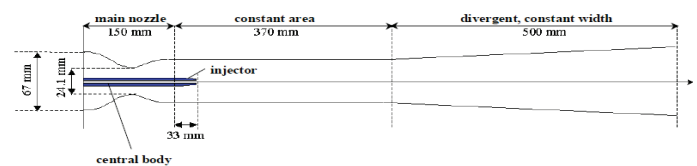


Fig. 1 The LAERTE Scramjet engine combustor [2]

Table 1 The Inlet Conditions.

	Fuel	Air
Inlet pressure(Pa)	80000	80000
Inlet Mach number	2	2
Total temperature(K)	300	1840
The composition	H <sub>2</sub> : 0.51 CH <sub>4</sub> : 0.49	N <sub>2</sub> : 0.627 O <sub>2</sub> : 0.252 H <sub>2</sub> O : 0.121

## 2 Results and Discussion

The predicted wall-pressure distributions of the non-reactive flow, where the fuel is replaced by nitrogen, are shown in Fig. 2, in comparison with the experimental data of Quintilla *et al.* [2] and the simulation results of Davidenko *et al.* [3]. The present predictions are in good agreement with the experimental data and the published numerical results.

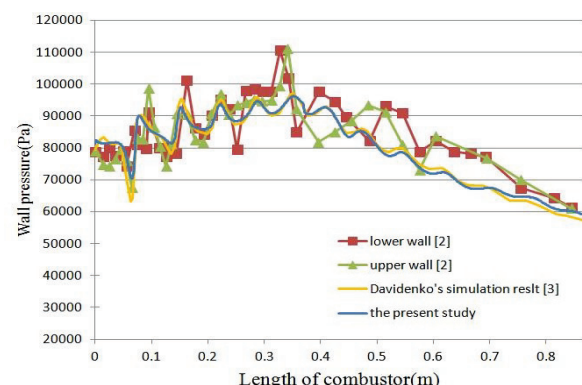
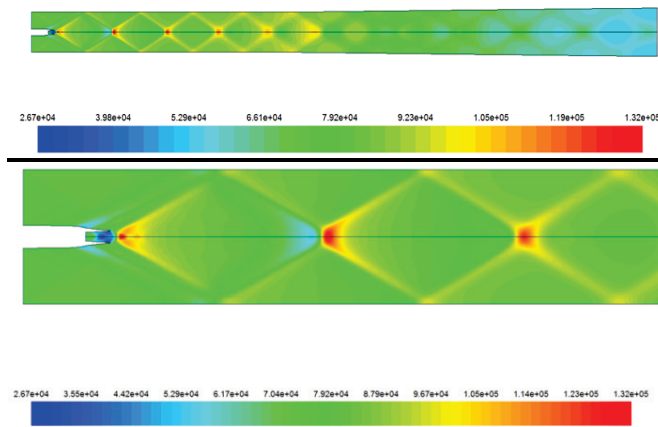


Fig. 2 Comparison of the predicted wall pressure of the non-reactive flow with the experimental data [2] and published numerical results [3]

The predicted pressure distributions of the non-reactive flow are shown in Fig. 3. When the supersonic free-stream flows into the combustion chamber, it expands due to the contraction of the side wall of the injector, resulting in a slight drop in pressure. However, as the fuel is injected at the chamber length of 33mm, the flow is compressed by the injected flow, resulting in an oblique shock and an increase in pressure. At the divergent section, the supersonic flow expands, and the pressure drops.

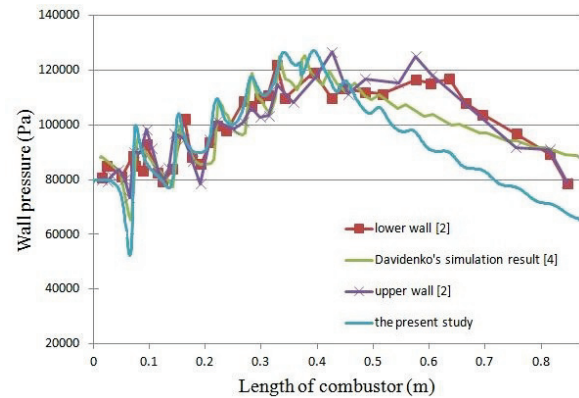


**Fig. 3** Pressure distributions of the non-reactive flow (Unit: Pa)

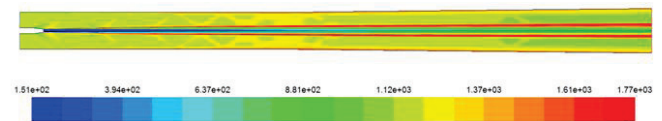
The predicted wall-pressure distributions for the combustion flow are shown in Fig. 4, in comparison with the experimental data of Quintilla *et al.* [2] and the simulation results of Davidenko *et al.* [4]. The predicted results are only qualitatively in agreement with the experimental data. The pressure is under-estimated in the downstream expansion region. This is because the flamelet model does not take into account the effect of the detailed finite-rate chemistry on the flow field. This results in a narrow diffusion-flame zone, as shown in Fig. 5, where the temperature distribution of the combustion flow is depicted. As the reaction zone is limited, the average flow temperature will be lower than it should be, leading to a lower pressure in the downstream expansion region. Therefore, the detailed finite-rate chemistry should be accounted for in the turbulent combustion model for supersonic combustion.

### 3 Conclusions

In the present study, a computational model, employing the software Fluent adopting the SST- $k-\omega$  turbulent model and the flamelet turbulent-combustion model, has been successfully developed for the supersonic combustion flow. The predictions for the non-reactive flow are in agreement with the experimental data and the published numerical results. However, due to the oversimplification of the flamelet model for the complex detailed chemistry, the predicted pressure is lower in the downstream expansion region. The detailed finite-rate chemistry is suggested to be accounted for in the turbulent combustion model for supersonic combustion.



**Fig. 4** Comparison of the predicted wall pressure of the combustion flow with the experimental data [2] and published numerical results [4]



**Fig. 5** Temperature distribution of the combustion flow (Unit: K)

## 4 References

- [1] Fluent Incorporated, Fluent 6.3, User Guide, 2006.
- [2] Quintilla, V., Magre, P., Scherrer, D., Destors, P.E., Dufour, E., France, M., "Experimental and Numerical Investigation of Supersonic Reacting Hydrogen/Methane Jets in Hot Air Co-Flows". AIAA/CIRA 13<sup>th</sup> International Space Planes and Hypersonics Systems and Technologies Conference, 2005: p. 1-16.
- [3] Davidenko, DM., Gökalp, I., Dufour, E., Magre, P., "Numerical Simulation of Supersonic Combustion with CH<sub>4</sub>-H<sub>2</sub> Fuel". European Conference for Aerospace Sciences, 2005.
- [4] Davidenko, D., Gökalp, I., Dufour, E., Magre, P., "Ignition and Combustion of Hydrogen and Methane in a Model Supersonic Combustion Chamber", Proceedings of the European Combustion Meeting, 2005.

