Periodic Steady State Solution of Power Networks using the Current Injections Method and Parallel Processing based on GPUs

Marcolino H. Diaz-Araujo¹, Aurelio Medina-Rios¹, Ernesto Magaña-Lemus¹, Antonio Ramos-Paz¹

Abstract — This paper details an efficient, fast and non-iterative algorithm for the simulation of the steady state response of power networks under non-sinusoidal conditions in the frequency domain. The algorithm uses the injection current method, LU decomposition, and parallel processing techniques based on Graphic Processing Units (GPUs). It is shown that for the explicit harmonics representation, the implementation on a GPU-based platform becomes an efficient computational resource to find the steady state solution since floating-point operations and repetitive calculations increase in proportion to the number of harmonics and size of the network, both related with the computer effort.

Keywords — Injection current method, LU decomposition, parallel processing, Graphic Processing Unit.

1 Introduction

The complexity of power systems has increased in direct proportion with the network size and the presence and continuous incorporation of nonlinear elements. In addition, the need for more efficient, accurate and robust simulation techniques has also increased. Engineers need to determine many variables and quantities of interest, such as harmonic levels in the power network [1].

Admittedly, the periodic steady state solution of power systems can be obtained in three main frameworks, i.e. frequency domain, time domain and hybrid frequency-time domain, respectively [2]. A concise review is given in [2] regarding the main advantages, drawbacks, formulation and convergence characteristics related to the different methods belonging to the frames of reference above indicated. In particular, this contribution deals with a method for the efficient periodic steady state solution of power networks in the frequency domain.

A widely used methodology to compute the steady state solution of power networks is based on a unified iterative approach where phases, linear and nonlinear components, number of harmonics explicitly represented, harmonic cross-coupling and unbalance effects are combined together for the entire system [3]. Its application to larger scale three-phase systems may lead, however, to excessive computer effort as high dimension problems may need to be solved [4].

On the other hand, phasor equivalents of linear and nonlinear power system waveforms consist of an infinite number of terms. For the purposes of simulation, this representation is truncated to a finite number of terms. This procedure is intuitive and it should be verified that the frequency set used in the analysis provides accurate results by increasing the number of frequencies, then repeating the simulation, and checking that the simulation results change by a negligible amount. In practical terms, it is sufficient for harmonic analysis to account for the first 50 harmonics. However, it may result on a time consuming process.

Due to the time critical nature of such computation, the need for parallel processing for the simulation of realistic systems becomes a necessity. Parallel processing is a technique that allows one program to execute multiple tasks concurrently. A thread consists of a stream of control that can execute its instructions independently so a multi-threaded process, or program, can perform numerous tasks concurrently [5].

Over the last 15 years, significant changes have occurred to summarize and review the relationship between power system analysis and high performance computing. By way of example, in [6] a study for large-scale transient stability simulation based on the massively parallel architecture of multiple GPUs is made.

In this paper, the current injection method [1] is applied to obtain the periodic steady state analysis of power systems under non-sinusoidal conditions by means of parallel processing based on GPUs and LU decomposition.

2 Harmonic Analysis

2.1 Linear Circuit Analysis

The steady state solution of power systems operating under sinusoidal and non-sinusoidal conditions can be obtained using phasor analysis. The power network is solved for each frequency of interest as opposed to only the fundamental frequency [7].

In general, a linear circuit analysis operating under non-sinusoidal conditions can be represented by the following set of linear equations:

\[ \begin{bmatrix} \mathbf{v}_h^1 \\ \vdots \\ \mathbf{v}_h^N \end{bmatrix} = \mathbf{I}_h^1 \cdot \mathbf{x}_h^1 + \cdots + \mathbf{I}_h^N \cdot \mathbf{x}_h^N \]

(1)
where \( i \) is the current phasor for frequency \( h \) injected at node \( j \); \( y^{ij} \) is the equivalent admittance for frequency \( h \) between nodes \( i \) and \( j \) (mutual when \( i \neq j \) and self when \( i = j \)); \( v \) is the voltage phasor for frequency \( h \) at node \( j \), and \( N \) is the number of nodes of the network.

In compact form (1) can be written as,

\[
I_h = Y_h V_h
\]

where \( h \) is the harmonic, \( I_h \) is the harmonic current injection vector, \( Y_h \) is the harmonic equivalent admittance matrix, and \( V_h \) is the harmonic voltage vector.

2.2 The Harmonic Current Injection Method

The harmonic current injection method is widely used to carry out harmonic propagation studies in distribution systems [7]. The salient features of the method are outlined below:

- Build \( Y_h \) of the power system including the contribution for all sources and loads. A different \( Y_h \) must be calculated for each harmonic \( h \).
- Obtain \( I_h \) by extracting the term of the appropriate frequency from each nonlinear load.
- Use (2) to calculate \( V_h \). Both magnitude and phase information are important. If a time domain solution required for each bus voltage, the calculated harmonics are superimposed.

2.3 LU Decomposition

Since (2) should be repetitively used, once for each harmonic. It is advisable to form \( Y_h \) with an algorithm being time and memory efficient. For instance, triangular factorization may be applied. The triangular factors of \( Y_h \) and the voltages are calculated by forward and backward substitution, respectively.

In this expression \( L \) and \( U \) are the lower left and upper right triangular factors of \( Y_h \). The vector \( W \) is solved by forward substitution, and the vector \( V_h \) is subsequently calculated by backward substitution.

2.4 Parallel Processing Based on GPU

NVIDIA is one of the leading manufactures of GPUs. Architectures Fermi and Kepler are the most widely used for parallel processing. The GPU used in this research is the Tesla C2075 with Fermi architecture [8], discussed next.

The first Fermi architecture based GPU, implemented 3.0 billion transistors and features up to 512 CUDA cores. A CUDA core executes a floating point or integer instruction per clock for a thread. The 512 CUDA cores are organized in 16 streaming multiprocessors (SM) of 32 cores each. The GPU has 64-bit memory partitions, for a 384-bit memory interface; supporting up to a total of 6 GB of GDDR5 DRAM memory. A host interface connects the GPU to the CPU via PCI-Express. The GigaThread global scheduler distributes thread blocks to SM thread schedulers. Fig. 1, shows the elements of SM and a core. CUDA is the hardware and software architecture that enables NVIDIA GPUs to execute programs written with C, C++, Fortran, Open CL, DirectCompute, and other languages [9]. A CUDA program calls parallel kernels. A kernel executes a process in parallel across a set of parallel threads. The programmer or compiler organizes these threads in thread blocks and grids of thread blocks. The GPU instantiates a kernel program on a grid of parallel thread blocks. Each thread within a thread block executes an instance of the kernel, and has a thread ID within its thread block, program counter, registers, per-thread private memory, inputs, and output results.

A CUDA program has two parts: the serial part and the parallel part. In the serial part, no parallelism exists and the instructions are executed in the CPU. In the parallel part, which involves massive data parallelism, instructions are executed in the GPU. A high-level view of the CUDA programming model is illustrated in Fig. 2.

2.5 Component Modelling

To assemble the elements of a power system into a bus impedance matrix for each harmonic, a frequency dependent model for each element must be developed [11]. This section summarizes the typical representations of common network components for harmonic analysis.

1) Transmission lines: For the case of a transmission line, the total resistance and inductive reactance of the line is included in the series arm of the equivalent-\( r \) and the total capacitance to neutral is divided equally between its shunt arms.

2) Generators: These are considered to be linear components whose harmonic impedance is

\[
Z_G = R v_h + j X_h
\]

where \( R \) is derived from the generator power losses and \( X_h \) is the generator subtransient reactance.

3) Transformer: These are considered to be linear components whose harmonic impedance is

\[
Z_T = R v_h + j X_h
\]

where \( R \) is derived from the tranformer power losses and \( X_t \) is the transformer’s short-circuit reactance.

![Figure 1. SM and core processor scheme.](Image 314x91 to 552x268)
4) **Capacitor banks:** These are considered to be passive components, where

\[ Z_C = -jV^2/Q \]  

where \( V \) is the line voltage and \( Q \) is the reactive power.

5) **Passive loads:** Linear passive loads do not produce harmonics but have a significant effect on the system frequency response. A general model for passive load is given in [12]

\[ Z_L = R X_L / (R + X_L) + X_S \]  

where

\[ R = V^2/P \]  
\[ X_L = jhR / 6.7(Q/P - 0.74) \]  
\[ X_S = j0.073hR \]

6) **Non-linear loads:** These are represented by a harmonic current injection source. Harmonic current injection sources are used to represent the harmonic contributions from static VAR compensators (SVCs), induction arc furnaces, rectifiers and electronic devices. For example, a SVC is represented by the harmonic current injection given by,

\[ I_n = (\%)I_1 \]  

where \( \% \) is a percentage of the current at fundamental frequency given by

\[ I_1 = (Q / \sqrt{3}V) e^{j(\theta + \pi/2)} \]  

Where \( \theta \) is the voltage angle obtained from a conventional power flow study, and \( \pi/2 \) is the required phase shift, since the current leads or lags the voltage by 90°.

### 3 Harmonic Propagation Method

The algorithm for the harmonic propagation in the power network combines a conventional power flow study with the injection current method and LU decomposition.

Fig. 3 shows the flow chart for the harmonic propagation method. It is basically composed by three blocks, i.e. the data block that reads the parameters of the power system, the power flow block that performs a conventional power flow study and the harmonic voltage block that determines the harmonic propagation throughout the system.

Some parts of the algorithm are executed sequentially and some parts in parallel. The system data block is programmed sequentially. Then two tasks are simultaneously run. Each task is performed by one thread in the CPU (OpenMP). One of the threads (thread 1) performs the power flow study meanwhile the other thread (thread 0) copy the system data from the CPU to the GPU. These two tasks are executed in parallel and have different computation time, so they have to be synchronized. To synchronize this part of the algorithm a flag is used. It is initially flag = 0 and changes to flag = 1 when the power flow concludes. If the power flow study has no finished yet, thread 0 will have to wait until thread 1 finishes its process. The last part of the algorithm is executed in the GPU (CUDA). For each harmonic, it is necessary to obtain the equivalent admittance matrix, the current injection vector, and solve for the harmonic voltage vector. Superposition effects are accounted to obtain the final result. The following steps summarize the complete method:

- Find the steady state solution given by a conventional power flow study.
- With the resultant voltages at fundamental frequency, compute the passive equivalent circuit.
- Obtain the driving point impedance seen from node where the non-linear load is connected.
- Solve (2) for each frequency to get the final result by superposition of effects.

![Figure 3. Algorithm of the proposed method.](image-url)
4 Test Case

The test system of Fig. 4 has been used to illustrate the performance of the implemented method. The electric power system includes three SVCs connected in three different buses. The maximum magnitudes of harmonic current injected by the SVCs are given in Table I. The fundamental power flow solution is given in Table II. The SVCs are considered to be delta connected, hence no zero sequence harmonic current is injected into the system.

By using the same procedure, the parameters for the rest of load buses are obtained. The generator reactance for the slack bus is $Z_G = jh0.0001$ and for regulated buses are $Z_G = jh0.001$.

For the SVC at bus 14, the fundamental frequency current is given by

$$I_{SVC} = \left(\frac{0.40}{\sqrt{3} \cdot 1.1704}\right) e^{j(-0.3331 + \frac{3.1415}{2})} = 0.1973 \angle 70.914^\circ$$

In Table I, the harmonic currents are given as percentage of the fundamental component.
4.1 Harmonic Propagation in Larger Systems

The parallel code has been applied to larger systems to show the relevance of the harmonic injection current method processed in parallel. The IEEE-57, 118 and 300 test bus systems have been solved. Table IV shows the results obtained by the sequential and the parallelized algorithm. The first column shows the node number, the second the time for data reading of power system, and the third is the time consumed by the power flow study. Columns fourth and fifth show the computation time of the current injection method processed en parallel (PCI) an sequential (SCI), respectively. Columns sixth and seventh show the processing time for the complete parallel (TPCI) and sequential (TSCI) code, respectively. Last two columns show the speed-up of the complete algorithm and the speed-up of the current injection method, respectively.

The comparison of the fourth column to the fifth shows the importance of using parallel processing based on GPU for the current injection method. The best speed-up is 12.95 times for the IEEE-300 bus test system and 4.84 times for the complete algorithm. For the case of the IEEE 118 bus test system the results were 6.94 and 3.83, respectively, and for the case of the IEEE 57 bus test system were 1.11 and 1.07, respectively. It is clear that the speed-up significantly increases with the size of the power network. The speed-up can be improved if the entire algorithm and not just to the current injection algorithm are solved by using parallel processing.

5 Conclusions

The application of parallel GPU-computing in harmonic propagation studies in large electrical networks has been presented.

The algorithm is non-iterative, and a solution is always obtained. The algorithm exploits the injection current method and LU decomposition. The introduction of the parallel processing technique has been very effective in reducing the required CPU time and in increasing memory efficiency. Even for the small test system analyzed, it has shown that the parallel algorithm is faster than the sequential algorithm. The algorithm implemented on a Tesla C2075 GPU improves efficiency from 1.11 to 12.95 times for the injection current method and from 1.07 to 4.84 times in total simulation, as compared with a conventional algorithm processed in series.

6 References


7 Acknowledgment

Financial support by Consejo Nacional de Ciencia y Tecnología (CONACYT) is gratefully acknowledged. The first author wants to acknowledge the scholarship granted by CONACYT for his doctoral studies at the División de Estudios de Posgrado, Facultad de Ingeniería Eléctrica, Universidad Michoacana de San Nicolás de Hidalgo, Morelia, México.