Simulating Spiking Neurons by Hodgkin Huxley Model

Terje Kristensen\(^1\) and Donald MacNearney\(^2\)

\(^1\)Department of Computing, Bergen University College, Bergen, Norway, tkr@hib.no
\(^2\)Electrical Systems Integration, Aspin Kemp and Associates, Stratford, PE, Canada, donaldmaenearney@aka-group.com

Abstract - The Hodgkin Huxley model of the biological neuron is numerically solved in Java, using both an Euler and a 4th order Runge Kutta method. The theory behind the Hodgkin Huxley model is described and the basic spikes are investigated using a graphical program developed by the authors. This is the first step in a long-term goal to develop a tool in Java to simulate neuron interactions on a larger scale. In the future, the authors would like to simulate interactions between multiple neurons, with the aspiration to investigate large-scale neural behavior with visualization and total control of all parameters involved.

Keywords: Hodgkin-Huxley, Java, simulation, spiking neuron, numerical integration

I. INTRODUCTION

In 1952, Alan Lloyd Hodgkin and Andrew Huxley described their mathematical model, which explains the ionic interactions that characterize action potentials in a giant squid axon. Their model remains the most widely accepted model for describing neuron action potentials, and is used ubiquitously in neural modeling to this day. This project focuses on the Hodgkin Huxley model, and generating a graphical solution to their equations using Java. This graphical model will be used to draw conclusions about neural behavior, and to reinforce concepts presented in literature on such topics.

First, the objectives of this experiment will be formalized, followed by the background knowledge necessary to understand the Hodgkin Huxley model and achieve these objectives. A summary of the two numerical integration techniques to be used will be presented, and then a selection of observations made using the Java program will be discussed. As this project represents only the basics of a large and complex field of ongoing research, a summary of what may be realized in the future will be presented, pending more research in this area, notably in the area of multiple neuron interactions.

A. Objective

The purpose of this project is to investigate the Hodgkin Huxley model of a neuron action potential. The solution of the presented equations will be obtained using numerical methods in Java, and the solutions will then be presented graphically, using the Swing library in Java. First, the Euler numerical method will be used to obtain the solution, and then be compared to a solution found using a 4th order Runge Kutta method. Once this basic simulation has been completed, these simple solutions may be concatenated into more complex operations involving multiple neurons. When this is completed, the initial steps for integrating these spiking neurons into a two dimensional network of neuron modules may be investigated.

II. THEORY

The Hodgkin-Huxley model of neuron action potentials is based on measurements made by Hodgkin and Huxley on the axon of a giant squid. Hodgkin and Huxley were able to create a model of the electrical characteristics of a cellular neuron based on standardized circuit theory. They characterized the cellular membrane with a capacitive element and a series of varying resistive elements, the conductivities of which vary, based on the potential present across the membrane. As the system is disturbed by an input, or injected current, the membrane potential will attempt to regain its equilibrium through a set of equations defined empirically by Hodgkin and Huxley.

\[
I = C_m \frac{dV}{dt} + g_m n^4 (V - V_k) + g_N m^3 h (V - V_{Na}) + g_l (V - V_l) \tag{1}
\]

Biologically speaking, this equation may be understood as the net current passing through the cell membrane being made up of a charging current across the capacitance of the membrane itself, plus an ionic current component stemming from ionic charge carriers crossing the membrane. This may be summarized according to equation 2.

\[
I_{total} = I_{capacitive} + I_{ionic} \tag{2}
\]

The general Hodgkin-Huxley equation is fairly straightforward to understand, in that it follows basic circuit principles. The parameters \(g_k\) and \(g_{Na}\) are the maximum conductances of the potassium and sodium channels, respectively, across the cell membrane, determined experimentally by Hodgkin and Huxley. Because this model only explicitly addresses potassium and sodium as current conducting ions, the parameter \(g_l\) was introduced to incorporate all other leakage conductances across the membrane. It will be seen later that the current through the
potassium and sodium channels are in fact quite complex, while the leakage current is characterized by only three parameters: the leakage conductance, $g_L$, the leakage threshold voltage, $V_L$, both of which are constants, as well as the instantaneous potential across the membrane, $V$, which varies with time. The leakage parameters were selected by Hodgkin and Huxley to enable the entire system to approach the correct equilibrium point that matches with the observed equilibrium point in the physical experiments. In other words, the leakage current term adjusts for the offsets between the mathematical and observed membrane potential.

The parameter $C_M$ represents the capacitance of the cell membrane, and the other parameters of the system were adjusted by Hodgkin and Huxley to allow this capacitance to become unity. As the capacitance is effectively just a scaling term, this does not affect the general characteristics of the solution. The remaining parameters of the general equation, $m$, $n$, and $h$, are denoted ‘gating parameters’, and represent much of the complexity of the model, in that these parameters themselves are determined by additional differential equations, defined in general form by Equations 3, 4, and 5.

\[
\frac{dm}{dt} = \alpha_m (1-m) - \beta_m m
\]
\[
\frac{dn}{dt} = \alpha_n (1-n) - \beta_n n
\]
\[
\frac{dh}{dt} = \alpha_h (1-h) - \beta_h h
\]

Note that each of the gating parameters is defined in the same general form, however each is further characterized by two more parameters $\alpha$ and $\beta$, which are functions of the instantaneous potential across the cell membrane. The equations for these parameters are also determined experimentally by Hodgkin and Huxley and are defined by equations 6 to 11.

\[
\alpha_n = \frac{(0.1 - 0.01V)/(e^{(1.0V)} - 1)}
\]
\[
\beta_n = 0.125e^{V/80}
\]
\[
\alpha_m = (2.5 - 0.1V)/(e^{(2.5-0.1V)} - 1)
\]
\[
\beta_m = 4e^{-V/18}
\]
\[
\alpha_h = 0.07e^{V/20}
\]
\[
\beta_h = 1/(e^{(3.0-0.1V)} +1)
\]

More components of Hodgkin and Huxley’s equations, necessary for obtaining a solution, are the values of the constant parameters, experimentally obtained by Hodgkin and Huxley. The constant parameters of the problem are the membrane capacitance, the ionic channel threshold voltages, and the ionic channel conductances. Once again, note that the values for the leakage parameters were chosen by Hodgkin and Huxley to have a resting ionic current of zero and a resting membrane potential of 0 (mV). The values for these parameters used for this simulation are summarized below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_M$</td>
<td>1.0µF/cm²</td>
</tr>
<tr>
<td>$V_{Na}$</td>
<td>115mV</td>
</tr>
<tr>
<td>$V_K$</td>
<td>-12mV</td>
</tr>
<tr>
<td>$V_L$</td>
<td>10.613mV</td>
</tr>
<tr>
<td>$g_{Na}$</td>
<td>120mS/cm²</td>
</tr>
<tr>
<td>$g_K$</td>
<td>36mS/cm²</td>
</tr>
<tr>
<td>$g_L$</td>
<td>0.3mS/cm²</td>
</tr>
</tbody>
</table>

It should be noted that there are various publications that summarize Hodgkin Huxley’s equations, and offer their own solutions and parameter values [7]. These parameters differ in terms of units, or the adjustment for the leakage current and resting potential. When different values for the parameters are presented, the equations are also altered to account for the different parameters. The end result of the simulation should be the same, regardless of which representation of Hodgkin-Huxley’s equations is used, and the only real differences are how the equations are presented. For this project the equations presented are as in [1].

To summarize this background section, the ionic currents in Hodgkin-Huxley’s general equation (equation 1) are determined by time varying gating parameters, and the rates of change of these parameters are in turn determined by the present membrane potential. However, equation 1 also contains a capacitive current term, which is dependent on the rate of change of the membrane potential. Thus, it may be seen that equation 1 actually represents a differential equation relating the instantaneous membrane potential to the change in membrane potential. With this knowledge, it is possible to use numerical techniques to solve for the membrane function over time.

III. INTEGRATION METHODS

A. Euler’s method

Euler's method is the most basic of the explicit methods for solving differential equations through numerical integration [4]. The basic theory behind Euler's method is quite simple. Given an initial value which satisfies a differential equation, the given equation may be used to determine the instantaneous rate of change of the desired function at that point. Then, using a chosen step size and that rate of change,
the next value for the function may be approximated. As the step size approaches zero, this approaches a perfect integration, and the differential equation may be solved exactly. The step size for this method must be chosen to be small enough to allow convergence of the calculated solution with the true solution. Mathematically, Euler's method is shown below. Given a differential equation with an initial value, as shown below, and a step size \( h \), the solution of the problem may be found iteratively using Equation 15.

\[
y' = f(t, y(t)) \quad y(t_0) = y_0
\]

\[
y_{n+1} = y_n + hf(t_n, y_n)
\]  

(15)

**B. 4th order Runge Kutta's method**

The Runge Kutta method used in this project is the most common method for solving initial value problems in the Runge Kutta family of methods [Chapra 2008]. In fact, the Euler's method described earlier is also known as a first order Runge Kutta method. For a given initial value problem specified by the following conditions, may be solved numerically using Equations 16 and 17, where the values for \( k_1, k_2, k_3, \) and \( k_4 \) are specified by Equations 18 through 21.

\[
y' = f(t, y(t)) \quad y(t_0) = y_0
\]

\[
y_{n+1} = y_n + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4)
\]

(16)

\[
t_{n+1} = t_n + h
\]  

(17)

\[
k_1 = h * f(t_n, y_n)
\]

(18)

\[
k_2 = h * f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1)
\]

(19)

\[
k_3 = h * f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2)
\]

(20)

\[
k_4 = h * f(t_n + h, y_n + k_3)
\]

(21)

Essentially, the 4th order Runge Kutta method works in the same way as the Euler method, except that the slope used to determine the next value of the function under investigation is taken as a weighted average of the slopes across the step interval \( h \). In theory, this enables the function to be approximated more accurately, albeit at the cost of more computing time. In this paper, the advantages and disadvantages of using each method will be investigated, based on the processing time each technique utilizes, as well as the mathematically determined order of the error.

All coding for this project was done in Java, using the Java Swing library, as well as the open source FreeChart library, for graphical interfacing. Presented in the following sections are the main themes of the project: observations of the action potential characteristics using the Hodgkin Huxley model, a comparison of the two numerical techniques used to obtain these results, and a preliminary investigation of multiple neuron interactions.

The graphical user interface developed for the initial system analysis is shown in figure 1. The type of input stimulation may be selected on this GUI, as well as the integration method desired. Each possible input has a separate configuration screen, where initial parameters are set.

![Figure 1: GUI for Single Neuron Analysis](image1)

A pulse input indicates an input in the form of regular set of pulses, where the duration of each pulse and time between them may be altered. A step input indicates an input current as a Heaviside step function. A Gaussian input current consists of pseudo-random numbers generated by the program which follow a Gaussian distribution. The shape of the distribution depends on the mean and the variance of the input current which may be set to a desired level.

![Figure 2: GUI for Configuration of a Pulse Input Signal](image2)
variables of the selected input for the simulation. The screen displayed is for pulse input stimulation.

**Step input**

The basic synapse model is best displayed visually using a step input signal. Depending on the size of the step input, as well as the final value of the input current, three cases are possible. The input could produce no spike in the neuron, a single spike, or a chain of repeating spikes.

The expected result was verified using the step input signal definition in the GUI developed for this project. A typical test, yielding a single spike, may be seen in Figure 3.

![Image of neuron response to step input](image1)

**Figure 3:** Neuron Response to step from 0 to 3 (µA/cm²) – Single Spike

Also portrayed in each simulation case are the variations of the gating parameters \( m, n, \) and \( h \), with time. Visually showing each of the gating parameters along with the membrane potential output effectively demonstrates the role of these parameters in producing action potentials, and the interrelations present between each parameter.

If the step input is altered to begin at 0 and step to 10(µA/cm²), the result is a train of repeated spikes, as shown in Figure 4.

![Image of neuron response to step input](image2)

**Figure 4:** Neuron Response to step from 0 to 10 (µA/cm²) – Train of Spikes

In figure 5, the dependency on the rate of change of the input, and not just the value of the input, is clearly demonstrated. With amplitude that is the same as in Figure 3, we can see that, in the case of a pulsing input, a repeated spike train is generated, with a spike occurring at each pulse. In the step input with this amplitude (Figure 3), only one spike occurred, because once the spike happens, the membrane potential approaches a new resting potential, which in turn has a new threshold level for generating spikes. Because the steady state potential for this system is equal to the new resting potential, and thus below the new threshold potential, only one spike is generated.

![Image of neuron response to pulse input](image3)

**Figure 5:** Neuron response to Pulse Input with Amplitude = 3(µA/cm²), Pulse Duration = 10 (ms) and period = 20 (ms)

However, for a pulse input (Figure 5), once the pulse has generated a spike, the level goes back to the original resting

**Pulse Input**

To demonstrate other characteristics of Hodgkin-Huxley model we may use a series of regular pulses as input. In these tests the amplitude of the pulse, the duration of the pulse, and the period of the pulses (measured from the beginning of one pulse to the beginning of the next) are changed to demonstrate two characteristics of the synapse: membrane potential buildup and the refractory period.

It may be seen that the generation of a spike is dependent on the change in input stimulus, as well as the final value of the stimulus. In this paper the joint dependency on the size of the step and the duration of this step shall be referred to as membrane potential buildup, or just potential buildup.

In essence, this term encompasses the idea that the spike is generated by a buildup of voltage across the capacitance of the cell membrane, and as such both the level and duration of the input stimulus play a role in determining whether a spike will be created or not.
potential of zero, giving the neuron time to release the potential buildup and realize its original state, so that the next pulse generates the same response. The result is a continuous series of spikes, separated regularly by the period of the pulses.

IV. REFRACTORINESS

The refractory periods of the Hodgkin Huxley model neurons may be demonstrated by reducing the time between the input pulses. Figure 5 may be used as a reference point where no refractoriness is displayed, because there is a 10 (ms) delay between the end of one pulse and the beginning of the next.

In Figure 6, the refractory period inherent in the spiking neuron model may be clearly seen. The 10 (ms) delay between pulses from Figure 6 was reduced to 3 (ms) in Figure 6, and the result of this is that only a single spike is generated in the simulation. The reason for this is that 3 (ms) is not enough time to recover the resting state of the system, and as such applying a jump in current input is no longer enough to overcome the threshold limits for spiking after only 3 (ms) of recovery time.

Gaussian Input

An algorithm denoted Gaussian Input in this paper is used to simulate a noisy input, creating a pseudo-random number every 0.01 (ms), which follows a normal distribution based on a user input mean and variance, and uses this generated number as the input stimulation to the neuron. This is effective in demonstrating that spike generation increases with a larger mean input stimulus, that spike generation increases with larger variance of the input stimulus, and also reinforces the concept of the refractory periods of synapses. Each of these points has been touched on before, but a demonstration with noisy input signals can reinforce the point.

In Figure 7, a typical example of a spiking response to a Gaussian input may be seen. In this particular example, a mean input level of 8(µA/cm²) and a variance of 10(µA/cm²) were used for the simulation.

Notably with Figure 7, the refractory period of these neurons may be clearly seen. Once the input signal becomes large enough to generate repeated spikes, there is still a regularly spaced period between each of these spikes. Because the input signal does not follow any set pattern, it may be inferred that any regular spacing of the spikes is due to the refractory period inherent in the model.

V. COMPARISON BETWEEN EULER AND RUNGE KUTTA METHOD

To offer a basic comparison between the use of the Runge Kutta method and Euler's method, the same input signal was analyzed for each method, and the computational time used by each method was compared. For a step input from 0 to 10 (µA/cm²) at 20 (ms), with a simulation time of 100 (ms), 1000 iterations of the calculations were performed by each method, with the Runge Kutta method taking 38.105 (s), and the Euler method taking 41.230 (s). This implies a run time of 38.105(ms) for each iteration with the Runge Kutta method, and 41.230(ms) for each iteration with the Euler method. A second trial was performed, using a pulse input with a stimulus level of 10 (µA/cm²), a pulse duration of 20 (ms), a period of 20 (ms), and a simulation time of 100 (ms). The Runge Kutta performed 1000 iterations of this calculation in 39.584(s), while the Euler method took 39.522 (s) to perform the same number of iterations.

While these numbers are not very useful in determining which method is better to use, it should be noted that, mathematically, the error of the Runge Kutta method is on the order of $h^4$ [4], while the error for the Euler method is on the order of $h$. Thus, if the computational times are similar, as found in this project, the Runge Kutta's method is a more accurate way to determine a numerical solution, and should be used if possible. While analysis of the computation time and error of each method may determine which method is
best for this particular project, the computation time itself is too large to expect real time simulation of neurons, and thus indicates that neither method is necessarily fast enough. Thus, when discussion turns to linking these neuron models into networks, neither of the methods used here will suffice to perform the calculations in real time.

VI. MULTIPLE NEURONS

Until now only the modeling of a single neuron has been accomplished. However, it is known that the brain contains billions of neurons interacting with each other. Therefore, in the course of this project, a simple approach to modeling multiple neuron interactions in a linear network was undertaken. However, due to the aforementioned excessive computing time required to simulate a single neuron, it was only possible to simulate a chain series of ten neurons before the computing time became too great to generate any useful data. The modified GUI for multiple neuron analysis may be seen in figure 8.

![GUI for Multiple Neuron Analysis](image)

Fig. 8: GUI for Multiple Neuron Analysis

A model that can be used to describe the interactions between many neurons is the integrate-and-fire model [1]. The integrate-and-fire model differs from the Hodgkin Huxley model in that it satisfies accuracy and completeness in a both simple and computationally fast manner, allowing many neurons to interact, and even cross-couple, to create increasingly complex biological models. Also known as the leaky integrate-and-fire model, it models a neuron as a simple parallel RC circuit, charged by an input current pulse.

A threshold potential is introduced manually, and a resting potential may be defined, which \( V(t) \) is set to, after it reaches the threshold potential. In its simplest form, the response of this circuit resembles a saw-tooth wave with a time constant, for a constant input current. Once the threshold potential is exceeded, an output pulse is generated and the voltage on that neuron is set back to zero, or the defined resting potential. This model may be configured to demonstrate various response types for various input signals. Its flexibility and simplicity makes it a desirable approach to modeling systems of neurons, while the Hodgkin Huxley equations are desirable for accurately displaying real neuron characteristics. Integrate-and-fire models may also be modified to demonstrate the absolute refractory period shown by real synapses, as well as by Hodgkin and Huxley's model, by simply adding a delay period into the equations used in the model.

VII. CONCLUSION

From the simulations run in this project, it may be inferred that the generation of spiking action potentials in neurons depends on three major components: the level of input stimulation, the change in the level of input stimulation, and the refractory periods of the neurons themselves. Larger input stimulation levels, as well as larger rates of change in the input stimulations, both create more action potentials. Absolute refractory periods limit the minimum time between spiking, and make it impossible to initiate a second action potential before the first action potential has finished. Simulation of multiple neuron interactions was attempted, however the model used proved to be computationally inefficient, and therefore unable to generate useful data. It remains to study simulations of many neurons interacting based on the integrate-and-fire model. Such a model has not yet been implemented in Java.

VIII. FURTHER WORK

Ideally, this type of spiking network would present an alternative to the non-spiking artificial neural networks used most often in modern cognitive computing, offering increased accuracy in modeling a ‘real’ cognitive system using real time processing. The results of this project are a long way from providing a new system with which to perform such calculations.

However, this project provides a tool for increasing understanding of the biological neuron processes using a graphical interface, which is an important step towards realizing the full potential of cognitive computing. In the future it would be crucial to optimize the processing power used to perform the computations in this project, and use the increased time efficiency to implement the interactions of more neurons in a single system. Of course, the end goal would be to concatenate these neurons into a vast network, capable of emulating the human brain, but the methods for modeling these interactions used in this project are in themselves far too computationally exhaustive, and therefore slow, to be replicated on the order of billions of neurons, while still retaining any expectations that the system will perform in a reasonable representation of real time.

To take this project a step further, a thorough investigation would need to be conducted of each method available, and a simple yet accurate method of simulating the analog interactions of the neuron within a digital system would need to be implemented. This would hopefully enable the vast cognitive computing of the human brain to be simulated in the digital world. Advances have been made in the use of analog technologies to emulate the analog nature of the
neuron, which have been quite promising [3], however the space efficiency of digital technology still makes a digital simulation of neural processes desirable.

REFERENCES


