

The Hodgkin-Huxley Model

Its Extensions, Analysis and Numerics

Ryan Siciliano

6/3/2012

CONTENTS

Abstract	4
The Hodgkin-Huxley Model	5
Background Information	5
Anatomy of the Neuron	6
Action Potentials	7
Mathematical Model	8
Ionic Current.....	9
Voltage Clamp Experiments.....	11
Rate Constants	13
Corrections, Modifications and Extensions.....	14
Numerical Methods for Solving Ordinary Differential Equations	15
Euler Methods.....	15
Forward Euler Method.....	15
Modified Euler Method.....	16
Backward Euler Method	16
Runge-Kutta Method	16
Adams-Bashforth-Moulton Predictor-Corrector Method.....	17
Matlab ODE Built-In Function (ODE45)	17
Matlab Solutions using Numerical Methods	18
Euler Methods.....	18
Runge-Kutta Method	19
Adams-Bashforth-Moulton Predictor-Corrector Method.....	19
ODE45	20
Accuracy of Methods	20
Error.....	20
Order of the Method.....	21
Analysis of Hodgkin-Huxley Model.....	22
Solving Hodgkin-Huxley Differential Equation	22
Results	24

Exact Solution and Comparison of Methods	25
Personal Remarks	27
Conclusion	27
Acknowledgements	29
Works Cited	30
Appendix - Matlab Codes.....	32
1. Arbitrary ODE	32
1.1 Euler Methods	32
1.2 Runge-Kutta.....	33
1.3 Predictor-Corrector	34
1.4 ODE45.....	35
1.5 Order of the Method	35
2. Hodgkin Huxley Model	36
2.1 Comparison of Methods.....	36
2.2 HH function	40
2.3 Alpha and Beta Functions.....	41

ABSTRACT

A mathematical model for the initiation and propagation of an action potential in a neuron was named after its creators in 1952. Since then, the Hodgkin-Huxley model has been used vastly in the world of physiology. This project begins by introducing the background physiology of the model's origin. The model's derivation is given along with all supporting functions and constants.

The main argument of this project is to complete and compare different numerical methods to solve the Hodgkin-Huxley model. Six different numerical methods are first introduced and compared using a simple and arbitrary ordinary differential equation. The numerical methods used are: forward Euler, modified Euler, backward Euler, Runge-Kutta, Adams-Bashforth-Moulton predictor-corrector, and Matlab's ODE45 function. After initial analysis of the error compared to an exact solution, the abovementioned methods are listed in order of increasing accuracy. The order of each method was also calculated to compare speed.

The most significant result was creating Matlab code to solve the Hodgkin-Huxley model for each different numerical method. Each solution to the model is plotted to visually compare the differences. The methods were also statistically compared to the exact solution by setting the sodium and potassium conductances to zero. The action potential obtained from Matlab will be analyzed both physiologically and mathematically.

THE HODGKIN-HUXLEY MODEL

BACKGROUND INFORMATION

The Journal of physiology presented a series of papers in 1952 that would forever change the relationship between mathematics and physiology. Alan Lloyd Hodgkin and Andrew Huxley authored a succession of five papers describing the nonlinear ordinary differential equations that model how action potentials can be initiated and propagated through an axon [1-5]. This project will touch on the first paper in the series to give a background identity to the differential equation and its parameters and focus mainly on the fifth article which describes the mathematical model to be analyzed in future sections.

In 1963, the Nobel Prize for physiology and medicine was awarded to Hodgkin and Huxley for their ground-breaking research on the squid giant axon. The squid, with its tremendously large axon (up to 1mm diameter) and minimal conductance [6], allowed the pair to perform many experiments that have wide applications across organisms.

Cole and Curtis in 1939 were the first to identify the increase in membrane conductance during an action potential [7] following from a study in 1902 by Bernstein to correctly show the membrane separating two solutions of different ionic composition [8]. Bernstein studied the ionic composition of the solutions surrounding the membrane and proved there is a much greater concentration of potassium within the cell and an increased level of sodium outside. He used this information to conclude that the resting membrane potential is close to the potassium equilibrium value during rest, but rose to zero during activity. Bernstein reasoned that the activity caused the membrane to degrade, but this was proved incorrect in 1938.

During the year of 1938, Hodgkin collaborated with Cole to directly measure the membrane voltage [9]. This experiment was successful and they noticed that during activity the membrane potential will rise from the resting value (around -65mV) to well exceed the proposed value of zero (by Bernstein [8]) and

almost reverse sign at the peak (Figure 1). Hodgkin and Cole were able to attribute this rise (action potential upstroke) to the changing permeability of different ions.

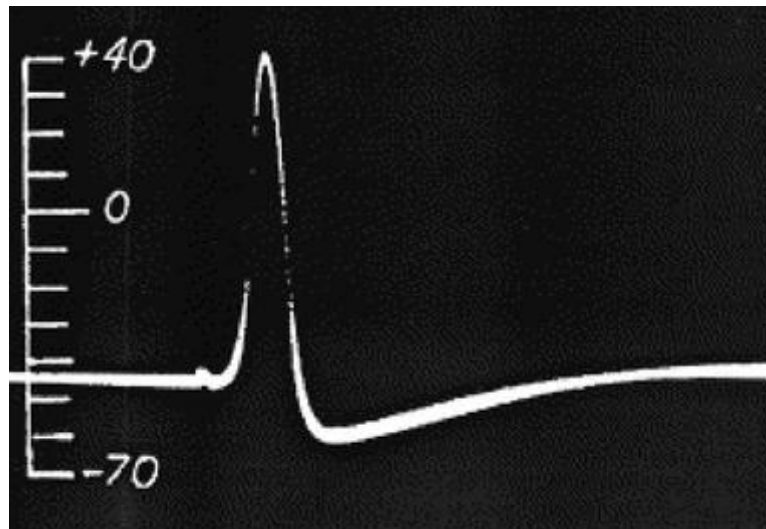


FIGURE 1 THE ORIGINAL DEPICTION OF A SINGLE ACTION POTENTIAL OVER TIME [9]

To further analyze the membrane changes, the voltage clamp experiment was created. Two pairs of electrodes read the voltage drop, while another pair injects current to clamp the voltage to a specific level. The changes in permeability and the dynamic link to membrane voltage were finally addressed and modeled in 1952 by Hodgkin and Huxley [1-5].

ANATOMY OF THE NEURON

Neurons receive signals through dendrites located on the main bulk of the cell. From there, the signal is propagated through the body and sent along the axon to the next neuron (Figure 2). Neurons send signals to other neurons via action potentials. An action potential is an explosion of depolarizing current that travels along the cell. For an action potential to occur, the depolarization must reach a minimum threshold voltage. Action potentials are only fired as an 'all-or-none' response. This means that action potentials do not vary in size and will not occur if threshold is not reached.

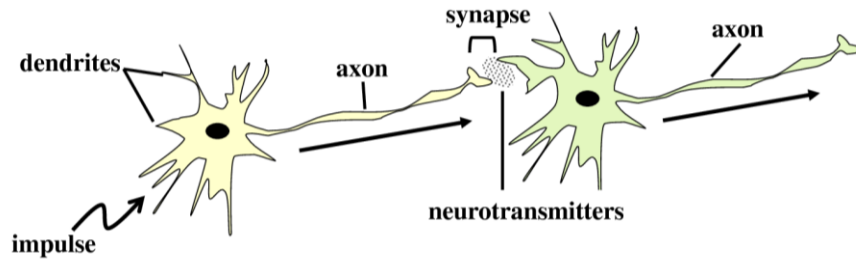


FIGURE 2 TWO NEURONS SIDE BY SIDE SHOWING THE PATH OF AN ACTION POTENTIAL. THE IMPULSE IS RECEIVED BY THE DENDRITES CLOSE TO THE BODY. IT THEN MAKES ITS WAY TO THE AXON WHERE IT TRAVELS TO THE NEXT NEURON OR ENDPOINT. THE SYNAPSE IS A LOCATION OF RELEASE OF NEUROTRANSMITTERS TO BEGIN THE ACTION POTENTIAL AT THE NEXT NEURON. TAKEN FROM A PUBLIC SOURCE.

ACTION POTENTIALS

Action potentials occur in excitable cells including neurons, muscle cells, and endocrine cells. In the case of neurons this is a signal that is propagated to relay information. The stepwise process of an action potential is described below.

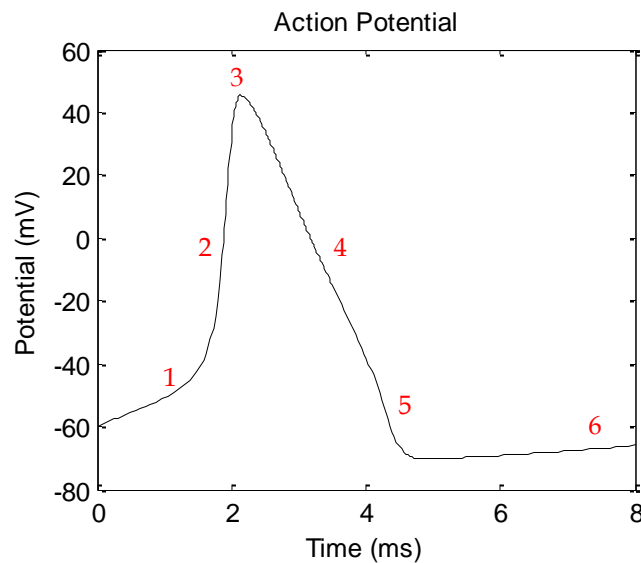


FIGURE 3 THIS IS AN ACTION POTENTIAL CREATED FROM THE MATLAB CODE PRODUCED LATER IN THIS PROJECT. EACH TIME POINT OF THE CURVE IS EXPLAINED IN THE TEXT BELOW.

Figure 3 above is a copy of the action potential created using the Matlab program to solve the Hodgkin-Huxley equation. Each part of the graph will be broken down according to the number sequence:

1. At this point in time, the firing threshold has been reached and the action potential begins to fire. The sodium channels are now open and an influx of sodium occurs.
2. The membrane becomes more positive at this point due to sodium entering the cell. Now, the potassium channels open and potassium leaves the cell.
3. At the peak of the action potential, the sodium channels have now become refractory and no more sodium is allowed to enter.
4. During the downfall phase of the action potential, the sodium channel is still in its refractory stage and only the potassium ions are passing through the membrane. As the potassium ions continue to leave the cell, the membrane potential moves toward the resting value.
5. At this point, the potassium channels close and the sodium channels begin to leave the refractory phase and reset to its resting phase.
6. The diffusion of extracellular potassium away from the cell causes a very slight increase in membrane voltage. It finally returns to its resting value where it can await another action potential. While the membrane is hyperpolarized (below resting) the cell cannot fire. This prevents the action potential from travelling backwards.

MATHEMATICAL MODEL

The Hodgkin-Huxley model is based on the parallel thought of a simple circuit with batteries, resistors and capacitors. A basic model of this circuit is shown in Figure 4. Current can be carried through the circuit as ions passing through the membrane (resistors) or by charging the capacitors of the membrane [5].

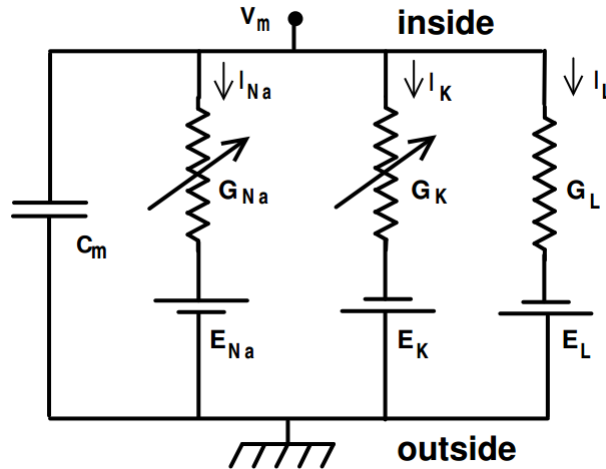


FIGURE 4 THIS FIGURE SHOWS THE MODEL OF THE CELL AND ITS FLOW OF CURRENT IN A SIMPLE CIRCUIT DIAGRAM. THE CAPACITOR OF THE MEMBRANE IS SHOWN ON THE FAR LEFT AND THE INDIVIDUAL IONIC CURRENT IS SEEN TO TRAVEL DOWN ITS RESPECTIVE BRANCH[5].

The electrical activity depicted in Figure 4 can be summarized as the famous Hodgkin-Huxley model:

$$C_m \frac{dV_m}{dt} + I_{ion} = I_{ext},$$

where C_m is the membrane capacitance, V_m is the intracellular potential, t is the time, I_{ion} is the net ionic current flowing across the membrane, and I_{ext} is the externally applied current [5]. The most significant advantage of the above equations is to determine the membrane capacitance in such a way that is independent of the sign or magnitude of the intracellular potential and minimally affected by the time course of V_m .

IONIC CURRENT

The ionic current described in Hodgkin and Huxley's giant squid axon can be divided into three different cases: sodium current, potassium current, and the leakage current [5]. The movement of each of these currents is proportional to the conductance times driving force [5]. For example, $I_k = G_k(V_m - E_k)$.

Conductance can easily be described by using the notion of the membrane channel. However, in 1952 when Hodgkin and Huxley were developing their model, there was very little knowledge about these ion

channels. The initial findings of the model properties still hold true for the additional information acquired about ion channels. For convenience, the 1952's Hodgkin Huxley model will be explained using today's knowledge of membrane ion channels.

The conductance is caused by the opening of many microscopic channels in the membrane. Each individual membrane contains many gates and when all gates are in the *permissive* state (allows ions to pass through) then the channel is considered to be *open* [6]. Once a channel is open, all ions of the specific channel can flow freely through. The probability of a gate being in the permissive state depends on the current value of the membrane voltage.

To develop the differential equations that describe the conductance, the probability of a gate being open is defined as p_i for any ion, i . Looking at the macroscopic scale, p_i can also be the fraction of gates that are in the permissive state. Also, the amount of gates closed are $(1 - p_i)$, by definition. To obey the first-order kinetics [6],

$$\frac{dp_i}{dt} = \alpha_i(V)(1 - p_i) - \beta_i(V)p_i,$$

where, α_i and β_i are the rate constants dependent on the voltage that describe the transient rates of permissive and non-permissive gates.

All gates must be permissive simultaneously for a channel to be considered open. When this occurs, the channel will contribute a miniscule amount of conductance on a large scale, otherwise it will not contribute anything. The macroscopic conductance depends on many channels being open, which depends on an even greater amount of gates being in the permissive state. This leads to the equation,

$$G_i = \bar{g}_i \prod_i p_i,$$

where, \bar{g}_i is the normalized constant that determines the maximum conductance when all the channels are open [5, 6].

Hodgkin and Huxley labeled the probability of a gate being in the permissive state as the name of the respective gate. To conform to this method, p_i is exchanged for the name of the gates - m , n , or h . Each channel is defined to have a specific amount of these gates. For example, the sodium channel has three m gates and one h gate [5]. Using this notation and the above theory, the ionic currents can be summarized by,

$$I_{ion} = \bar{g}_{Na}m^3h(V_m - E_{Na}) - \bar{g}_Kn^4(V_m - E_K) - \bar{g}_l(V_m - E_l),$$

$$\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n,$$

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m,$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h.$$

To completely describe the Hodgkin Huxley model, the only thing that remains is to define the six rate constants.

VOLTAGE CLAMP EXPERIMENTS

This section will describe the voltage clamp experiment for the potassium channels. Both the leakage and sodium channels follow the same methods to obtain similar results.

In the fifth paper of the series by Hodgkin and Huxley, they discovered that as they kept the voltage constant, the conductance of the channel increased to a steady value dependant on the initial voltage [5]. Their most significant finding was that the rate at which the conductance reached its maximum depended greatly on the clamped voltage. More specifically, at 20mV the half-maximum value occurs after 5 ms, but at 100mV the half-maximum value is reached after only 2 ms [5].

Hodgkin and Huxley were then able to find the time course of the step value n in the circumstance where the voltage is increased by a certain increment after the steady-state is achieved. Initially, the resting value of n is

$$n_{\infty}(0) = \frac{\alpha_n(0)}{\alpha_n(0) + \beta_n(0)}.$$

But as the voltage is clamped to a different voltage, V_c , the steady state becomes

$$n_{\infty}(V_c) = \frac{\alpha_n(V_c)}{\alpha_n(V_c) + \beta_n(V_c)}.$$

A solution can then be found mathematically to solve $\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n$ with the above restraints. This simple solution is an exponential of the form [6]

$$n(t) = n_{\infty}(V_c) - (n_{\infty}(V_c) - n_{\infty}(0))e^{-\frac{t}{\tau_n}},$$

where [6]

$$\tau_n(V_c) = \frac{1}{\alpha_n(V_c) + \beta_n(V_c)}.$$

After all this information, Hodgkin and Huxley still needed to determine the proper order and type of gates for each channel. They used a trial-and-error method to find the proper order of the gates to achieve the sigmoidal conductance found from previous results. Since the value of n is bound between zero and one, it first needed to be multiplied by the normalization constant \bar{g}_K . In the case of potassium, the best match is the n value brought to the power of four, such that

$$G_K = \bar{g}_K n^4.$$

The conductance for the sodium and leakage channels was obtained by the voltage clamp in the exact same manner. The results from this section help to form the desired rate constants.

RATE CONSTANTS

The values of $n_\infty(V_c)$ and $\tau_n(V_c)$ obtained above by fitting the conductance data will allow $\alpha_n(V_c)$ and $\beta_n(V_c)$ to be found by using the following relationships [5, 6]:

$$\alpha_n(V) = \frac{n_\infty(V)}{\tau_n(V)},$$

$$\beta_n(V) = \frac{1 - n_\infty(V)}{\tau_n(V)}.$$

In Figure 5, the plotted circles represent the experimentally acquired values of $n_\infty(V_c)$, $\tau_n(V_c)$, $\alpha_n(V_c)$, and $\beta_n(V_c)$ as a function of voltage. Hodgkin and Huxley then determined curves that fit this data to give the following expressions for the rate constants:

$$\alpha_n(V) = \frac{0.01(10 - V)}{\exp\left(\frac{10 - V}{10}\right) - 1},$$

$$\beta_n(V) = 0.125 \exp\left(\frac{-V}{80}\right).$$

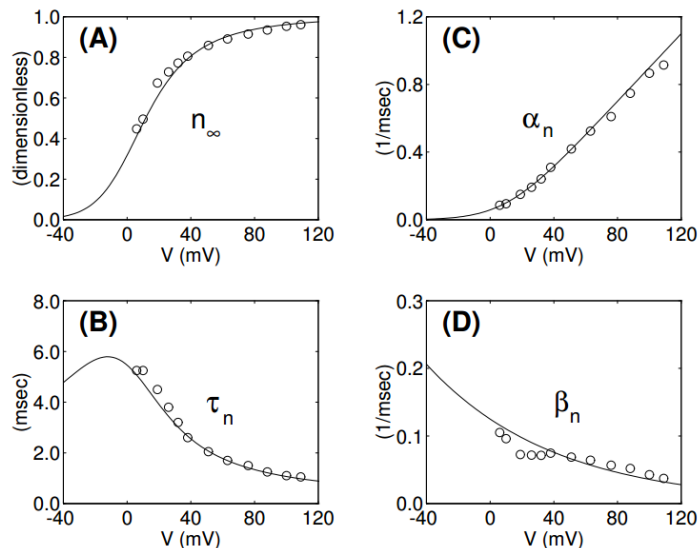


FIGURE 5 THE ASSOCIATED FUNCTION WITH THE N GATING VARIABLE. THE CIRCLES SHOW THE ACTUAL RECORDED VALUES WITH THE MODELED EQUATION AS THE SOLID LINE. (A) THE VALE OF N OVER 120MS. THE RATIO IS ALWAYS BETWEEN 0 AND 1. (B) THE TAU TIME VARIABLE. (C) SHOWS HOW ALPHA OF N CHANGES OVER TIME. (D) THE BETA VALUE OF N. [6]

To complete all the associated differential equations needed for the Hodgkin Huxley model the above methods are used to find the rate constants for the other gates. Thus,

$$\alpha_m(V) = \frac{0.1(25 - V)}{\exp\left(\frac{25 - V}{10}\right) - 1}$$

$$\alpha_h(V) = 0.07 \exp\left(\frac{-V}{20}\right)$$

$$\beta_m(V) = 4.0 \exp\left(\frac{-V}{18}\right)$$

$$\beta_h(V) = \frac{1}{\exp\left(\frac{30 - V}{10}\right) + 1}$$

That concludes all the necessary equations and constants that define the Hodgkin and Huxley model. The following sections deal with the different numerical methods used to solve the equations above.

CORRECTIONS, MODIFICATIONS AND EXTENSIONS

It has been sixty years since Hodgkin and Huxley made their ground-breaking discovery with minimal technology and background information. Since then, many have come forward to offer corrections and modifications to the model. There are many slight modifications to the original model to adjust to a different cell and different organism, but most follow the same original properties. For the remainder of this project, the original results will be respected, but listed below are some of the most common and obvious additions.

Hodgkin and Huxley showed the firing of an action potential with type 2 excitability. Type 2 excitability is sustained firing of action potentials from suprathreshold depolarizing current [10]. In 2008, John Clay et al. were able to separate this theoretical approach from experimental results by introducing a small modification to allow modeling of type 3 excitability. A simple modification provided in their 2008 paper slightly adjusts the potassium current [10].

There is an enormous amount of extensions to the Hodgkin-Huxley equation. Some examples are: bifurcations in the Hodgkin-Huxley model [11], speed dependent on frequency [12], and no external signal [13]. Many modifications to the model can be found by tweaking the results to match a different organism or different cell within an organism.

The most challenging article confronting the Hodgkin-Huxley equation is an article entitled 'Playing the Devil's advocate: is the Hodgkin-Huxley model useful?'. This article neatly points out that although there are many other models with the same function, for the most part they all incorporate basics of the Hodgkin-Huxley model [14]. The ideal and universal equation is a marriage between phenomenological equations and Hodgkin-Huxley-like models concluding that the Hodgkin-Huxley model will remain in the future and has earned its place in today's day and age [14].

Despite the numerous efforts to challenge, change and correct the original Hodgkin-Huxley model, it has a firm and factual heritage that will merit its existence for time to come.

NUMERICAL METHODS FOR SOLVING ORDINARY DIFFERENTIAL EQUATIONS

EULER METHODS

The Euler method is the simplest numerical integration method to solve first order ordinary differential equations. In practice, the Euler method is very quick to compute but prone to instability and inaccuracy. The results of this method will be compared to other numerical integration techniques in a future chapter.

FORWARD EULER METHOD

To find the shape of a curve that has a given starting point and satisfies a certain differential equation, the Euler method will calculate the slope of a given point once that point has been calculated. If the actual curve starts at the initial point, a_0 , that is given, the slope can be calculated at that point. Assuming that the next point, a_1 , is extremely close to a_0 it can be found on the tangential line. From a_1 a new slope and tangential line can be calculated leading to the arrival of the next point along the second tangential line called a_2 [15]. This iteration is completed over time to form a curve that will closely model the actual curve.

The differential equation $y'(t) = f(t, y(t))$ where the initial condition is $y(t_0) = y_0$ and the value of t_n is defined to be $t_0 + n \cdot h$, where h is the size of the step, can be solved at t_n using the Euler method. The Euler method yields the result $y_{n+1} = y_n + h \cdot f(t_n, y_n)$ [16].

MODIFIED EULER METHOD

The modified Euler method is very similar to the forward Euler method but it uses the trapezoidal rule to find the solution. The trapezoidal rule is an approximation of the integral from a to b as trapezoid, no matter the actual path taken. Instead of looking backwards to find the next point, this solution uses a combination of the past and current points to determine a solution. It is important to note that $f(t_n, y_n)$ is unknown and must be solved for implicitly. So the modified Euler estimation is given by $y_{n+1} = y_n + \frac{h}{2} [f(t_{n+1}, y_{n+1}) + f(t_n, y_n)]$.

BACKWARD EULER METHOD

The backward Euler method only uses the current value of the point (i.e. $f(t_n, y_n)$). The backward Euler method is given by $y_{n+1} = y_n + h \cdot f(t_{n+1}, y_{n+1})$.

RUNGE-KUTTA METHOD

The most commonly used Runge-Kutta method is the fourth-order Runge-Kutta method or simply "RK4". Given a differential equation $y'(t) = f(t, y(t))$ where the initial condition is $y(t_0) = y_0$, the RK4 method yields $y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ and $t_{n+1} = t_n + h$ [17]. y_{n+1} is the RK4 approximation where

$$k_1 = h \cdot f(t_n, y_n), k_2 = h \cdot f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right), k_3 = h \cdot f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right), k_4 = h \cdot f(t_n + h, y_n + k_3) \quad .$$

The next point on the curve is determined by the previous point in addition to the weighted averages of four different increments. The first increment, k_1 , is the Euler increment seen above. It takes into consideration the slope of the tangent of the previous point with the step size. The second increment is based on the slope of the tangent at the center of the step. The third increment is the slope in the middle,

again, with respect to the second increment. Finally, the fourth increment is the slope at the end of the step.

ADAMS-BASHFORTH-MOULTON PREDICTOR-CORRECTOR METHOD

The methods previously discussed are all *single-step* methods because they only depend on one previous value. The predictor-corrector method depends on several preceding values. For example, the Adams predictor-corrector method depends on $y_{n-3}, y_{n-2}, y_{n-1}, y_n$ to get the value of y_{n+1} . Unless the first four values are given, the next value of the curve cannot be found. This means that the predictor-corrector methods are not self-starting. Fortunately, any of the above single-step methods can be used to attain the first four values and then the predictor-corrector method can be used henceforth.

The Adams-Bashforth-Moulton fourth-order method will be discussed to represent the predictor-corrector method. This method yields the following formula: $y_{n+1} = y_c + \frac{19}{270}(y_p - y_c)$, where y_p is the predictor equation, $y_p = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$, and y_c is the corrector equation, $y_c = y_n + \frac{h}{24}(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$. In the Matlab algorithm provided appendix 1.3, the RK4 method is used to find the first four values and the subsequent values are found using the predictor-corrector method.

MATLAB ODE BUILT-IN FUNCTION (ODE45)

Matlab's most common native solver for ordinary differential equations is the function 'ODE45'. This function uses a form of Runge-Kutta-type solving with a variable time step for efficient solving [18]. This function will also be used in comparison to the above functions when analyzing the Hodgkin-Huxley model.

MATLAB SOLUTIONS USING NUMERICAL METHODS

EULER METHODS

Using the Euler methods described above (forward, modified, and backward), a Matlab file was created to analyze the three approximation methods on a given differential equation. The equation $y'(t) + 4 \cdot y(t) = 2 \cdot e^{-5t}$ has the solution $y(t) = -2 \cdot e^{-5t} + 3 \cdot e^{-4t}$ that was found using Matlab's built-in differential equation solver. The Matlab code for the for the Euler methods is appended as 1.1.

Figure 6 shows the output of the three Euler methods. It can be seen that the modified Euler method has the most accurate results with the forward and backward methods erring on either side of the exact solution.

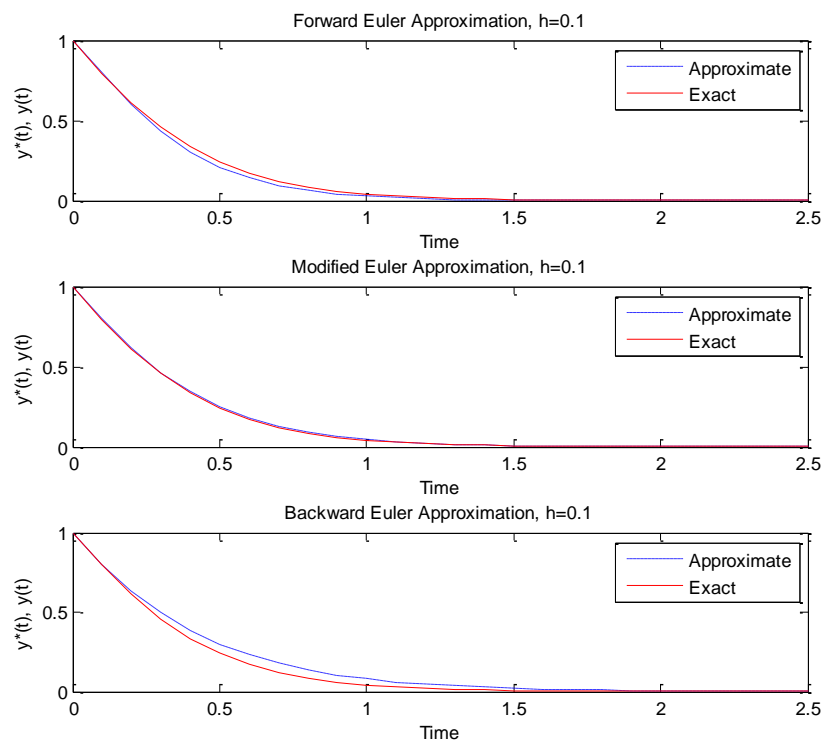


FIGURE 6 EULER METHODS PLOTTED WITH A STEP SIZE OF 0.1. THE RED SOLID LINE IS THE EXACT SOLUTION TO THE EQUATION AND THE BLUE DASHED LINE IS THE SOLUTION FOUND USING THE RESPECTIVE METHOD. TOP: FORWARD EULER METHOD; MIDDLE: MODIFIED EULER METHOD; BOTTOM: BACKWARD EULER METHOD.

RUNGE-KUTTA METHOD

Using the same differential equation above as an example, code to compute the RK4 approximation was produced in Matlab (appendix 1.2).

The resultant approximation is visibly more accurate than the previous Euler methods. In Figure 7 it seems that the RK4 approximation is identical to the actual solution.

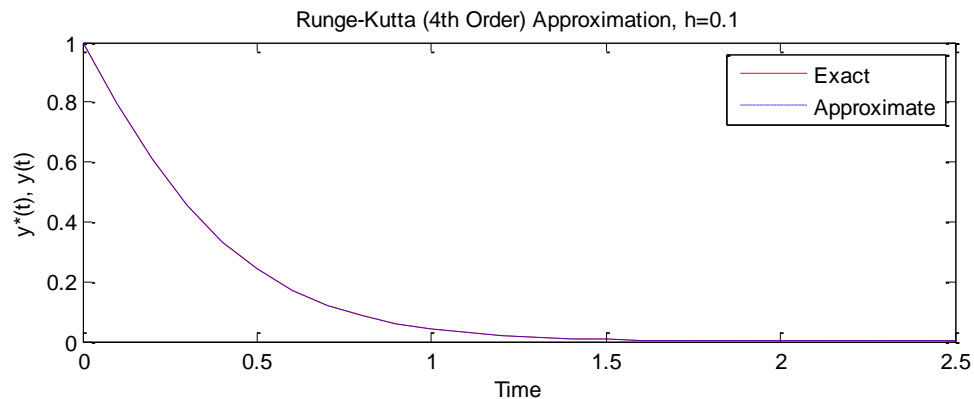


FIGURE 7 A PLOT OF THE EXACT SOLUTION AND THE APPROXIMATE SOLUTION FOUND USING THE 4TH ORDER RUNGE-KUTTA METHOD WITH A STEP SIZE OF 0.1.

ADAMS-BASHFORTH-MOULTON PREDICTOR-CORRECTOR METHOD

The predictor correct method used to find the solution to the above ordinary differential equation. See appendix 1.3 for the Matlab code.

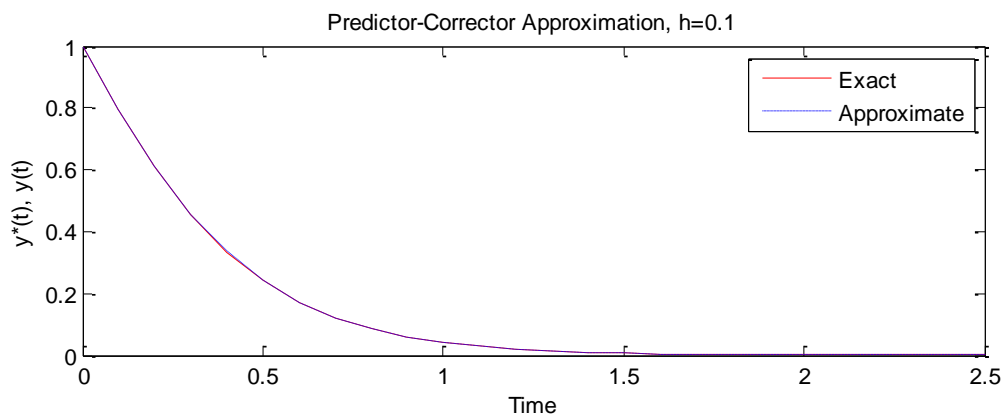


FIGURE 8 PLOT OF THE EXACT SOLUTION AGAINST THE APPROXIMATE SOLUTION FOUND USING THE PREDICTOR CORRECTOR METHOD

ODE45

The innate Matlab function for the ODE45 solution plotted below. The simple Matlab code using the function is found in appendix 1.4.

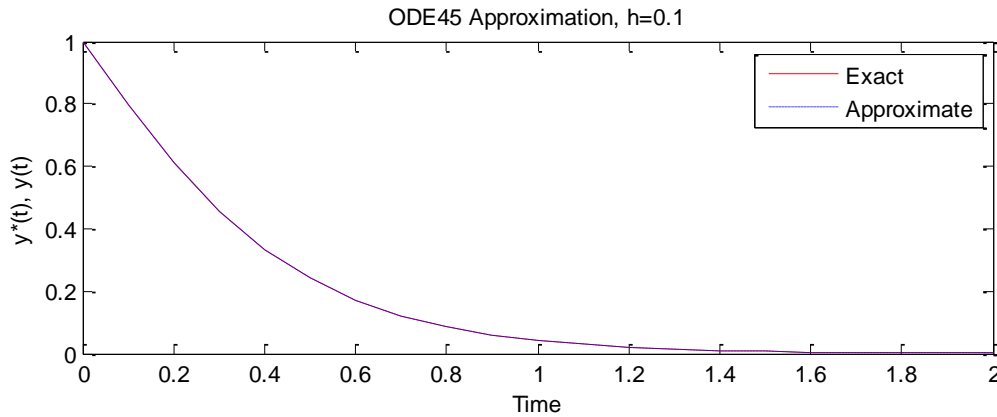


FIGURE 9 PLOT OF THE EXACT SOLUTION AGAINST THE APPROXIMATE SOLUTION FOUND USING THE ODE45 FUNCTION

ACCURACY OF METHODS

ERROR

Using the error formula, $error = |numerical\ result - exact\ value|$, the error will be calculated for the five methods mentioned above. The results are reported in Table 1.

TABLE 1 ERRORS BETWEEN EXACT SOLUTION AND APPROXIMATE SOLUTION

Method	Average Error
Forward Euler	0.6102
Modified Euler	0.1622
Backward Euler	1.4324
Runge-Kutta	0.0014
Predictor-Corrector	0.0083
ODE45	0.0011

From this very basic analysis using the differential equation $y'(t) + 4 \cdot y(t) = 2 \cdot e^{-5t}$ it can easily be seen that the Euler methods are very different from the actual curve when the size of the interval is 0.1. Both the Runge-Kutta and predictor-corrector methods serve as a much better approximation to the real curve.

ORDER OF THE METHOD

This section involves taking each method (excluding ODE45 because it is an adaptive method) and varying the step size while observing the change in error. The step size, h , was taken as $h = 0.1, 0.05, 0.025, 0.0125, 0.00625 \dots$ for the same solvable differential equation above ($y'(t) + 4 \cdot y(t) = 2 \cdot e^{-5t}$). Figure 10 shows the graph for each numerical method where the step size is plotted against the error on a log scale. The log scale is used because the error depends on some exponent of h thus rendering a straight-line. This value of the exponent (obtained from the slope of the graph) describes the speed of the method. This quantity is known as the order of the method and is shown in Table 2 for each method.

TABLE 2 ORDER OF THE METHODS

Method	Order of Method
Forward Euler	0.9958
Modified Euler	2.0115
Backward Euler	1.0607
Runge-Kutta	4.0000
Predictor-Corrector	4.9075

It can be seen from the order of the methods that the higher order methods are faster. The Runge-Kutta method and predictor-corrector method are on about the 4th and 5th order meaning the speed of these methods are much faster than the Euler methods. The *fastness* or *speed* of a method is determined by the gain in accuracy when a computational resource is doubled. Of the Euler methods, the modified Euler is twice as fast as the forward and backward versions. Table 2 shows the order (as slope) on the log scales as compared to reference orders.

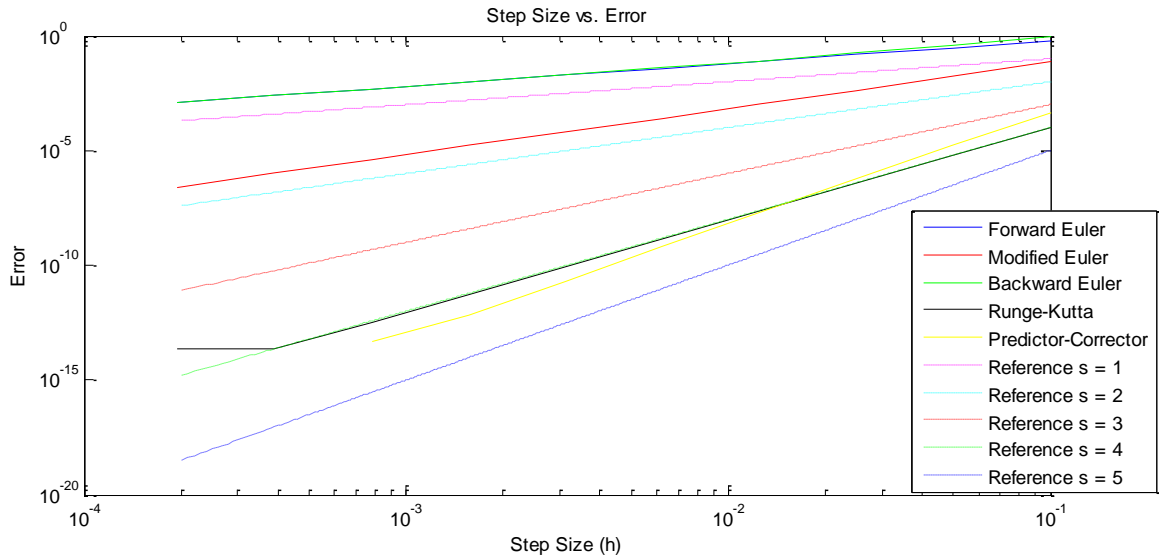


FIGURE 10 PLOT OF THE ERRORS FOR DIFFERENT STEP SIZES ON A DOUBLE LOG SCALE. THE SOLID LINES ARE THE RESULTS OF THE RESPECTIVE NUMERICAL METHODS. THE DOTTED LINES ARE REFERENCE ORDERS TO VISUALLY COMPARE

Also included in the above plot are reference lines of several different orders to be used as comparison. The general formula of $r(h) = h^s$, where h is the step size and s is the order of the method, is used to visually compare the different numerical methods with the order values from one to four.

An example (predictor-corrector) of the Matlab code used to compute the graph of h versus error is appended as 1.5. All the other m-files follow the same algorithm using the respective method.

ANALYSIS OF HODGKIN-HUXLEY MODEL

SOLVING HODGKIN-HUXLEY DIFFERENTIAL EQUATION

From the numerical methods discussed and analyzed above, four were chosen to solve the Hodgkin-Huxley model described in the introduction. The variety of methods is: forward Euler, Runge-Kutta, predictor-corrector, and ODE45.

The Hodgkin-Huxley model can be summarized neatly into four separate ordinary differential equations with some supporting functions. These equations and their constants solved by the above methods are listed below [19]:

$$\frac{dv}{dt} = \frac{1}{C_m} \cdot [I - g_{Na}m^3h(v - E_{Na}) - g_Kn^4(v - E_K) - g_l(v - E_l)]$$

$$\frac{dn}{dt} = \alpha_n(v)(1 - n) - \beta_n(v)n$$

$$\frac{dm}{dt} = \alpha_m(v)(1 - m) - \beta_m(v)m$$

$$\frac{dh}{dt} = \alpha_h(v)(1 - h) - \beta_h(v)h$$

Where,

$$\alpha_n = \frac{0.01(v + 50)}{1 - \exp\left(\frac{-(v + 50)}{10}\right)}$$

$$\beta_m = 4.0 \exp(-0.0556(v + 60))$$

$$\alpha_h = 0.07 \exp(-0.05(v + 60))$$

$$\beta_n = 0.125 \exp\left(\frac{-(v + 60)}{80}\right)$$

$$\beta_h = \frac{1}{1 + \exp(-0.1(v + 30))}$$

$$\alpha_m = \frac{0.1(v + 35)}{1 - \exp\left(\frac{-(v + 35)}{10}\right)}$$

$$C_m = 0.01 \mu\text{F}/\text{cm}^2$$

$$g_{Na} = 1.2 \text{ mS}/\text{cm}^2$$

$$E_{Na} = 55.17 \text{ mV}$$

$$g_K = 0.36 \text{ mS}/\text{cm}^2$$

$$E_K = -72.14 \text{ mV}$$

$$g_l = 0.003 \text{ mS}/\text{cm}^2$$

$$E_l = -49.42 \text{ mV}$$

RESULTS

Solving the Hodgkin-Huxley model with numerical methods provided a solution familiar to the action potentials discussed in the introduction. All parts of the action potential are present including the rapid uprise, downfall and unexcitable phase. As can be seen in Figure 11, each different numerical method provides extremely similar action potentials. This is expected since the errors obtained for each method compared to the exact solution above were relatively small.

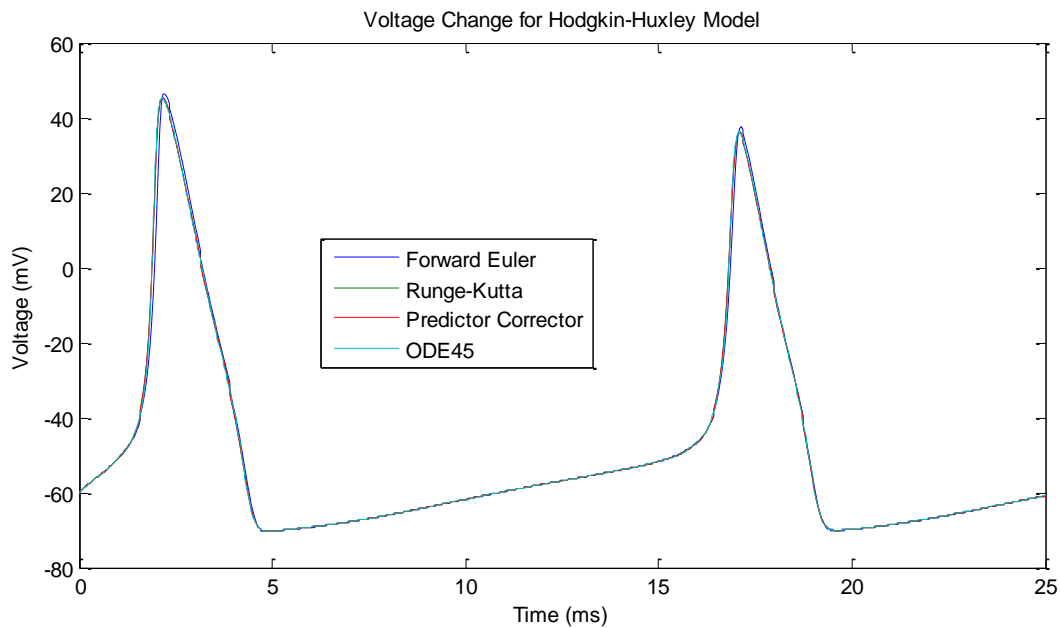


FIGURE 11 THE SOLUTION OF THE HODGKIN-HUXLEY MODEL BY FOUR DIFFERENT METHODS. TWO ACTION POTENTIALS ARE SEEN OVER THE 25SEC TIME SPAN.

The gating variables are correctly constrained between 0 and 1 in Figure 12. The variables are shown for five milliseconds but repeat in the same manner over longer periods of time. All methods are seen to have almost identical gating variables except for the ODE45 method.

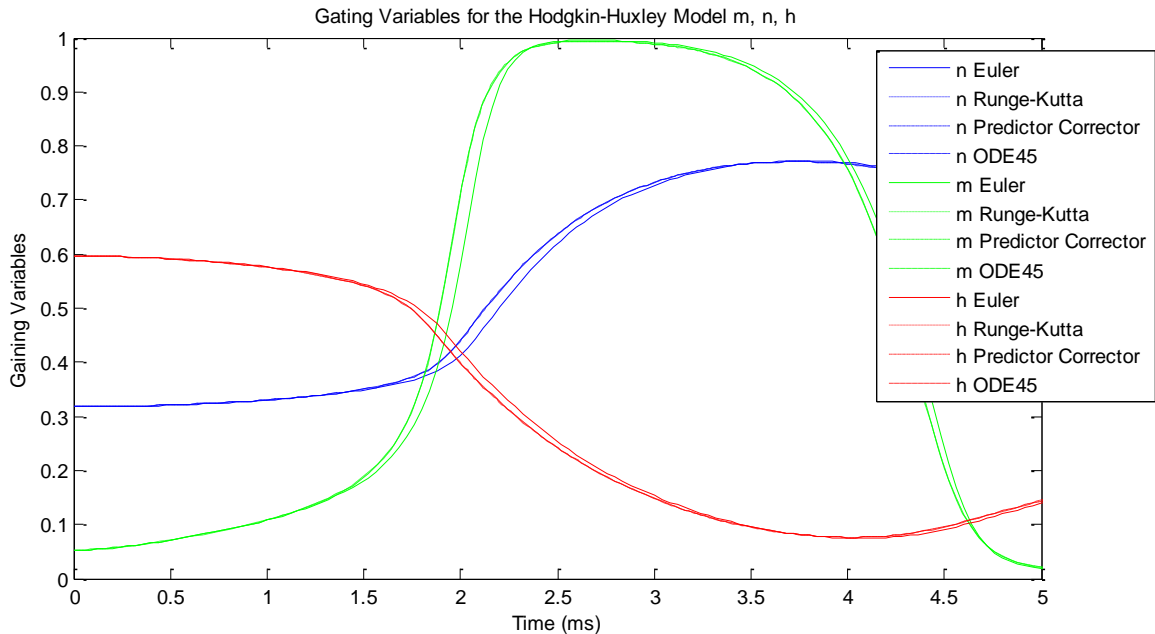


FIGURE 12 A PLOT OF THE DIFFERENT GATING VARIABLES FOUND USING DIFFERENT NUMERICAL METHODS

The Matlab code written to solve this complex set of equations was written in three parts. The first file (appendix 2.1) demonstrates all the methods in succession using the basic form to call the HH function (appendix 2.2). This very important function was used for three of the four methods. It allows the calling code to be written in its simplest form and provides access to easy manipulation. The third and final part is a series of functions that define the alpha and beta characteristics (appendix 2.3). These functions are called from both the main code and the HH function.

EXACT SOLUTION AND COMPARISON OF METHODS

An interesting way to compare the numerical solutions of the Hodgkin-Huxley equation is to use the exact solution. An exact solution can be obtained if the sodium and potassium conductance are set equal to zero. The new equation is

$$\frac{dv}{dt} = \frac{1}{C_m} [I - g_l(v - E_l)]$$

This ordinary differential equation can be solved easily using the method of separation of variables. The initial voltage is set to be -60mV. The solution obtained is [19]

$$v = \frac{1}{g_l} \left[-\exp\left(-\frac{g_l}{C_m} t\right) (I + 60g_l + g_l E_l) + I + g_l E_l \right]$$

Using the same Matlab code for voltage change, the variables g_{Na} and g_K are set equal to zero and plotted on the same axis as the exact solution.

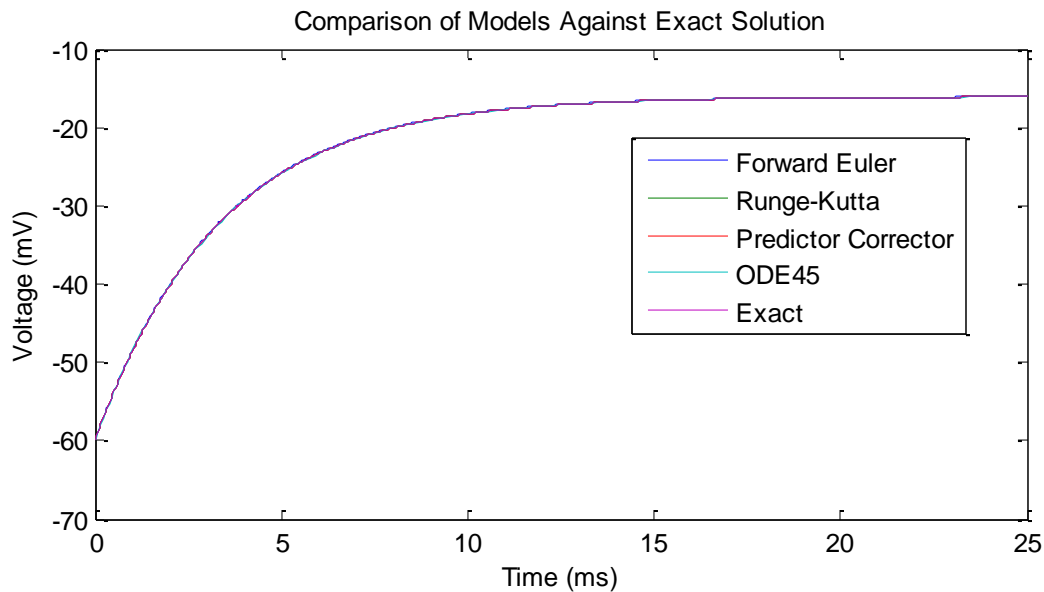


FIGURE 13 A PLOT OF THE EXACT SOLUTION TO THE HODGKIN-HUXLEY MODEL USED TO COMPARE THE ACCURACY OF THE DIFFERENT NUMERICAL METHODS.

TABLE 3 ERROR OF NUMERICAL METHODS AS COMPARED TO THE EXACT SOLUTION

Method	Average Error
Forward Euler	0.034984
Runge-Kutta	0.00000010155
Predictor Corrector	0.000000012004
ODE45	0.00030036

It can be easily demonstrated that the forward Euler method is the weakest when the results of the different numerical methods are compared to the exact solution. This is what was expected based on the initial analysis and resultant speed of computation. Both the Runge-Kutta and predictor-corrector method

were significantly more accurate than the forward Euler method. To the contrary of previous results and intuition, the ODE45 function was found to be less accurate than the Runge-Kutta and Predictor Corrector methods.

PERSONAL REMARKS

The essential communication from neuron to neuron is nothing short of amazing. The fact that Hodgkin and Huxley were able to create an equation to model this complex behavior, yet simple enough to solve and predict might be even more amazing.

The numerical methods implemented here truly show the simplicity of the model, but the resultant action potentials show so much complexity. It was extremely interesting and very satisfying to try the different numerical methods to successfully solve the Hodgkin-Huxley model.

The forward Euler method was the easiest to code, besides, of course, the ODE45 function. The most difficult part of the coding was to create the HH function. It was a fairly short code, but it branched out to many other functions and returned the current value of the membrane voltage and the three gating parameters. Once the HH function was created, the coding was much simpler and involved the use of the numerical methods' basic definitions.

It was very satisfying to see the code produce action potentials that I have studied in extreme depth in my physiology classes. To know what each moment of the simple line means physiologically makes the underlying math seem more sophisticated.

CONCLUSION

This project offered an initial physiological background to the Hodgkin-Huxley model but focused mainly on the mathematics and numerics behind it. The primary objective was to solve the Hodgkin-

Huxley model by different numerical methods and compare the speed and accuracy of each method using different techniques.

Forward Euler, Runge-Kutta, predictor-corrector, and ODE45 numerical methods were used to first calculate a theoretical action potential and associated variables followed by a direct comparison using an exact solution. The action potential solved using the numerical methods through Matlab gave a textbook result of membrane voltage change over time.

The Runge-Kutta and predictor-corrector methods gave the most accurate results when compared to the exact solution by setting sodium and potassium conductances equal to zero for a step size of 0.04. The Matlab ODE45 function produced a slightly less accurate result. The default parameters of ODE45 proved to be less accurate than the small step sized used for the former methods. In theory, when comparing apples to apples, the ODE45 method will solve the equation closer to the exact result. The Euler method proved, again, to be the least accurate. After producing the HH function, all numerical methods become quite simple to code and produce similar results.

The Hodgkin-Huxley model produced customizable and predictable action potentials that reveal a tremendous amount of information about neuron signaling. The results in this project confirm the hierarchy of numerical methods and simultaneously display the power and beauty of the Hodgkin-Huxley model.

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to Dr. Gantumur Tsogtgerel. Dr. Tsogtgerel offered abundant support and time to this project. His vast knowledge, guidance, patience, and motivation helped me to create this project.

WORKS CITED

1. Hodgkin, A.L., A.F. Huxley, and B. Katz, *Measurement of current-voltage relations in the membrane of the giant axon of Loligo*. J Physiol, 1952. **116**(4): p. 424-48.
2. Hodgkin, A.L. and A.F. Huxley, *Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo*. J Physiol, 1952. **116**(4): p. 449-72.
3. Hodgkin, A.L. and A.F. Huxley, *The components of membrane conductance in the giant axon of Loligo*. J Physiol, 1952. **116**(4): p. 473-96.
4. Hodgkin, A.L. and A.F. Huxley, *The dual effect of membrane potential on sodium conductance in the giant axon of Loligo*. J Physiol, 1952. **116**(4): p. 497-506.
5. Hodgkin, A.L. and A.F. Huxley, *A quantitative description of membrane current and its application to conduction and excitation in nerve*. J Physiol, 1952. **117**(4): p. 500-44.
6. Rinzel, M.N.a.J., *Chapter 4 The Hodgkin-Huxley Model*, in *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural Simulation System*1998, TELOS/Springer-Verlag.
7. Cole, K.S. and H.J. Curtis, *ELECTRIC IMPEDANCE OF THE SQUID GIANT AXON DURING ACTIVITY*. J Gen Physiol, 1939. **22**(5): p. 649-670.
8. Bernstein, J., *Untersuchungen zur Thermodynamik der bioelektrischen Ströme*. Pflügers Archiv European Journal of Physiology, 1902. **92**(10): p. 521-562.
9. Cole, K.S. and A.L. Hodgkin, *Membrane and Protoplasm Resistance in the Squid Giant Axon*. J Gen Physiol, 1939. **22**(5): p. 671-87.
10. Clay, J.R., D. Paydarfar, and D.B. Forger, *A simple modification of the Hodgkin and Huxley equations explains type 3 excitability in squid giant axons*. Journal of The Royal Society Interface, 2008. **5**(29): p. 1421-1428.
11. Feudel, U., et al., *Homoclinic bifurcation in a Hodgkin-Huxley model of thermally sensitive neurons*. Chaos: An Interdisciplinary Journal of Nonlinear Science, 2000. **10**(1): p. 231-239.
12. Miller, R.N. and J. Rinzel, *The dependence of impulse propagation speed on firing frequency, dispersion, for the Hodgkin-Huxley model*. Biophysical Journal, 1981. **34**(2): p. 227-259.
13. Lee, S.-G., A. Neiman, and S. Kim, *Coherence resonance in a Hodgkin-Huxley neuron*. Physical Review E, 1998. **57**(3): p. 3292-3297.
14. Meunier, C. and I. Segev, *Playing the Devil's advocate: is the Hodgkin-Huxley model useful?* Trends in Neurosciences, 2002. **25**(11): p. 558-563.
15. Atkinson, K., *An Introduction to Numerical Analysis*1989, New York: Wiley.
16. Butcher, J., *Numerical Methods for Ordinary Differential Equations*, 2003, J. Wiley: Chinchester, West Sussex, England Hoboken, NJ.

17. Press, W.H.F., Brian P.; Teukolsky, Saul A.; Vetterling, William T., *Nmerical Recipes: The Art of Scientific Computing*. Section 17.1 Runge-Kutta Method2007: Cambridge University Press.
18. Senan, N.A.F. *A brief introduction to using ode45 in MATLAB*. May 12th 2012].
19. Aaby, D., *A Comparitive Study of Numerical Methods for the Hodgkin-Huxley Model of Nerve Cell Action Potentials*, U.o. Dayton, Editor 2009.

APPENDIX - MATLAB CODES

1. ARBITRARY ODE

1.1 EULER METHODS

```
%% Euler Methods

%DE:= y'(t)+4*y(t)=2*exp(-5*t)

%Exact solution found using Matlab: y(t)=-2*exp(-5*t)+3*exp(-4*t)
%Used initial condition of y(0)=1

%=====
%% Forward Euler Method for 1st Order ODE
%=====

h=0.15;      %h is the step size
t=0:h:4;     %initialize time array

clear ystar;

ystar(1)=1.0; %initial condition

for i=1:length(t)-1,
    k=2*exp(-5*t(i))-4*ystar(i); %Calculates derivative at t(i)
    ystar(i+1)=ystar(i)+h*k;      %Estimates new value of y;
end

%exact solution
y=-2*exp(-5*t)+3*exp(-4*t);

%Plot approximate and exact solution.
subplot(1,3,1)
plot(t,ystar,'b--',t,y,'r-');
legend('Approximate','Exact');
title('Forward Euler Approximation, h=0.015');
xlabel('Time');
ylabel('y*(t), y(t)');

clear all;

%=====
%% Modified Euler Method for 1st Order ODE
%=====

h=0.15;      %h is the step size
t=0:h:4;     %initialize time array
```



```

clear ystar;

ystar(1)=1.0;    %initial condition

for i=1:length(t)-1,
    ynew=ystar(i)+h*(2*exp(-5*t(i))-4*ystar(i))
    k=2*exp(-5*t(i+1))-4*ynew
    ystar(i+1)=ystar(i)+h/2*((2*exp(-5*t(i))-4*ystar(i))+k);
%Estimates new value of y;
end

%exact solution
y=-2*exp(-5*t)+3*exp(-4*t);

%Plot approximate and exact solution.
subplot(1,3,2)
plot(t,ystar,'b--',t,y,'r-');
legend('Approximate','Exact');
title('Modified Euler Approximation, h=0.015');
xlabel('Time');
ylabel('y*(t), y(t)');

clear all;

%=====
%Backward Euler Method for 1st Order ODE
%=====

h=0.15;    %h is the step size
t=0:h:4;    %initialize time array

clear ystar;

ystar(1)=1.0;    %initial condition

for i=1:length(t)-1,
    ynew=ystar(i)+h*(2*exp(-5*t(i))-4*ystar(i))
    k=2*exp(-5*t(i+1))-4*ynew
    ystar(i+1)=ystar(i)+h*(k);    %Estimates new value of y;
end

%exact solution
y=-2*exp(-5*t)+3*exp(-4*t);

%Plot approximate and exact solution.
subplot(1,3,3)
plot(t,ystar,'b--',t,y,'r-');
legend('Approximate','Exact');
title('Backward Euler Approximation, h=0.015');
xlabel('Time');
ylabel('y*(t), y(t)');

```

1.2 RUNGE-KUTTA

```

%=====
%% Runge-Kutta Method
%=====

%DE:= y'(t)+4*y(t)=2*exp(-5*t)

%Exact solution found using Maple: y(t)=-2*exp(-5*t)+3*exp(-4*t)
    %Used initial condition of y(0)=1

h=0.1;      %h is the step size
t=0:h:2.5;  %initialize time array

clear ystar;

ystar(1)=1.0; %initial condition

for i = 1:length(t)-1
    k1 = 2*exp(-5*t(i))-4*ystar(i);
    k2 = 2*exp(-5*(t(i)+h/2))-4*(ystar(i)+(h/2)*k1);
    k3 = 2*exp(-5*(t(i)+h/2))-4*(ystar(i)+(h/2)*k2);
    k4 = 2*exp(-5*(t(i)+h))-4*(ystar(i)+h*k3);
    ystar(i+1) = ystar(i) + h/6*(k1 + 2*k2 + 2*k3 + k4);
end

%exact solution
y=-2*exp(-5*t)+3*exp(-4*t);

plot(t,y,'r-',t,ystar,'b-.');
legend('Exact','Approximate');
title('Runge-Kutta (4th Order) Approximation, h=0.15');
xlabel('Time');
ylabel('y*(t), y(t)');

```

1.3 PREDICTOR-CORRECTOR

```

%=====
%% Predictor-Corrector Method
%=====

%DE:= y'(t)+4*y(t)=2*exp(-5*t)

%Exact solution found using Maple: y(t)=-2*exp(-5*t)+3*exp(-4*t)
    %Used initial condition of y(0)=1

h=0.1;      %h is the step size
t=0:h:2.5;  %initialize time array

clear ystar;

ystar(1)=1.0; %initial condition

for i = 1:3 % find the first four elements using RK4

```

```

k1 = 2*exp(-5*t(i))-4*ystar(i);
k2 = 2*exp(-5*(t(i)+h/2))-4*(ystar(i)+(h/2)*k1);
k3 = 2*exp(-5*(t(i)+h/2))-4*(ystar(i)+(h/2)*k2);
k4 = 2*exp(-5*(t(i)+h))-4*(ystar(i)+h*k3);
ystar(i+1) = ystar(i) + h/6*(k1 + 2*k2 + 2*k3 + k4);
end

for i = 4:length(t)-1 % P-C Method
    yp=ystar(i)+(h/24)*(55*(2*exp(-5*t(i))-4*ystar(i))-59*(2*exp(-
5*t(i-1))-4*ystar(i-1))+37*(2*exp(-5*t(i-2))-4*ystar(i-2))-
9*(2*exp(-5*t(i-3))-4*ystar(i-3)));
    yc=ystar(i)+(h/24)*(9*(2*exp(-5*t(i+1))-4*yp)+19*(2*exp(-
5*t(i))-4*ystar(i))-5*(2*exp(-5*t(i-1))-4*ystar(i-1))+1*(2*exp(-
5*t(i-2))-4*ystar(i-2)));
    ystar(i+1)=yc+(19/270)*(yp-yc);
end

%exact solution
y=-2*exp(-5*t)+3*exp(-4*t);

plot(t,y,'r-',t,ystar,'b-.');
legend('Exact','Approximate');
title('Predictor-Corrector Approximation, h=0.15');
xlabel('Time');
ylabel('y*(t), y(t)');

```

1.4 ODE45

```

%=====
%% ODE45 Method
%=====

clear all;
h=0.05 %set initial step size
tspan=0:h:2;
xinit=1;
[t,x] = ode45(@DE, tspan, xinit) %DE is -4*y+2*exp(-5*t);

yact=-2*exp(-5*t)+3*exp(-4*t); %exact solution

plot(t,yact,'r-',t,x,'b-.');
legend('Exact','Approximate');
title('ODE45 Approximation, h=0.15');
xlabel('Time');
ylabel('y*(t), y(t)');

err(x,yact) %error between method and exact solution

```

1.5 ORDER OF THE METHOD

```

%% Find the Order of Method for Predictor-Corrector Method

clear all;
h=0.1 %set initial step size
t=0:h:2;
ystar(1)=1.0; %initial condition
y=-2*exp(-5*t)+3*exp(-4*t); %exact solution

for j=1:8; %h is the step size
    harray(j)=h; %record the h-value
    h=h/2; %create next h-value
    t=0:h:2;
    clear ystar;
    ystar(1)=1;
    y=-2*exp(-5*t)+3*exp(-4*t);
    %PC method for finding ystar
    for i = 1:3 % find the first four elements using RK4
        k1 = 2*exp(-5*t(i))-4*ystar(i);
        k2 = 2*exp(-5*(t(i)+h/2))-4*(ystar(i)+(h/2)*k1);
        k3 = 2*exp(-5*(t(i)+h/2))-4*(ystar(i)+(h/2)*k2);
        k4 = 2*exp(-5*(t(i)+h))-4*(ystar(i)+h*k3);
        ystar(i+1) = ystar(i) + h/6*(k1 + 2*k2 + 2*k3 + k4);
    end
    for i = 4:length(t)-1 % P-C Method
        yp=ystar(i)+(h/24)*(55*(2*exp(-5*t(i))-4*ystar(i))-
59*(2*exp(-5*t(i-1))-4*ystar(i-1))+37*(2*exp(-5*t(i-2))-4*ystar(i-
2))-9*(2*exp(-5*t(i-3))-4*ystar(i-3)));
        yc=ystar(i)+(h/24)*(9*(2*exp(-5*t(i+1))-4*yp)+19*(2*exp(-
5*t(i))-4*ystar(i))-5*(2*exp(-5*t(i-1))-4*ystar(i-1))+1*(2*exp(-
5*t(i-2))-4*ystar(i-2)));
        ystar(i+1)=yc+(19/270)*(yp-yc);
    end
    p(j)=err(ystar,y) %record the error at the corresponding h-
value
end

%plot on a loglog scale
loglog(harray,p)
title('Predictor-Corrector Method - Step Size vs. Error');
xlabel('Step Size (h)');

```

2. HODGKIN HUXLEY MODEL

2.1 COMPARISON OF METHODS

```

%=====
%% HH Comparison of all Methods
%=====

%% Forward Euler Method

clc; clear;

```

```

%Constants set for all Methods
Cm=0.01; % Membrane Capacitance uF/cm^2
dt=0.04; % Time Step ms
t=0:dt:25; %Time Array ms

I=0.1; %External Current Applied

ENa=55.17; % mv Na reversal potential
EK=-72.14; % mv K reversal potential
El=-49.42; % mv Leakage reversal potential
gbarNa=1.2; % mS/cm^2 Na conductance
gbarK=0.36; % mS/cm^2 K conductance
gbarl=0.003 % mS/cm^2 Leakage conductance

V(1)=-60; % Initial Membrane voltage
m(1)=am(V(1))/(am(V(1))+bm(V(1))); % Initial m-value
n(1)=an(V(1))/(an(V(1))+bn(V(1))); % Initial n-value
h(1)=ah(V(1))/(ah(V(1))+bh(V(1))); % Initial h-value

for i=1:length(t)-1

    %Euler method to find the next m/n/h value
    m(i+1)=m(i)+dt*((am(V(i))*(1-m(i)))-(bm(V(i))*m(i)));
    n(i+1)=n(i)+dt*((an(V(i))*(1-n(i)))-(bn(V(i))*n(i)));
    h(i+1)=h(i)+dt*((ah(V(i))*(1-h(i)))-(bh(V(i))*h(i)));

    gNa=gbarNa*m(i)^3*h(i);
    gK=gbarK*n(i)^4;
    gl=gbarl;

    INa=gNa*(V(i)-ENa);
    IK=gK*(V(i)-EK);
    Il=gl*(V(i)-El);

    %Euler method to find the next voltage value
    V(i+1)=V(i)+(dt)*((1/Cm)*(I-(INa+IK+Il)));

end

%Store variables for graphing later
FE=V;
FEm=m;
FEn=n;
FEh=h;

clear V m n h;

%% Runge-Kutta Method

V(1)=-60; % Initial Membrane voltage
m(1)=am(V(1))/(am(V(1))+bm(V(1))); % Initial m-value
n(1)=an(V(1))/(an(V(1))+bn(V(1))); % Initial n-value
h(1)=ah(V(1))/(ah(V(1))+bh(V(1))); % Initial h-value

```

```

for i=1:length(t)-1 % Loop through each step until time is
finished

    %4 step method of Runge-Kutta
    K1=dt*HH(i,[V(i); n(i); m(i); h(i)]);
    k1=K1(1,1);n1=K1(2,1);m1=K1(3,1);h1=K1(4,1);% obtain 4 k
variables (V,m,n,h) from HH function

K2=dt*HH(i+(0.5*dt),[V(i)+(0.5*k1);n(i)+(0.5*n1);m(i)+(0.5*m1);h(i)
)+(0.5*h1)]);
    k2=K2(1,1);n2=K2(2,1);m2=K2(3,1);h2=K2(4,1);

K3=dt*HH(i+(0.5*dt),[V(i)+(0.5*k2);n(i)+(0.5*n2);m(i)+(0.5*m2);h(i)
)+(0.5*h2)]);
    k3=K3(1,1);n3=K3(2,1);m3=K3(3,1);h3=K3(4,1);
    K4=dt*HH(i+dt,[V(i)+k3;n(i)+n3;m(i)+m3;h(i)+h3]);
    k4=K4(1,1);n4=K4(2,1);m4=K4(3,1);h4=K4(4,1);

    %create next step for each variable
    V(i+1)=V(i)+1/6*(k1+2*k2+2*k3+k4);
    n(i+1)=n(i)+1/6*(n1+2*n2+2*n3+n4);
    m(i+1)=m(i)+1/6*(m1+2*m2+2*m3+m4);
    h(i+1)=h(i)+1/6*(h1+2*h2+2*h3+h4);
end

%set variables for graphing later
RK=V;
RKn=m;
RKn=n;
RKh=h;

clear V m n h;
%% PC Method

V(1)=-60; % Initial Membrane voltage
m(1)=am(V(1))/(am(V(1))+bm(V(1))); % Initial m-value
n(1)=an(V(1))/(an(V(1))+bn(V(1))); % Initial n-value
h(1)=ah(V(1))/(ah(V(1))+bh(V(1))); % Initial h-value

%First four steps are found using the Runge-Kutta Method
for i=1:3

    K1=dt*HH(i,[V(i); n(i); m(i); h(i)]);
    k1=K1(1,1);n1=K1(2,1);m1=K1(3,1);h1=K1(4,1);

K2=dt*HH(i+(0.5*dt),[V(i)+(0.5*k1);n(i)+(0.5*n1);m(i)+(0.5*m1);h(i)
)+(0.5*h1)]);
    k2=K2(1,1);n2=K2(2,1);m2=K2(3,1);h2=K2(4,1);

K3=dt*HH(i+(0.5*dt),[V(i)+(0.5*k2);n(i)+(0.5*n2);m(i)+(0.5*m2);h(i)
)+(0.5*h2)]);
    k3=K3(1,1);n3=K3(2,1);m3=K3(3,1);h3=K3(4,1);
    K4=dt*HH(i+dt,[V(i)+k3;n(i)+n3;m(i)+m3;h(i)+h3]);
    k4=K4(1,1);n4=K4(2,1);m4=K4(3,1);h4=K4(4,1);

```

```

V(i+1)=V(i)+1/6*(k1+2*k2+2*k3+k4);
n(i+1)=n(i)+1/6*(n1+2*n2+2*n3+n4);
m(i+1)=m(i)+1/6*(m1+2*m2+2*m3+m4);
h(i+1)=h(i)+1/6*(h1+2*h2+2*h3+h4);
end

for i = 4:length(t)-1 % P-C Method

    %predictor

yp=[V(i);n(i);m(i);h(i)]+(dt/24)*(55*HH(t(i),[V(i);n(i);m(i);h(i)]
)-59*(HH(t(i-1),[V(i-1);n(i-1);m(i-1);h(i-1)]))+37*(HH(t(i-
2),[V(i-2);n(i-2);m(i-2);h(i-2)]))-9*(HH(t(i-3),[V(i-3);n(i-
3);m(i-3);h(i-3)])));
    %corrector

yc=[V(i);n(i);m(i);h(i)]+(dt/24)*(9*(HH(t(i+1),yp))+19*(HH(t(i),[V
(i);n(i);m(i);h(i)]))-5*HH(t(i-1),[V(i-1);n(i-1);m(i-1);h(i-
1)]))+HH(t(i-2),[V(i-2);n(i-2);m(i-2);h(i-2)])));
    C=yc+(19/270)*(yp-yc);
    V(i+1)=C(1,1);
    n(i+1)=C(2,1);
    m(i+1)=C(3,1);
    h(i+1)=C(4,1);
end

%Store variables for graphing
PC=V;
PCm=m;
PCn=n;
PCh=h;

clear V m n h;

%% ODE45 Method

V=-60; % Initial Membrane voltage
m=am(V)/(am(V)+bm(V)); % Initial m-value
n=an(V)/(an(V)+bn(V)); % Initial n-value
h=ah(V)/(ah(V)+bh(V)); % Initial h-value

y0=[V;n;m;h];

tspan = [0,max(t)];

%Matlab's ode45 function
[time,V] = ode45(@HH,tspan,y0);

OD=V(:,1);
ODn=V(:,2);
ODm=V(:,3);
ODh=V(:,4);

```

```

clear V;

%% Plots

%Plot the functions
plot(t,FE,t,RK,t,PC,time,OD);
legend('Forward Euler','Runge-Kutta','Predictor
Corrector','ODE45');
xlabel('Time (ms)');
ylabel('Voltage (mV)');
title('Voltage Change for Hodgkin-Huxley Model');

figure
plot(t,FEn,'b',t,RKn,'b:',t,PCn,'b-.',time,ODn,'b--
',t,FEm,'g',t,RKm,'g:',t,PCm,'g-.',time,ODm,'g--
',t,FEh,'r',t,RKh,'r:',t,PCh,'r-.',time,ODh,'r--');
ylabel('Gaining Variables')
xlabel('Time (ms)')
axis([0 5 0 1])
legend('n Euler','n Runge-Kutta','n Predictor Corrector','n
ODE45','m Euler','m Runge-Kutta','m Predictor Corrector','m
ODE45','h Euler','h Runge-Kutta','h Predictor Corrector','h
ODE45');

```

2.2 HH FUNCTION

```

function dydt = HH(t,y)

% Constants
ENa=55.17; % mv Na reversal potential
EK=-72.14; % mv K reversal potential
El=-49.42; % mv Leakage reversal potential
gbarNa=1.2; % mS/cm^2 Na conductance
gbarK=0.36; % mS/cm^2 K conductance
gbarl=0.003; % mS/cm^2 Leakage conductance

I = 0.1; %Applied Current
Cm = 0.01; %Membrane Capacitance

% Values set to equal input values
V = y(1);
n = y(2);
m = y(3);
h = y(4);

gNa=gbarNa*m^3*h;
gK=gbarK*n^4;
gl=gbarl;

INa=gNa*(V-ENa);
IK=gK*(V-EK);
Il=gl*(V-El);

```



```

%Hodgkin-Huxley Model Equation
dydt = [ ((1/Cm)*(I-(INa+IK+Il))); an(V)*(1-n)-bn(V)*n; am(V)*(1-
m)-bm(V)*m; ah(V)*(1-h)-bh(V)*h];

```

2.3 ALPHA AND BETA FUNCTIONS

```

function a=am(v) %Alpha for Variable m
a=0.1*(v+35)/(1-exp(-(v+35)/10));
end

```

```

function b=bn(v) %Beta for variable m
b=4.0*exp(-0.0556*(v+60));
end

```

```

function a=an(v)%Alpha for variable n
a=0.01*(v+50)/(1-exp(-(v+50)/10));
end

```

```

function b=bn(v) %Beta for variable n
b=0.125*exp(-(v+60)/80);
end

```

```

function a=ah(v) %Alpha value for variable h
a=0.07*exp(-0.05*(v+60));
end

```

```

function b =bh(v) %beta value for variable h
b=1/(1+exp(-(0.1)*(v+30)));
end

```