The Hodgkin Huxley Model:
Simulation of the Action Potential in the Giant Squid Axon

Akwasi Darkwah Akwaboah*

Abstract— **The Hodgkin-Huxley model is by far one of the most groundbreaking work of the twentieth century and beyond; it elucidated the underpinnings of cell electrophysiology. In a series of articles published in the 1952, this work presented the ionic basis of action potential in a giant squid axon. The mechanism discovered is however not limited to squid axon but all excitable cells. This report succinctly presents the details of this model and goes further to implement a simulation in this respect. The kinetics of the ionic currents, effects of modifying the constituent currents as well as the computational time step are presented.**

Index Terms— **Hodgkin-Huxley model, cell electrophysiology, biophysics, computational biology**

I. Introduction

In a series of 5 articles on work by Alan Lloyd Hodgkin and Andrew Fielding Huxley and their assistants aimed at uncovering the ionic basis of the cell excitability, the Hodgkin-Huxley (HH) model was formulated.[1] In their somewhat serendipitous choice of the giant squid axon as the model cell for this purpose, Hodgkin and Huxley gathered significant empirical data on the kinetics of three ionic currents, namely sodium, potassium and leakage currents, which they hypothesized was the basis of excitability in the giant squid axon. They postulated the *parallel conductance model* (shown in figure 1 below) that a cell membrane patch possesses capacitance (C_m) as a result of the intracellular and extracellular electrolytic media separated by an insulating hydrophobic bilipid layer. Also, the interspersed ion channels were modeled as conductances (g_i). More so, the individual ionic intracellular and extracellular concentrations were represented as electrochemical potentials (batteries, E_i) given by the Nernst equation below.

$$E_{ion} = \frac{RT}{z_{ion}F}\ln\left(\frac{[ion]_o}{[ion]_i}\right) \text{ ----------- } 1$$

The resulting potential across the network gives the transmembrane voltage (V_m). In the presence of a stimulus, I_{stim}, the time course of V_m produces the iconic spiking action potential, which in the case of axon, encodes information to be delivered to the synapse to be perceived by adjacent neurons as some sort of request for propagating the excitation. In the case of cardiac and other muscle cells this forms the basis of the electro-mechanical coupling to elicit contraction and relaxation.

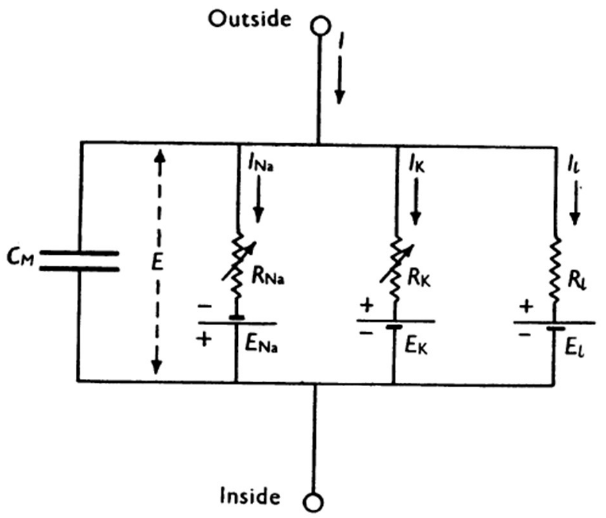

Figure 1. The Parallel-Conductance Model. (Retrieved from [1]*)*

Circuit analysis of the parallel conductance model yields a first order differential equation relating V_m to the various ionic currents, I_{ion} and the membrane capacitance, C_m. This is shown in equation 2 below;

$$Cm\frac{dv_m}{dt} = I_{stim} - I_{ion} \text{ ------ } 2$$

Upon gathering results from voltage and space clamping of the squid axon. Hodgkin and Huxley proposed the underlying ion channel kinetics and the overall conductance. Three gating probabilities were presented, two – activation, *m* and inactivation, *h* for sodium conductance and one activation, *n* for potassium conductance. All three gating probabilities are also governed by a first order equation presented below;

$$\frac{dx_i}{dt} = \alpha_i(1 - x_i) - \beta_i x_i \text{ ----------- } 3$$

Where x_i = *m, h* or *n* and α_i and β_i are activation and inactivation rate constants respectively and represented by Boltzmann equations. The individual activation and inactivation rate constants can be obtained from [1][2]. Also, the two conductance equations which Hodgkin and Huxley formulated were derived from curve fitting to experimental data

Akwasi Darkwah Akwaboah is with the Electronics Engineering Department, Norfolk State University

$$G_{Na} = \overline{g_{Na}} m^3 h \text{ -------------------- } 4$$

$$G_K = \overline{g_K} n^4 \text{ ---------------------- } 5$$

Where $\overline{g_{Na}}$ and $\overline{g_K}$ are the maximum sodium and potassium currents respectively given in literature[1] as 120mS/cm^2 and 36mS/cm^2 respectively, the leakage conductance, G_L is however assumed to be ohmic and given as 0.3mS/cm^2

$$I_{ion} = G_{Na}(v_m - E_{Na}) + G_{Na}(v_m - E_{Na}) + G_L(v_m - E_L)$$

Where $E_{Na} = 52.4mV$, $E_K = -74.7mV$ are the Nernst potentials for sodium and potassium respectively for the squid axon. Given the resting membrane potential, V_{rest} = -60mV, the leakage potential, E_L=43.256mV is computed using the Goldman's equation[2] shown below;

$$v_{\text{res}} = \frac{g_k E_k + g_{Na} E_{Na} + g_L E_L}{g_k + g_{Na} + g_L} \text{ ------ } 7$$

II. Action Potential: Simulation

The action potential along with the corresponding plots for the time course of the gating parameters have been presented in this section. These plots were obtained from computer simulations, with differential equations handled using the forward Euler approximation method. A time step of $1\mu s$ is used. MATLAB code (and an alternative python code) for this implementation have been included in the appendix. The action potential of the HH model simulated with an applied stimulus between t = 0ms and 2ms shown in figure 2 below depicts the rapid upstroke (depolarization) resulting from increased sodium conductance, which eventually inactivates at ~60mV, followed by repolarization as result of increased potassium currents which proceeds to sub- resting membrane potential – hyperpolarization, and an eventual return and beyond the V$_{\text{rest}}$ = -60mV

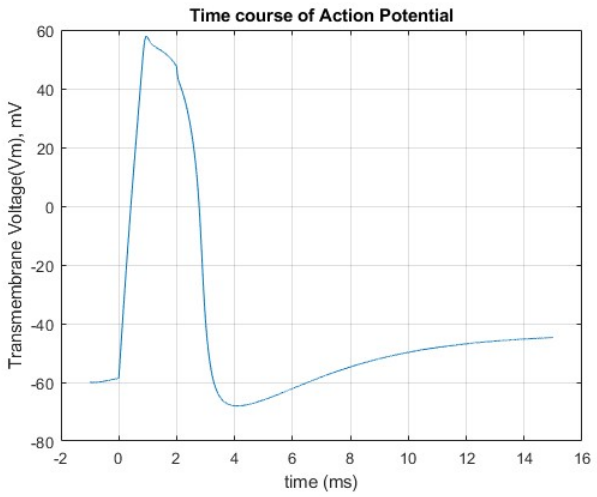

Figure 2. Action Potential of the HH model, with stimulus applied between t = 0ms and 2ms

The channel kinetics for above the action potential is provided by the gating probabilities – *m, h* and *n*. A plot for all three have been shown in figure 3 below

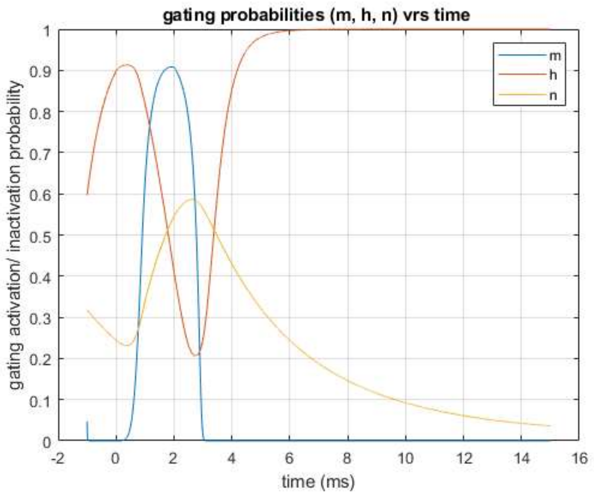

Figure 3. The time course of the gating probabilities - m, h, and n

III. Effect of Differential Time Step Changes on Computation

As mentioned earlier, the forward Euler approximation method is employed in solving the differential equations. Mathematically, this method is given as;

$$y_{i+1} = y_i + (\Delta t \times y'_{i+1}) \text{ ------------- } 8$$

Where y is the variable being solved for and Δt is the chosen time. The derivative, y'_{i+1} is computed using forward difference method, ie.

$$y_{it} := \frac{y^1_{i+1} - y_i}{t_{i+1} - t_i} \text{ ---------------- } 9$$

Thus, if the multiplicative Δt is way larger than the denominator of equation 9, then a scaling error is applied to the update and over the propagation of the computation, the solution system becomes unstable. In the case of this simulation, the effect on computational stability for five successive time updates - $1\mu s$, $5\mu s$, $10\mu s$, $20\mu s$ and $50\mu s$ are investigated. Computational stability was observed for the first three smaller time steps $1\mu s$, $5\mu s$ and $10\mu s$. The two higher times steps were however unstable and exploded with 'NANs' observed. Figure 4 shows how all three smaller time steps recapitulate the action potential whereas figure 5 presents the instability, emphasis is on the scale of the vertical axis. More so, the entire time course of 15ms is not completed as potentials diverge beyond the time presented.

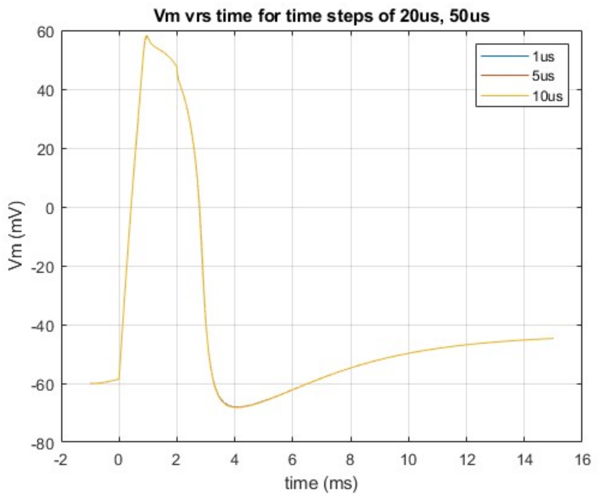

*Figure 4. Smaller three time steps 1*µs, 5µs, *10*µs *recapitulating the action potential*

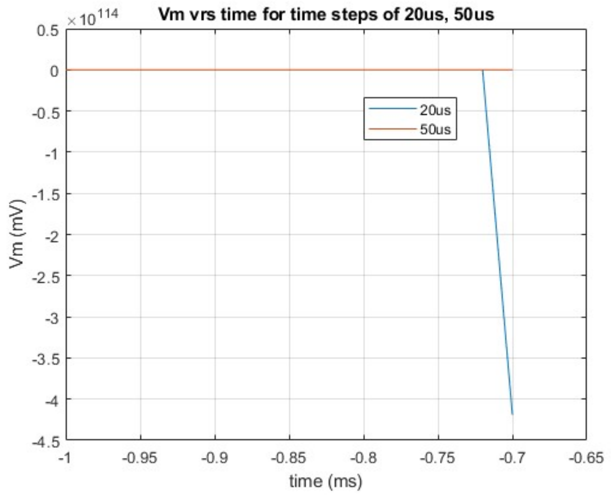

*Figure 5. Computational Instability of the two higher time steps - 20*µs *and 50*µs*. Observe the scale of the vertical axis.*

IV. Ionic Contribution to Action Potential

In the section, investigation into the effects of modifying the depolarizing sodium current and the repolarizing potassium currents is done. It must be noted that since potassium is an outward current modification of ionic current proportions that favor it tends to increase the repolarization rate hence reducing the action potential duration (APD) and vice versa. The sodium current on the other hand is favors depolarization, hence a reduced sodium current tends to reduce the depolarization rate and as such the upstroke gradient reduces with reducing sodium current resulting in right shift for the depolarization course depicted in the zoomed image in figure 8.

Four curves (figure 6-9) demonstrating these effects, though somewhat subtle, elucidates the above reasoning.

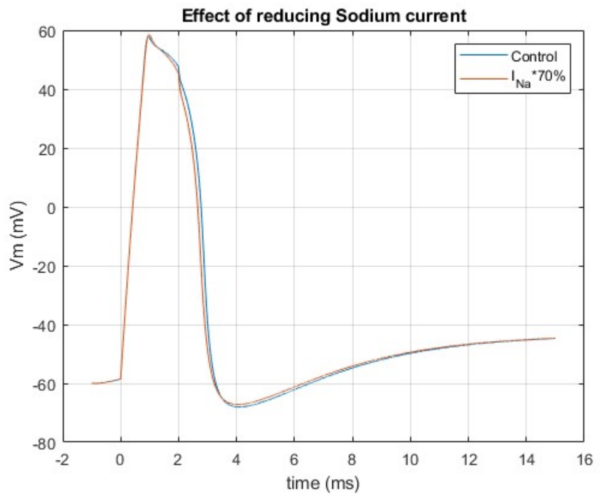

Figure 6. 30% reduced sodium current, favors outward repolarizing potassium current proportions hence reduced APD with respect to control

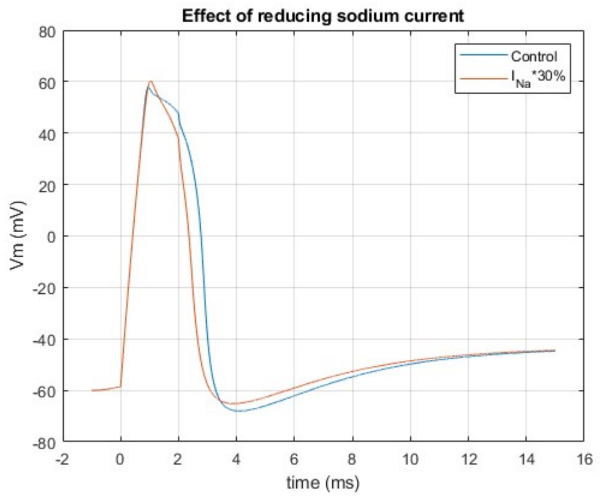

Figure 7. 70% reduced sodium current, a more drastic reduction shows a significant APD reduction with respect to the control

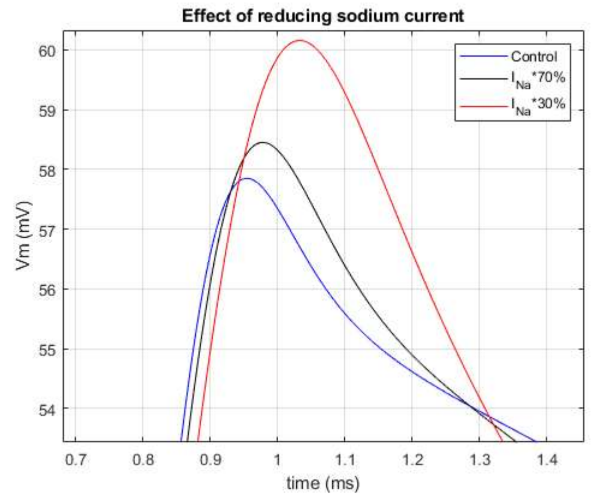

Figure 8. Decreasing upstroke/ depolarization rate shown as right shift as sodium current is reduced

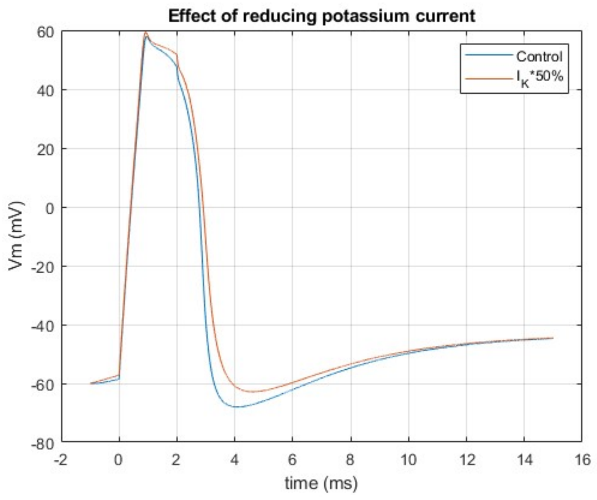

Figure 9. 50% decreased potassium current presents contrasting effect with a rather increased action potential duration with respect to the control

V. Conclusion

The Hodgkin-Huxley model since it was postulate in early 1950s has served as the basis for complex ionic cell models in recent times. This report briefly discusses the details and presents simulation results based on the underlying concepts. Access code alternatively via github link: https://github.com/Adakwaboah/Hodgkin_Huxley_Model/blob/master/HH_main.py

References

[1] A. L. HODGKIN and A. F. HUXLEY, "A quantitative description of membrane current and its application to conduction and excitation in nerve.," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, Aug. 1952.

[2] R. Plonsey and R. C. Barr, *Bioelectricity: a quantitative approach*. Springer Science & Business Media, 2007.

VI. Appendix

A. Matlab Code

Github Link:

```
close all
clear all

% Define constants here
Vrest = -60 ; % resting potential, mV
ENa = 54.2 ; % sodium Nernst potential, mV
corrected from 52.4
gNa_max = 120  ; % sodium saturation
conductance, mS/cm2
%gNa_max = 120*0.3
%% 1) Define other constant related to IK and
IL here
%%
EK = -74.7 %potassium Nernst potential, mV
gK_max = 36 %potassium saturation conductance,
mS/cm2
%gK_max = 36 *0.5
EL = -43.256 %leak Nernst potential,
calculated using the goldmann
gL_max = 0.3 % leakage saturation conductance,
mS/cm2
Cm = 1 %membrane capacitance 1uF/cm2

% time stepping and stimulus related constants
deltaT = 0.001 ; % time step, millisec
tStart = -1.000 ; % start time, millisec
tEnd = 15.000 ; % end time, millisec
nStep = ceil((tEnd-tStart)/deltaT) % number of
time steps
outputInterval = 20 ; % number of time steps
between screen output
StimDur=2; % duration of the stimulus,
millisec
Jstim=150; % strength of stimulus


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set initial value of state variables
Vm = Vrest ; % membrane potential, mV
m = 0.05293 ; % initial value of m gate
%% 2) similarly, declare initial values of h
and n gates here
%%

h = 0.59612; %initial value for h gate
n = 0.31768; %initial value of n gate



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Storage for variables to plot
plot_Vm = zeros(nStep,1) ;
plot_m = zeros(nStep, 1);
plot_h = zeros(nStep, 1);
plot_n = zeros(nStep, 1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Print a heading for the screen output
display('Hodgkin-Huxley squid axon model');
fprintf('\t Time \t Vm \t n \t\t m \t\t h\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start the simulation
tNow = tStart ;
for iStep = 1:nStep
% Compute ion currents & stimulus current
JNa = gNa_max*m*m*m*h*(Vm-ENa) ;
%% 3) write similar expressions for JK and IL
here
%%
%JK = gK_max*n*n*n*n*(Vm - EK);
JK = gK_max*n*n*n*n*(Vm - EK);
JL = gL_max*(Vm-EL);

if( 0<=tNow && tNow<StimDur ) % apply stimulus
current at t=0
Jm = Jstim ;
else
Jm = 0 ;
end

% Compute gates' opening and closing rates

alpha_m = 0.1*(25-Vm)/(exp(0.1*(25-Vm))-1);
beta_m = 4*exp(-Vm/18);
m = m + deltaT*(alpha_m*(1-m)-beta_m*m);
```

```
%% 4) Write similar expressions for n and h
gates here
%%
alpha_n = (0.01*(10-Vm))/(exp((10-Vm)/10)-1);
beta_n = 0.125*exp(-Vm/80);
n = n + deltaT*(alpha_n*(1-n) - beta_n*n);

alpha_h = 0.07*exp(-Vm/20);
beta_h = 1/(exp((30-Vm)/10)+1);
h = h + deltaT*(alpha_h*(1-h) - beta_h*n);

% Compute change in state variables
deltaVm = -deltaT * (JNa + JK + JL - Jm)/Cm ;

% Record/display state variables & other
values
plot_Vm(iStep) = Vm ;
plot_m(iStep) = m;
plot_n(iStep) = n;
plot_h(iStep) = h;
plot_time(iStep) = tNow;
if mod(iStep,outputInterval) == 0
fprintf('%8.2f %7.3f %7.5f %7.5f %7.5f\n', ...
tNow, Vm, n, m, h) ;
end % if mod(tNow)

% Update state variables
Vm = Vm + deltaVm ; % new Vm = current Vm +
change in Vm
tNow = tStart + iStep*deltaT ; % increment the
time
end % for iStep

DataNa30(:,1) = plot_time
DataNa30(:,2) = plot_Vm
% Plot the gates, probabilities, currents, Vm,
etc
% subplot(2,1,1)
plot(plot_time, plot_Vm); grid on ;
xlabel('time (ms)')
ylabel('Transmembrane Voltage(Vm), mV')
title('Time course of Action Potential')

%% 5) Similarly plot gates n, m and h against
time
%%
figure
plot(plot_time, plot_m); grid on ;
hold on
plot(plot_time, plot_h);
hold on
plot(plot_time, plot_n);
legend('m', 'h', 'n')
xlabel('time (ms)')
ylabel('gating activation/ inactivation
probability')
title('gating probabilities (m, h, n) vrs
time')

% end of file
```

B. Alternative Python Code

Github link:
https://github.com/Adakwaboah/Hodgkin_Huxley_Model/blob/master/HH_main.py

```
'''
Written by: AKwasi Darkwah Akwaboah
Description: Forward Euler Implementation of the Hodgkin
Huxley Model
Date: October 6, 2019
'''

import numpy as np
import matplotlib.pyplot as plt

Vrest = -60
ENa = 54.2
EK = -74.7
EL = -43.256

Cmem = 1 #uF/cm2
gK_bar = 36
gNa_bar = 120
gL_bar = 0.3

dt = 0.001
tStart = -1.000
tEnd = 15.000
nStep = np.int(np.ceil((tEnd-tStart)/dt)) #number of
steps
print(nStep)
StimDur = 2
Is = 150 #stimulus strength

Vm = Vrest;

m = 0.05293
h = 0.59612; #initial value for h gate
n = 0.31768; #initial value of n gate

#create storage for state variables
plot_Vm = np.zeros((nStep, 1), dtype=np.float)
plot_m = np.zeros((nStep, 1), dtype=np.float)
plot_h = np.zeros((nStep, 1), dtype=np.float)
plot_n = np.zeros((nStep, 1), dtype=np.float)
plot_time = np.empty((nStep, 1), dtype=np.float)

print('The Hodgkin Huxley Model')
tNow = tStart
for iStep in range(nStep):
    JNa = gNa_bar*m*m*m*h*(Vm - ENa)
    JK = gK_bar*n*n*n*n*(Vm - EK)
    JL = gL_bar*(Vm - EL)

    Iss = (0.0 < tNow < 2.0) * Is + (tNow >= 2.0 or tNow
< 0) * 0
    alpha_m = (0.1 * (25 - Vm)) / (np.exp((25 - Vm) / 10)
- 1)
    beta_m = 4 * np.exp(-Vm / 18)
    m = m + (dt*((alpha_m*(1-m)) - (beta_m*m)))

    alpha_n = (0.01 * (10 - Vm)) / (np.exp((10 - Vm) /
10) - 1)
    beta_n = 0.125 * np.exp((- Vm) / 80)
    n = n + (dt * ((alpha_n * (1 - n)) - (beta_n * n)))

    alpha_h = 0.07 * np.exp((- Vm) / 20)
    beta_h = 1 / (1 + np.exp((30 - Vm) / 10))
    h = h + dt * ((alpha_h * (1 - h)) - (beta_h * h))
    dVm = -dt * (JNa + JK + JL - Iss) / Cmem
    plot_Vm[iStep] = Vm
    plot_m[iStep] = m
    plot_n[iStep] = n
    plot_h[iStep] = h
    plot_time[iStep] = tNow

    Vm = Vm + dVm
    tNow = tStart + iStep * dt

plt.plot(plot_time, plot_Vm)
plt.title('time course of Transmembrane Potential')
plt.xlabel('time(ms)')
plt.ylabel('Vm (mV)')
plt.grid()
plt.show()

plt.figure()
plt.plot(plot_time, plot_m)
plt.plot(plot_time, plot_h)
plt.plot(plot_time, plot_n)
plt.title('time course of Gating probabilties, (m, h,
n)')
plt.xlabel('time (ms)')
plt.ylabel('gating (activation/ inactivation)
probability')
plt.legend(['m', 'h', 'n'], loc='best')
plt.grid()
plt.show()
```