EL 473

Biomedical Instrumentation

Lab 1:

Hudgkin-Huxley Simulation

Using Matlab

R. Yağız Mungan 8288

INTRODUCTION

Hudgkin-Huxley circuit, models the operation of action potentials in a neuron cell, via the use Na, K pumps and concentrations. This modelling has made Alan Lloyd Hodgkin and Andrew Huxley win the nobel prize in 1963.

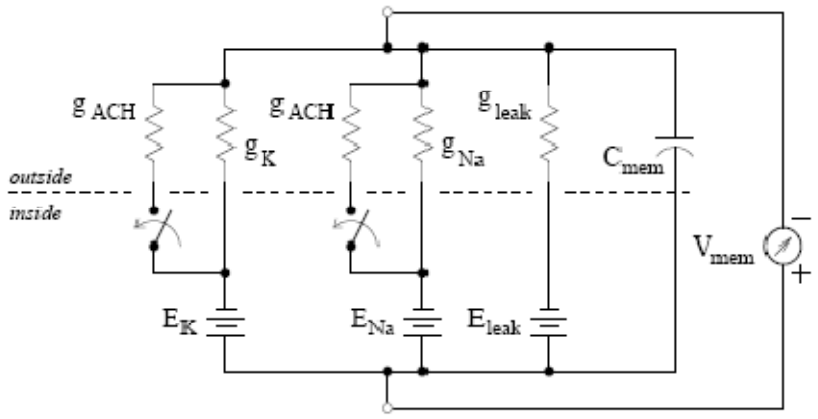

Figure 1: Hodgkin-Huxley Model

Shown in Figure 1, the circuit is composed of switches and resistances (conductors in figure). While ACH is absent in the cell, the circuit acts as the switches are open. In that condition the circuit stays at equilibrium with the leakage current playing an important role.

When ACH is existent in the cell, the switches are closed and the equilibrium of the cell/circuit is disturbed. This causes change in currents flowing, which changes the voltage difference.

The disturbance in Vmem is such that it first increases from a value like -60mV to 20mV then; it drops to its initial value after making an overshoot. This behaviour is simulated with the conductors gACH, gK and gNa through the given equations.

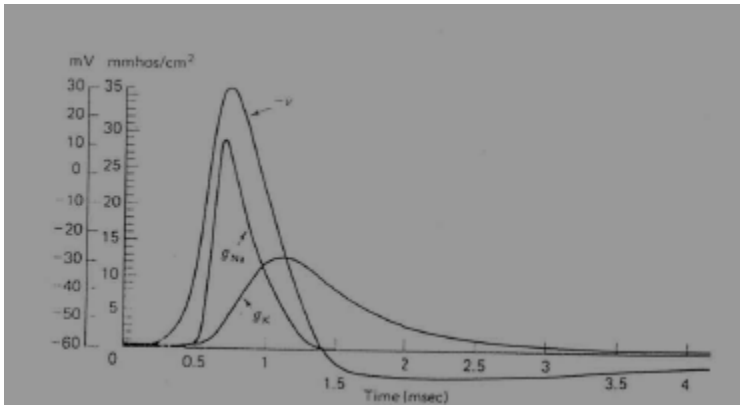

Figure 2: Action-Potential

$$\frac{\partial m}{\partial t} = \frac{1}{\tau_m(\nu)}\left(m_\infty(\nu) - m\right)$$

$$\frac{\partial n}{\partial t} = \frac{1}{\tau_n(\nu)}\left(n_\infty(\nu) - n\right)$$

$$\frac{\partial h}{\partial t} = \frac{1}{\tau_h(\nu)}\left(h_\infty(\nu) - h\right)$$

$$\tau_m(\nu) = \frac{1}{\alpha_m(\nu) + \beta_m(\nu)}, \quad m_\infty(\nu) = \frac{\alpha_m(\nu)}{\alpha_m(\nu) + \beta_m(\nu)}$$

$$\tau_n(\nu) = \frac{1}{\alpha_n(\nu) + \beta_n(\nu)}, \quad n_\infty(\nu) = \frac{\alpha_n(\nu)}{\alpha_n(\nu) + \beta_n(\nu)}$$

$$\tau_h(\nu) = \frac{1}{\alpha_h(\nu) + \beta_h(\nu)}, \quad h_\infty(\nu) = \frac{\alpha_h(\nu)}{\alpha_h(\nu) + \beta_h(\nu)}$$

$$\alpha_m(\nu) = \frac{0.1(25-\nu)}{e^{0.1(25-\nu)} - 1}, \quad \beta_m(\nu) = 4e^{-\nu/18}$$

$$\alpha_n(\nu) = \frac{0.01(10-\nu)}{e^{0.1(10-\nu)} - 1}, \quad \beta_n(\nu) = 0.125e^{-\nu/80}$$

$$\alpha_h(\nu) = 0.07e^{-\nu/20}, \quad \beta_h(\nu) = \frac{1}{e^{0.1(30-\nu)} + 1}$$

Hodgkin-Huxley Equations

MATLAB CODES

1) Determining Threshold

Here is the code used in the first two parts of the simulation:

```
clear all;


EK=-82;                          %constans are defined by scaling all the
values to 10^(-3)
ENa=45;
Eleak=-59.4011;
Vrest=-70;
gKbar=36;
gNabar=120;
gleak=0.3;
Cmem=1;
dt=1*10^(-3);


tmax = 10;                       %pointer for time simulations
```

```
t = 0:dt:tmax;

V(1)=0;                          %initial values of the Hudkin-Huxley
functions
Vmem(1) = Vrest;
m(1) = am(0) / (am(0)+bm(0));
h(1) = ah(0) / (ah(0)+bh(0));
n(1) = an(0) / (an(0)+bn(0));
gach(1) = 0.00;
I(1)=0;
Ic(1)=0;
INa(1)= ((gNabar * (m(1)^3*h(1))*(Vrest-ENa)));
IK(1)=(gKbar * (n(1)^4) * (Vrest-EK));
Ileak(1)=(gleak * (Vrest-Eleak));
Iach(1)=gach(1)*(Vrest-EK)+(gach(1)*(Vrest-ENa));
x=0;
Gach=0;

while(x==0)                      %in each loop gach value is taken constant
  Gach= Gach+0.001;              %after in a for loop if the action
potential is achivied
                                 %x becomes 1, thus the program exits the
                                 %while loop and the last values which the
                                 %action potential is achieved are saved
  for i=1:length(t)-1
  if (t(i)<1)||(t(i)>4)           %gach pulse is created here ( for the
first part (t(i)<1)||(t(i)>2)
  gach(i+1)=0;                                              %for the
second parts (t(i)<1)||(t(i)>2.5)
  else
%(t(i)<1)||(t(i)>3) and (t(i)<1)||(t(i)>4)
  gach(i+1)=Gach;
  end

  m(i+1) = m(i) + dt * (am(V(i))*(1-m(i)) - bm(V(i))*m(i));
%Hudgkin-Huxley functions and
  h(i+1) = h(i) + dt * (ah(V(i))*(1-h(i)) - bh(V(i))*h(i));
%Voltages&Currents are updated here
  n(i+1) = n(i) + dt * (an(V(i))*(1-n(i)) - bn(V(i))*n(i));
  Vmem(i+1) = Vmem(i) + (dt/Cmem) * I(i);
  V(i+1)=Vmem(i+1)-Vrest;
  Ic(i+1)=Cmem*((Vmem(i+1)-Vmem(i))*dt);
  INa(i+1)= ((gNabar * (m(i+1)^3*h(i+1))*(Vmem(i+1)-ENa)));
  IK(i+1)=(gKbar * (n(i+1)^4) * (Vmem(i)-EK));
  Ileak(i+1)=(gleak * (Vmem(i+1)-Eleak));
  Iach(i+1)=gach(i+1)*(Vmem(i+1)-EK)+(gach(i+1)*(Vmem(i+1)-ENa));
  I(i+1)=Cmem*((Vmem(i+1)-Vmem(i))*dt)-(gach(i+1)*(Vmem(i+1)-EK))-
(gach(i+1)*(Vmem(i+1)-ENa))-((gNabar * (m(i+1)^3*h(i+1)) * (Vmem(i+1)-
ENa))+(gKbar * (n(i+1)^4) * (Vmem(i+1)-EK))+(gleak * (Vmem(i+1)-Eleak)));

  if(Vmem(i)>(-50))
      x=1;                       %signal for exiting the loop when action
potential is initiated
```

```
  end
  end
gach(i+1)=0;                    %to match the size of t(i) to enable
plotting
  end

subplot (3,1,1); plot(t,I);                                          %total
current is plotted
title('I(i) vs. time'); axis([t(1) t(length(t)) -100 200])
ylabel('I')

subplot (3,1,2); plot(t,Vmem);                                       %Vmem is
plotted
title('Vmem vs. time'); axis([t(1) t(length(t)) -100 100])
xlabel('time (ms)'); ylabel('Vmem (mV)')

subplot (3,1,3); plot(t,gach);                                       %gach is
plotted
title('Vmem vs. time'); axis([t(1) t(length(t)) 0 1])
xlabel('time (ms)'); ylabel('gach')
return
```

Output of the simulation for the first part:

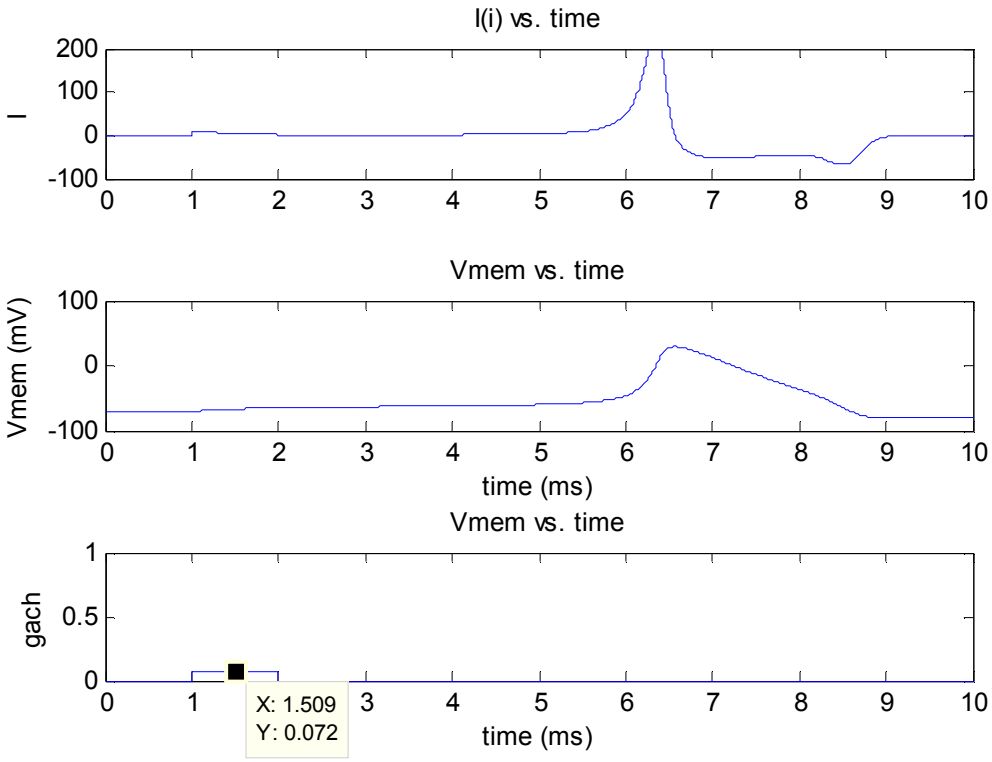

Figure 3: Action potential with t=1ms;

Here gach is 0.072 and maximum value of Vmem is 29.35mV for a pulse width of 1 ms.

2) Strength/Duration Curve

In the second part the same code is used with only changing the pulse width. As the pulse width increased the minimum amount of gach required for an action potential has decreased in the sense the area under the curve relatively stayed stable.

Output of the simulation of the first segment of the second part:

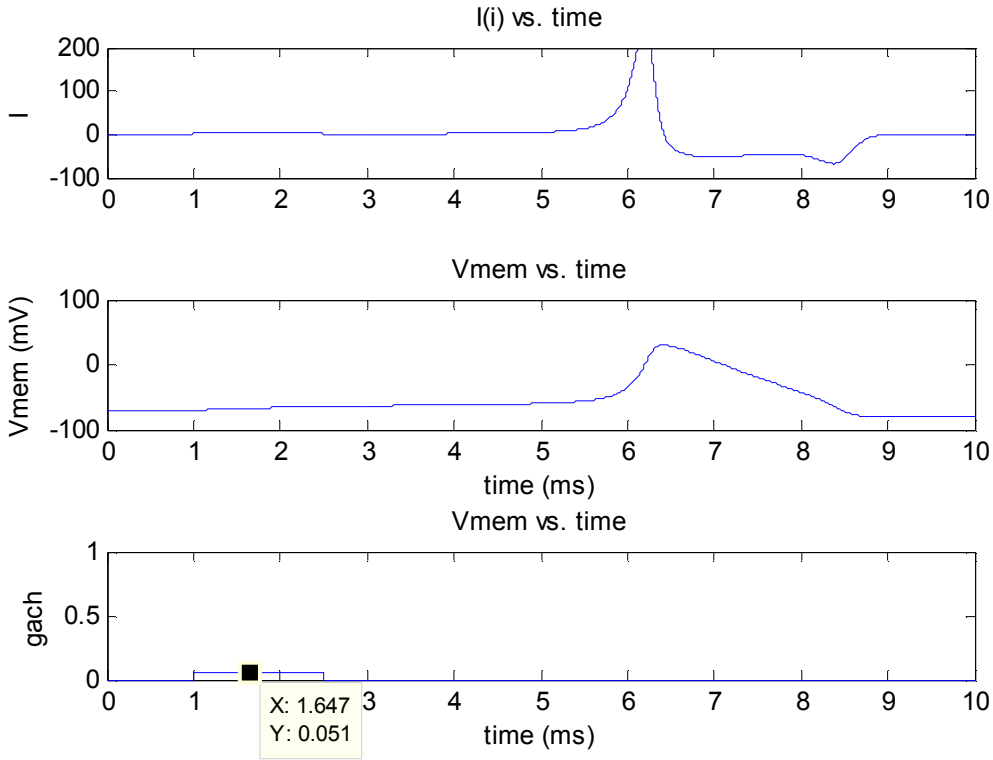

Figure 4: Action potential with t=1.5ms;

Here gach level is 0.051 and maximum value of Vmem is 30.04mV for a pulse width of 1.5 ms.

Output of the simulation of the second segment of the second part:

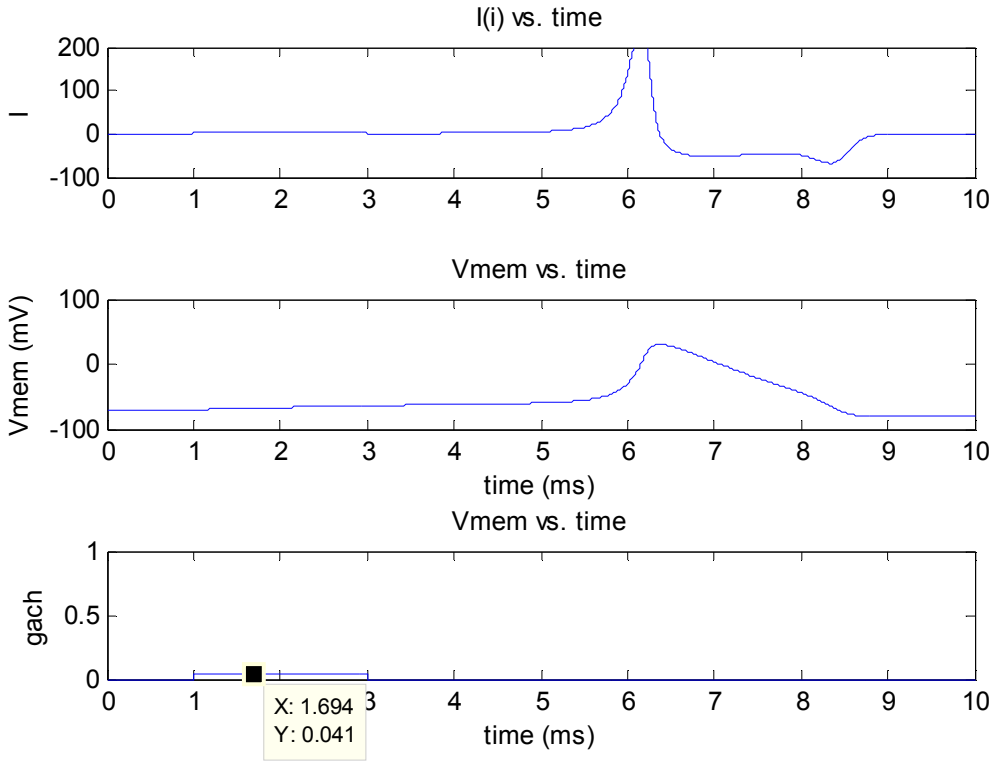

Figure 5: Action potential with t=2ms;

Here gach is 0.041 and maximum value of Vmem is 30.46mV for a pulse width of 2ms.

Output of the simulation of the third segment of the second part:

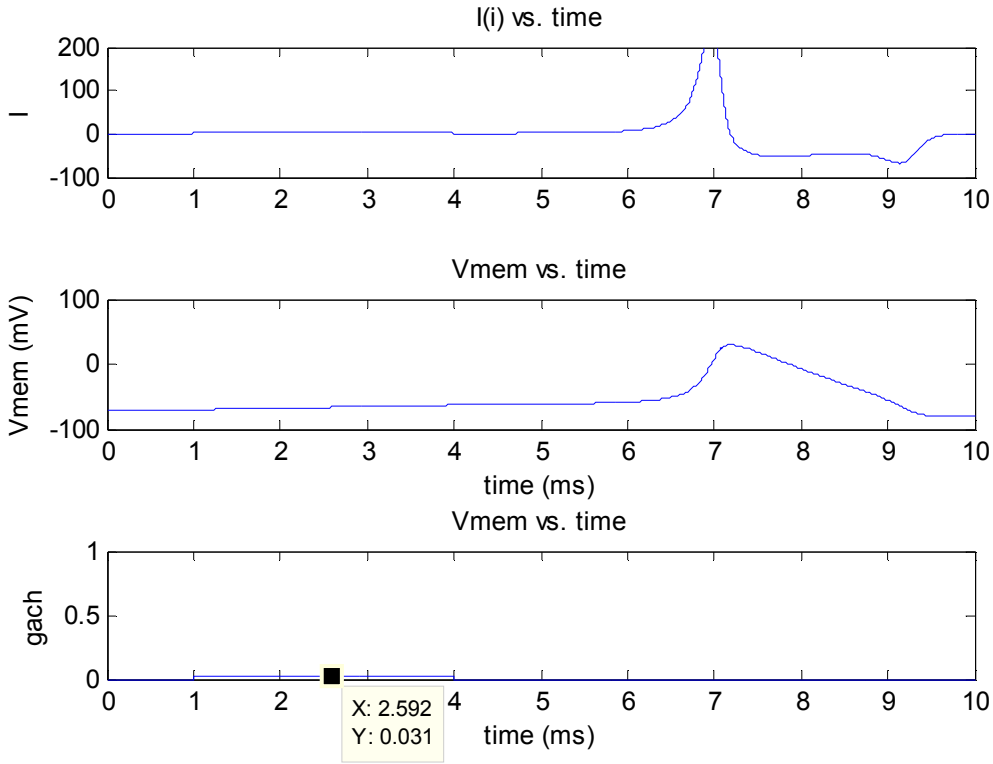

Figure 5: Action potential with t=3ms;

Here gach is 0.031 and maximum value of Vmem is 29.8mV for a pulse width of 2ms.

3) Refractory Period

To have another action potential a certain time must pass, this is the reason that neuron cells can not send signals continuously, rather they some kind of tire. Here this time difference period is simulated taking gach level as 1.5 multiple of the minimum value for the 1 second pulse.

Here is the code used in the first two parts of the simulation:

```
clear all;


EK=-82;
ENa=45;
Eleak=-59.4011;
Vrest=-70;
gKbar=36;
gNabar=120;
```

```
gleak=0.3;
Cmem=1;
dt=1*10^(-3);


tmax = 22;
t = 0:dt:tmax;                          %maximum time is increased to be able
to observe the second
                                        %action potential

V(1)=0;
Vmem(1) = Vrest;
m(1) = am(0) / (am(0)+bm(0));
h(1) = ah(0) / (ah(0)+bh(0));
n(1) = an(0) / (an(0)+bn(0));
gach(1) = 0.00;
I(1)=0;
Ic(1)=0;
INa(1)= ((gNabar * (m(1)^3*h(1))*(Vrest-ENa)));
IK(1)=(gKbar * (n(1)^4) * (Vrest-EK));
Ileak(1)=(gleak * (Vrest-Eleak));
Iach(1)=gach(1)*(Vrest-EK)+(gach(1)*(Vrest-ENa));
x=0;
Gach=0;


  for i=1:length(t)-1

if ((1<t(i))&&((2>t(i))||(16<t(i)))&&(17>t(i)))      %for refractory period
gach(i+1)=0.108;                                    %the second condition
of the or is changed
else                                                %while searching for
the second action-potential
gach(i+1)=0;
end
m(i+1) = m(i) + dt * (am(V(i))*(1-m(i)) - bm(V(i))*m(i));
h(i+1) = h(i) + dt * (ah(V(i))*(1-h(i)) - bh(V(i))*h(i));
n(i+1) = n(i) + dt * (an(V(i))*(1-n(i)) - bn(V(i))*n(i));
Vmem(i+1) = Vmem(i) + (dt/Cmem) * I(i);
V(i+1)=Vmem(i+1)-Vrest;
Ic(i+1)=Cmem*((Vmem(i+1)-Vmem(i))*dt);
INa(i+1)= ((gNabar * (m(i+1)^3*h(i+1))*(Vmem(i+1)-ENa)));
IK(i+1)=(gKbar * (n(i+1)^4) * (Vmem(i)-EK));
Ileak(i+1)=(gleak * (Vmem(i+1)-Eleak));
Iach(i+1)=gach(i+1)*(Vmem(i+1)-EK)+(gach(i+1)*(Vmem(i+1)-ENa));
I(i+1)=Cmem*((Vmem(i+1)-Vmem(i))*dt)-(gach(i+1)*(Vmem(i+1)-EK))-
(gach(i+1)*(Vmem(i+1)-ENa))-((gNabar * (m(i+1)^3*h(i+1)) * (Vmem(i+1)-
ENa))+(gKbar * (n(i+1)^4) * (Vmem(i+1)-EK))+(gleak * (Vmem(i+1)-Eleak)));

end

subplot (3,1,1); plot(t,I);
title('I(i) vs. time'); axis([t(1) t(length(t)) -100 200])
ylabel('I')

subplot (3,1,2); plot(t,Vmem);
title('Vmem vs. time'); axis([t(1) t(length(t)) -100 100])
xlabel('time (ms)'); ylabel('Vmem (mV)')
```

```
subplot (3,1,3); plot(t,gach);
title('Vmem vs. time'); axis([t(1) t(length(t)) 0 1])
xlabel('time (ms)'); ylabel('gach')
return
```

Output of the simulation of the third part:

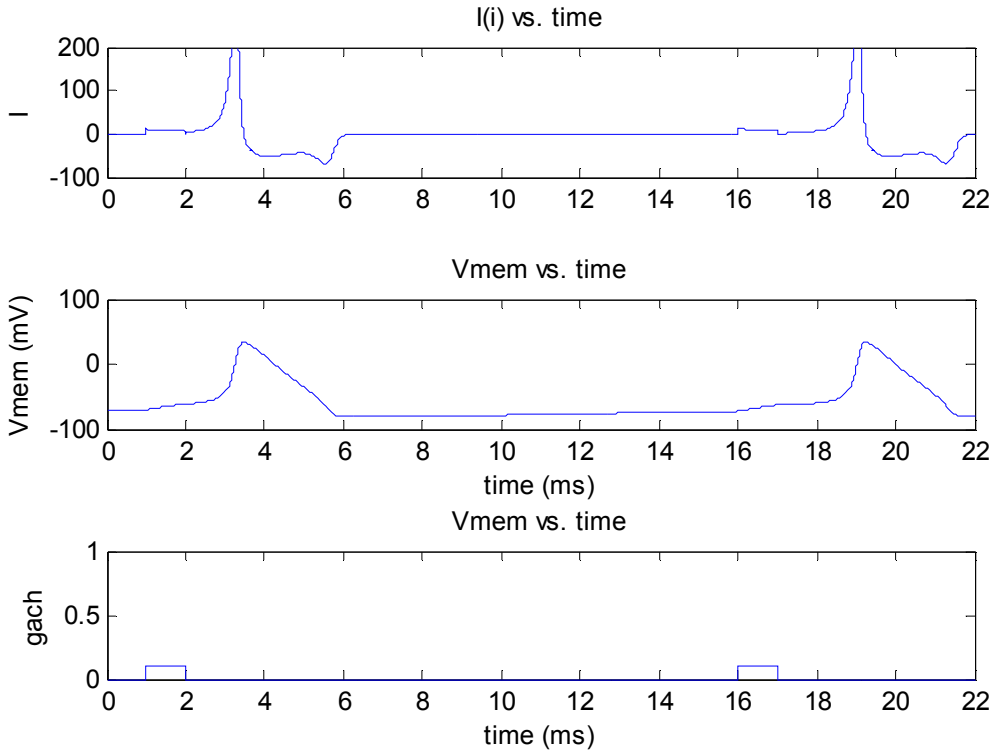

Figure 6: Refractory Period with gach=1.5gachmin and t=1ms;

After 14 ms another action-potential can be observed. Before that time applying gach makes small fluctuations in the Vmem however can not cause another action-potential of same power and duration.

Supplementary MATLAB Codes:

These codes are used to implement the functions αm, αh, αn, βm, βh and βn.

for αm:

```
function [am] = am(V)
```

```
am=(0.1*(25-V))/(exp(0.1*(25-V))-1);
return
```

for αh:

```
function [ah] = ah(V)

ah = 0.07*exp(-V/20);
return
```

for αn:

```
function [an] = an(V)

an = (0.01*(10-V))/(exp(0.1*(10-V))-1);
return
```

for βm:

```
function [bm] = bm(V)
bm=4*exp(-V/18);
return
```

for βh:

```
function [bh] = bh(V)

bh = 1/(exp(0.1*(30-V))+1);
return
```

for βn:

```
function [bn] = bn(V)

bn = 0.125*exp(-V/80);
return
```