[Tutorial 2.2]

Contents

Question 1: voltage gated channel

```
%setting up the parameters
EL = -70; %leaky = -70mV
Rm = 5; % resistence = 5ohms
Cm = 2; % membrane capacitance = 5nF
Vth = -50; % threshold -50mV
Vreset = -65; % reset voltage -65mV
Vpeak = 50; %voltage peak = -50mV

%setting up time
dt = 0.0001; %0.1ms step size
tvec = 0:dt:2; % setting up the tvector 2seconds
tau_ref = 25; % tau reference 2.5*1ms= 2.5ms

%setting up voltage vector
vvec1 = zeros(size(tvec)); %voltage size same as tvec
vvec1(1) = EL; %initialize first element as leaky
```

setting up Euler's method

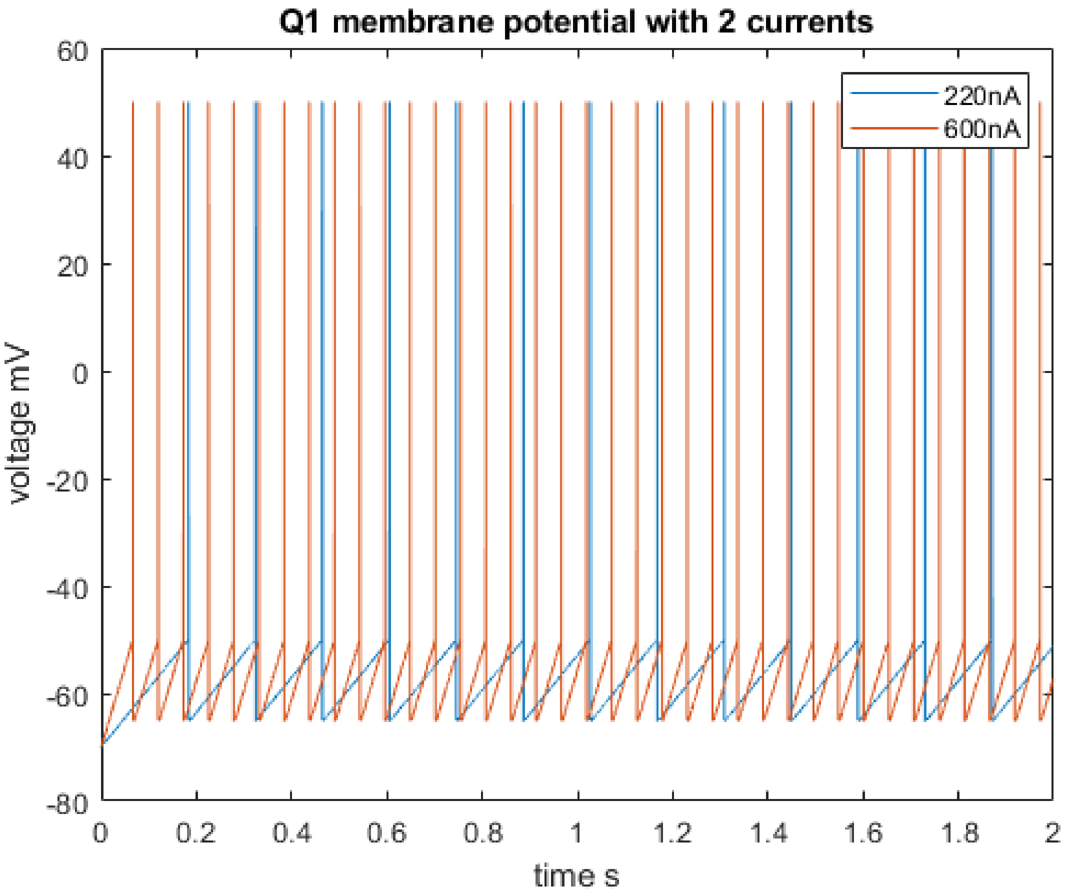
```
for range = 100:100:600 %setting up applied current constants
    pos = 1; %pos of iapp vec
       countvec1 = zeros(6);% define the spike vector
   iapp1 = zeros(6); %current vector
   vvec1avg = zeros(6);% the vm average vector
```

```
      vvec1avg(pos) = 0; % initialize = 0
   iapp1(pos) = range; % first element as the first applied current range
   refrac = 1 + tau_ref;
   countvec1(pos) = 0; % spike array counts
   vvec1add = 0; % define an sum vector for membrane potential
for i = 2:length(tvec)
    if (refrac > tau_ref) %if time > refractory period, start firing
    %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
    dVm_over_dt = ((EL-vvec1(i-1)) / Rm + iapp1(pos))/Cm;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    vvec1(i) = vvec1(i-1) + dVm_over_dt * dt;
    else %or it's just gonna stay at reset, while time increases
        refrac = refrac+1;
        vvec1(i) = Vreset;
    end
    if (vvec1(i) > Vth)% if vvec element is higher than V threshold
        vvec1(i-1) = Vpeak; %force the element to peak;
        vvec1(i) = Vreset;
        countvec1(pos)= countvec1(pos)+1; %spike count +1
        refrac = 0; % resset refractory period.
    end
    vvec1add = vvec1(i-1)+vvec1(i);
end
vvec1avg(pos) = vvec1add / length(tvec); % get the average
pos = pos + 1;
vvec1add = 0; % reset the sum to 0 to start another 'for' loop.
end
```

setting up Euler's method (membrane potential)

```
for range = 220:380:600 %setting up applied current constants
       pos = 1; %pos of iapp vec
    iapp = zeros(2); %current vector
   iapp(pos) = range; % first element as the first applied current range
   refrac = 1 + tau_ref; %refractory time initialization

for i = 2:length(tvec)
    if (refrac > tau_ref) % if time achieves tau, start firing
    %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
    dVm_over_dt = ((EL-vvec1(i-1)) / Rm + iapp(pos))/Cm;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    vvec1(i) = vvec1(i-1) + dVm_over_dt * dt;
    else%or it's just gonna stay at reset, while time increases
        refrac = refrac+1;
        vvec1(i) = Vreset;
    end
    if (vvec1(i) > Vth)% if vvec element is higher than V threshold
        vvec1(i-1) = Vpeak; %force the element to peak;
        vvec1(i) = Vreset;
        refrac = 0; %reset the refractory period
    end
end
pos = pos + 1; % move on to the next position array

figure(1)
```

```
plot(tvec,vvec1)
hold on
title('Q1 membrane potential with 2 currents')
xlabel('time s')
ylabel('voltage mV')
end
legend({'220nA','600nA'})
```



Q2 Threshold Increase

```
%implement the new voltage threshold vector.
Vthvec = zeros(size(tvec)); %Vth is a variable
Vth0 = -50;%initial condition is -50mV
Vthvec(1) = Vth0; %initial condition is -50mV
Vthmax = 200; % max condition is 200mV
vvec2 = zeros(size(tvec));
vvec2(1) = EL;
%new parameters
tau_vth = 1;%refractory time const = 1ms
```

setting up Euler's method

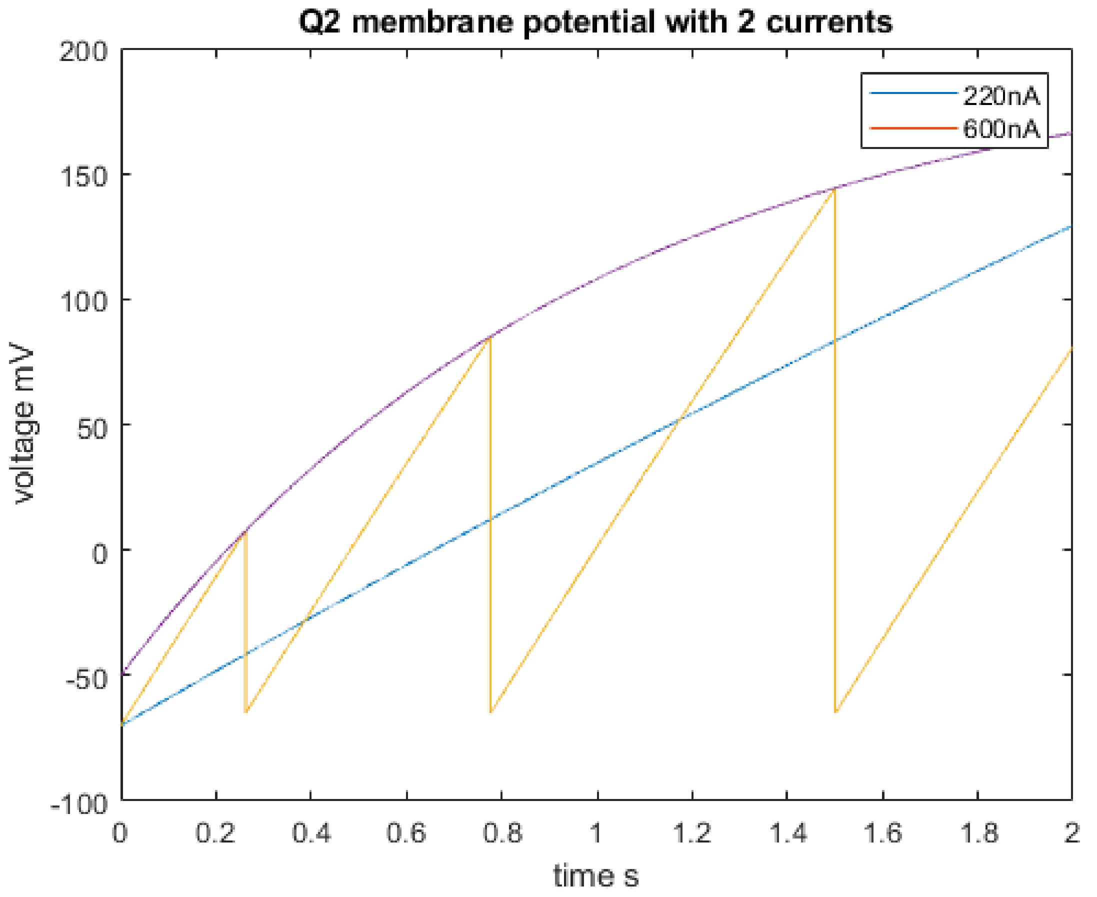
```
for range = 100:100:600 %setting up applied current constants
       pos = 1; %pos of iapp vec
          countvec2 = zeros(6);% define the spike vector
   iapp2 = zeros(6); %current vector
   iapp2(pos) = range; % first element as the first applied current range
      vvec2avg = zeros(6);% the vm average vector
```

```
      vvec2avg(pos) = 0; % initialize = 0
      vvec2add = 0;
   countvec2(pos) = 0; % spike array counts
for i = 2:length(tvec)
    %Euler's for V threshold
    %dVth/dt = (Vth0 - Vth)/tau_vth
    dVth_over_dt = (Vth0-Vthvec(i-1))/tau_vth;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    Vthvec(i) = Vthvec(i-1) + dVth_over_dt * dt;
    %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
    dVm_over_dt = ((EL-vvec2(i-1)) / Rm + iapp2(pos))/Cm;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    vvec2(i) = vvec2(i-1) + dVm_over_dt * dt;

    if (vvec2(i) > Vthvec(i))% if vvec element is higher than V threshold
        vvec2(i) = Vreset;
        Vth0 = Vthmax;
        countvec2(pos) = countvec2(pos) + 1;
    end
    vvec2add = vvec2(i-1)+vvec2(i);
end
vvec2avg(pos) = vvec2add/length(tvec);
pos = pos + 1; % move on to the next position array
vvec2add = 0;
end
```

setting up Euler's method (membrane potential)

```
for range = 220:380:600 %setting up applied current constants
      pos = 1; %pos of iapp vec
    iapp = zeros(2); %current vector
   iapp(pos) = range; % first element as the first applied current range
for i = 2:length(tvec)
    %Euler's for V threshold
    %dVth/dt = (Vth0 - Vth)/tau_vth
    dVth_over_dt = (Vth0-Vthvec(i-1))/tau_vth;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    Vthvec(i) = Vthvec(i-1) + dVth_over_dt * dt;
    %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
    dVm_over_dt = ((EL-vvec2(i-1)) / Rm + iapp(pos))/Cm;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    vvec2(i) = vvec2(i-1) + dVm_over_dt * dt;

    if (vvec2(i) > Vthvec(i))% if vvec element is higher than V threshold
        vvec2(i) = Vreset;
        Vth0 = Vthmax;
    end
end
pos = pos + 1; % move on to the next position array

figure(2)
plot(tvec,vvec2)
hold on
plot(tvec,Vthvec)
title('Q2 membrane potential with 2 currents')
```

```
xlabel('time s')
ylabel('voltage mV')
end
legend({'220nA','600nA'})
```



Q3 Refractory Conductance with Threshold Increase

```
% declare additional parameters
EK = -80; %Epotassium = -80mV
tau_gref = 0.2; %tau conductance refractory0.2ms
%declare refractory conductance vectors
Gvec = zeros(size(tvec)); % declare Gvec
Gvec(1) = 0; %initialize first value to 0;
deltag = 2; %delta g = 2uS
vvec3 = zeros(size(tvec));
vvec3(1) = EL;
```

setting up Euler's method

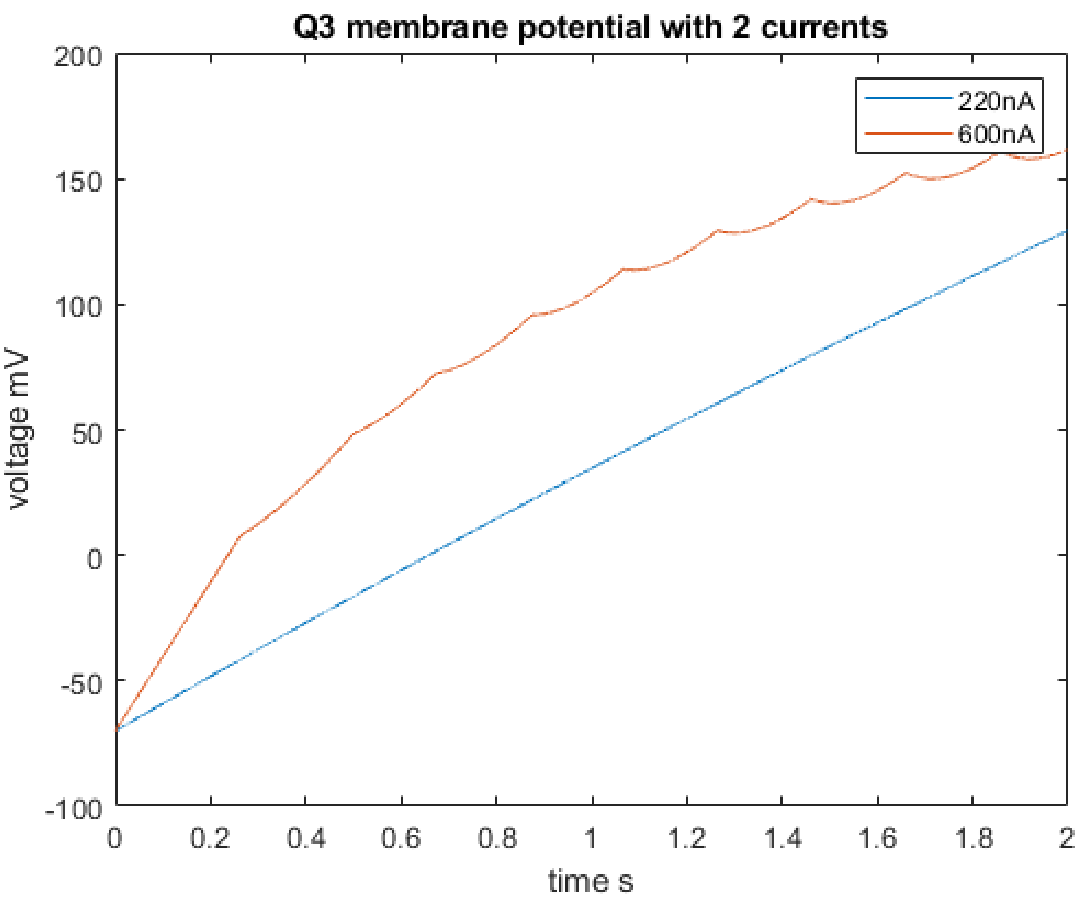
```
   countvec3 = zeros(6);% define the spike vector
   iapp3 = zeros(6); %current vector
for range = 100:100:600 %setting up applied current constants
       pos = 1; %pos of iapp vec
   iapp3(pos) = range; % first element as the first applied current range
   countvec3(pos) = 0; % spike array counts
   vvec3avg = zeros(6);
   vvec3avg(pos) = 0;
   vvec3add = 0;
```

```
for i = 2:length(tvec)
    %Euler's for Gthreshold
    %dGref(t)/dt = -Gref(t)/tau_gref
    dGref_over_dt = -Gvec(i-1)/tau_gref;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    Gvec(i) = Gvec(i-1) + dGref_over_dt *dt;
    %Euler's for V threshold
    %dVth/dt = (Vth0 - Vth)/tau_vth
    dVth_over_dt = (Vth0-Vthvec(i-1))/tau_vth;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    Vthvec(i) = Vthvec(i-1) + dVth_over_dt * dt;
    %dVm/dt = ((El - Vm) /Rm + Gref*(EK-Vm) Iapp)/Cm.
    dVm_over_dt = ((EL-vvec3(i-1)) / Rm + Gvec(i-1)*(EK-vvec3(i-1)) +iapp3(pos))/Cm;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    vvec3(i) = vvec3(i-1) + dVm_over_dt * dt;
    if (vvec3(i) > Vthvec(i))% if vvec element is higher than V threshold
        Gvec(i) = Gvec(i-1) + 2;
        Vth0 = Vthmax;
        countvec3(pos) = countvec3(pos) + 1;
    end
    vvec3add = vvec3(i-1)+vvec3(i);
end
vvec3avg(pos) = vvec3add / length(tvec);
pos = pos + 1; % move on to the next position array
vvec3add = 0;
end
```

setting up Euler's method (membrane potential)

```
for range = 220:380:600 %setting up applied current constants
       pos = 1; %pos of iapp vec
    iapp = zeros(2); %current vector
   iapp(pos) = range; % first element as the first applied current range
for i = 2:length(tvec)
    %Euler's for Gthreshold
    %dGref(t)/dt = -Gref(t)/tau_gref
    dGref_over_dt = -Gvec(i-1)/tau_gref;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    Gvec(i) = Gvec(i-1) + dGref_over_dt *dt;
    %Euler's for V threshold
    %dVth/dt = (Vth0 - Vth)/tau_vth
    dVth_over_dt = (Vth0-Vthvec(i-1))/tau_vth;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    Vthvec(i) = Vthvec(i-1) + dVth_over_dt * dt;
    %dVm/dt = ((El - Vm) /Rm + Gref*(EK-Vm) Iapp)/Cm.
    dVm_over_dt = ((EL-vvec3(i-1)) / Rm + Gvec(i-1)*(EK-vvec3(i-1)) +iapp(pos))/Cm;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    vvec3(i) = vvec3(i-1) + dVm_over_dt * dt;
    if (vvec3(i) > Vthvec(i))% if vvec element is higher than V threshold
        Gvec(i) = Gvec(i) + 2;
        Vth0 = Vthmax;
    end
end
pos = pos + 1; % move on to the next position array
figure(3)
```

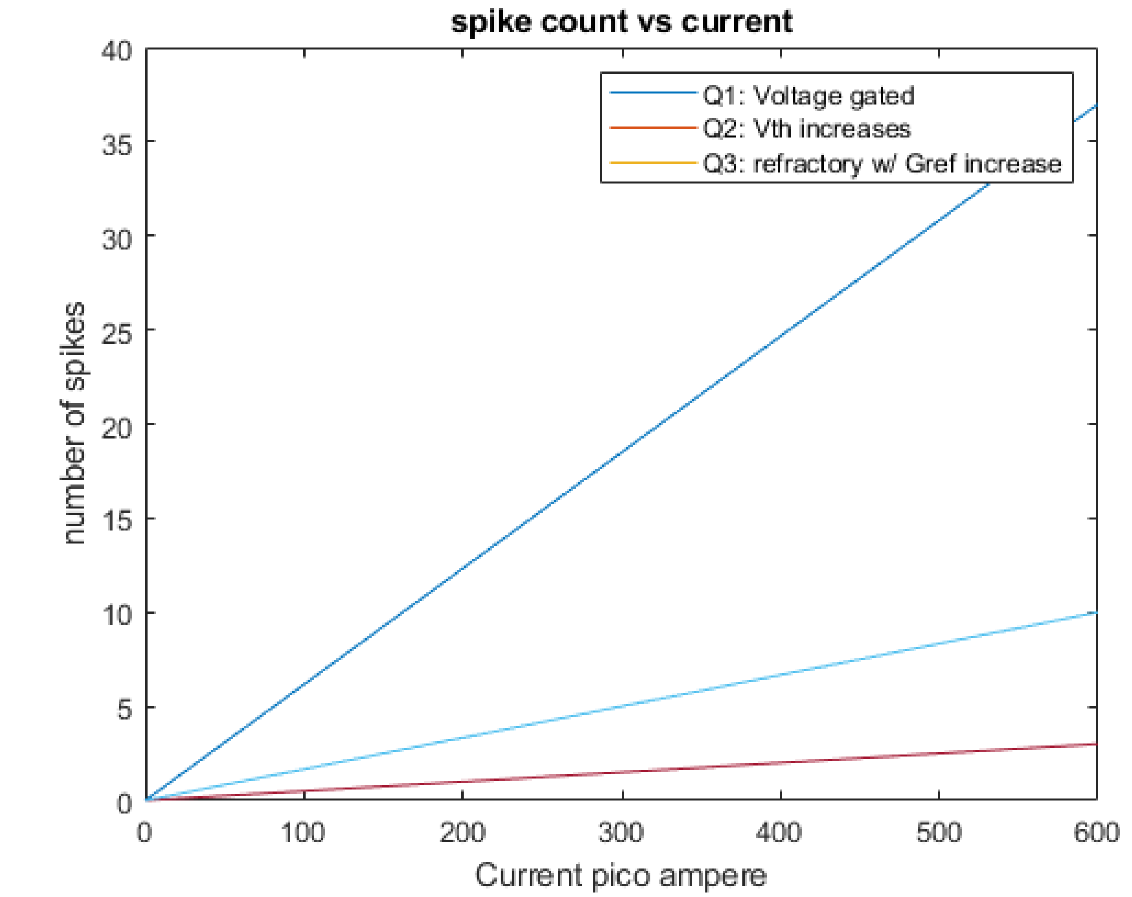
```
plot(tvec,vvec3)
hold on
title('Q3 membrane potential with 2 currents')
xlabel('time s')
ylabel('voltage mV')
end
legend({'220nA','600nA'})
```

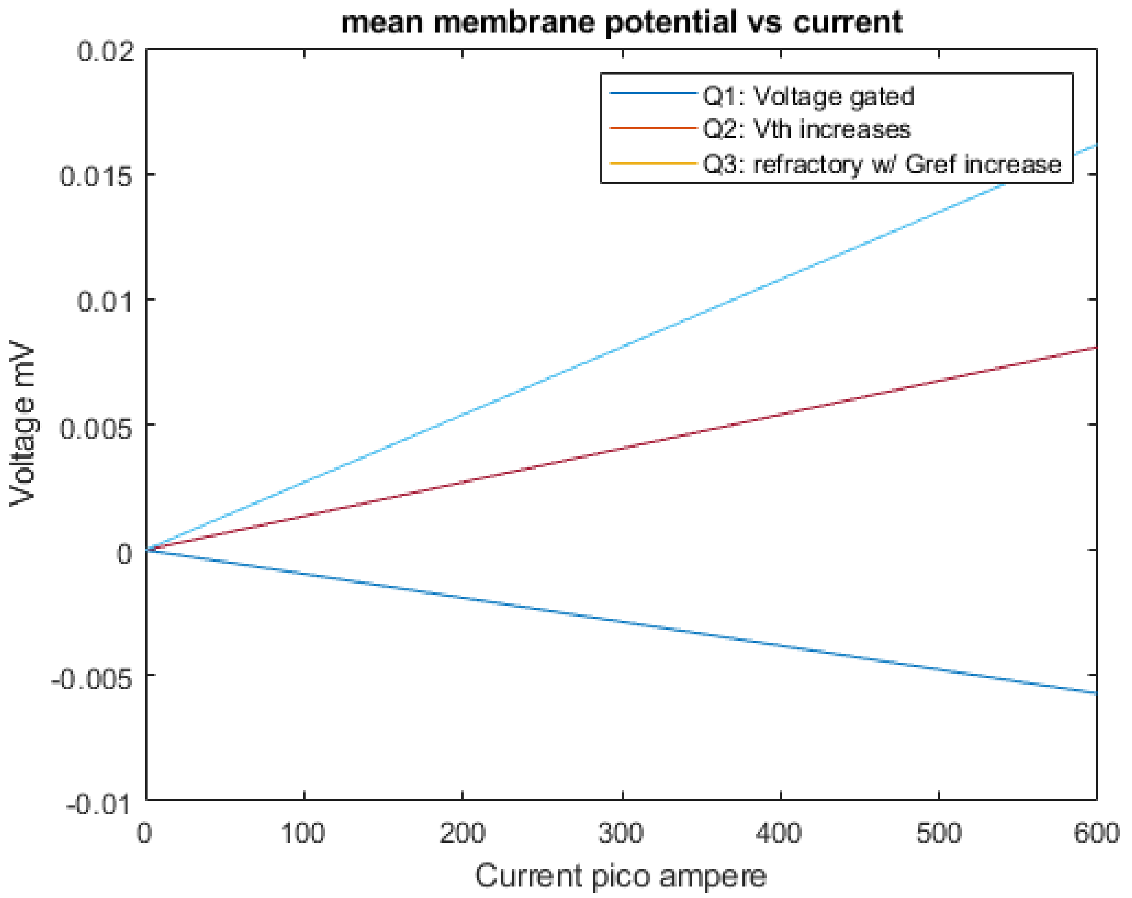


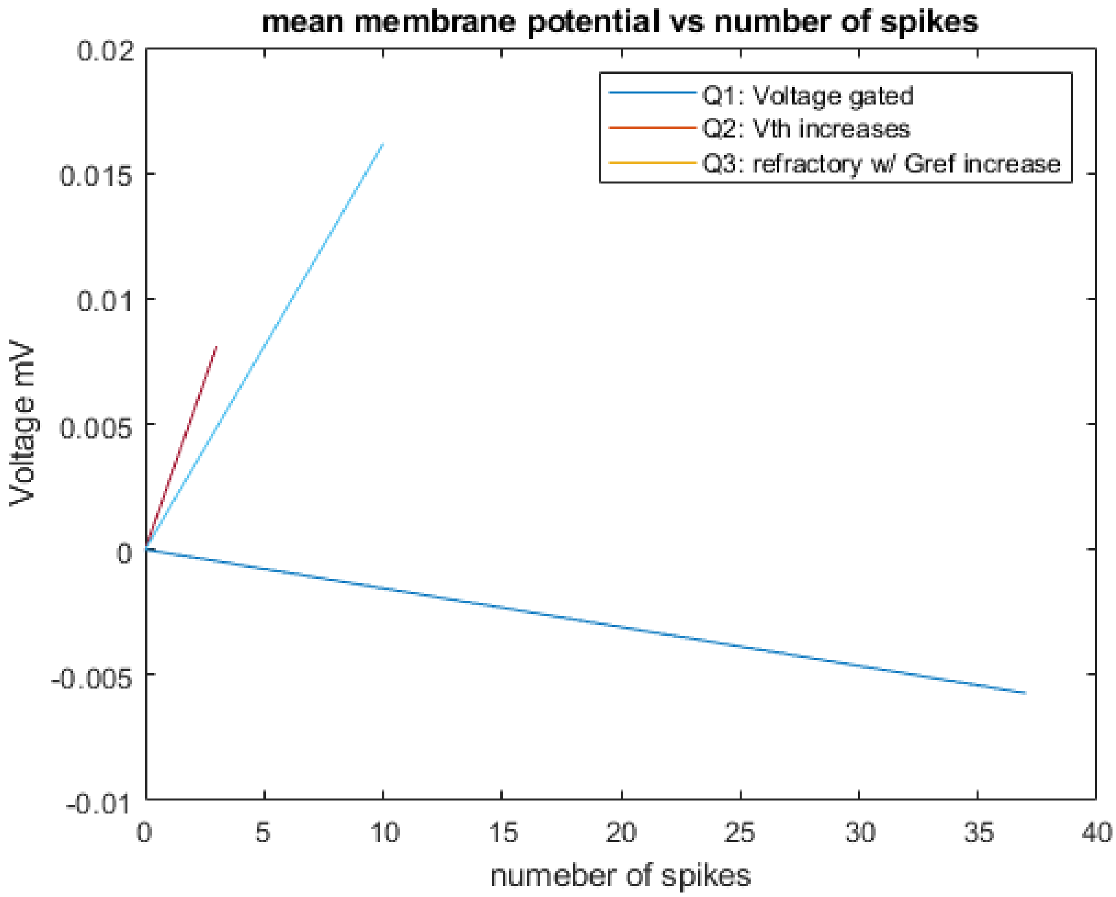
Drawing Requirements

```
% number of spikes vs applied current
figure(4)
plot(iapp1, countvec1)
hold on
plot(iapp2, countvec2)
hold on
plot(iapp3, countvec3)
title('spike count vs current')
xlabel('Current pico ampere')
ylabel('number of spikes')
legend({'Q1: Voltage gated','Q2: Vth increases','Q3: refractory w/ Gref increase'})

%mean membrane potential vs applied current.
figure(5)
plot(iapp1, vvec1avg)
hold on
plot(iapp2, vvec2avg)
hold on
plot(iapp3, vvec3avg)
```

```
title('mean membrane potential vs current')
xlabel('Current pico ampere')
ylabel('Voltage mV')
legend({'Q1: Voltage gated','Q2: Vth increases','Q3: refractory w/ Gref increase'})

% mean membrane vs spikes fired
figure (6)
plot(countvec1, vvec1avg)
hold on
plot(countvec2, vvec2avg)
hold on
plot(countvec3, vvec3avg)
title('mean membrane potential vs number of spikes')
xlabel('numeber of spikes')
ylabel('Voltage mV')
legend({'Q1: Voltage gated','Q2: Vth increases','Q3: refractory w/ Gref increase'})
```


spike count vs current
Q1: Voltage gated
Q2: Vth increases
Q3: refractory w/ Gref increase
number of spikes
40
35
30
25
20
15
10
5
0
0
100
200
300
400
500
600
Current pico ampere


mean membrane potential vs current
Q1: Voltage gated
Q2: Vth increases
Q3: refractory w/ Gref increase
Voltage mV
0.02
0.015
0.01
0.005
0
-0.005
-0.01
0
100
200
300
400
500
600
Current pico ampere



Explanation

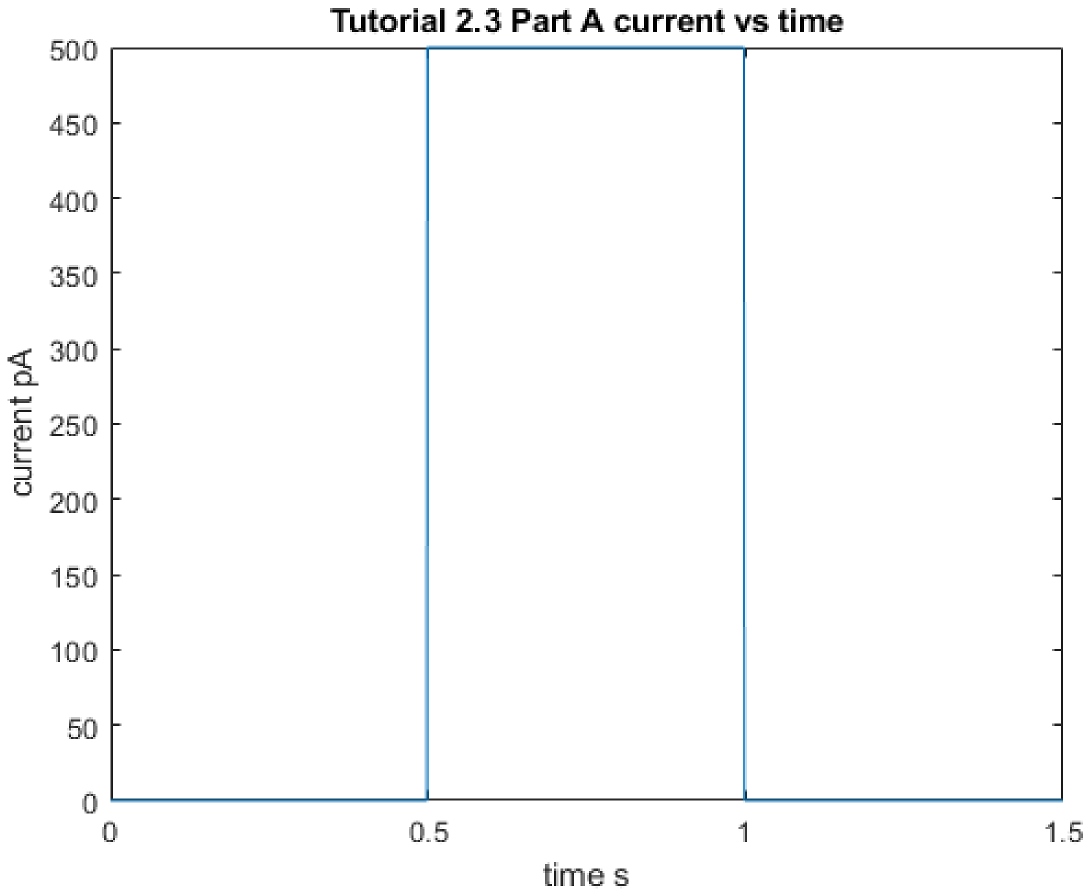
```
%there are obvious differences in both three methods of computation. On
%figure 4 where spike vs current, we can see that voltage gated has a
%higher slope than the other two, which means that the the larger applied
%current is, the voltage gated one is more prone to increases the number of
%spike by a whole margin. Therfore, the draweback of having a lot of action
%potentials is to have a more negative mean membrane potential as opposed to
%applied current, as can be seen by figure 5. The only negative slope is the
%voltage gated one. The other two still maintains a positive slope because
%they don't fire as much action potentials. As for figure 6, since we have
%different amounts of spikes, the line is, accordingly, in different length
%where voltage gated is the longest, and the shortest line has the highest
%slope. All of them are linear.
```

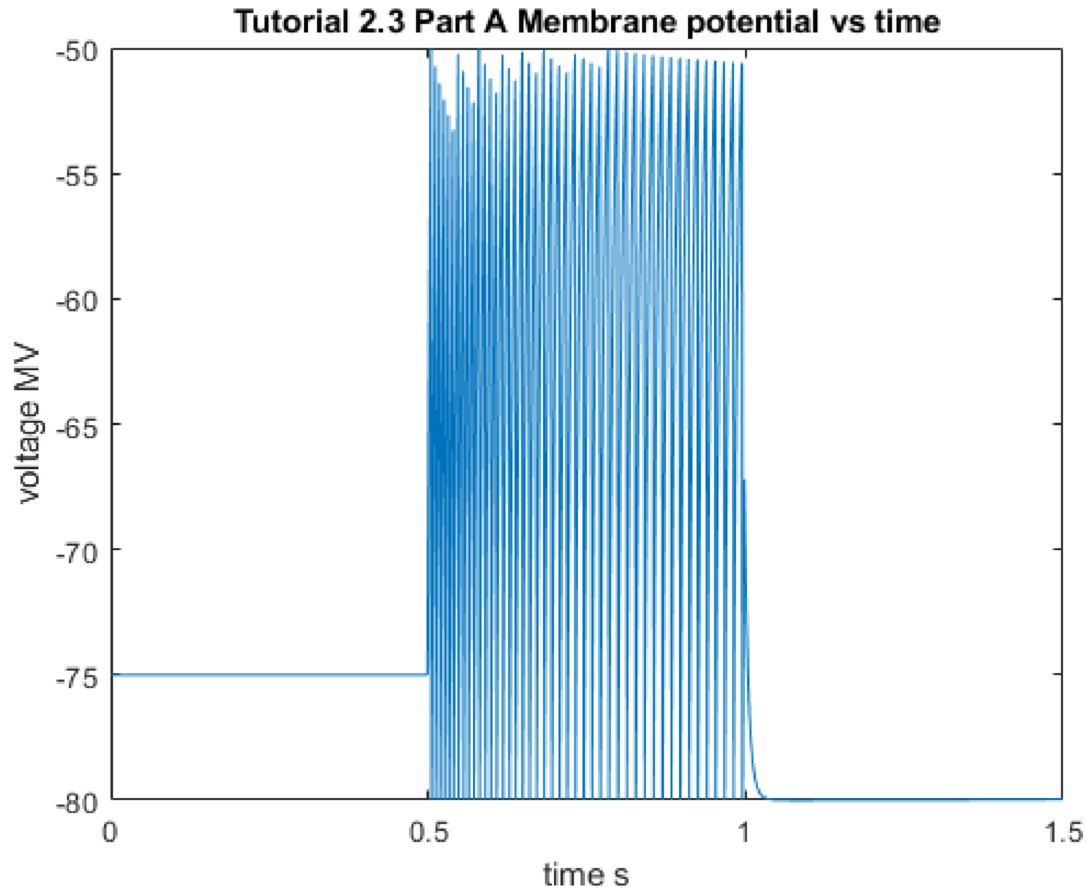
[Tutorial 2.3]

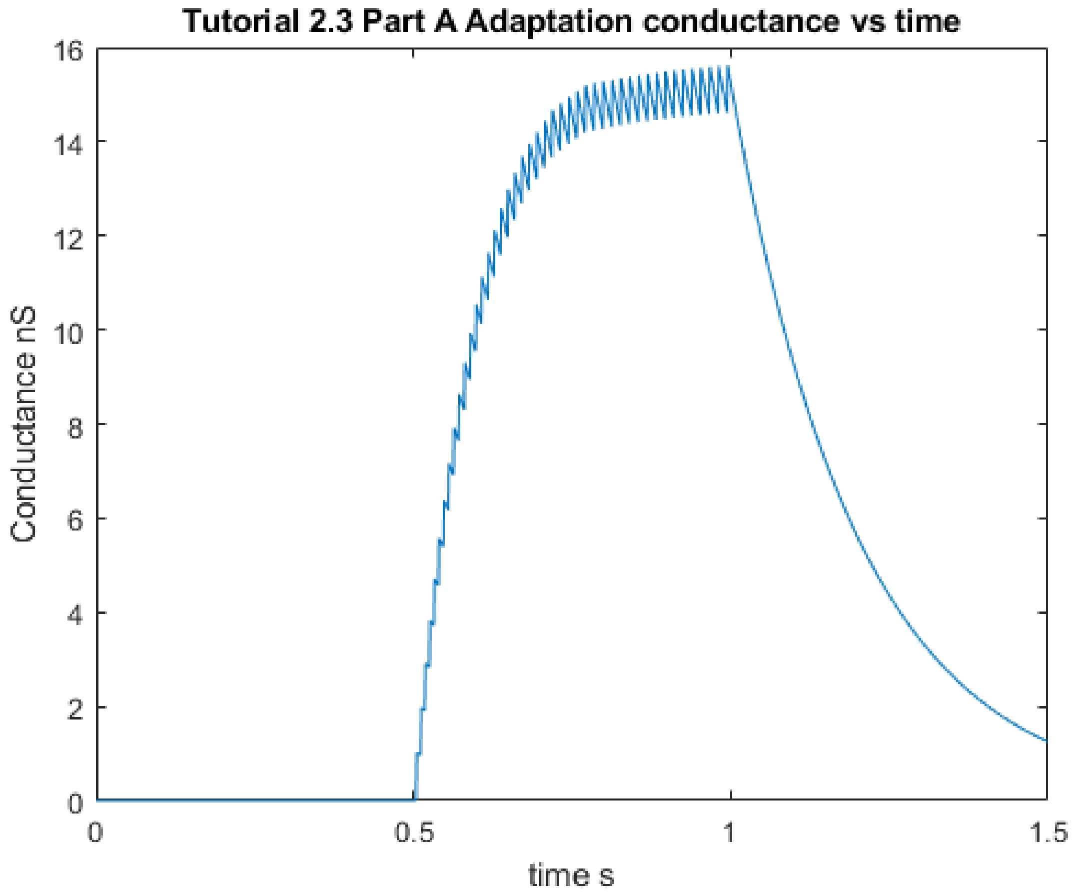
```
%Setting up parameters. (everything 10^3 for clarity)
EL = -75; %Leaky = -75mV
Vth = -50; % Threshold = -50mV
Vreset = -80; %the reset is 80mV
Rm = 100; % resistence is 100Mohms
Cm = 0.1; % Capacitance is 0.1pF
Ek = -80; %Equilibrium potassium = -80mV
deltaG = 1; % deltaGSRA = 1nS
tau = 0.2; % tau = 200ms
```

Part a

```
%setting up time vec
dt = 0.001; % step size = 0.1ms
tvec = 0:dt:1.5; % until 1.5s
%setting up Voltage vector
vvec = zeros(size(tvec));
vvec(1) = EL; %initialize the first term
%setting up conductance vector
gvec = zeros(size(tvec));
gvec(1) = 0; % initalize the first term
%Setting up Euler's method
iapp = zeros(size(tvec)); % have a variable iapp.
for i = 2:length(tvec)
    if (i > 500 && i < 1000)
        iapp(i-1) = 500;%500pA for applied curve
    else
        iapp(i-1) = 0;
    end
    %dG/dt = -G/tau
    dG_over_dt = -gvec(i-1)/tau;
    gvec(i) = gvec(i-1) + dG_over_dt * dt;
    %dV/dt = ((EL - V) /Rm + G(EK-V)+Iapp)/Cm.
    dV_over_dt = ((EL-vvec(i-1)) / Rm + gvec(i-1)*(EK-vvec(i-1))+iapp(i-1))/Cm;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    vvec(i) = vvec(i-1) + dV_over_dt * dt;
    if (vvec(i) > Vth)% if vvec element is higher than V threshold
        vvec(i) = Vreset; %force the element to reset to Vreset
        gvec(i) = gvec(i-1) + deltaG;
    end
end
figure (7)
plot(tvec,iapp)
title('Tutorial 2.3 Part A current vs time')
xlabel('time s')
ylabel('current pA')

figure (8)
plot(tvec,vvec)
title('Tutorial 2.3 Part A Membrane potential vs time')
xlabel('time s')
ylabel('voltage MV')

figure (9)
plot(tvec,gvec)
title('Tutorial 2.3 Part A Adaptation conductance vs time')
xlabel('time s')
ylabel('Conductance nS')
```


Tutorial 2.3 Part A current vs time
current pA
time s
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5

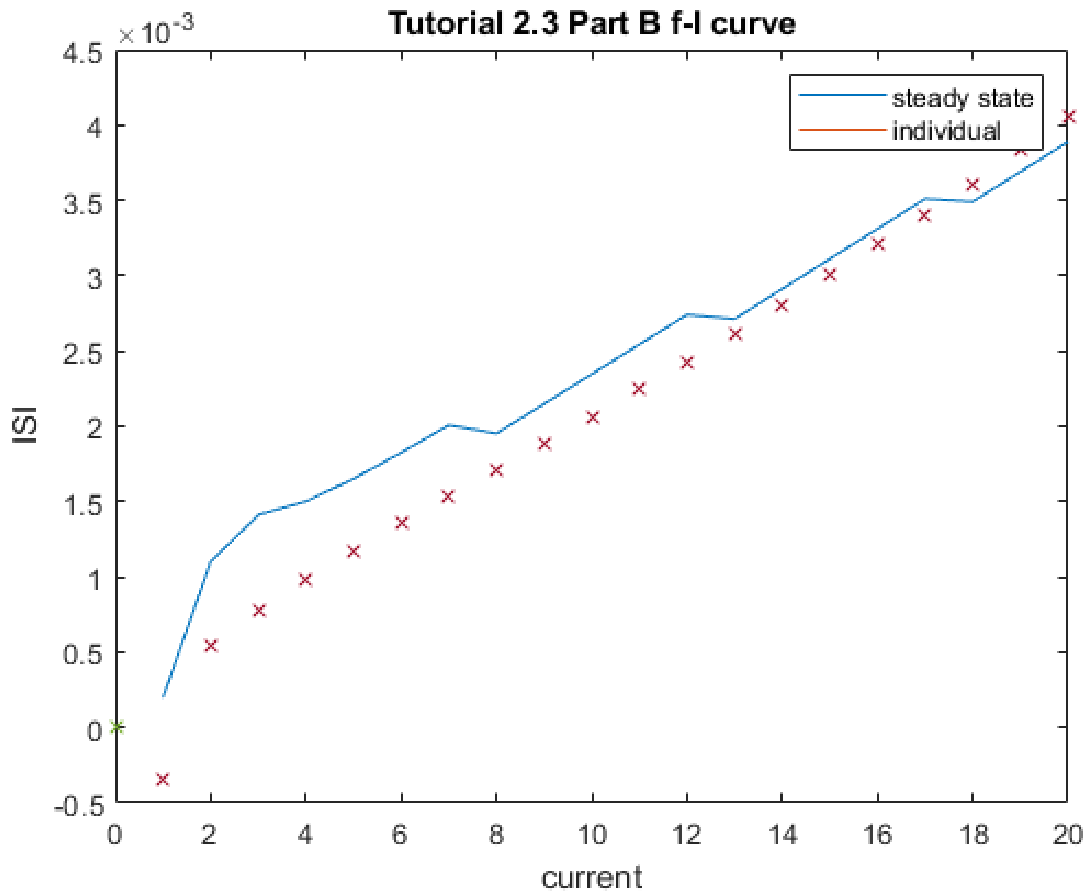

Tutorial 2.3 Part A Membrane potential vs time
voltage MV
time s
-50
-55
-60
-65
-70
-75
-80
0
0.5
1
1.5



Part B

```
%time changes
tvec2 = 0:dt:5; % until 5s
%setting up Voltage vector
vvec4 = zeros(size(tvec2));
vvec4(1) = EL; %initialize the first term
%setting up conductance vector
gvec2 = zeros(size(tvec2));
gvec2(1) = 0; % initalize the first term
%Setting up Euler's method
inv_isi_time1 = zeros(20);
inv_isi_steady = zeros(20);
pos = 1;
iapp = 0;
irange = zeros(20);
for range = 1:1:20
    iapp = range;
    t1 = 0;
    t2 = 0;
    k = 0;
    j1 = 0;
    j2 = 0;
for i = 2:length(tvec2)
    %dG/dt = -G/tau
    t1 = t1 + 1;
    t2 = t2+1;
    dG_over_dt = -gvec2(i-1)/tau;
    gvec2(i) = gvec2(i-1) + dG_over_dt * dt;
    %dV/dt = ((EL - V) /Rm + G(EK-V)+Iapp)/Cm.
```

```
        dV_over_dt = ((EL-vvec4(i-1)) / Rm + gvec2(i-1)*(EK-vvec4(i-1))+iapp)/Cm;
        % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
        vvec4(i) = vvec4(i-1) + dV_over_dt * dt;
        if (vvec4(i) > Vth)% if vvec element is higher than V threshold
            vvec4(i) = Vreset; %force the element to reset to Vreset
            gvec2(i) = gvec2(i-1) + deltaG;
            k = k+1;
           if k == 1
            j1 = t1;
           elseif k == 2
            j2 = t2;
           end
        end
    end
    inv_isi_time1(pos) = 1 /(j2-j1);
    inv_isi_steady(pos) = 1/ ((length(tvec2)-j2)/ k);
    irange(pos) = range;
     hold on
     pos = pos+1;
    end
    figure (10)
    plot(irange,inv_isi_steady)
    hold on
    plot(irange,inv_isi_time1,('x'))
    legend('steady state','individual')
    xlabel('current')
    ylabel('ISI')
    title('Tutorial 2.3 Part B f-I curve')
```



comment:

the initial firing frequency is less than the stady state frequency, yet when applied current is in becoming of a greater degree the initial firing frequency becomes larger.

Pod Times

We met on Thursday from 7 to 9PM to work and clear up settting parameters

```
% I helped by sharing my codes and graphs to them and clearing up the
% values of parameters because of margin of error.

% I had questions on the theoretical curves and what the graphs should
% actually look like to confirm if my plot is correct.
```

Published with MATLAB® R2022b