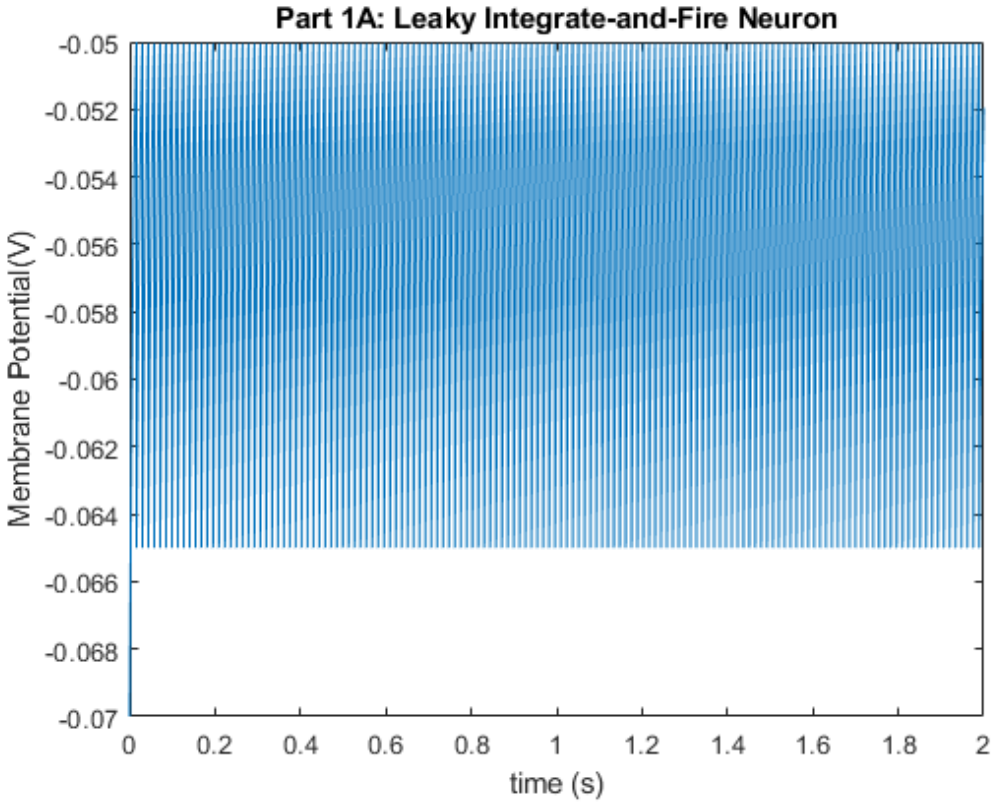
Contents

[Part 1A] write a code to simulate Cm * dVm/dt = (El - Vm) /Rm + Iapp.

```
% (I) setting up parameters
EL = -70e-3; % -70mV leak Nernst potential
Rm = 5e6; % 5 Mohms membrane resistence
Cm = 2e-9; % 2nF membrane capacitance
Vth = -50e-3; %-50mV for membrane potential
Vreset = -65e-3; %-65mV for resert membrane potential

% (II) create time vectors;
t0 = 0; % t = 0
tmax = 2; % max time is t = 2s
dt = 0.0001; % step size 0.1ms
tvec = t0:dt:tmax; %time vector

%(III) voltage vector;
vvec = zeros(size(tvec)); % set voltage vector same as tvec size.
%(IV) set initial value of vvec to leak potential EL
vvec(1) = EL;

%(V) vector for iapp.
iapp = zeros(size(tvec)); % same size as tvec
I0 = 5e-9; %include I0 as a random number
for i = 1:length(tvec)
iapp(i) = I0; %every increment of iapp is I0.
end

%(VI,VII,VIII) write a for loop.
for i = 2:length(tvec) % from two to length tvec (VI)
    %dVm/dt = ((El-Vm)/Rm+Iapp)/Cm.
    dVm_over_dt = ((EL-vvec(i-1)) / Rm + iapp(i-1))/Cm;
    % V(i) = V(i-1) + f(t(i-1),v(i-1))*dt
    vvec(i) = vvec(i-1) + dVm_over_dt * dt;
    if (vvec(i) > Vth)% if vvec > V threshold
        vvec(i) = Vreset; %force the vvec = Vreset
    end
end

%Graphing out 1A
figure(1)
plot(tvec,vvec)
title('Part 1A: Leaky Integrate-and-Fire Neuron')
xlabel('time (s)')
```

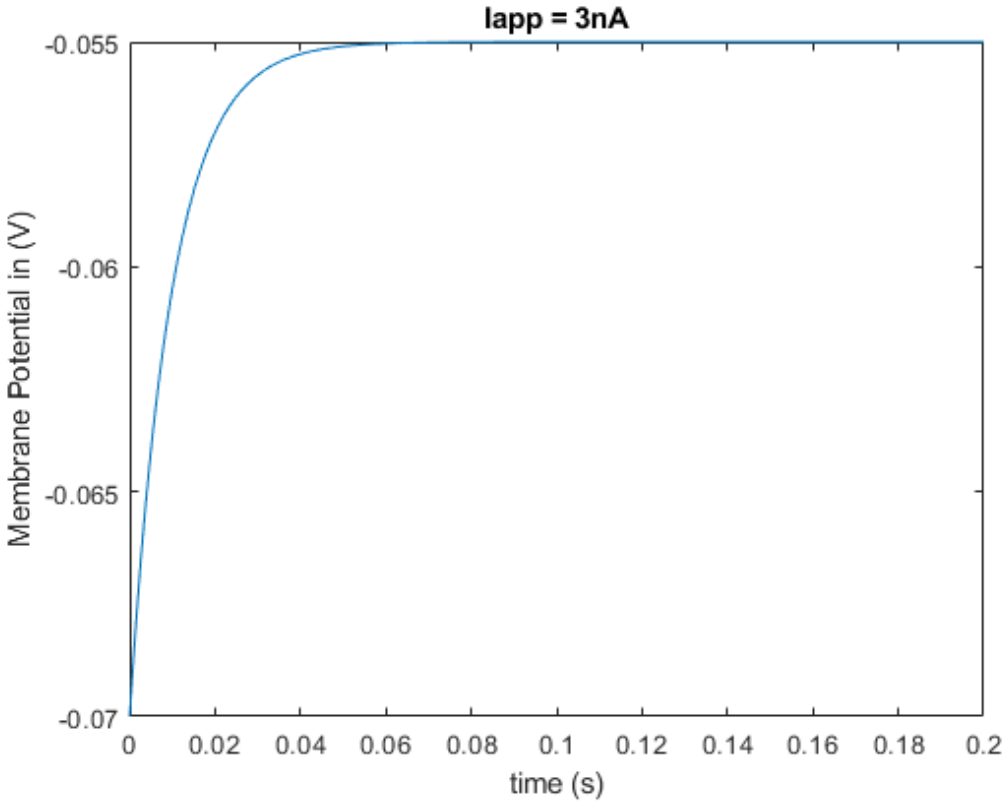
```
ylabel('Membrane Potential(V)')
hold off
```

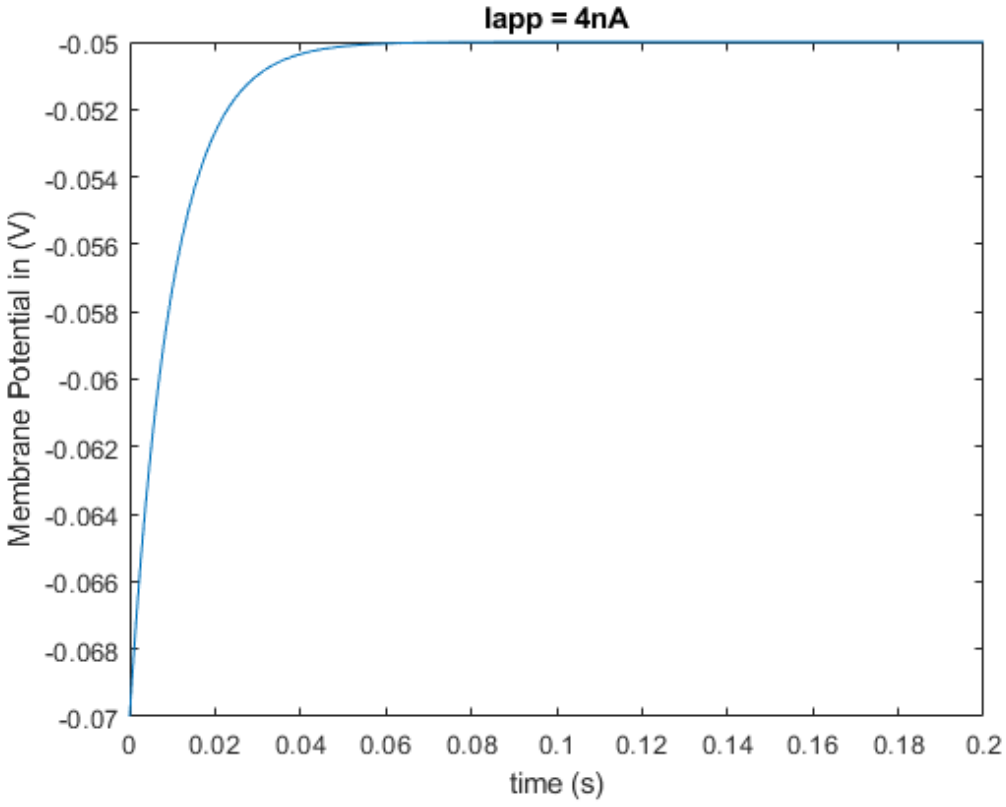


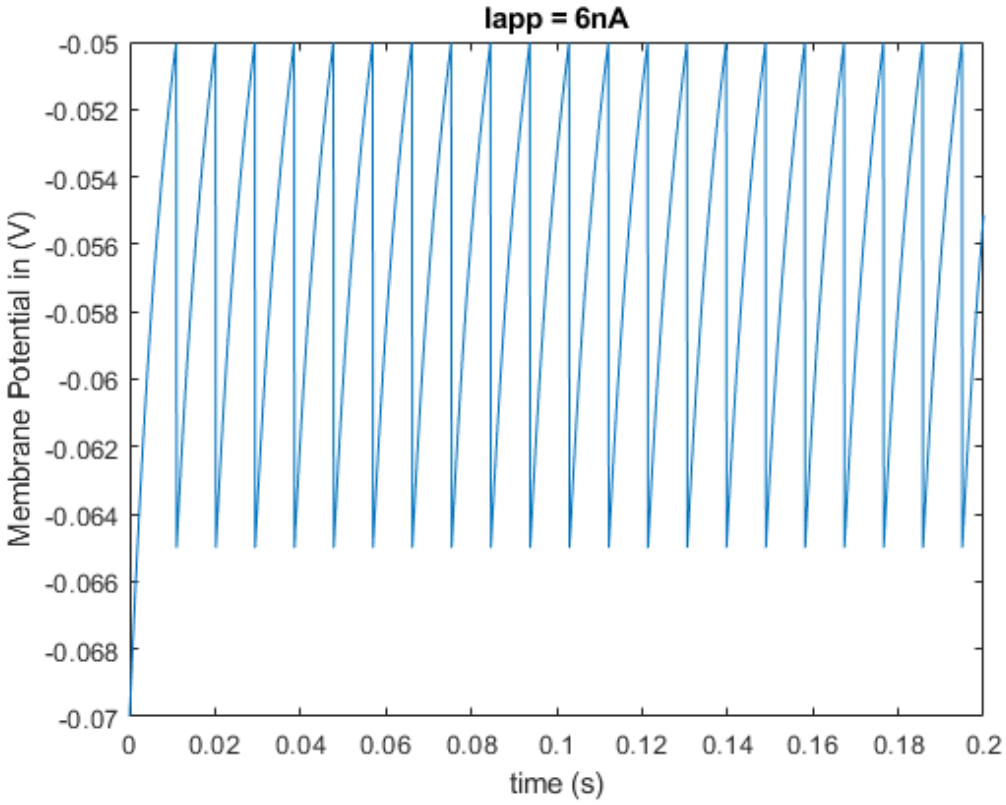
[Part 1B] Find Ith and compare them.

```
%equation from 2.11 textbook. Ith = GL(Vth - EL) Which equals to
%Ith = 1/Rm * (Vth-EL). Ith's answer is 4e-9 Amperes.
Ith = 4e-9;
%setup Iapp < Ith, Iapp = Ith, Iapp > Ith.
Icompare = zeros(3);
Icompare(1) = 3e-9; % Iapp < Ith
Icompare(2) = 4e-9; % Iapp = Ith
Icompare(3) = 6e-9; % Iapp > Ith
%Setup new tvec from 0 to 0.2s (200ms) from size 0.1ms
tf = 0.2;
dt = 0.0001;
tvec = 0:dt:tf;
%Setup Voltage Vector
vvec = zeros(size(tvec));% same size as tvec
vvec(1) = EL; % set first element to leaky equilibrium potential
%Setup Iapp Vector
iapp = zeros(size(tvec)); % same size as tvec
%Setup Euler's Method
for i = 1:3 %for loop for three scenarios
    for j = 1:length(tvec)
        iapp(j) = Icompare(i); % set iapp vector constant.
    end
    %Euler's Method
    for k = 2:length(tvec)
        %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
        dVm_over_dt = ((EL-vvec(k-1)) / Rm + iapp(k-1))/Cm;
        % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
        vvec(k) = vvec(k-1) + dVm_over_dt * dt;
        if (vvec(k) > Vth)% if vvec > V threshold
        vvec(k) = Vreset; %force the element to Vreset
```

```
        end
    end
    % plotting the figures
    figure (i+1)
    plot (tvec,vvec)
    xlabel('time (s)')
    ylabel('Membrane Potential in (V)')
    % different names in different graphs to match Iapp.
    if i == 1
    title ('Iapp = 3nA')
    hold off
    end
    if i == 2
    title ('Iapp = 4nA')
    hold off
    end
    if i == 3
    title ('Iapp = 6nA')
    hold off
    end
end
```

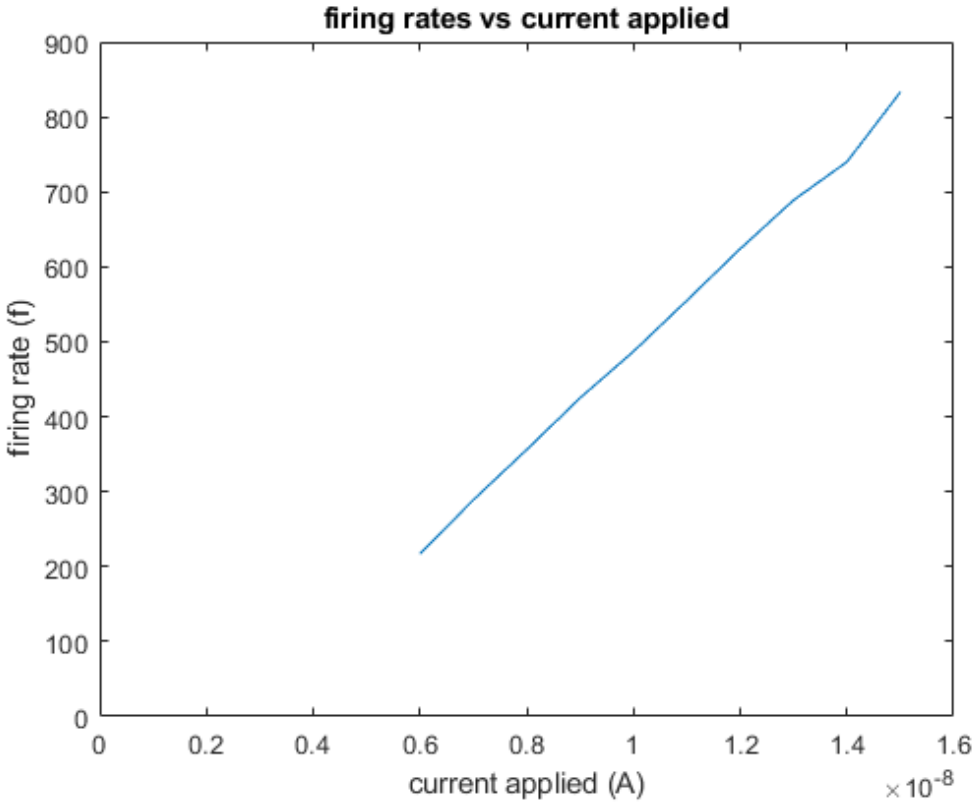

Iapp = 3nA
Membrane Potential in (V)
-0.055
-0.06
-0.065
-0.07
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
time (s)


Iapp = 4nA
Membrane Potential in (V)
-0.05
-0.052
-0.054
-0.056
-0.058
-0.06
-0.062
-0.064
-0.066
-0.068
-0.07
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
time (s)



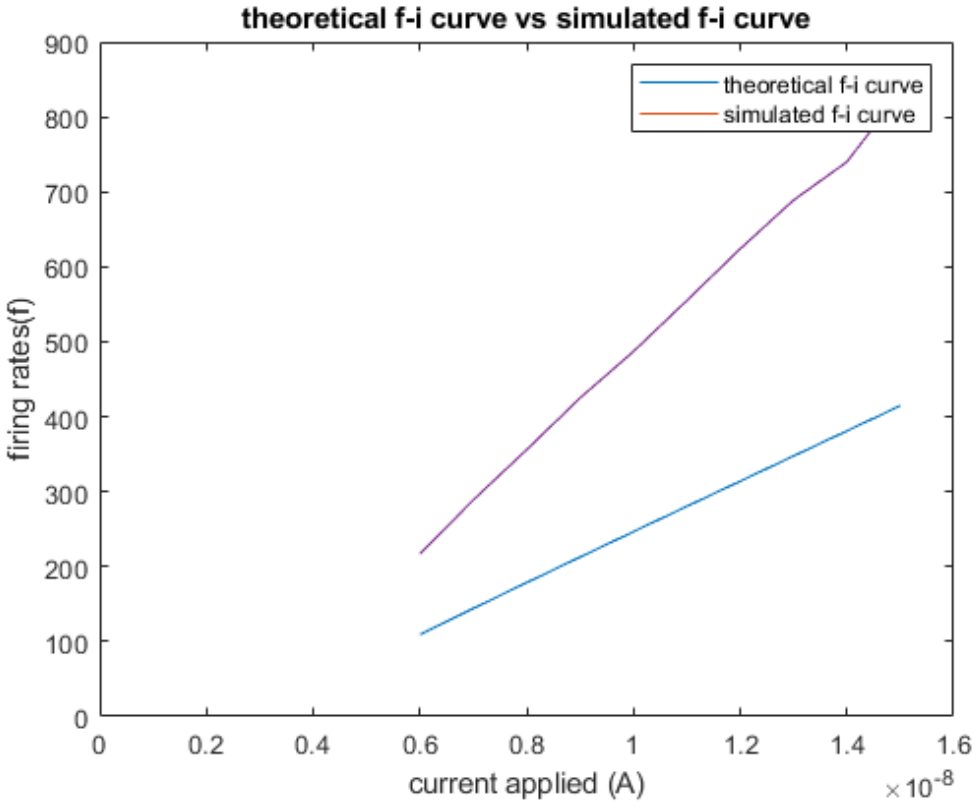
[Part 1C] simulate 10 times with different Iapp values;

```
%setting up time
tf = 2; % 2 seconds
dt = 0.0001; % 0.1ms
tvec = 0:dt:tf;
% setting up vectors
vvec = zeros(size(tvec));
vvec(1) = EL; %set voltage vector initial =EL
iapp =zeros(10); % setup Iapp size 10 for 10 increments.
firing_rate = zeros(10);% setup firing rate size 10 for 10 increments.
pos = 1;%setup dummy variable
for range = 6e-9:1e-9:15e-9 % increment from 6nA to 15nA by 1nA.
    iapp(pos) = range;
        for j = 2:length(tvec)
        %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
        dVm_over_dt = ((EL-vvec(j-1)) / Rm + iapp(pos))/Cm;
        % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
        vvec(j) = vvec(j-1) + dVm_over_dt * dt;
            if (vvec(j) > Vth)% if vvec element is higher than V threshold
                vvec(j) = Vreset; %force the element to reset to Vreset
                firing_rate(pos) = firing_rate(pos) + 1; % once peak add 1.
            end
        end
        pos = pos + 1;
end
%plot the relationship
figure (5)
plot (iapp,firing_rate)
title('firing rates vs current applied')
xlabel('current applied (A)')
ylabel('firing rate (f)')
hold off
```



Part [1D] simulate 1/f

```
%define variables
tau = Cm * Rm;
%define theoretical firing rate
f = zeros(10);
%iapp
newiapp = zeros(10);
pos = 1; %dummy var

%calculate theoretical firing rate
for range = 6e-9:1e-9:15e-9 %same iapp from above
newiapp(pos) = range;
f(pos) = 1/(tau * reallog(newiapp(pos) * Rm + EL - Vreset) - tau * reallog(newiapp(pos) * Rm + EL - Vth));
pos = pos+1; % next position
end

figure (6)
plot(newiapp, f)
hold on
plot(iapp, firing_rate)
title('theoretical f-i curve vs simulated f-i curve')
xlabel('current applied (A)')
ylabel('firing rates(f)')
legend({'theoretical f-i curve','simulated f-i curve'})
hold off
```



[Part 1D] Explanation

From the graph, it is clearly known that the slope of the simulated curve is visibly larger than that of theoretical one. In addition, it seemed more like that the simulated one shifts upward as well. In this case, the theoretical is underestimating simulation.

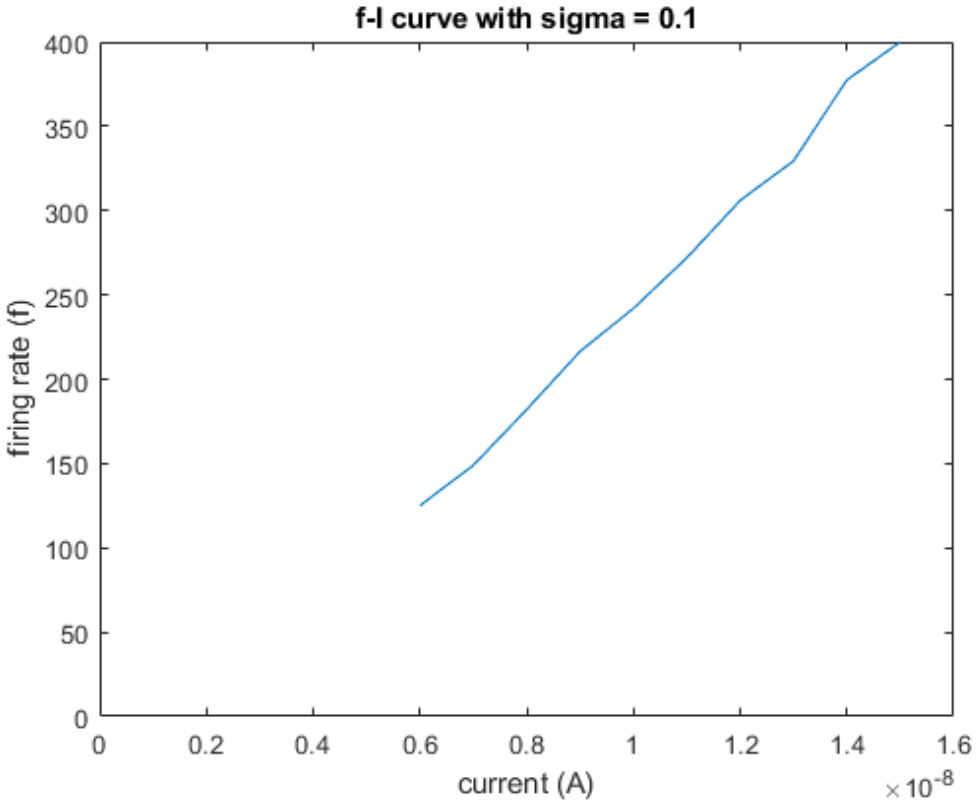
[Part 2A] setting up the sigma factor noise

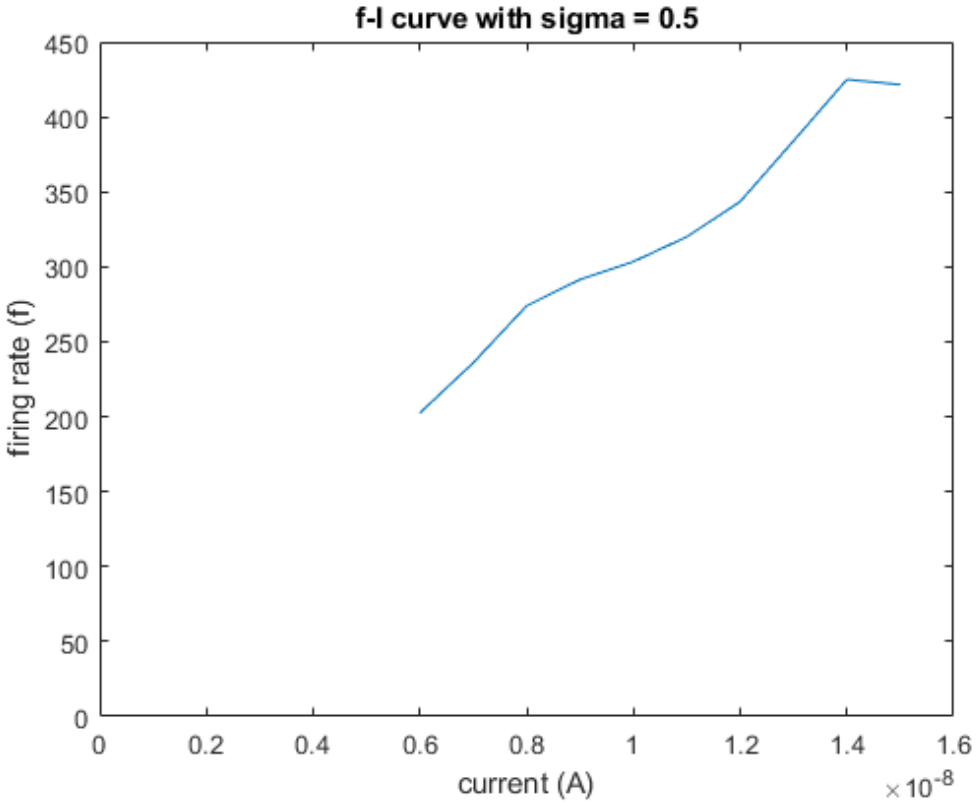
```
% setting up parameters
EL = -70e-3; % -70mV leak Nernst potential
Rm = 5e6; % 5 Mohms membrane resistence
Cm = 2e-9; % 2nF membrane capacitance
Vth = -50e-3; %-50mV for membrane potential
Vreset = -65e-3; %-65mV for resert membrane potential
t0 = 0; % t = 0
tmax = 2; % max time is t = 2s
dt = 0.0001; % step size 0.1ms

%Set up vectors
tvec = t0:dt:tmax; %time vector that takes in from 0-2 with step size 0.1ms
vvec = zeros(size(tvec)); % set voltage vector same tvec size
vvec(1) = EL; %initialize first term
% set up noise vec
sigma_I = 0.5; % set up noise scale
iapp = zeros(size(tvec)); % same size as tvec
I0 = 5e-9;% initial constant
for i = 1:length(tvec)
iapp(i) = I0; %fill in
end
%Euler methods
for i = 2:length(tvec)
    %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
    dVm_over_dt = ((EL-vvec(i-1)) / Rm + iapp(i-1))/Cm;
    % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
    vvec(i) = vvec(i-1) + dVm_over_dt * dt + randn(1)*sigma_I*sqrt(dt);
    if (vvec(i) > Vth)% if vvec element is higher than V threshold
        vvec(i) = Vreset; %force the element to reset to Vreset
```

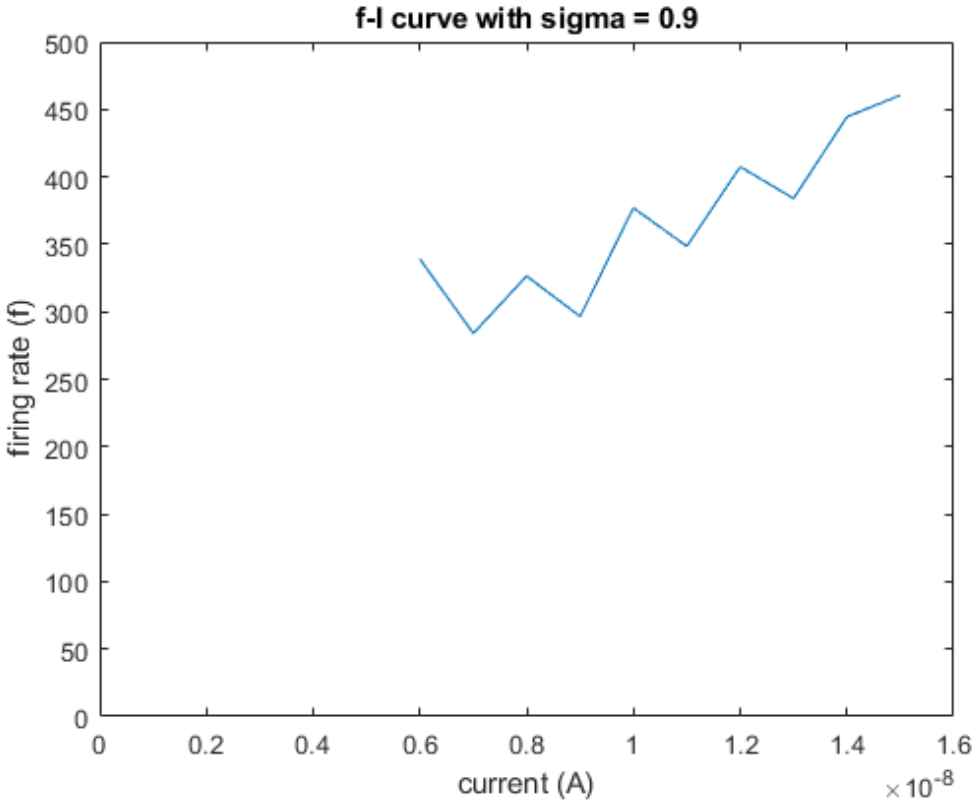
```
    end
end
```

[Part 2B] setting up for different noise values

```
%set up sumulations for sigma = 0.1, 0.5, 0.9
for sigma_range = 0.1:0.4:0.9
sigma_I = sigma_range; % noise
fire = zeros(10);% firing rate
iapp = zeros(10); % setting up the current vector
pos = 1; % dummy position value
for range = 6e-9:1e-9:15e-9 % same iapp range from part 1
    iapp(pos) = range;
    for i = 2:length(tvec)
        %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
        dVm_over_dt = ((EL-vvec(i-1)) / Rm + iapp(pos))/Cm;
        % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
        vvec(i) = vvec(i-1) + dVm_over_dt * dt + randn(1)*sigma_I*sqrt(dt);
            if (vvec(i) > Vth)% if vvec element is higher than V threshold
            vvec(i) = Vreset; %force the element to reset to Vreset
            fire(pos) = fire(pos)+ 1; %action potential + 1
            end
    end
    fire(pos) = fire(pos) / tmax; %get the rate
    pos = pos + 1;
end
    if sigma_range == 0.1
        figure(7)
        plot(iapp, fire)
        title ('f-I curve with sigma = 0.1')
        xlabel('current (A)')
        ylabel('firing rate (f)')
    elseif sigma_range == 0.5
        figure(8)
        plot(iapp, fire)
        title ('f-I curve with sigma = 0.5')
        xlabel('current (A)')
        ylabel('firing rate (f)')
    elseif sigma_range == 0.9
        figure(9)
        plot(iapp, fire)
        title ('f-I curve with sigma = 0.9')
        xlabel('current (A)')
        ylabel('firing rate (f)')
    end
end
hold off
```


f-I curve with sigma = 0.1
firing rate (f)
current (A)
×10^-8
0
50
100
150
200
250
300
350
400
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


f-I curve with sigma = 0.5
firing rate (f)
current (A)
×10^-8
0
50
100
150
200
250
300
350
400
450
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6



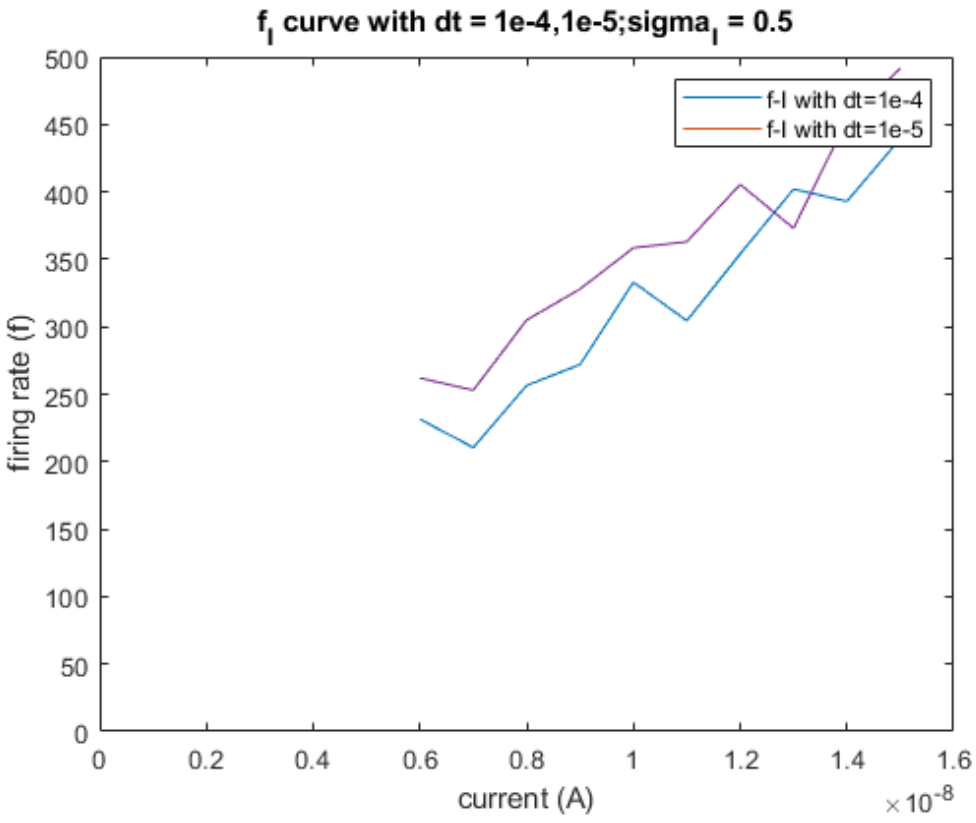
[Part 2B] Explanation

```
%The closer sigma goes to 1, the more nonlinear the graph looks. For
%example, we have the sigma = 0.1 case, you can see that it remains almost
%linear. For the sigma = 0.9 case, you can see that the more instable the
%graph looks.
```

[Part 2C] change dt

```
%set up sumulations for sigma = 0.5 and dt = default 0.0001
sigma_I = 0.5;
dt = 0.0001; % step size 0.1ms
tvec = t0:dt:tmax; %time vector that takes in from 0-2 with step size 0.0001.
fire = zeros(10);% firing rate
iapp = zeros(10); % setting up the current vector
pos = 1; % dummy position value
for range = 6e-9:1e-9:15e-9 % same iapp range from part 1
    iapp(pos) = range;
    for i = 2:length(tvec)
        %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
        dVm_over_dt = ((EL-vvec(i-1)) / Rm + iapp(pos))/Cm;
        % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
        vvec(i) = vvec(i-1) + dVm_over_dt * dt + randn(1)*sigma_I*sqrt(dt);
            if (vvec(i) > Vth)% if vvec element is higher than V threshold
            vvec(i) = Vreset; %force the element to reset to Vreset
            fire(pos) = fire(pos)+ 1; %action potential + 1
            end
    end
    fire(pos) = fire(pos) / tmax; %get the rate
    pos = pos + 1;
end

%set up sumulations for sigma = 0.5 and dt = 0.00001
dt = 0.00001; % step size 0.1ms
tvec = t0:dt:tmax; %time vector that takes in from 0-2 with step size 0.00001.
```

```
fire2 = zeros(10);% firing rate
pos = 1;
for range = 6e-9:1e-9:15e-9 % same iapp range from part 1
    iapp(pos) = range;
    for i = 2:length(tvec)
        %dVm/dt = ((El - Vm) /Rm + Iapp)/Cm.
        dVm_over_dt = ((EL-vvec(i-1)) / Rm + iapp(pos))/Cm;
        % Vmi = Vmi-1 + dt*f(ti-1,vmi-1)
        vvec(i) = vvec(i-1) + dVm_over_dt * dt + randn(1)*sigma_I*sqrt(dt);
            if (vvec(i) > Vth)% if vvec element is higher than V threshold
            vvec(i) = Vreset; %force the element to reset to Vreset
            fire2(pos) = fire2(pos)+ 1; %action potential + 1
            end
    end
    fire2(pos) = fire2(pos) / tmax; %get the rate
    pos = pos + 1;
end
figure (10)
plot(iapp, fire)
hold on
plot(iapp, fire2)
title ('f_I curve with dt = 1e-4,1e-5;sigma_I = 0.5')
xlabel('current (A)')
ylabel('firing rate (f)')
legend({'f-I with dt=1e-4','f-I with dt=1e-5'})
hold off
```



[Part 2C] Explanation

There is a significant difference when you have dt on default and that of 10 times smaller. It can be interpreted that the ten times smaller on if higher than that of the original. The noise is also different so the graph's slope looks different as well. They also have siginificant different frequencies.

Collaboration requirement

```
%My Pod members are Anthony and Mauricio.

%To discuss this lab, I met with my pod on Thursday 4/27 for about 2 hours
%from 8PM to 10PM.

%We discussed the confusion we had on part B. We had a I threshold as 4,
%which is correct but there is not a spike. The problem is solved when we
%increase the time from 2 to 20.

%I got clarification on the window of time t which solved the issue of not
%seeing the reset near 4nA.

%We clarified from problem two with the sigma I noise. Anthony was asking
%about what that means so I read through the appendix with him and we added
%noise together.
```

Published with MATLAB® R2022b