LIF Networks

In this experiment I simulated 3 leaky integrate-and-fire neuron networks by adding weights between neurons. The weights follow Dale's Principle. I used the following formula to calculate the membrane potential. I take 10 neurons throughout the experiment.

$$\frac{dV_i}{dt} = -\frac{1}{\tau}(V_i - V_{rest}) + \sum_j W_{ij}\delta_j$$

Where $\delta_j = \begin{cases} 1 & \textit{if the jth neuron spiked in the previous time bin} \\ 0 & \textit{elsewise} \end{cases}$

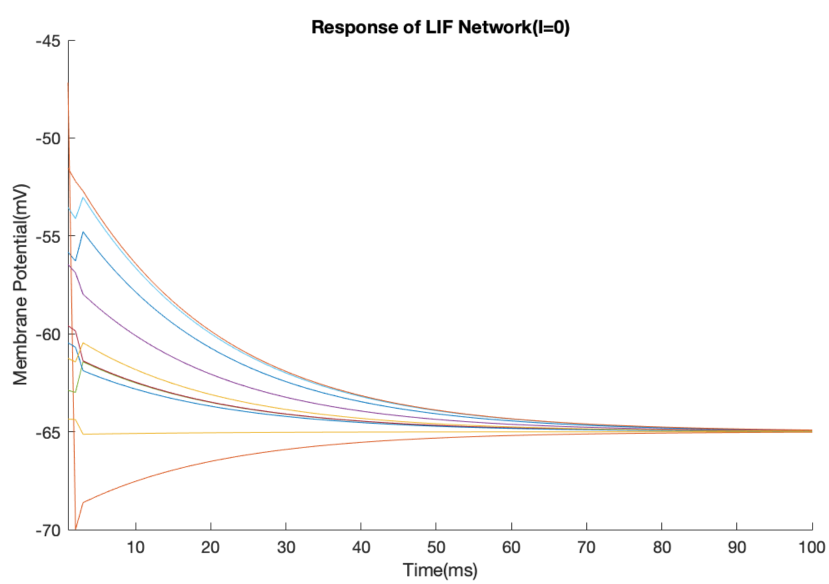
PART I: Random Network

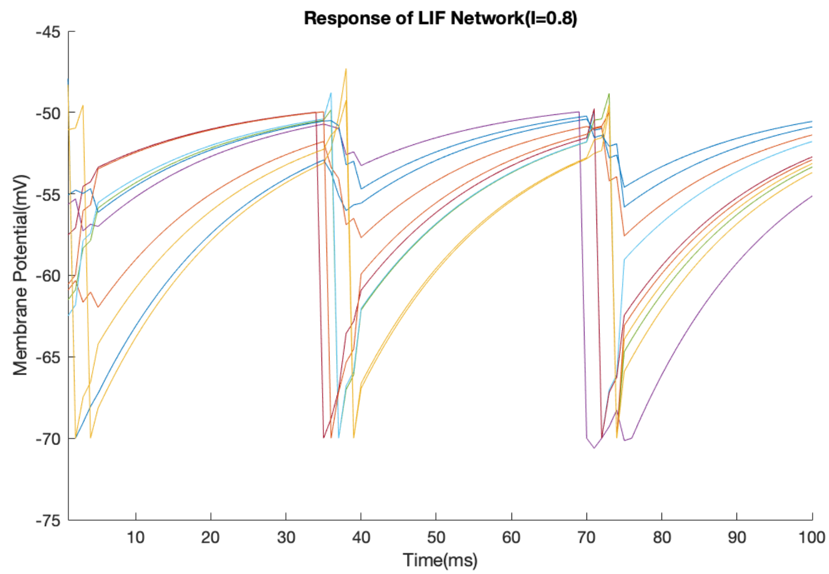
In the random network I draw the weights uniformly from -w to w.

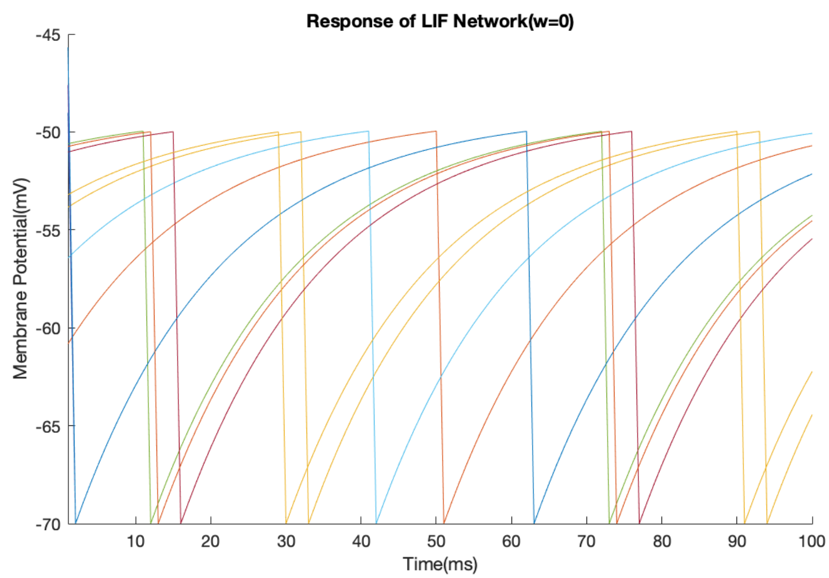
```
function [V, spks, spkcnt] = randNet(time, input, w, n)
% Random LIF network by given time, constant input,
% Weight vairable w, and number of neurons n
% Returns the membrane potential V
% And the binary spike sequence for each neuron spks
% And the spike count for each neuron spkcnt

    decay = 20;
    v_rest = -65;
    v_thres = -50;
    v_reset = -70;
    dt = 1;    % ms
    bins = time * 1000 / dt;
    spkcnt = zeros(n, 1);

    % Calculate the connection strength matrix
    weights = zeros(n, n);
    for i = 1 : n
        sign = (-1) ^ binornd(1, 0.5);
        for j = 1 : n
            weights(i, j) = sign * abs(-w + rand() * 2 * w);
        end
    end

    % Initialize membrane potential V
    V = zeros(n, bins);
    spks = zeros(n, bins);
    for i = 1 : n
        V(i, 1) = v_rest + rand() * (v_thres - v_reset);
        spks(i, 1) = 0;
    end

    % Simulate for given time
    for t = 2 : bins
        for i = 1 : n
```

```
            v_prev = V(i, t - 1);

            if v_prev > v_thres
                V(i, t) = v_reset;
                spks(i, t) = 1;
                spkcnt(i, 1) = spkcnt(i, 1) + 1;

            elseif v_prev < v_reset
                V(i, t) = v_reset;

            else
                stim = 0;  % total weighted spike stimuli from other neurons
                for j = 1 : n
                    if spks(j, t - 1) == 1
                        stim = stim + weights(i, j);
                    end
                end
                V(i, t) = v_prev + dt * (-1 / decay * (v_prev - v_rest) + input +
stim);
            end

        end
    end

end
```

The following are the plot of membrane potential of the neurons when I = 0, I = 0.8, both with w = 2, and when I = 0.8, w = 0. The behavior is as expected. When I = 0, they all get asymptotic to Vrest. When they fire, the weights makes them somewhat on sync, and when w = 0, they just act as separate neurons.





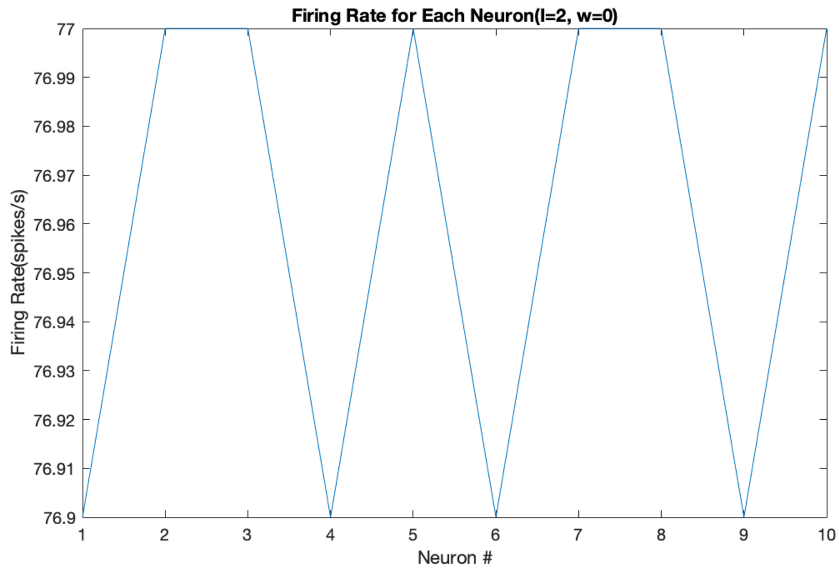


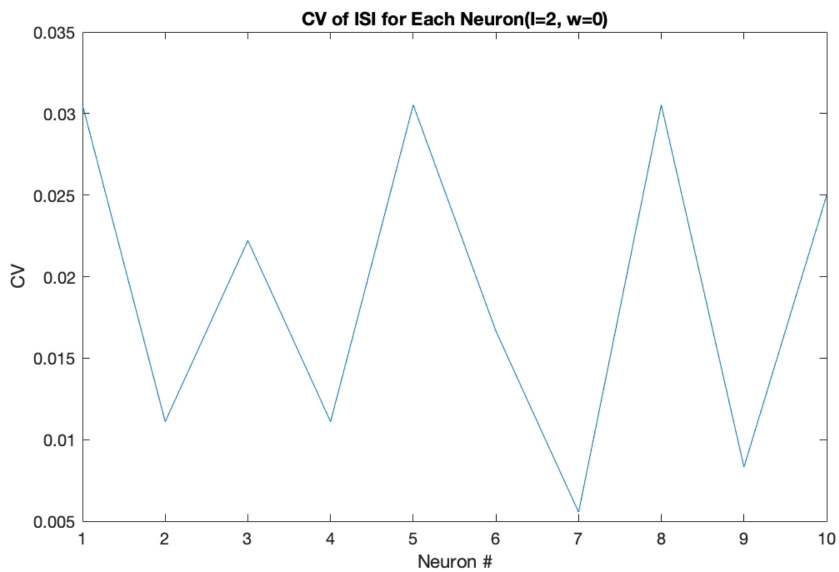
The following are the functions regarding the firing rate and ISI.

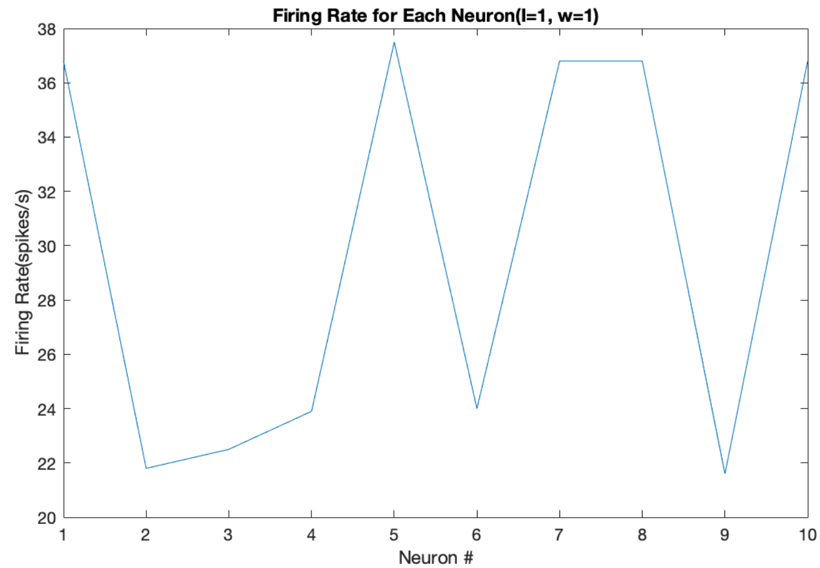
```
function [fr, mFr, cvFr] = getFR(spkcnt, time)
% Given spike count and time of the network
% Returns the firing rate of each neuron(fr)
% And the mean firing rate of all neurons(mFR)
% And the CV of firing rates(cvFR)

    n = size(spkcnt, 1);
    fr = zeros(n, 1);
    for i = 1 : n
        fr(i, 1) = spkcnt(i, 1) / time;
    end
    mFr = mean(fr);
```

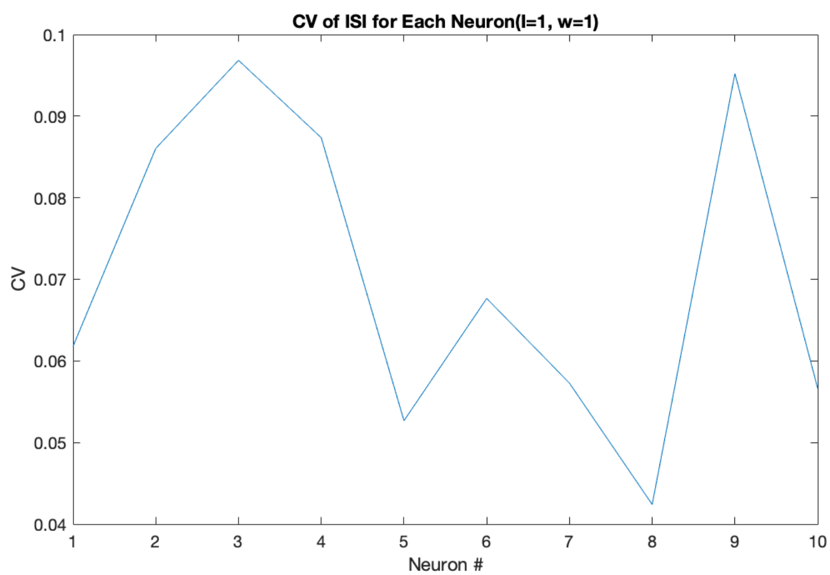
```
    cvFr = std(nonzeros(fr)) / mean(nonzeros(fr));
end

function [cv, meanCV] = getISI(spks, spkcnt)
% Given binary spike sequence and spike count for each neuron
% Returns the CV of ISIs for each neuron and their mean

    n = size(spkcnt, 1);
    dt = 1;
    bins = size(spks, 2);
    cv = zeros(n, 1);

    for i = 1 : n
        if spkcnt(i, 1) < 10
            continue
        else
            isi = zeros(1, spkcnt(i, 1));
            time = 0;
            cnt = 1;
            for t = 1 : bins
                time = time + dt;
                if spks(i, t) == 1
                    isi(1, cnt) = time;
                    time = 0;
                    cnt = cnt + 1;
                end
            end
        end
        cv(i, 1) = std(isi) / mean(isi);
    end

    meanCV = mean(nonzeros(cv));
end
```

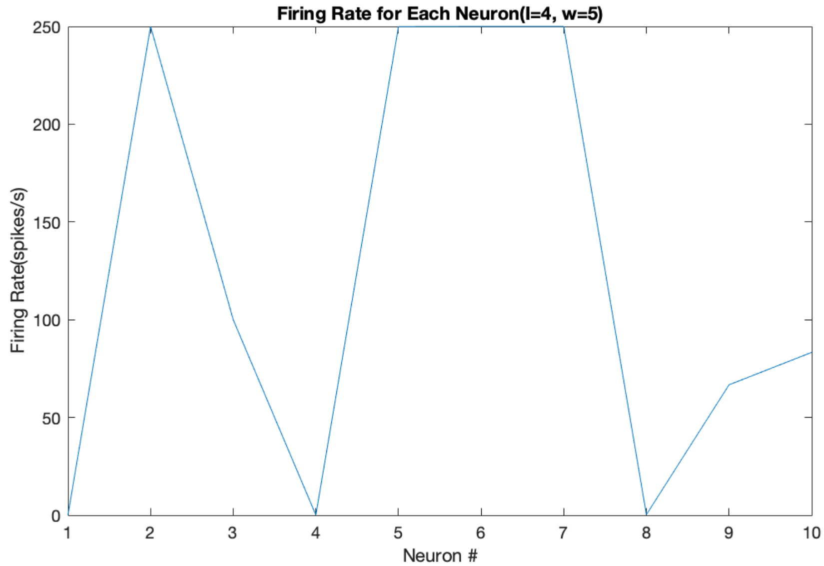
I plotted the firing rate and CV of ISI's of different neurons for 3 different combinations of w and I. When w increases, the firing rates are less similar.

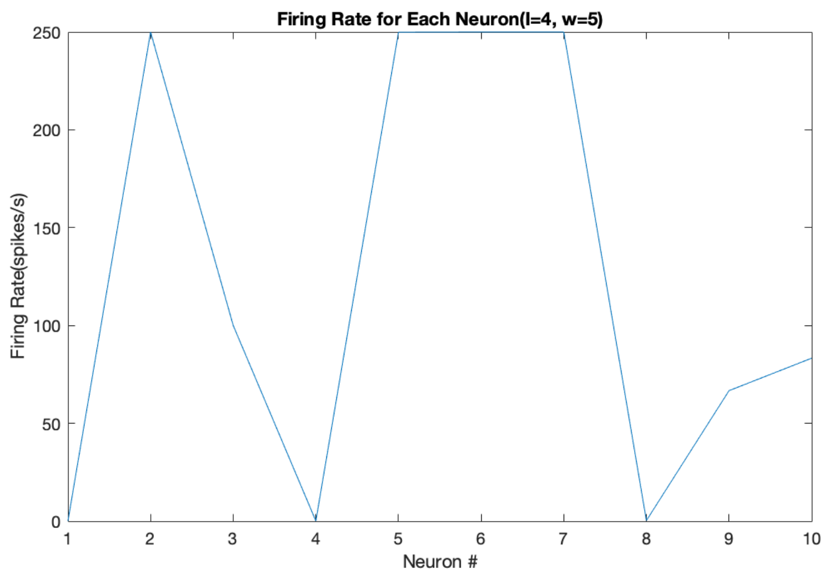

Firing Rate for Each Neuron(I=2, w=0)
Firing Rate(spikes/s)
77
76.99
76.98
76.97
76.96
76.95
76.94
76.93
76.92
76.91
76.9
1
2
3
4
5
6
7
8
9
10
Neuron #


CV of ISI for Each Neuron(I=2, w=0)
CV
0.035
0.03
0.025
0.02
0.015
0.01
0.005
1
2
3
4
5
6
7
8
9
10
Neuron #


Firing Rate for Each Neuron(I=1, w=1)
Firing Rate(spikes/s)
38
36
34
32
30
28
26
24
22
20
1
2
3
4
5
6
7
8
9
10
Neuron #


CV of ISI for Each Neuron(I=1, w=1)
CV
0.1
0.09
0.08
0.07
0.06
0.05
0.04
1
2
3
4
5
6
7
8
9
10
Neuron #



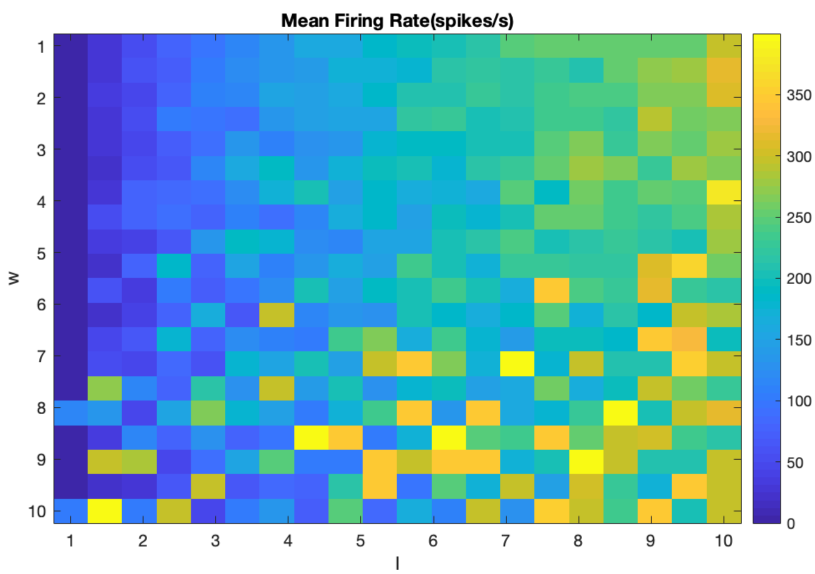


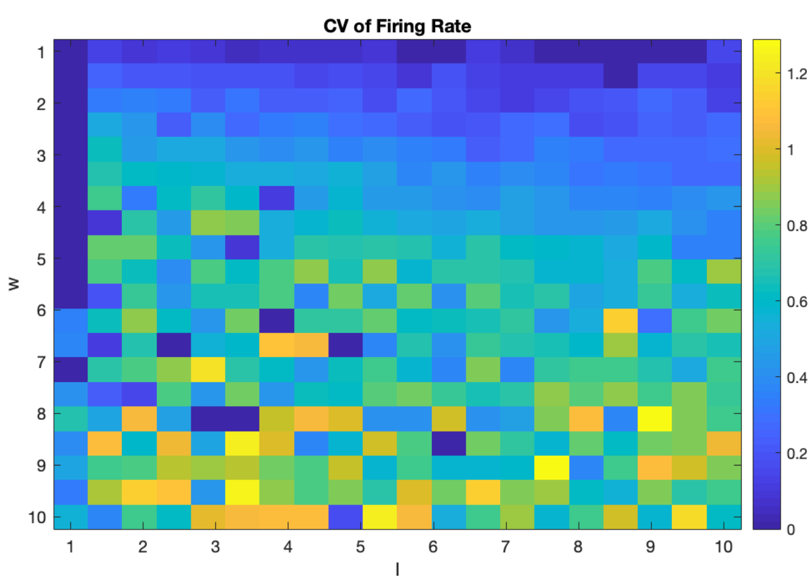
Now I used the following function to get the 2d relation between firing rate, CV and w & I.

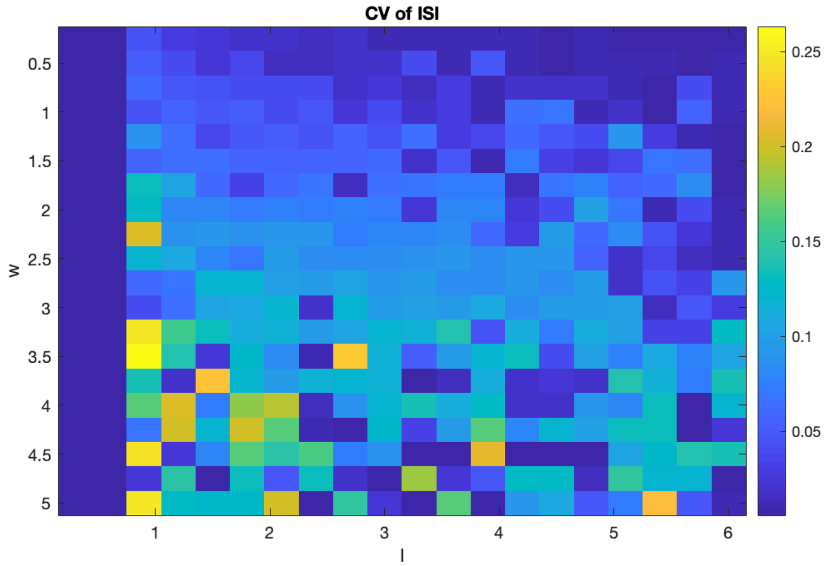
```
function [mFR, cvFR, cvISI] = getRelation(num, wmax, Imax)
% Vary w and I by given num and given limits
% Returns mean firing rate, CV of firing rate, and CV of ISI across the network

    time = 10;
    n = 10;
    w = [wmax / num : wmax / num : wmax];
    I = [Imax / num : Imax / num : Imax];
    mFR = zeros(num, num);
    cvFR = zeros(num, num);
    cvISI = zeros(num, num);
```

```
    for iw = 1 : length(w)
        for iI = 1 : length(I)
            [~, spks, spkcnt] = randNet(time, I(1, iI), w(1, iw), n);
            [~, cvISI(iw, iI)] = getISI(spks, spkcnt);
            [~, mFR(iw, iI), cvFR(iw, iI)] = getFR(spkcnt, time);
        end
    end

end
```

In all 3 plots, w adds randomness to the values. The firing rate increases as I increases. The CV of firing rate and the CV of ISI's increase as w increases. When w = 8 and I = 8, the CV of the firing rate is about 1.







PART II: Sensory Network

In the sensory network, I separate the neurons into 2 pools, where the neurons in the same pool excites each other with weight wE > 0, and the neurons in different pools inhibit each other with weight wI < 0. Here's the function to simulate a sensory network.

```
function [V, spks, spkcnt] = sensNet(time, I1, I2, wE, wI, n)
% 2-Pool sensory LIF network by given time, given neuron number n
% Constant input for 1st pool I1, for 2nd pool I2
% Excite weight wE, inhibit weight wI
% Returns the membrane potential V
% And the binary spike sequence for each neuron spks
% And the spike count for each neuron spkcnt

    decay = 20;
    v_rest = -65;
    v_thres = -50;
    v_reset = -70;
    dt = 1;    % ms
    bins = time * 1000 / dt;
    spkcnt = zeros(n, 1);

    % Calculate the connection strength matrix for two pools of neurons
    weights = zeros(n, n);
    for i = 1 : n
        for j = 1 : n
            if (i > n/2 && j > n/2) || (i <= n/2 && j <= n/2)
                weights(i, j) = wE;
            else
                weights(i, j) = wI;
            end
```

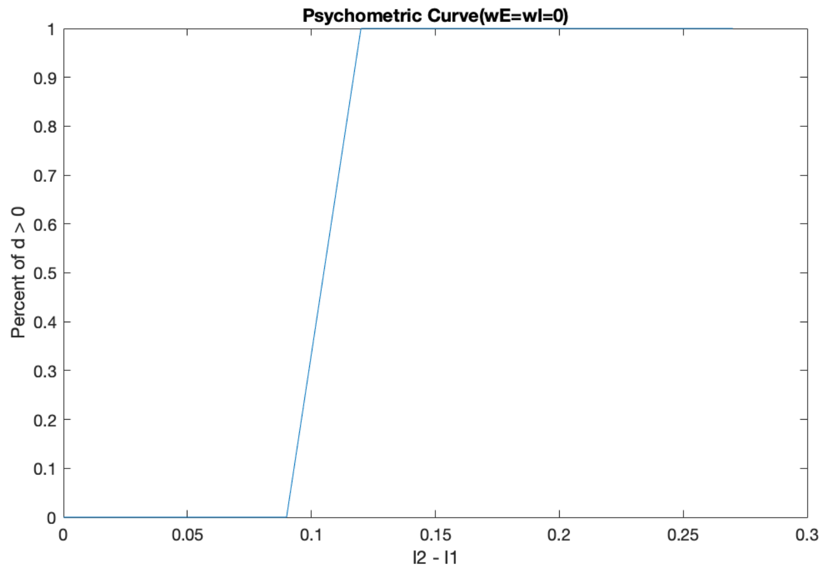
```
        end
    end

    % Initialize membrane potential V
    V = zeros(n, bins);
    spks = zeros(n, bins);
    for i = 1 : n
        V(i, 1) = v_rest + rand() * (v_thres - v_reset);
        spks(i, 1) = 0;
    end

    % Simulate for given time
    for t = 2 : bins
        for i = 1 : n

            v_prev = V(i, t - 1);

            if v_prev > v_thres
                V(i, t) = v_reset;
                spks(i, t) = 1;
                spkcnt(i, 1) = spkcnt(i, 1) + 1;

            elseif v_prev < v_reset
                V(i, t) = v_reset;

            else
                stim = 0;  % total weighted spike stimuli from other neurons
                for j = 1 : n
                    if spks(j, t - 1) == 1
                        stim = stim + weights(i, j);
                    end
                end

                if(i > n/2)
                    input = I2;
                else
                    input = I1;
                end
                V(i, t) = v_prev + dt * (-1 / decay * (v_prev - v_rest) + input +
stim);
            end

        end
    end
end
```

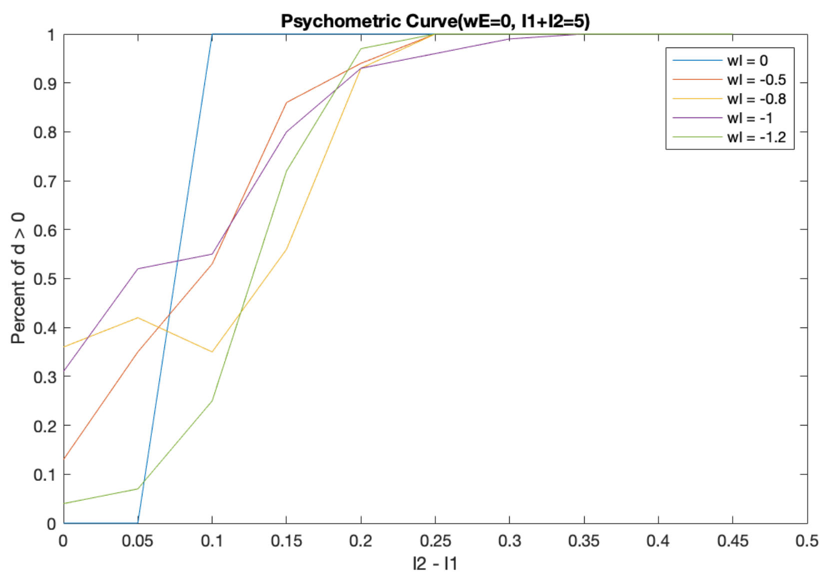
The average firing rate is 10 spikes/s when I1 = I2 = 0.757. That's the inputs that makes the neurons reach the spiking threshold if we keep I1 = I2. Next I have the function to plot the psychometric curve: percentage of d > 0 vs. I2 – I1. This is the decision variable of deciding whether I2 > I1. The read out weight v = 1 for neurons in pool 1, v = 0 for neurons in pool 2. d is calculated as follows.

$$d = \sum v_i spkcnt_i$$

```
function [diff, percentCorrect] = psychometric(wE, wI, sumI, diffrange, diffnum)
% Psychometric curve where
% percentCorrect is the percent when d > 0 (Choose I2 > I1)
% diff is the increasing values of I2 - I1 following the given constraints
% Given excite weight wE, inhibit weight wI
% sumI is the fixed value of I2 + I1
% diffrange is the limited range of I2 - I1
% diffnum is the number of values of I2 - I1 we take (length of diff)

    time = 1;
    n = 10;
    trlnum = 100;
    v = [-1, -1, -1, -1, -1, 1, 1, 1, 1, 1];

    I1 = zeros(1, diffnum);
    I2 = zeros(1, diffnum);
    percentCorrect = zeros(1, diffnum);
    halfsum = sumI / 2;
    ddiff = diffrange / 2 / diffnum;
    for i = 1 : diffnum
        I1(1, i) = halfsum - (i - 1) * ddiff;
        I2(1, i) = halfsum + (i - 1) * ddiff;

        correctcnt = 0;
        for trial = 1 : trlnum
            [~, ~, spkcnt] = sensNet(time, I1(1, i), I2(1, i), wE, wI, n);
            d = 0;
            for j = 1 : n
                d = d + v(1, j) * spkcnt(j, 1);
            end
            if d > 0
                correctcnt = correctcnt + 1;
            end
        end
        percentCorrect(1, i) = correctcnt / trlnum;
    end
    diff = I2 - I1;

end
```

We fix wE = 0, I1 + I2 = 5, and vary wI to see the change in the psychometric curve as follows. We see that a smaller wI yields a worse decision. We take wI = -0.5 to make the curve steepest and continue.



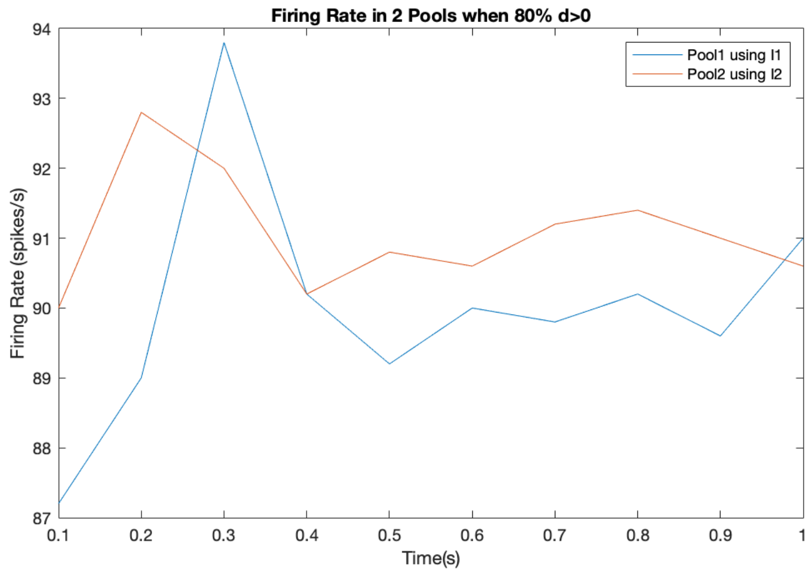


The next function returns the change of average firing rates for both pools in one 1-s-long trial.

```
function [fr1, fr2] = fr1trl(wE, wI, I1, I2)
% Given excite weight wE, inhibit weight wI
% Input for 1st pool I1, for 2nd pool I2
% Return the mean firing rate for each pool across the network in 10 bins of 1s

    time = 1;   % s
    blength = 100; % ms
    bins = time * 1000 / blength;
    n = 10;
    trlnum = 10;
    fr1 = zeros(1, bins);
    fr2 = zeros(1, bins);
```

```
    allfr1 = zeros(trlnum, bins);
    allfr2 = zeros(trlnum, bins);

    for trial = 1 : trlnum
        [~, spks, ~] = sensNet(time, I1, I2, wE, wI, n);
        tcnt = 0;
        bin = 1;
        spksum1 = 0;
        spksum2 = 0;
        for t = 1 : time * 1000
            tcnt = tcnt + 1;
            for i = 1 : n
                if i <= n/2
                    spksum1 = spksum1 + spks(i, t);
                else
                    spksum2 = spksum2 + spks(i, t);
                end
            end

            if tcnt >= blength
                tcnt = 0;
                allfr1(trial, bin) = spksum1 / (n / 2 * (blength / 1000));
                allfr2(trial, bin) = spksum2 / (n / 2 * (blength / 1000));
                bin = bin + 1;
                spksum1 = 0;
                spksum2 = 0;
            end

        end
    end

    for bin = 1 : bins
        fr1(1, bin) = mean(allfr1(:, bin));
        fr2(1, bin) = mean(allfr2(:, bin));
    end

end
```

Fix wI = -0.5, I1 = 2.43, I2 = 2.57 (I2 – I1 = 0.14), so that the percent of d > 0 is about 80%. The firing rate average from 10 trials is as follows. We see that the neurons in pool 2 have a higher firing rate for most of the times, but not always, which is expected because the percent correct is 80%.



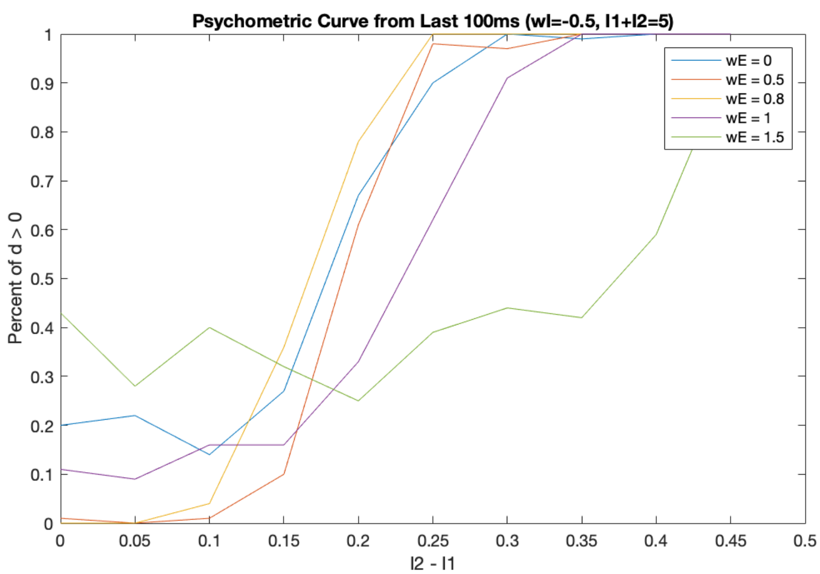
PART III: Decision-Making (Attractor) Network

In this part I calculated the psychometric curve where only the spikes in the last time bin(100ms) of a trial is taken into account.

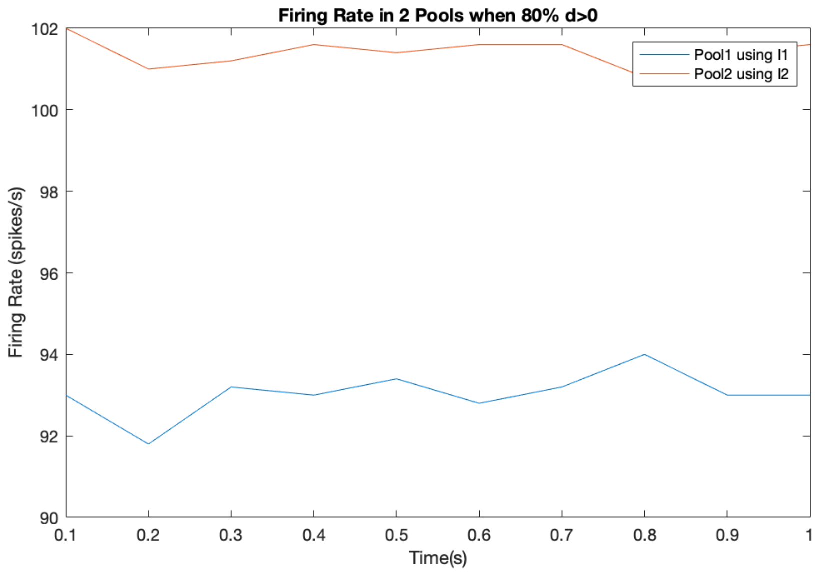
```
function [diff, percentCorrect] = psymelastbin(wE, wI, sumI, diffrange, diffnum)
% Psychometric curve
% Only the last 100ms of the simulation's spikes are taken into account
% percentCorrect is the percent when d > 0 (Choose I2 > I1)
% diff is the increasing values of I2 - I1 following the given constraints
% Given excite weight wE, inhibit weight wI
% sumI is the fixed value of I2 + I1
% diffrange is the limited range of I2 - I1
% diffnum is the number of values of I2 - I1 we take (length of diff)

    time = 1;
    n = 10;
    trlnum = 100;
    blength = 100; % ms
    v = [-1, -1, -1, -1, -1, 1, 1, 1, 1, 1];

    I1 = zeros(1, diffnum);
    I2 = zeros(1, diffnum);
    percentCorrect = zeros(1, diffnum);
    halfsum = sumI / 2;
    ddiff = diffrange / 2 / diffnum;
    for i = 1 : diffnum
        I1(1, i) = halfsum - (i - 1) * ddiff;
        I2(1, i) = halfsum + (i - 1) * ddiff;

        correctcnt = 0;
```

```
        for trial = 1 : trlnum
            [~, spks, ~] = sensNet(time, I1(1, i), I2(1, i), wE, wI, n);
            d = 0;
            for t = time * 1000 - blength + 1 : time * 1000
                for j = 1  : n
                    d = d + v(1, j) * spks(j, t);
                end
            end
            if d > 0
                correctcnt = correctcnt + 1;
            end
        end
        percentCorrect(1, i) = correctcnt / trlnum;
    end
    diff = I2 - I1;

end
```

Fix wI = -0.5, I1 + I2 = 5, and vary wE. We see that lower wE gives us a worse percentage when the contrast of I1 and I2 is low, but increases very fast as the contrast increases. We fix wE = 0.5 to make the curve the steepest.



Fix wE = 0.5, wI = -0.5, I1 = 2.39, I2 = 2.61 (I2 – I1 = 0.22), so that the percent of d > 0 is about 80%. The firing rate average from 10 trials is as follows. This time the difference of the firing rates in the 2 pools becomes much larger than when wE = 0. It means that the connections between the neurons in the same pool excite the neurons and this makes the difference larger.


Firing Rate in 2 Pools when 80% d>0
Firing Rate (spikes/s)
102
100
98
96
94
92
90
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time(s)
Pool1 using I1
Pool2 using I2