BE 521: Homework 5

Vision

Spring 2019

42 points

Due: Thursday, 02/27/2020 11:59pm

Objective: Visual responses and likelihood

Andrew Clark
Collaborators: Jack Bellwoar

V1 Dataset

In this homework, you will work with data from 18 cells recorded from mouse primary visual cortex (also known as V1). Cells in this area are responsive to specific angles. Hence, a common stimulation paradigm is to show the subject a sinusoidal grating drifting at a specific angle (see Figure 1). This data was collected

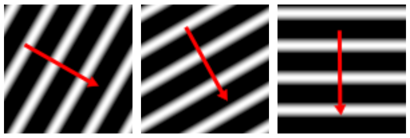


Figure 1: Example sinusoidal drifting grating at (in order left to right) 30, 60, and 90 degrees, where the red arrows shows the direction of the drift.

and graciously provided by Daniel Denman in the Contreras Lab, University of Pennsylvania. The file `mouseV1.mat` contains two variables: `neurons`, a cell array representing all the times that each of the 18 cells fired a spike during the approximately 7 minute long experiment, and `stimuli`, which provides the time (first column, in milliseconds) that a given angle (second column) was presented in this experiment. Note that each stimulus in `stimuli` is presented for exactly 2 seconds, after which a gray screen is presented for approximately 1.5 seconds (therefore each trial is approximately 3.5 seconds in duration.)

1 Stimulus Response (11 pts)

In this section, you will explore the response of the cells to different stimulus angles.

1. How many unique grating angles, m, are there in `stimuli`? (1 pt)

```
unique_angles_col=unique(stimuli(:,2));

num_unique_angles=length(unique_angles_col)
```

```
num_unique_angles =

    12
```

As computed by the code above, there are 12 unique grating angles in stimuli.

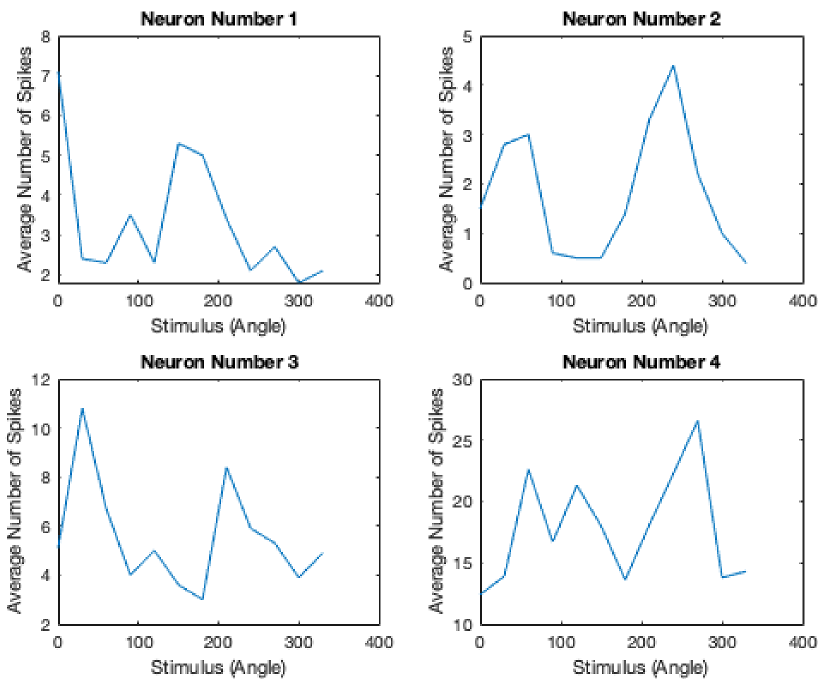
2. A *tuning curve* is frequently used to study the response of a neuron to a given range of input stimuli. To create tuning curves for this data, calculate the average number of spikes each cell fires in response to each grating angle. Store the result in an $18 \times m$ dimensional matrix, where each element represents the response of a single neuron to a particular input stimulus angle, with each neuron assigned a row and each angle assigned a column. In a 2×2 Matlab subplot, plot the tuning curve for the first four cells. Place the stimulus angle on the x-axis and the number of spikes on the y-axis. (6 pts)

```
%calculate the average number of spikes each cell fires in response to each
%grating angle

%store in an 18 x 12 matrix with each neuron assigned a row and each angle
%assigned a column

%neurons is a cell array representing all of the times that each of the 18
%cells fired a spike during the

%stimuli provides the time that a given angle. each of the 12 stimulus
%angles appear 10 times

%create matrix with angles as columns and stimulus times as rows

zero_list=[];
thirty_list=[];
sixty_list=[];
ninety_list=[];
one_twenty_list=[];
one_fifty_list=[];
one_eighty=[];
two_ten=[];
two_forty=[];
two_seventy=[];
three_hundo=[];
three_thirty=[];

for i=1:length(stimuli)
    if stimuli(i,2) == 0
        zero_list(i)=stimuli(i,1);
    elseif stimuli(i,2)==30
        thirty_list(i)=stimuli(i,1);
    elseif stimuli(i,2) == 60
        sixty_list(i)=stimuli(i,1);
    elseif stimuli(i,2) == 90
        ninety_list(i) =stimuli(i,1);
    elseif stimuli(i,2) == 120
        one_twenty_list(i) = stimuli(i,1);
    elseif stimuli(i,2) == 150
        one_fifty_list(i)= stimuli(i,1);
    elseif stimuli(i,2) == 180
        one_eighty(i)= stimuli(i,1);
    elseif stimuli(i,2) == 210
        two_ten(i)=stimuli(i,1);
    elseif stimuli(i,2) == 240
        two_forty(i) =stimuli(i,1);
    elseif stimuli(i,2) == 270
        two_seventy(i) = stimuli(i,1);
    elseif stimuli(i,2) == 300
        three_hundo(i) = stimuli(i,1);
    elseif stimuli(i,2) == 330
        three_thirty(i) = stimuli(i,1);
    end

    zero_list=zero_list(zero_list~=0);
    thirty_list=thirty_list(thirty_list~=0);
    sixty_list=sixty_list(sixty_list~=0);
    ninety_list=ninety_list(ninety_list~=0);
    one_twenty_list=one_twenty_list(one_twenty_list~=0);
    one_fifty_list=one_fifty_list(one_fifty_list~=0);
    one_eighty=one_eighty(one_eighty~=0);
    two_ten=two_ten(two_ten~=0);
    two_forty=two_forty(two_forty~=0);
    two_seventy=two_seventy(two_seventy~=0);
    three_hundo=three_hundo(three_hundo~=0);
    three_thirty=three_thirty(three_thirty~=0);
end



angles_mat=zeros(10,12);

angles_mat(:,1)=zero_list;
angles_mat(:,2)=thirty_list;
angles_mat(:,3)=sixty_list;
angles_mat(:,4)=ninety_list;
angles_mat(:,5)=one_twenty_list;
angles_mat(:,6)=one_fifty_list;
angles_mat(:,7)=one_eighty;
angles_mat(:,8)=two_ten;
angles_mat(:,9)=two_forty;
angles_mat(:,10)=two_seventy;
angles_mat(:,11)=three_hundo;
angles_mat(:,12)=three_thirty;

%angles_mat now has stim times of all the angles in it
```

```
%r=1;
%col=neurons{r}
tuning_curve_mat=zeros(18,12);
%counter=0
 for r=1:length(neurons)
     col=neurons{r};
        for s=1:12
            counter=0;
            %avg_spike=0
            for k=1:10
                for i=1:length(col)
                    if col(i) <= angles_mat(k,s)+3.5e3 && col(i) >= angles_mat(k,s)%check upper bound first
                        counter=counter+1;
                    end
                end
            end
            %avg_spike=counter/10
            tuning_curve_mat(r,s)=counter;
        end
 end
```

```
final_tuning=tuning_curve_mat/10;
```

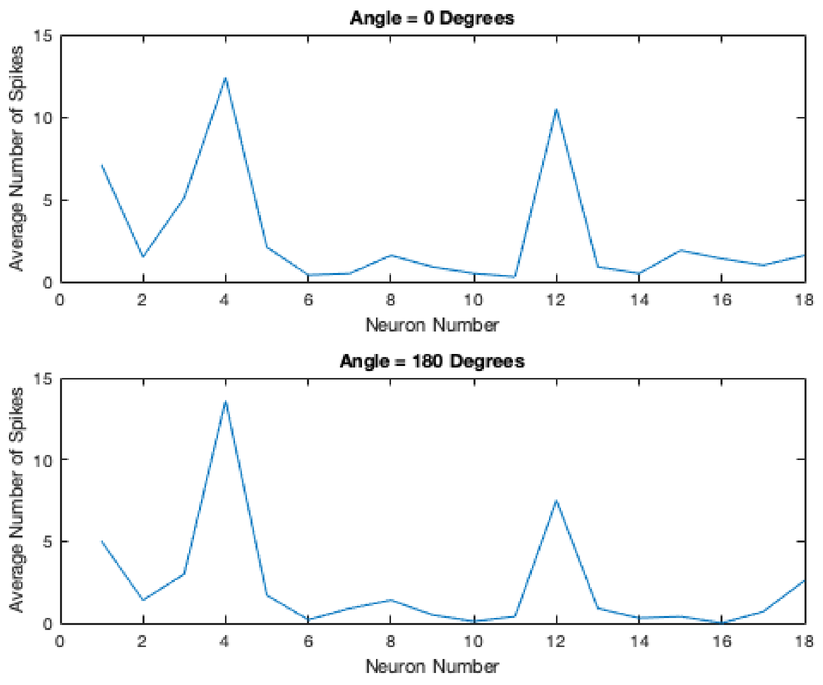
```
%create plots
figure
subplot(2,2,1)
plot(unique_angles_col, final_tuning(1,:))
xlabel('Stimulus (Angle)')
ylabel('Average Number of Spikes')
title('Neuron Number 1')
subplot(2,2,2)
plot(unique_angles_col, final_tuning(2,:))
xlabel('Stimulus (Angle)')
ylabel('Average Number of Spikes')
title('Neuron Number 2')
subplot(2,2,3)
plot(unique_angles_col, final_tuning(3,:))
xlabel('Stimulus (Angle)')
ylabel('Average Number of Spikes')
title('Neuron Number 3')
subplot(2,2,4)
plot(unique_angles_col, final_tuning(4,:))
xlabel('Stimulus (Angle)')
ylabel('Average Number of Spikes')
title('Neuron Number 4')
```

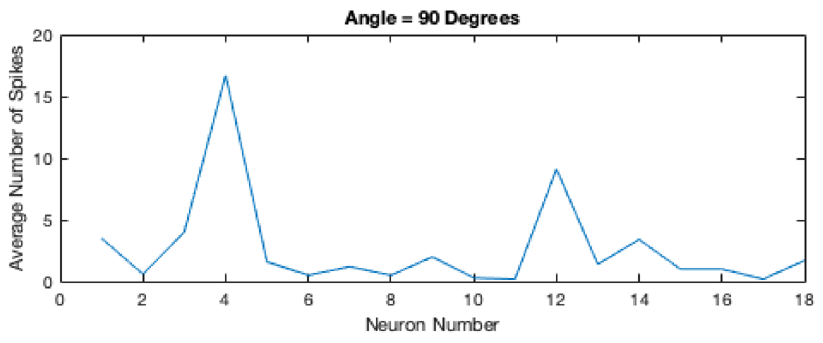


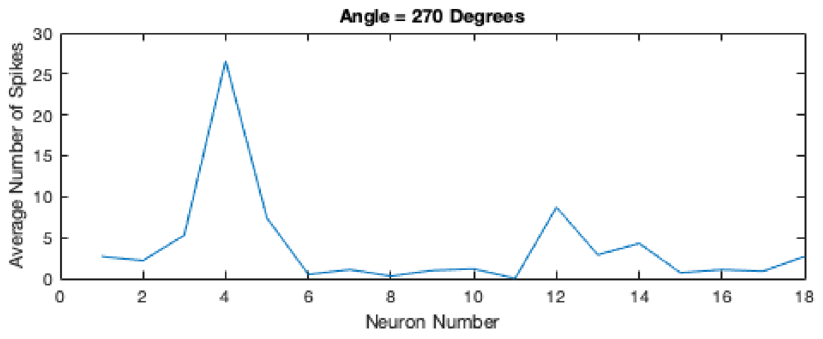
(a) Look through the tuning response curves of each of the 18 cells. How reasonable is it to assume that the response of a given cell to angle θ is the same as its response to angle $\theta + 180$? Include at least a few tuning curves to back up your answer. (2 pts)

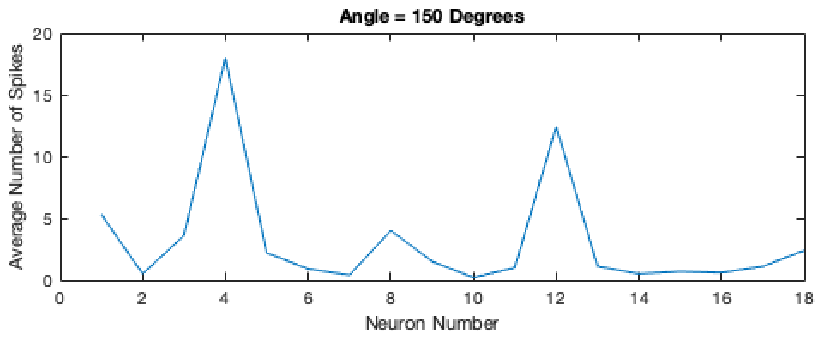
```
%show subplots with theta and theta + 180
neuron_list=1:18;
%Figure for 0 degrees
figure
subplot(2,1,1)
```

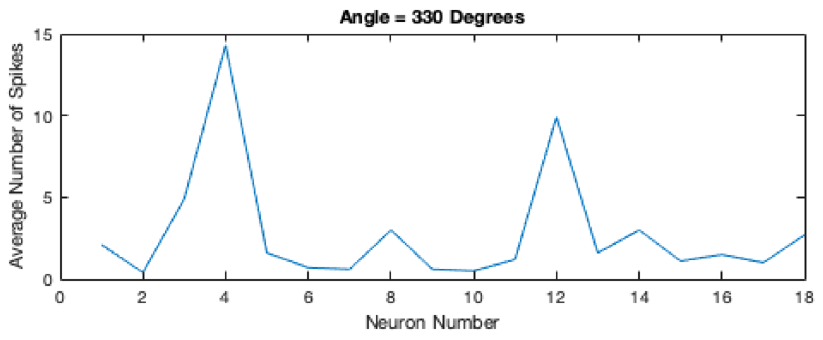
```
plot(final_tuning(:,1))
title('Angle = 0 Degrees')
xlabel('Neuron Number')
ylabel('Average Number of Spikes')
subplot(2,1,2)
plot(final_tuning(:,7))
title('Angle = 180 Degrees')
xlabel('Neuron Number')
ylabel('Average Number of Spikes')

figure
subplot(2,1,1)
plot(final_tuning(:,4))
title('Angle = 90 Degrees')
xlabel('Neuron Number')
ylabel('Average Number of Spikes')
subplot(2,1,2)
plot(final_tuning(:,10))
title('Angle = 270 Degrees')
xlabel('Neuron Number')
ylabel('Average Number of Spikes')
figure
subplot(2,1,1)
plot(final_tuning(:,6))
title('Angle = 150 Degrees')
xlabel('Neuron Number')
ylabel('Average Number of Spikes')
subplot(2,1,2)
plot(final_tuning(:,12))
title('Angle = 330 Degrees')
xlabel('Neuron Number')
ylabel('Average Number of Spikes')
```











It is reasonable to assume that the response of a given cell to the angle theta is equal to theta + 180. This is illustrated in each of the subplots above, each of the curves in the top of the subplot are very, very similar to the plot in the bottom. We can see from these plots that the curves mostly match for many neurons. Thus, it is reasonable to assume that the response of a given cell to the angle theta is equal to theta + 180 as the neurons responded mostly in the same way to both theta and theta + 180.

(b) Does this assumption have any physiological justification (given what you know about the types of cells in V1)? (2 pts)

This assumption does have physiological justification because the types of cells in V1 (the primary visual cortex) are cells intrepretating visual stimuli and the stimuli at theta and theta + 180 visually look exactly the same (the gratings appear the same), so they appear to the cells elliciting a response. Thus, the response of the cells in V1 will be the same if they are presented with a visual stimulus that appears the same. As such, it is concievable to say any variation in neural cell response is due to small inherent differences in the behavior of V1 cells when they are presented with any stimulus that is the same.

2 Neural Decoding (31 pts)

Suppose we would like to work backwards - that is, for a given neural response, can we predict what the grating angle was? This process is called "neural decoding," and is especially of interest to the BCI motor control community (as we'll see in a later homework). In this section, we will try out an approach which is detailed in Jazayeri & Movshon 2006 [1]. The method we will use involves finding the maximum likelihood of the data.

Here, the data is the number of spikes s_i that cell i fires when the subject sees a stimulus with grating angle θ. One way to think about our likelihood function is to ask the question "given a stimulus angle θ, how many spikes would I expect this cell to fire?" We can represent this number of spikes s_i using a Poisson process with parameter $f_i(\theta)$ for a stimulus θ, where f_i represents neuron i's tuning function. A Poisson distribution is often used to model count data that occurs at a constant rate, and in this case the rate is given by $f_i(\theta)$. In other words, our likelihood function $L_i(\theta)$ for each neuron i is the probability $p(s_i|\theta)$ of neuron i firing s_i spikes for a given value of θ. The idea in this method is to calculate the log likelihood[2] function of each neuron and then add them all together to get the log likelihood function of the entire population of (n) neurons. We often work with the *log* likelihood because it allows adding of probabilities instead of multiplying, which can lead to numerical problems.

$$\begin{aligned}
p(s_i|\theta) &\sim Pois(f_i(\theta)) = \frac{f_i(\theta)^{s_i}}{s_i!}e^{-f_i(\theta)} && \text{(Poisson probability density)}\\
Ł_i(\theta) &= p(s_i|\theta) && \text{(Likelihood of a given neuron firing at } s_i\text{)}\\
Ł(\theta) &= \prod_{i=1}^{n} p(s_i|\theta) && \text{(Joint likelihood of all n neurons)}\\
\log Ł(\theta) &= \sum_{i=1}^{n} \log L_i(\theta) = \sum_{i=1}^{n} \log p(s_i|\theta) && \text{(Take log)}\\
&\propto \sum_{i=1}^{n} s_i \log f_i(\theta) && \text{(evaluation of PDF and simplifying)}
\end{aligned}$$

Thus, we can define the log likelihood for each neuron i as the log of its tuning curve $f_i(\theta)$ times the number of spikes s_i it fires for a particular stimulus θ, and the population log likelihood is simply the summation across all cells. This tells us that, given a set of tuning curves $f_i(\theta)$, we can compute the likelihood of observing our data s. But we already have those tuning curves for each cell from question 1.2, so all we need to know for a new (hidden) stimulus is how many spikes each neuron fires. Let $\mathbf{s}$ be the n-dimensional column vector of the number of spikes each cell fires after the subject is presented with a new stimulus θ' and let $\mathbf{F}$ be the $n \times m$ matrix representing the tuning curves of each neuron at each of the m stimuli (for us,

[1] A copy of this paper is included if you are curious, but we walk you through the method in this homework.

[2] log = natural log unless otherwise specified

m is the number of stimuli between 0 and 150 degrees because we assume that all neurons respond equally to θ and $\theta + 180$ degrees.) We can then compute the log likelihood of the new stimulus θ' easily using the inner product of **s** and **F**: `L = s'*log(F)`.

1. Compute the matrix `F` by recalculating the tuning curves you calculated in question 1.2 using only the **first 70** trials (this is akin to our "training" data). You will use the remaining 50 trials (as "testing" data) to make predictions. Make a histogram of the number of stimulation angles for the first 70 trials to ensure that each angle (0 to 150) is presented at least a few times. (4 pts)

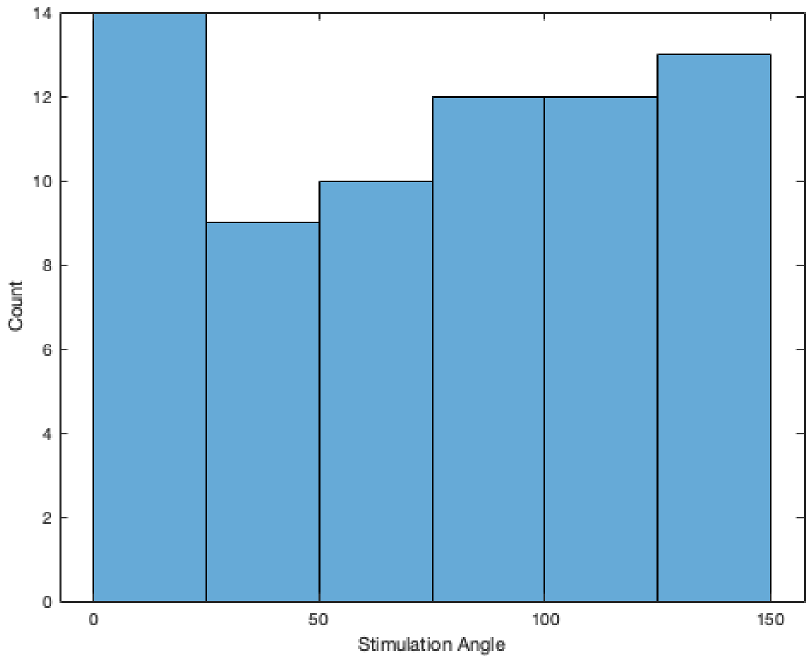
```
% make a new tuning curve using only the first 70 datapoints in the stimuli
% matrix

%if you are above 180 subtract 180 from the angle to make sure it all lies
%between 0 and 180

% first make a new matrix and subtract 180 from all the values 180 and above in the
% angles_mat matrix

% F should be 18 x 6

% recalculate tuning curve matrix with only the first 70 trials

% divide by number of unique times the angles show up

% six bins for the histogram

%create matrix with angles as columns and stimulus times as rows
```

```
F_mat=zeros(18,6);
for r=1:18
    col=neurons{r};
    angle_list=[];
    for k=1:length(col)
        for i=1:70
            if col(k)<= stimuli(i,1)+3.5e3 && col(k) >= stimuli(i,1)%check upper bound first and lower bound
                %grab angle
                angle_list=[angle_list stimuli(i,2)];
            end
        end
    end

    for a=1:6
    F_mat(r,a)=length(find(angle_list == 30*(a-1) | angle_list == 30*(a-1)+180))/length(find(stimuli(1:70,2)

    end

end
```

```
F_mat
```

```
F_mat =

    6.9286    2.4444    1.1000    2.7500    1.5833    2.7692
    1.5714    2.5556    4.1000    1.4167    1.0833    0.4615
    4.4286    7.4444    4.9000    4.0833    3.2500    4.2308
   15.7857   12.0000   20.1000   19.1667   16.7500   15.0769
    2.2857    1.1111    1.3000    3.6667    2.4167    0.9231
    0.3571    1.1111    1.9000    0.7500    0.7500    1.1538
    0.7143    4.1111    4.5000    1.2500    1.0000    0.6154
    1.8571    0.1111    0.2000    0.2500    0.5000    2.4615
    0.8571    0.5556    1.8000    1.0833    2.1667    0.6154
    0.2857    0.7778    0.4000    0.4167    0.0833    0.3077
    0.5000    0.4444    0.4000    0.1667    0.2500    1.3077
   10.2143    9.8889   10.5000   10.4167   12.3333   11.0769
    0.7857    0.7778    1.4000    1.5000    1.3333    0.5385
    0.5714    0.3333    0.7000    2.4167    4.0833    2.3846
    1.2143    1.5556    2.0000    0.8333    0.4167    0.4615
    0.8571    1.2222    1.5000    1.3333    1.3333    0.7692
    1.2143    0.4444    0.8000    0.6667    0.6667    1.0000
    2.6429    2.3333    2.6000    1.9167    1.1667    2.4615
```

```
%create histogram

angles_hist=stimuli(1:70,2);
for i=1:length(angles_hist)
    if angles_hist(i)>=180
        angles_hist(i)=angles_hist(i)-180;
    end
end


figure
histogram(angles_hist,6)
xlabel('Stimulation Angle')
ylabel('Count')
```



2. For the 50 “testing” trials, compute a $n \times 50$ matrix `S` where each row represents the number of spikes one neuron fired in response to a each of the 50 trials. With this, you can easily compute the log likelihood functions for all the trials at once with the command: `L_test = S'*log(F)`. (Hint: add a small number to `F` to avoid taking the log of 0)

(a) Plot the likelihood functions for the first four testing trials in a 2×2 subplot. In the title of each plot, give the trial number (1, 2, 3, or 4) and the true stimulation angle. Make sure to label your axes correctly. (5 pts)

S is 18 x 50. Each row represents the number of times one neuron fired in response to each of the 50 trials S transpose is 50 x 18.

```
% L_test = S' * log(F)

%For 50 testing trials = stimuli(71,:), create a 18 x 50 matrix
```

```
%generate S matrix
S=zeros(18,50);

for r=1:18
    col=neurons{r};
    angle_list=[];
    firing_vector=zeros(1,50);
    count=0;
    for k=71:120 % iterate through the stimuli vector

        for i=1:length(col)%iterate through the neurons vector
            %if stimuli(i,1)
            if col(i)<= stimuli(k,1)+3.5e3 && col(i) >= stimuli(k,1)%check upper bound first and lower
                %update count
                count=count+1;
                firing_vector(k-70)=count;
                %angle_list=[angle_list stimuli(i,2)];


            end
        end

        count=0;%reset count
    end
    for t=1:50
        S(r,t)=firing_vector(t);
    end
end
```

```
S
```

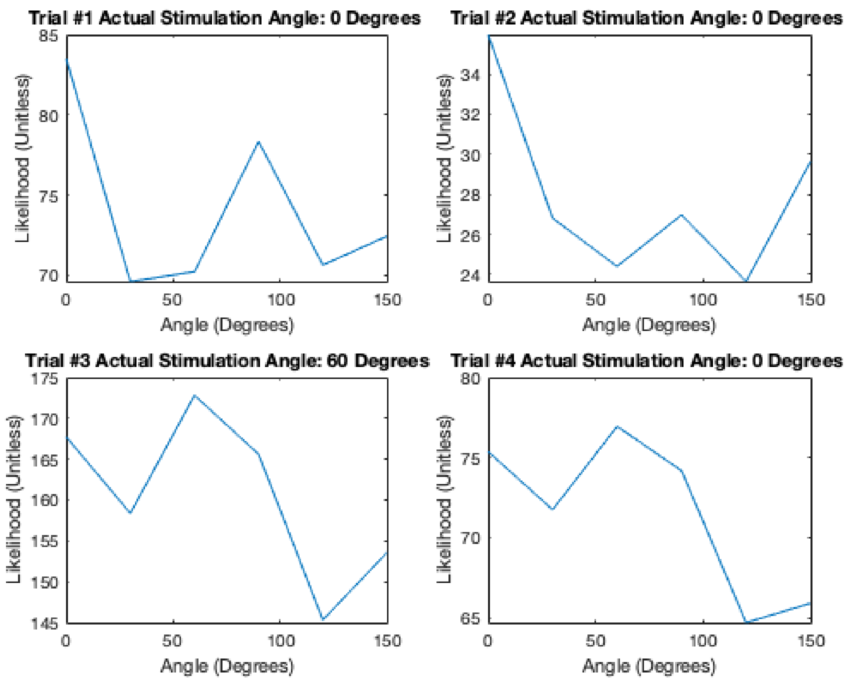
```
S =

  Columns 1 through 13

     9     6    10     2     1     5     4     5     0     3     2     0     4
     0     1     3     1     0     0     5     1     0     5     2     0     1
     5     3     9     9     9     4    11     1     4     4     1     0     3
    13     3    35    12    18    16    29     5     3    19     5     0     8
     3     0     0     3     7     7     0     0     2     5     0     0     0
     1     0     3     0     2     0     1     0     0     0     0     0     0
     0     1     2     0     0     0     2     0     0     1     2     0     0
     1     2     2     1     2     6     0     0     0     1     0     0     2
     0     0     1     2     3     0     0     0     0     0     0     0     3
     0     0     4     2     0     0     0     0     0     3     0     0     1
     0     0     0     0     3     1     0     0     2     0     0     0     0
     9     4    18     9     6    20    18     7    13     8     7     0     5
```

```
      1      0      1      0      2      2      0      1      0      0      0      0      4
      0      0      2      0      6      0      0      0      2      0      0      1      3
      1      0      7      3      0      1      5      0      0      0      1      0      0
      1      0      0      1      3      0      0      0      0      0      0      0      1
      0      0      0      0      2      0      0      0      0      0      0      0      1
      0      1      3      3      4      0      0      1      0      0      0      0      7

  Columns 14 through 26

      0      0      1      0      3      5      2      5      6      1      3     14      1
      0      1      0      2      3      0      0      0      1      1      0      1      3
      1      1      1      0      4      7      5      4      9     15      5      2     30
      2      2     12      1     12      1      3     12     20      8     23     23     19
      0      0      1      0      1      0      1      2      1      0      1      4      2
      0      0      1      0      0      0      0      0      0      0      0      0      2
      1      0      0      1      1      0      0      0      0      1      0      0      3
      0      0      1      0      1      0      1      1      3      2      5      6      0
      0      1      0      0      0      4      0      0      1      4      3      1      1
      0      2      0      0      1      1      0      3      0      1      1      0      1
      0      0      0      0      0      1      1      0      0      0      0      1      0
      4      3     13      1      7      7      5     10     15      1     14      8      6
      0      0      2      5      2      3      4     11      8      2      4      0      1
      2      0      2      0      0      2      0     12      4      1      3      0      0
      2      1      3      1      4      0      7      2      4     13      0      0     20
      0      0      0      0      0      0      1      0      0      0      0      0      0
      1      0      3      0      0      1      1      0      1      0      1      1      2
      1      0      3      0      2      0      5      1      1      0      2      2      0

  Columns 27 through 39

      3      4      2      5      6      8      3      3      0      0      4      2      4
      2      5      1      5      0      6      2      3      1      2      0      2      7
     19     17      9      7     11     24      7      5     12      5      6     14      6
     36     29     35     25     23     25     29     30     34     17     38     18     27
      0      0      2      1      9      1     10      7     11      4      1      0      4
      2      3      0      0      1      3      4      2      0      0      1      0      3
      4      2      1      0      0      2      4      3      1      1      0      1      1
      0      0      3      1      6      0      0      0      0      0      1      0      0
      1      0      2      0      1      2      0      0      1      0      3      2      1
      0      1      2      0      1      1      1      1      1      2      0      1      0
      0      0      0      0      1      2      0      0      0      0      0      0      0
     12     11     18      4     18      8      5     12      8      7     16      8     11
      2      1      4      0      3      2      3      2      0      1      3      0      5
      0      0     11      0      0      0      0      0     10      4      7      0      1
     15     17      1      0      0      8      9      4      1      0      1     12      2
      0      0      0      0      5      0      0      1      0      0      0      0      2
      1      0      0      0      2      2      1      2      0      1      3      5      1
      0      5      2      0      6      2      3      7      2      3      2      1      6

  Columns 40 through 50

      2      8      6      2      0      2      7      3      6      4      6
      4      1      0      4      5      1      4      1      4      0      3
     10      3      6      5      4      8      7      4      0      4      5
     25     34     30     30     26     24     28     26     51     17     18
      2      5      1      1      0      8      1      0     11      2      7
      0      0      0      4      1      1      0      0      0      0      0
      2      1      0      3      2      0      1      0      1      1      4
      0     11      2      0      0      1      0      8      1      0      0
      0     11      5      2      2      0      0      0      4      5      2
      0      1      1      2      0      0      2      1      1      1      1
      1      0      0      0      0      0      0      1      0      0      0
      7     16     11      4      8      5      4      8      7      2      3
      1      6      2      3      1      3      0      5      2      2      4
      1      0      4      0      0      8      1      2      6      2      2
      1     1     3     8     2     0     0     1     1     0     0
      1     3     0     0     0     1     1     2     0     1     3
      0     3     0     1     0     1     2     0     1     0     0
      7     1     2     3     0     3     3     4     2     2     6
```

```
%need to keep track of angles for each trial as well in a 1 x 50 matrix

angle_labels_mat=stimuli(71:120,2);

for i=1:length(angle_labels_mat)
    if angle_labels_mat(i)>=180
        angle_labels_mat(i)=angle_labels_mat(i)-180;
    end
end
```

```
%compute L_test

S_transpose=S';

L_test= S_transpose* log(F_mat);% this is the log likelihood function

L_test_with_angle=[L_test angle_labels_mat];

L=exp(L_test); %exponentiaing?

%need to keep track of the angles that each
```

```
%generate plots
angles=[0,30,60,90,120,150];
figure
subplot(2,2,1)
plot(angles,L_test(1,:))
title('Trial #1 Actual Stimulation Angle: 0 Degrees ')
xlabel('Angle (Degrees)')
ylabel('Likelihood (Unitless)')
subplot(2,2,2)
plot(angles, L_test(2,:))
title('Trial #2 Actual Stimulation Angle: 0 Degrees ')
xlabel('Angle (Degrees)')
ylabel('Likelihood (Unitless)')
subplot(2,2,3)
plot(angles, L_test(3,:))
title('Trial #3 Actual Stimulation Angle: 60 Degrees')
xlabel('Angle (Degrees)')
ylabel('Likelihood (Unitless)')
subplot(2,2,4)
plot(angles, L_test(4,:))
title('Trial #4 Actual Stimulation Angle: 0 Degrees ')
xlabel('Angle (Degrees)')
ylabel('Likelihood (Unitless)')
```



(b) How well do these four likelihood functions seem to match the true stimulation angle? Explain in a few sentences. (3 pts)
These four likelihood functions seem to match the true stimulation angle reasonable well, three out of the four subplots above have the highest likelihood at the true stimulation angle. However, the likelihood functions also sometimes show relatively high likelihoods for angles that are not the true stimulation angle (as can be seen in the upper left panel) and also sometimes the highest likelihood is not always the true stimulation angle (as can be seen in the bottom right panel). As such, it appears that these four likelihood functions are better predictors than chance (when considering the max likelihood given as the prediction of true stimulus angle), but still have considerable error.

(c) Compute the maximum likelihood estimate (MLE) for each of the 50 trials. This is another way of asking which angle θ has the highest probability.

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \; L(\theta) = \arg\max_{\theta} \; \log L(\theta)$$

In what percentage of the 50 trials did your MLE correctly predict the stimulation angle [0-150]? (5 pts)

```
%create two columns MLE_mat

mle_mat=zeros(50,1);

mle_mat=[mle_mat angle_labels_mat];

for i=1:length(mle_mat)
    max_row=0;
    %need grab column number of the max value
    max_row=max(L_test(i,:));
    [row,col]=find(L_test(i,:)==max_row);
    mle_mat(i,1)=col;
end
```

```
for h=1:length(mle_mat)
    if mle_mat(h,1)==1
        mle_mat(h,1)=0;
    elseif mle_mat(h,1)==2
        mle_mat(h,1)=30;
    elseif mle_mat(h,1)==3
        mle_mat(h,1)=60;
    elseif mle_mat(h,1)==4
        mle_mat(h,1)=90;
    elseif mle_mat(h,1)==5
        mle_mat(h,1)=120;
    elseif mle_mat(h,1)==6
        mle_mat(h,1)=150;
    end

end

num_correct=0;
for y=1:length(mle_mat)
    if mle_mat(y,1) == mle_mat(y,2)
        num_correct=num_correct+1;
    end

end

percent_correct = (num_correct/ 50)* 100
```

```
percent_correct =

    42
```

As computed by the code above, the MLE correctly predicted the true stimulation angle 42 percent of the time.

(d) In a few sentences, discuss how well this method worked for predicting the input angle from the response of the 18 neurons. What might you change in the experimental protocol to try and get a better prediction? (3 pts)

This method was better than chance (which would have been 16.67% correct) but not incredible (say accuracy higher than 95%) at making predictions. In order to get a better prediction I would change the experimental protocol so that more trials are run (and more data is collected for each stimulus angle). As more data is collected the ability of outliers in the collected data to skew the predictions calculated will decrease. As such, more data will allow for better inputs to the Poisson probability density function that is being utilized to create predictions via a series of calculations. In addition to increasing the number of trials for a given stimulus angle and a given neuron, I would also increase the actual number of neurons tested. This will help to give a better prediction because, in the method used here, the log likelihood function for the entire population of n neurons is computed (so increasing the number of neurons will yield more data for this calculation which will increase its accuracy as it is utilizing the Poisson distribution).

3. It is important to show that your findings are not a result of chance. One way to demonstrate this is using a “permutation test.” Here, we will perform a permutation test by randomly reassigning new grating angles to the 50 test responses and then calculating how often the new grating angles match the true stimulation angles.

   (a) Simulate the chance prediction (the “null” distribution) by randomly reassigning the stimulation angles 1000 times. For each permutation, calculate the percent accuracy of each label. Create a

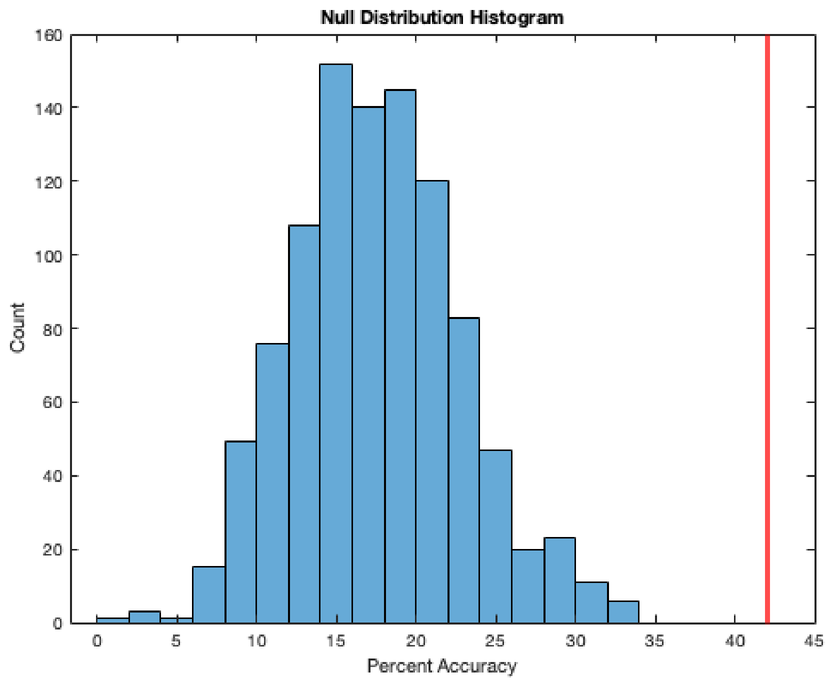
histogram of the 1000 accuracy measurements for the null distribution. Add a red vertical line with the accuracy calculated from using the Jazayeri method above. (4 pts)
simulate the chance prediction by randomly reassigning the stimulation angles 1000 times.

```
%select indices randomly from the angles matrix 50 times and do this 1000
%times


accuracy_measure_mat=zeros(1000,1);
for e=1: 1000
    r=randi(6,50,1);

    for h=1:length(r)
        if r(h,1)==1
            r(h,1)=0;
        elseif r(h,1)==2
            r(h,1)=30;
        elseif r(h,1)==3
            r(h,1)=60;
        elseif r(h,1)==4
            r(h,1)=90;
        elseif r(h,1)==5
            r(h,1)=120;
        elseif r(h,1)==6
            r(h,1)=150;
        end

    end

    num_correct=0;
    for y=1:length(mle_mat)
        if r(y,1) == mle_mat(y,2)%second column is the labels column
            num_correct=num_correct+1;
        end

    end

    accuracy_measure_mat(e,1)=(num_correct/50)*100;
end
%for each permutation calculate the percent accuracy of each label
```

```
%create a histogram of the 1000 accuracy measures
figure
histogram(accuracy_measure_mat)
hold on
xline(42,'r', 'LineWidth',3)
xlabel('Percent Accuracy')
ylabel('Count')
title('Null Distribution Histogram')
```

```
ans =

  ConstantLine with properties:

    InterceptAxis: 'x'
            Value: 42
            Color: [1 0 0]
        LineStyle: '-'
        LineWidth: 3
```

```
          Label: ''
    DisplayName: ''

  Use GET to show all properties
```



(b) Is the null distribution what you expected? Explain. (1 pt)

The null distribution is what I expected because the null distribution is approximately centered around 16.67% (which is 1/6 accuracy). Thus the most common percent accuracy generated by the null distribution is 16.67%, which makes sense because that is the expected value of the accuracy that would get from just picking one angle out of 6 and comparing it to the actual angle.

(c) What is the probability that your actual accuracy measurement comes from the null-distribution? That is, calculate the fraction of permutation samples with accuracy *more extreme* relative to the mean of the null distribution (less than your measurement if your value is less than the mean, or more than your measurement if your value is more than the mean). (2 pts)

```
%calculate p value
count_p=0;
for u=1:length(accuracy_measure_mat)
    if accuracy_measure_mat(u,1) >= 42
        count_p=count_p+1;
    end
end

p_val=(count_p/1000)*100
```

```
p_val =

     0
```

As calculated by the code above, this probability is 0 percent.

(d) What would the probability be if your accuracy had been 25%? (1 pt)

```
%calculate p value
count_p=0;
for u=1:length(accuracy_measure_mat)
    if accuracy_measure_mat(u,1) >= 25
        count_p=count_p+1;
    end
end

p_val_25=(count_p/1000)*100
```

```
p_val_25 =

     6
```

As calculated by the code above, if the accuracy had been 25 percent this probability is 6.9 percent.

(e) What is this value typically called? (1 pt)

This value is typically called the p-value.

4. The tuning curves and neuron responses to a new trial were calculated using the number of spikes each neuron fired after the stimulation. But what if a particular neuron just happens to fire a lot and fires even more when it gets a stimulus to which it's tuned? Those neurons could "swamp" the log likelihood estimate given in Equation 1 by virtue of just having a larger average s_i and $f_i(\theta)$. How might we correct for this type of behavior? Suggest a possible method. (2 pts)

We might correct for this type of behavior by normalizing each neuron to itself before proceeding with calculations using s_i and $f_i(\theta)$. We could do this by seeing how many times each neuron fired in response to each stimuli and then dividing all these firing counts by the max number of firings each particular neuron completed. This way we have normalized the tendicies of each neuron to itself (each number of firings will now be a number between zero and one). As a result, a particular neuron that happens to fire alot will not "swamp out" the log likelihood estimate given in Equation 1.