EE239AS.2, Spring 2017 **Homework #3 Solutions**

Department of Electrical Engineering Prof. J.C. Kao

University of California, Los Angeles Special Reader M. Srinivasan

Due Friday, 05 May 2017, uploaded to Gradescope.

Covers material up to Poisson Processes III.

100 points total.

1. (13 points) You are recording the activity of a neuron, which is spiking according to a Poisson process with rate λ. At some point during your experiment, the recording equipment breaks down and begins dropping spikes randomly with probability p.

   (a) (10 points) Let the random variable M be the number of recorded spikes with the broken equipment. Show that the distribution of M is Poisson($(1-p)\lambda s$). (Hint: If N is a random variable denoting the number of actual spikes, what is $\mathbf{Pr}(M = m|N = n)$?)

   Solution: We want to calculate $\mathbf{Pr}(M = m)$. By the law of total probability,

$$\begin{aligned}
\mathbf{Pr}(M = m) &= \sum_{n=0}^{\infty} \mathbf{Pr}(M = m, N = n) \\
&= \sum_{n=m}^{\infty} \mathbf{Pr}(M = m, N = n) \\
&= \sum_{n=m}^{\infty} \mathbf{Pr}(M = m|N = n)\mathbf{Pr}(N = n)
\end{aligned}$$

   where we used the fact that $\mathbf{Pr}(M = m, N = n) = 0$ if $n < m$ (it's not possible to have more dropped spikes than actual spikes).

   The distribution of $M = m|N = n$ is Binomial with parameter $1-p$, i.e., $M|N = n \sim$ Binom$(n, 1-p)$. Thus,

$$\mathbf{Pr}(M = m|N = n) = \begin{pmatrix} n \\ m \end{pmatrix} (1-p)^m p^{n-m}$$

   Now, we calculate our above sum.

$$\begin{aligned}
\mathbf{Pr}(M = m) &= \sum_{n=m}^{\infty} \mathbf{Pr}(M = m|N = n)\mathbf{Pr}(N = n) \\
&= \sum_{n=m}^{\infty} \frac{n!}{m!(n-m)!}(1-p)^m p^{n-m} \frac{(\lambda s)^n \exp(-\lambda s)}{n!} \\
&= \frac{(1-p)^m (\lambda s)^m \exp(-\lambda s(1-p)}{m!} \sum_{n=m}^{\infty} \frac{p^{n-m}(\lambda s)^{n-m} \exp(-\lambda s p)}{(n-m)!} \\
&= \frac{((1-p)\lambda s)^m \exp(-\lambda s(1-p))}{m!}
\end{aligned}$$

   where for the third line, we separated out all the m terms that give us the Poisson distribution with parameter $(1-p)\lambda s$ and left all the residual terms in the sum. By performing a change of variables, $k = n - m$ on the sum, we see that it is the sum over a Poisson distribution, and hence equal to 1.

(b) (1 points) What is the rate of the Poisson process in part (a)?

Solution: From the previous part, it is clear the rate is $(1-p)\lambda$.

(c) (2 points) What is the distribution on the number of spikes dropped within a τ second interval?

Solution: Following the first two parts of the question, it is Poisson with parameter $p\lambda\tau$.

2. (35 points) Homogeneous Poisson process
We will consider a simulated neuron that has a cosine tuning curve described in equation (1.15) in *TN*[1]:

$$\lambda(s) = r_0 + (r_{\max} - r_0)\cos(s - s_{\max}), \tag{1}$$

where λ is the firing rate (in spikes per second), s is the reaching angle of the arm, $s_{\max}$ is the reaching angle associated with the maximum response $r_{\max}$, and r_0 is an offset that shifts the tuning curve up from the zero axis. Let $r_0 = 35$, $r_{\max} = 60$, and $s_{\max} = \pi/2$.

(a) (6 points) Spike trains
For each of the following reaching angles ($s = k \cdot \pi/4$, where $k = 0, 1, \ldots, 7$), generate 100 spike trains according to a homogeneous Poisson process. Each spike train should have a duration of 1 second. Plot 5 spike trains for each reaching angle in the same format as shown in Figure 1.6(A) in *TN*. (To do this, make a subplot of dimension 5×3 and populate the appropriate subplots) (To further simplify things, we have also provided the helper function `PlotSpikeRaster.m`; this will likelys simplify your raster plotting.)

Solution: The following code gets the job done:

```
%% 2a

clear all; close all; clc;
r_0 = 35;
% (spikes/s)
r_max = 60; % (spikes/s)
s_max = pi/2; % (radians)
T = 1000; % trial length (ms)
bin_width = 20; % (ms)
bin_centers = bin_width/2:bin_width:T; % (ms)
s = (0:7)*pi/4; % (radians)
s_labels = ...
    {'0', '\pi/4', '\pi/2', '3\pi/4', '\pi', '5\pi/4', '3\pi/2'
      , '7\pi/4'};
lambda = r_0 + (r_max-r_0)*cos(s-s_max); % tuning curve
num_cons = length(s);
num_reps = 100; % per condition
num_rasters_to_plot = 5; % per condition
```

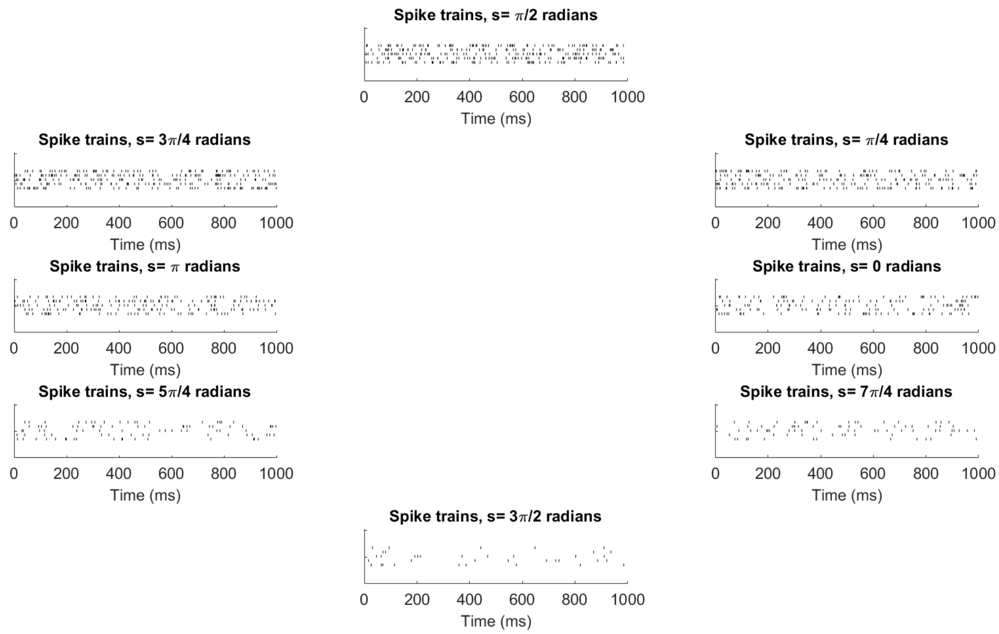
[1] *TN* refers to *Theoretical Neuroscience* by Dayan and Abbott.

```
% These variables help to arrange plots around a circle
num_plot_rows = 5;
num_plot_cols = 3;
subplot_indx = [9, 12, 14, 10, 7, 4, 2, 6];

spike_counts = zeros(num_cons, num_reps);
spike_times = cell(num_cons, num_reps);
% Generate and plot homogeneous Poisson process spike trains
figure;
for con=1:num_cons
    for rep=1:num_reps
        spike_times{con, rep} = ...
            GeneratePoissonSpikeTrain(T, lambda(con));
        spike_counts(con, rep) = length(spike_times{con, rep});
    end
    % Plot spike rasters
    subplot(num_plot_rows, num_plot_cols, subplot_indx(con));
    PlotSpikeRaster({spike_times{con, 1:num_rasters_to_plot}});
    title(['Spike trains, s= ', s_labels{con}, ' radians']);
end
```

The function `PlotSpikeRaster.m` was provided. For `GeneratePoissonSpikeTrain.m` we used the following code:

```

function spike_train = GeneratePoissonSpikeTrain( T, rate )
%GENERATEPOISSONSPIKETRAIN Summary of this function goes here
%   T in ms
%   r in spikes/s
%   returns spike_train, a collection of spike times

    spike_train = [];
    time = 0;

    while (time <= T)
        time_next_spike = exprnd(1/rate * 1000);
        time = time + time_next_spike;
        spike_train = [spike_train time];
    end

    %discard last spike if happens after T
    if (spike_train(length(spike_train)) > T)
        spike_train = spike_train(1:length(spike_train)-1);
    end

end
```

This is the output.



(b) (5 points) Spike histogram
For each reaching angle, find the spike histogram by taking spike counts in non-overlapping 20 ms bins, then averaging across the 100 trials. Plot the 8 resulting spike histograms around a circle, as in part (a). The spike histograms should have firing rate (in spikes / second) as the vertical axis and time (in msec, not time bin index) as the horizontal axis. The `bar` command in Matlab can be used to plot histograms.

Solution:

The following code gets the job done:

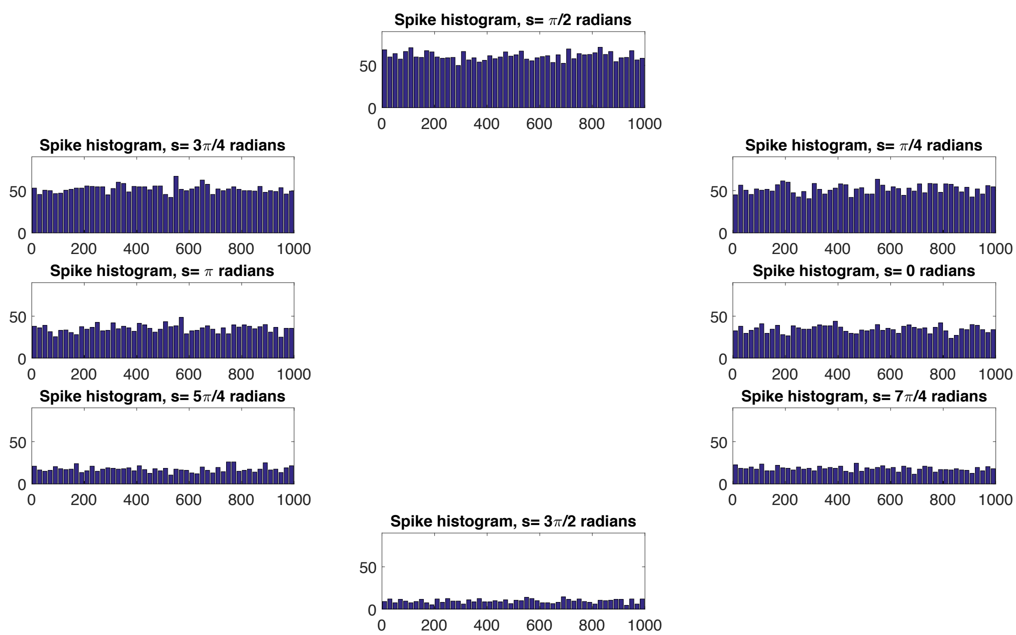
```
  %% 2b

figure;
for con = 1:num_cons
    subplot(num_plot_rows,num_plot_cols,subplot_indx(con));
    SpikeHistogram({spike_times{con,:}}, T, bin_width);
    title(['Spike histogram, s= ', s_labels{con}, ' radians']);
    axis([0, T, 0, 1.5*max(lambda)])
end
```

The function `SpikeHistogram.m` is below:

```
  function SpikeHistogram(S, T, bin_width)
% S is a cell of spike trains, T is the time horizon.

    total_trials = numel(S);
    incr = bin_width;
    start = 0;
```

```
    stop = start+bin_width;
    b = 1;
    spikeCount = [];

    while (stop <= T)

        spikeCount(b) = 0;
        for t = 1:total_trials
            spikeCount(b) = spikeCount(b) + length(intersect(
              find(S{t}>=start), find(S{t}<stop)));
        end
        spikeCount(b) = spikeCount(b)/(bin_width*(10^(-3))*
          total_trials);
        start = start+incr;
        stop = start+bin_width;
        b = b+1;

    end

    bin_centers = bin_width/2:bin_width:T;
    bar(bin_centers, spikeCount);
```

This is the output:



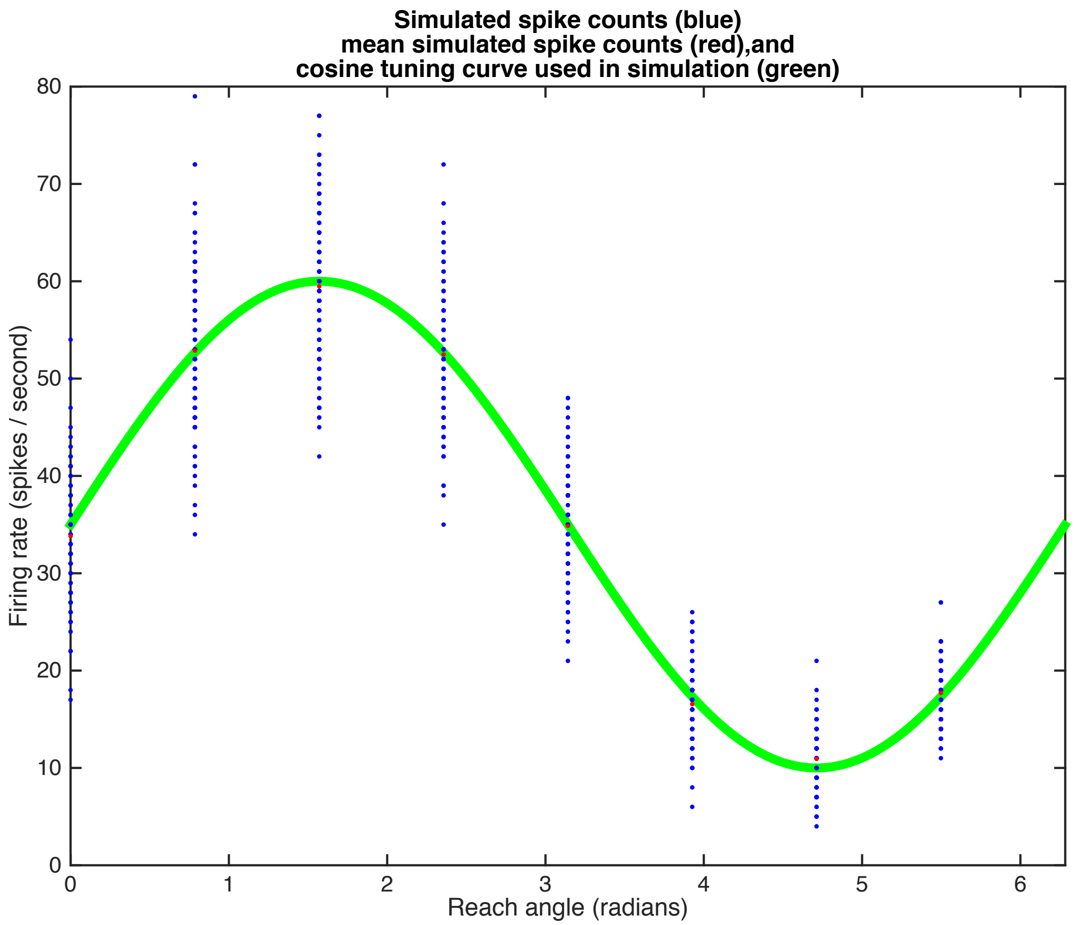
(c) (4 points) Tuning curve
For each trial, count the number of spikes across the entire trial. Plots these points on the axes like shown in Figure 1.6(B) in *TN*, where the x-axis is reach angle and the y-axis is firing rate. There should be 800 points in the plot (but some points may be on top of each other due to the discrete nature of spike counts). For each reaching angle,

find the mean firing rate across the 100 trials, and plot the mean firing rate using a red point on the same plot. Now, plot the tuning curve (defined in (1)) of this neuron in green on the same plot. Do the mean firing rates lie near the tuning curve?

Solution: The following code gets the job done:

```
mean_firing_rates = mean(spike_counts,2);
figure;
s_fine = 0:.01:2*pi;
plot(s_fine,r_0+(r_max-r_0)*cos(s_fine-s_max),'g-','linewidth',4);
hold on
plot(s,spike_counts,'b.')
plot(s,mean_firing_rates,'r.')
xlabel('Reach angle (radians)');
ylabel('Firing rate (spikes / second)');
title({'Simulated spike counts (blue)',...
    'mean simulated spike counts (red),and',...
    'cosine tuning curve used in simulation (green)'});
xlim([0, 2*pi]);
% ******************************************************
% Yes, the mean firing rates lie near the tuning curve.
% ******************************************************
```

This is the output:

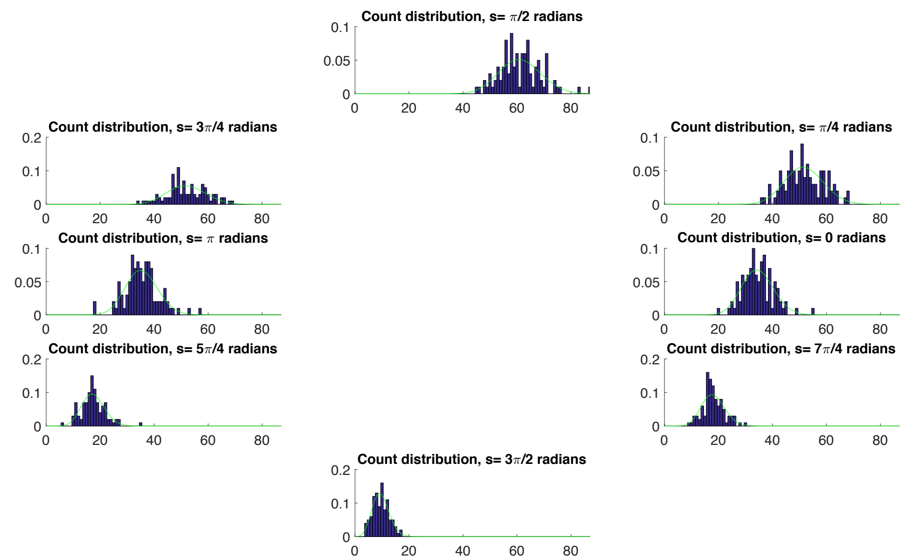


(d) (6 points) Count distribution
For each reaching angle, plot the *normalized* distribution (i.e., normalized so that the area under the distribution equals one) of spike counts (using the same counts from part (c)). Plot the 8 distributions around a circle, as in part (a). Fit a Poisson distribution to each empirical distribution and plot it on top of the corresponding empirical distribution. Are the empirical distributions well-fit by Poisson distributions?

Solution: See following code:

```

%% 2d

figure;
max_count = max(max(spike_counts));
spike_count_bin_centers = 0:max_count;
for con=1:num_cons
    % Maximum likelihood estimation for the Poisson distribution:
    lambda_hat_poisson = mean(spike_counts(con,:));
    Poisson_fit = exp(-lambda_hat_poisson)*(lambda_hat_poisson.^ ...
        spike_count_bin_centers)./factorial(spike_count_bin_centers);
```

```
    subplot(num_plot_rows,num_plot_cols,subplot_indx(con));
    xlim([0, max_count])
    hold on
    spike_count_histogram = hist(spike_counts(con,:),...
        spike_count_bin_centers);
    bar(spike_count_bin_centers,spike_count_histogram/num_reps,1);
    plot(spike_count_bin_centers,Poisson_fit,'g');
    title(['Count distribution, s= ', s_labels{con}, ' radians']);
end
% ******************************************************
% Yes, the empirical distributions are well-fit by Poisson distributions.
% ******************************************************
```



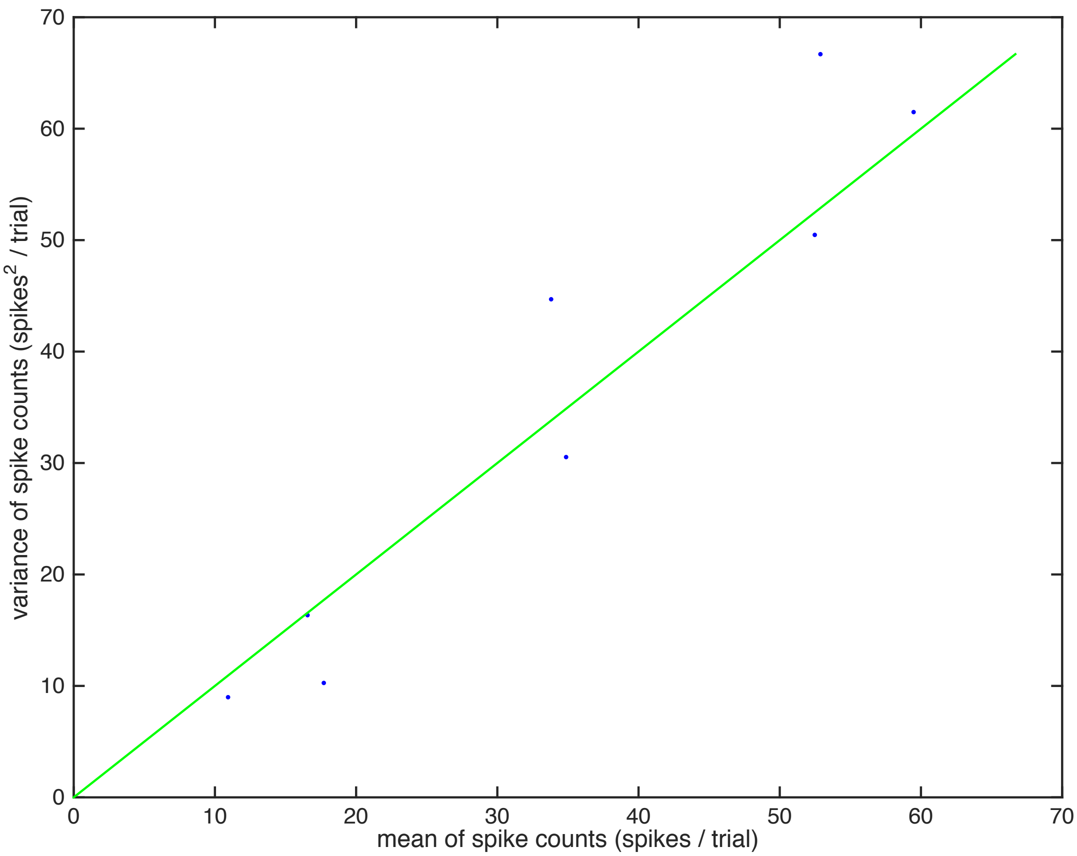
(e) (4 points) Fano factor

For each reaching angle, find the mean and variance of the spike counts across the 100 trials (using the same spike counts from part (c)). Plot the obtained mean and variance on the axes shown in Figure 1.14(A) in *TN*. There should be 8 points in this plot – one per reaching angle. Do these points lie near the 45 deg diagonal, as would be expected of a Poisson distribution?

Solution:

```
mean_spike_counts = mean(spike_counts,2);
var_spike_counts = var(spike_counts,1,2);
figure;
plot(mean_spike_counts,var_spike_counts,'b.')
hold on
max_plot_val = max([mean_spike_counts; var_spike_counts]);
plot([0, max_plot_val],[0, max_plot_val]','g-')
```

```
xlabel('mean of spike counts (spikes / trial)');
ylabel('variance of spike counts (spikes^2 / trial)');
% ********************************************
% Yes, the points lie near the 45 degree diagonal, as is expected of a
% Poisson distribution.
% ********************************************
```

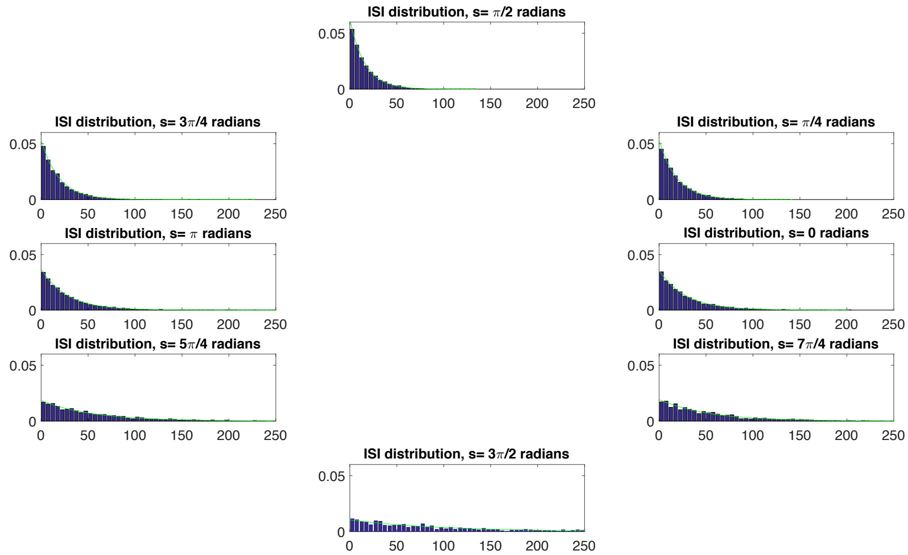


(f) (5 points) Interspike interval (ISI) distribution
For each reaching angle, plot the normalized distribution of ISIs. Plot the 8 distributions around a circle, as in part (a). Fit an exponential distribution to each empirical distribution and plot it on top of the corresponding empirical distribution. Are the empirical distributions well-fit by exponential distributions?

Solution:

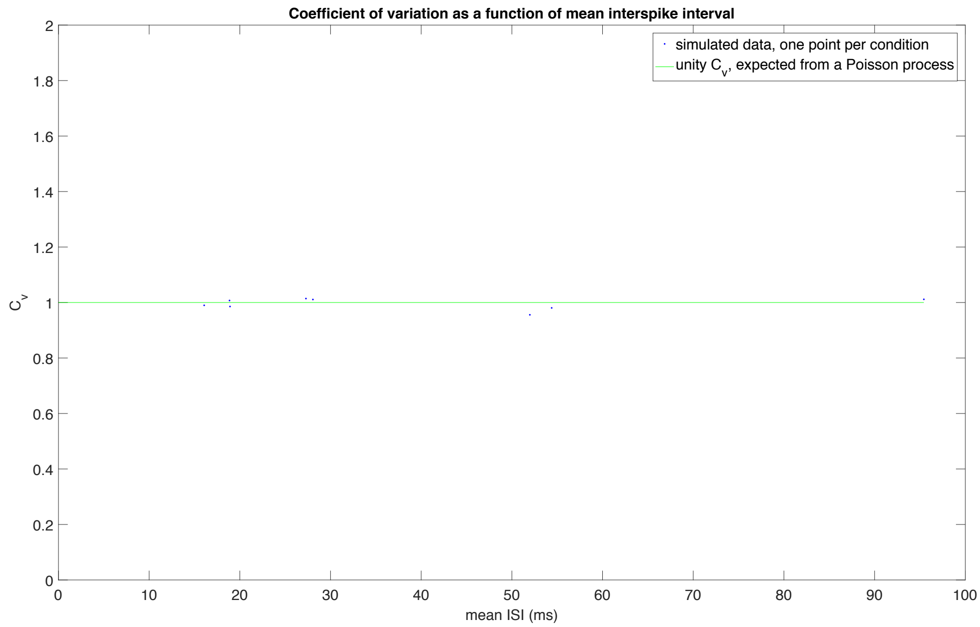
```
%% 2f

figure;
ISIs = cell(num_cons,1);
num_ISI_bins = 200;

for con=1:num_cons
    subplot(num_plot_rows,num_plot_cols,subplot_indx(con));
```

```matlab
    ISIs{con} = InterspikeIntervalHistogram({spike_times{con,:}},...
        1000 / num_ISI_bins);
    hold on
    % Maximum likelihood estimation of the rate parameter from the ISIs:
    lambda_hat_exp = 1/mean(ISIs{con});
    t = linspace(0,max(ISIs{con}),50);
    exp_fit = lambda_hat_exp*exp(-lambda_hat_exp*t);
    plot(t,exp_fit,'g');
    title(['ISI distribution, s= ', s_labels{con}, ' radians']);
    axis([0, T/4, 0, max(lambda)/1000])
end
% ****************************************
% Yes, the empirical distributions are well-fit by exponential
% distributions.
% ****************************************

function isi = InterspikeIntervalHistogram(S, bin_width)
% S is a cell of spike trains, num_bins is how many bins to have the ISI

    total_trials = numel(S);
    isi = [];

    trial_spikes    = cellfun(@(a) length(a), S);
    idx_most        = find(trial_spikes - max(trial_spikes) == 0);
    rows            = isrow(S{idx_most(1)});

    for i = 1:total_trials
        if rows
            isi = [isi diff(cell2mat(S(i)))];
        else
            isi = [isi; diff(cell2mat(S(i)))];
        end
    end

    bin = bin_width;
    [n,x] = hist(isi,(bin/2:bin:max(isi)));
    bar(x, n/sum(n)/bin_width);

end
```



(g) (5 points) Coefficient of variation (C_V)
For each reaching angle, find the average ISI and C_V of the ISIs. Plot the resulting values on the axes shown in Figure 1.16 in *TN*. There should be 8 points in this plot. Do the C_V values lie near unity, as would be expected of a Poisson process?

```
%% 2g

figure;
C_v = zeros(num_cons,1);
mean_ISI = zeros(num_cons,1);
std_ISI = zeros(num_cons,1);
for con=1:num_cons
    mean_ISI(con) = mean(ISIs{con});
    std_ISI(con) = std(ISIs{con},1);
    C_v(con) = std_ISI(con)/mean_ISI(con);
end
plot(mean_ISI,C_v,'b.');
hold on
plot([0, max(mean_ISI)],[1, 1],'g'); % unity
legend('simulated data, one point per condition', ...
    'unity C_v, expected from a Poisson process');
title('Coefficient of variation as a function of mean interspike interval');
xlabel('mean ISI (ms)');
ylabel('C_v');
ylim([0, 2]);
% ********************************************
% Yes, C_v values lie near unity, as is expected of a Poisson process.
% ********************************************
```



3. (22 points) Inhomogeneous Poisson process
In this problem, we will use the same simulated neuron as in Problem 2, but now the reaching angle s will be time-dependent with the following form:

$$s(t) = t^2 \cdot \pi, \tag{2}$$

where t ranges between 0 and 1 second.

(a) (6 points) Spike trains
Generate 100 spike trains, each 1 second in duration, according to an inhomogeneous Poisson process with a firing rate profile defined by (1) and (2). Plot 5 of the generated spike trains.

Solution:

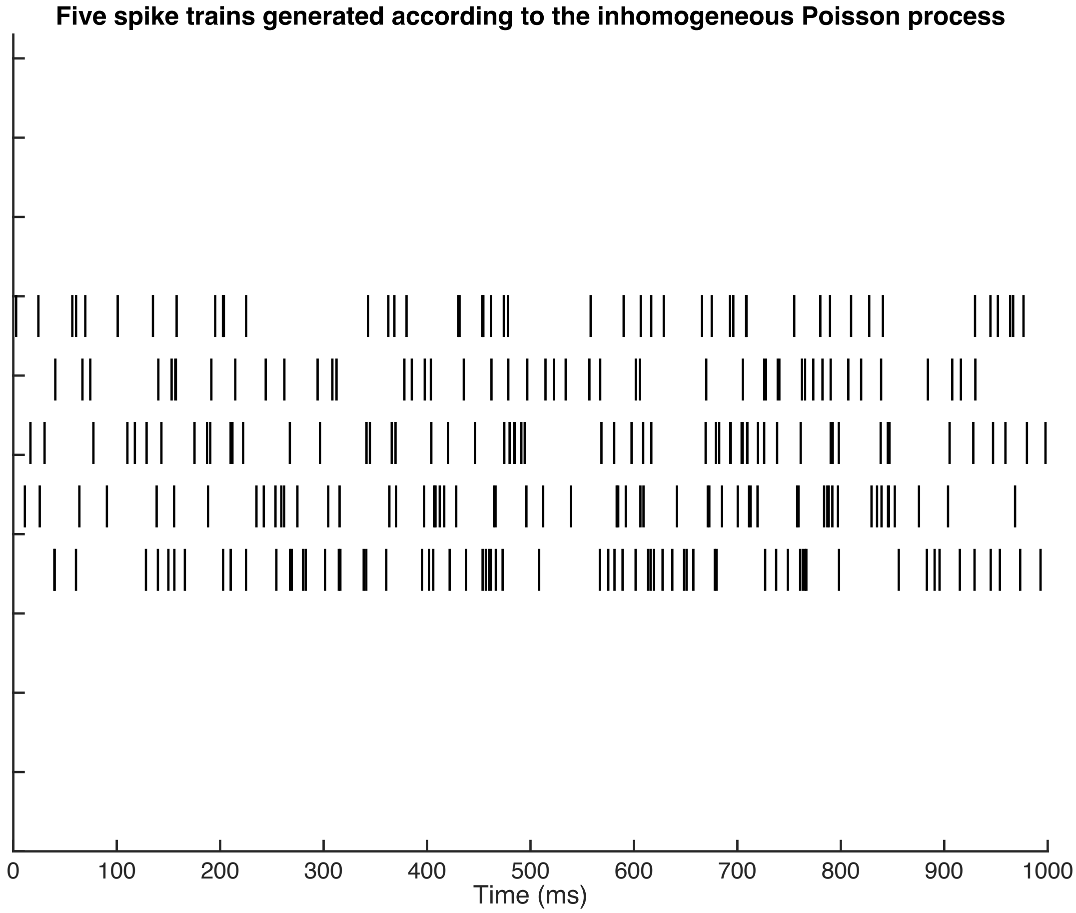
```

%% 3a

clear all; close all; clc;
r_0 = 35; % (spikes/s)
r_max = 60; % (spikes/s)
s_max = pi/2; % (radians)
T = 1000; % trial length (ms)
ms_per_s = 1000;
bin_width = 20; % (ms)
bin_centers = bin_width/2:bin_width:T; % (ms)

%

num_reps = 100;
```

```
num_rasters_to_plot = 5; % per condition
% Allocate space for the spike counts and spike times
spike_counts = zeros(1,num_reps);
spike_times = cell(1,num_reps);
% Generate and plot homogeneous Poisson process spike trains
for rep=1:num_reps
    spike_times{rep} = ...
        GeneratePoissonSpikeTrainInhomogeneous(T,r_max,@(t) r_0 + (r_max - r_0) * cos(t^2*pi - s_max));
    spike_counts(rep) = length(spike_times{rep});
end

% Plot spike rasters
figure;
PlotSpikeRaster({spike_times{1:num_rasters_to_plot}});
title('Five spike trains generated according to the inhomogeneous Poisson process');
```

with the following function:

```
function [ spike_train ] = GeneratePoissonSpikeTrainInhomogeneous( T, r_max, f )
%GENERATEPOISSONSPIKETRAININHOMOGENEOUS Summary of this function goes here
%   T in ms
%   r_max in spikes/s
%   f is the inhomogeneous function

    spike_train = [];
    time = 0;

    while (time <= T)
        time_next_spike = exprnd(1/r_max * 1000);
        time = time + time_next_spike;
        spike_train = [spike_train time];
    end

    %discard last spike if happens after T
    if (spike_train(length(spike_train)) > T)
        spike_train = spike_train(1:length(spike_train)-1);
    end

    % now throw away spikes.
    U           = rand(1, numel(spike_train));
    threshold   = arrayfun(@(a) f(a), spike_train / 1000);

    keep        = U < threshold / r_max;
    spike_train = spike_train(keep);

end
```

This produces the following output:



(b) (5 points) Spike histogram

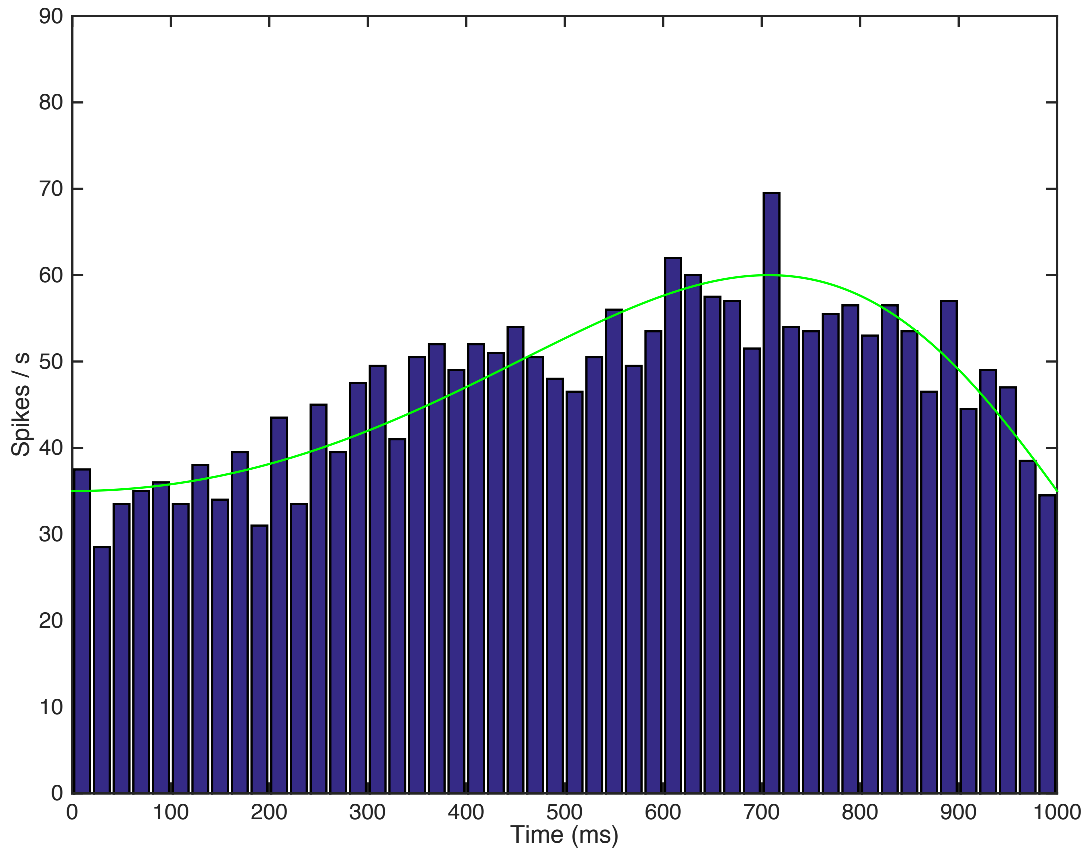
Plot the spike histogram by taking spike counts in non-overlapping 20 ms bins, then averaging across the 100 trials. The spike histogram should have firing rate (in spikes / second) as the vertical axis and time (in msec, not time bin index) as the horizontal axis. Plot the expected firing rate profile defined by equations (1) and (2) on the same plot. Does the spike histogram agree with the expected firing rate profile?

Solution:

```
%% 3b

figure;
SpikeHistogram({spike_times{1,:}}, T, bin_width);
hold on
% Here is the expected firing rate profile
t = linspace(0,T,1000);
s = pi*(t./ms_per_s).^2;
lambda = r_0 + (r_max-r_0)*cos(s-s_max);
plot(t,lambda,'g');
ylabel('Spikes / s')
xlabel('Time (ms)')
axis([0, T, 0, 1.5*r_max])
```

```
% ****************************************
% Yes, the spike histogram agrees with the expected firing rate
  profile.
% ****************************************
```

This produces the following output:



(c) (6 points) Count distribution

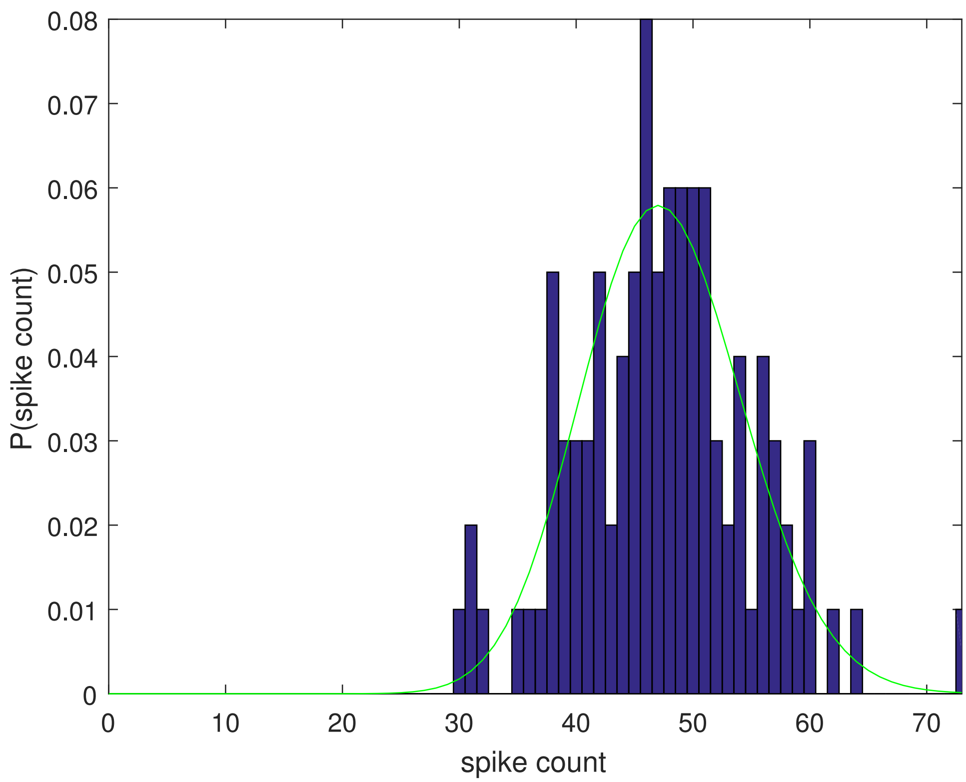
For each trial, count the number of spikes across the entire trial. Plot the normalized distribution of spike counts. Fit a Poisson distribution to this empirical distribution and plot it on top of the empirical distribution. Should we expect the spike counts to be Poisson-distributed?

Solution:

```
%% 3c

figure;
max_count = max(max(spike_counts));
spike_count_bin_centers = 0:max_count;
% Maximum likelihood estimation for the Poisson distribution:
lambda_hat_poisson = mean(spike_counts);
Poisson_fit = exp(-lambda_hat_poisson)*(lambda_hat_poisson.^
  ...
    spike_count_bin_centers)./factorial(spike_count_bin_centers
```

```
      );
spike_count_histogram = hist(spike_counts,
  spike_count_bin_centers);
bar(spike_count_bin_centers,spike_count_histogram/num_reps,1);
hold on
plot(spike_count_bin_centers,Poisson_fit,'g');
xlabel('spike count')
ylabel('P(spike count)')
xlim([0, max_count])
% **************************************
% As we showed in class, the total number of counts in an
  inhomogeneous
% Poisson process is distributed according to a Poisson
  distribution. This
% plot agrees with the theoretical result.
% **************************************
```

This produces the following output:



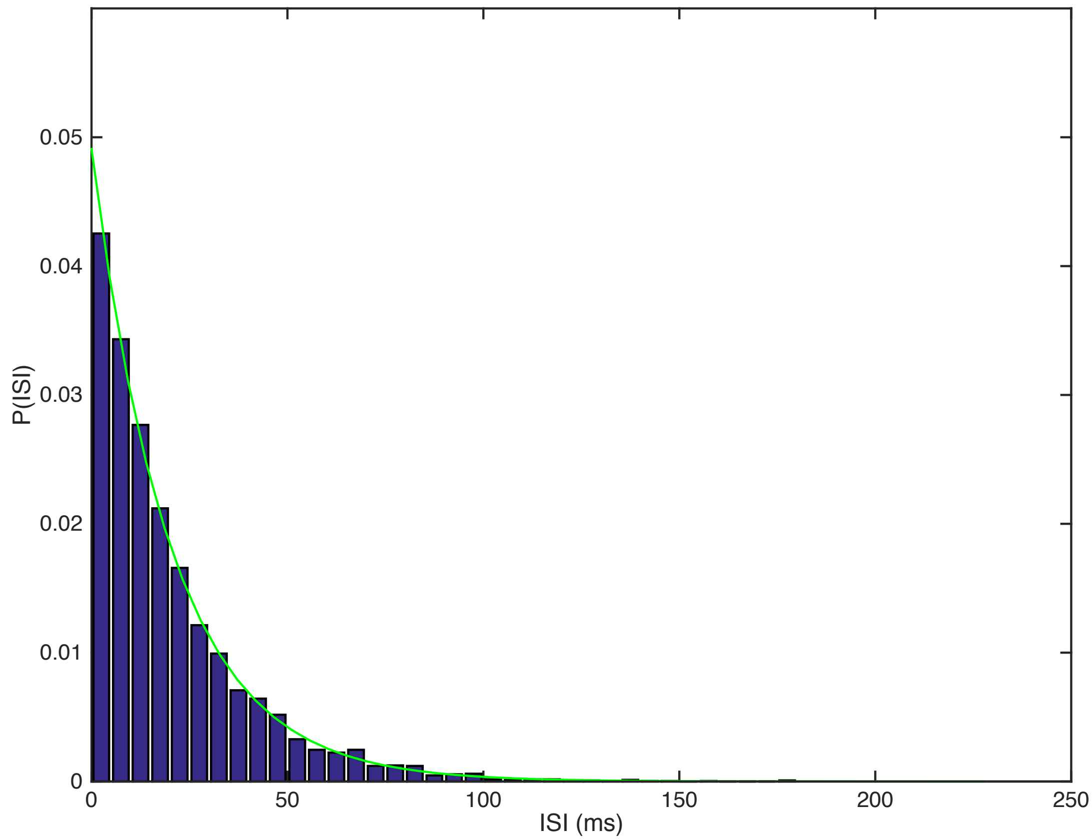
(d) (5 points) ISI distribution
Plot the normalized distribution of ISIs. Fit an exponential distribution to the empirical distribution and plot it on top of the empirical distribution. Should we expect the ISIs to be exponentially-distributed?

Solution:

```
%% 3d

```

```
figure;
num_ISI_bins = 200;
ISIs = InterspikeIntervalHistogram(spike_times, 1000 / num_ISI_bins);
hold on
% Maximum likelihood estimation of the rate parameter from the ISIs:
lambda_hat_exp = 1/mean(ISIs);
t = linspace(0,1.25*max(ISIs),50);
exp_fit = lambda_hat_exp*exp(-lambda_hat_exp*t);
plot(t,exp_fit,'g');
xlabel('ISI (ms)');
ylabel('P(ISI)')
axis([0, T/4, 0, max(lambda)/1000])
% *****************************
% As we showed in class, the ISIs from an inhomogeneous Poisson process are
% NOT exponentially distributed. This may be difficult to see in the plot
% for this problem--increasing num_repetitions to 500 may help, and you
% need a good number of bins in your histogram (we used 200 bins). You
% should see that an exponential distribution underfits for small values of
% t.
% *****************************
```

This produces the following output:



4. (30 points) Real neural data
We will analyze real neural data recorded using a 100-electrode array in premotor cortex of a macaque monkey[2]. The dataset can be found on CCLE as `'ps3_data.mat'`.

The following describes the data format. The .mat file has a single variable named `trial`, which is a structure of dimensions (182 trials) $\times$ (8 reaching angles). The structure contains spike trains recorded from a single neuron while the monkey reached 182 times along each of 8 different reaching angles (where the trials of different reaching angles were interleaved). The spike train for the `n`th trial of the `k`th reaching angle is contained in `trial(n,k).spikes`, where `n` $= 1, \ldots, 182$ and `k` $= 1, \ldots, 8$. The indices `k` $= 1, \ldots, 8$ correspond to reaching angles $\frac{30}{180}\pi$, $\frac{70}{180}\pi$, $\frac{110}{180}\pi$, $\frac{150}{180}\pi$, $\frac{190}{180}\pi$, $\frac{230}{180}\pi$, $\frac{310}{180}\pi$, $\frac{350}{180}\pi$, respectively. The reaching angles are not evenly spaced around the circle due to experimental constraints that are beyond the scope of this homework.

A spike train is represented as a sequence of zeros and ones, where time is discretized in 1 ms steps. A zero indicates that the neuron did not spike in the 1 ms bin, whereas a one indicates that the neuron spiked once in the 1 ms bin. Due to the refractory period, it is not possible for a neuron to spike more than once within a 1 ms bin. Each spike train is 500 ms long and is, thus, represented by a 1×500 vector.

(a) (6 points) Spike trains
Plot 5 spike trains for each reaching angle in the same format as shown in Figure 1.6(A)

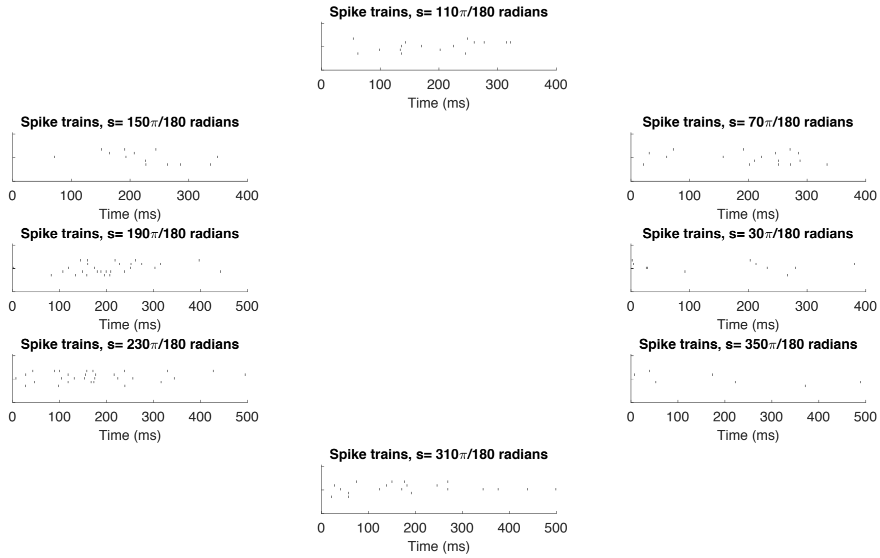
[2]The neural data have been generously provided by the laboratory of Prof. Krishna Shenoy at Stanford University. The data are to be used exclusively for educational purposes in this course.

in *TN*.

Solution:

```

%% 4a

clear all; close all; clc;
load ps3_data
[num_reps, num_cons] = size(trial);
T = 500; % trial length (ms)
bin_width = 20; % (ms)
bin_centers = bin_width/2:bin_width:T; % (ms)
num_rasters_to_plot = 5; % per condition
s = pi*[30/180 70/180 110/180 150/180 190/180 230/180 310/180 350/180]'; % (radians)
s_labels = {'30\pi/180', '70\pi/180', '110\pi/180', '150\pi/180', ...
    '190\pi/180', '230\pi/180', '310\pi/180', '350\pi/180'};
% These variables help to arrange plots around a circle
num_plot_rows = 5;
num_plot_cols = 3;
subplot_indx = [9 12 14 10 7 4 2 6 ];

%%
spike_counts = zeros(num_cons,num_reps);
spike_times = cell(num_cons,num_reps);
figure;
for con=1:num_cons
    for rep=1:num_reps
        spike_times{con,rep} = (find(trial(rep,con).spikes==1)'-1);
        spike_counts(con,rep) = length(spike_times{con,rep});

    end
    % Plot spike rasters
    subplot(num_plot_rows,num_plot_cols,subplot_indx(con));
    PlotSpikeRaster({spike_times{con,1:num_rasters_to_plot}});
    title(['Spike trains, s= ', s_labels{con}, ' radians']);
end
```

This produces the following output:

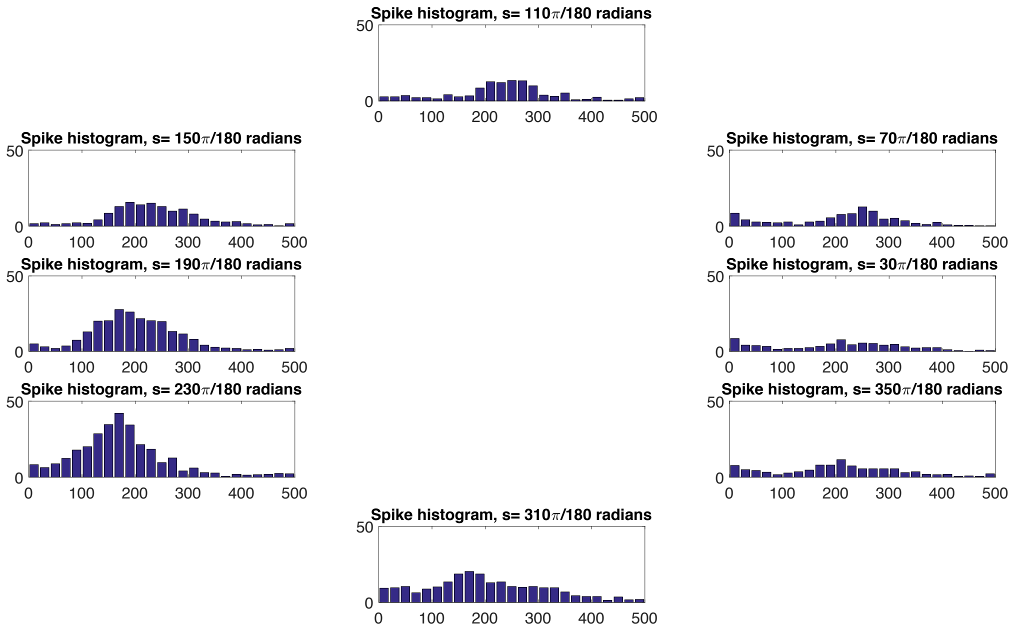


(b) (5 points) Spike histogram
For each reaching angle, find the spike histogram by taking spike counts in non-overlapping 20 ms bins, then averaging across the 182 trials. The spike histograms should have firing rate (in spikes / second) as the vertical axis and time (in msec, not time bin index) as the horizontal axis. Plot the histogram for 500ms worth of data. Plot the 8 resulting spike histograms around a circle, as in part (a).

Solution:

```
 %% 4b

figure;
T = max(vertcat(spike_times{:}));
T = ceil(T / 100) * 100;

for con=1:num_cons
    subplot(num_plot_rows,num_plot_cols,subplot_indx(con));
    SpikeHistogram({spike_times{con,:}}, T, bin_width);
    title(['Spike histogram, s= ', s_labels{con}, ' radians']);
    axis([0, T, 0, 50])
end
```

This produces the following output:



(c) (4 points) Tuning curve

For each trial, count the number of spikes across the entire trial. Plots these points on the axes shown in Figure 1.6(B) in *TN*. There should be $182 \cdot 8$ points in the plot (but some points may be on top of each other due to the discrete nature of spike counts). For each reaching angle, find the mean firing rate across the 182 trials, and plot the mean firing rate using a red point on the same plot.

Then, fit the cosine tuning curve (1) to the 8 red points by minimizing the sum of squared errors

$$\sum_{i=1}^{8} \big(\lambda(s_i) - r_0 - (r_{\max} - r_0)\cos(s_i - s_{\max})\big)^2$$

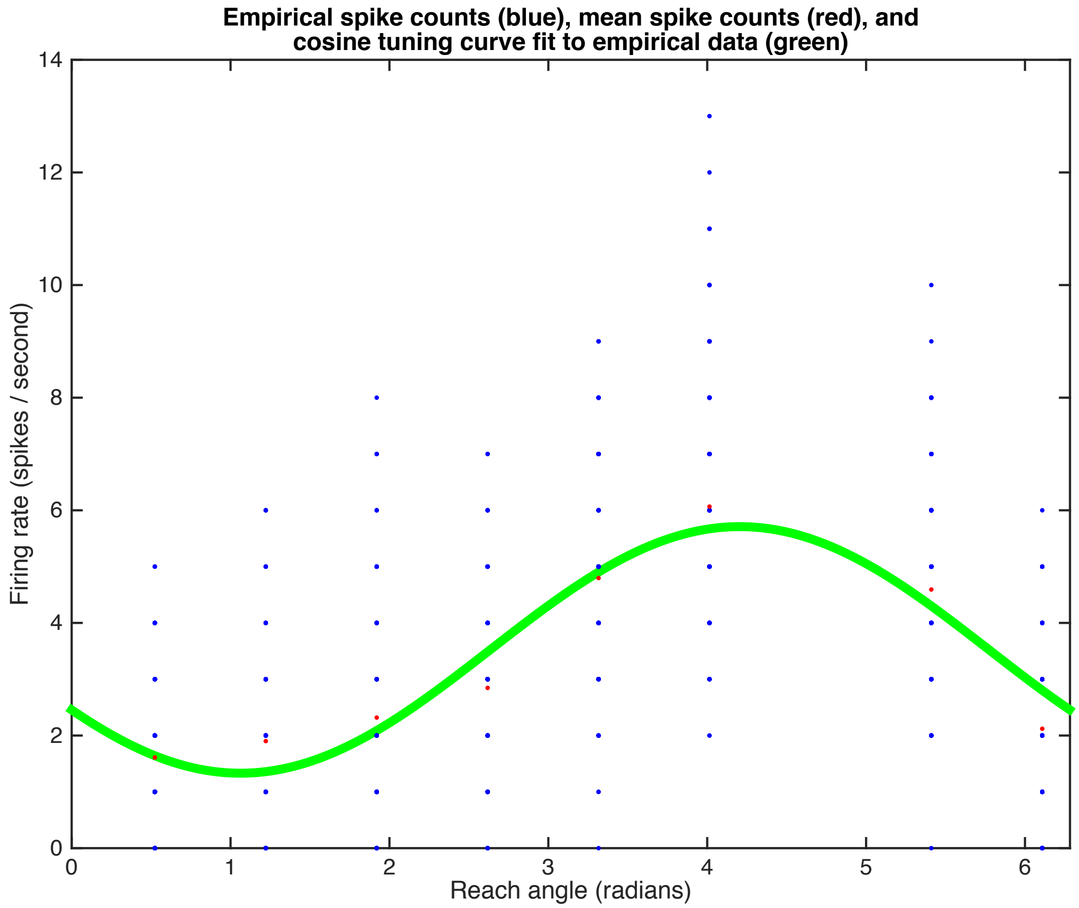
with respect to the parameters r_0, $r_{\max}$, and $s_{\max}$. (Hint: this can be done using linear regression; refer to Homework # 2.) Plot the resulting tuning curve of this neuron in green on the same plot.

Solution:

```

%% 4c

mean_firing_rates = mean(spike_counts,2);
% Set up a linear regression
lambda = mean_firing_rates;
A = [ones(num_cons,1), cos(s), sin(s)];
x = A\lambda; % closed form solution that minimizes norm(A*x-
  lambda,2)
% To obtain the tuning curve parameters from the x variables,
  note we have:
```

```
% r_0-(r_max-r_0)*cos(s_i-s_max) = x_1+x_2*cos(s_i)+x_3*sin(s_i).
% Any cosine with a phase offset can be uniquely written as the weighted linear
% combination of a cosine and sine with no offset. So we first get that
% r_0 = x_1. Expanding the LHS using cos(x-y) = cos(x)cos(y)+sin(x)sin(y) and
% finding like terms, we get x_2 = (r_0-r_max)*cos(s_max) and
% x_3 = (r_0-r_max)*sin(s_max). Dividing these gets us the next line of code.
s_max = atan(x(3)/x(2));
r_0 = x(1); % the offset on both sides of the equation must be same
r_max = r_0+x(2)/cos(s_max);
figure;
s_fine = 0:.01:2*pi;
plot(s_fine,r_0+(r_max-r_0)*cos(s_fine-s_max),'g-','linewidth',4);
hold on
plot(s,spike_counts,'b.')
plot(s,mean_firing_rates,'r.')
xlabel('Reach angle (radians)');
ylabel('Firing rate (spikes / second)');
title({'Empirical spike counts (blue), mean spike counts (red), and', ...
    'cosine tuning curve fit to empirical data (green)'});
xlim([0, 2*pi]);
```

This produces the following output:



(d) (6 points) Count distribution

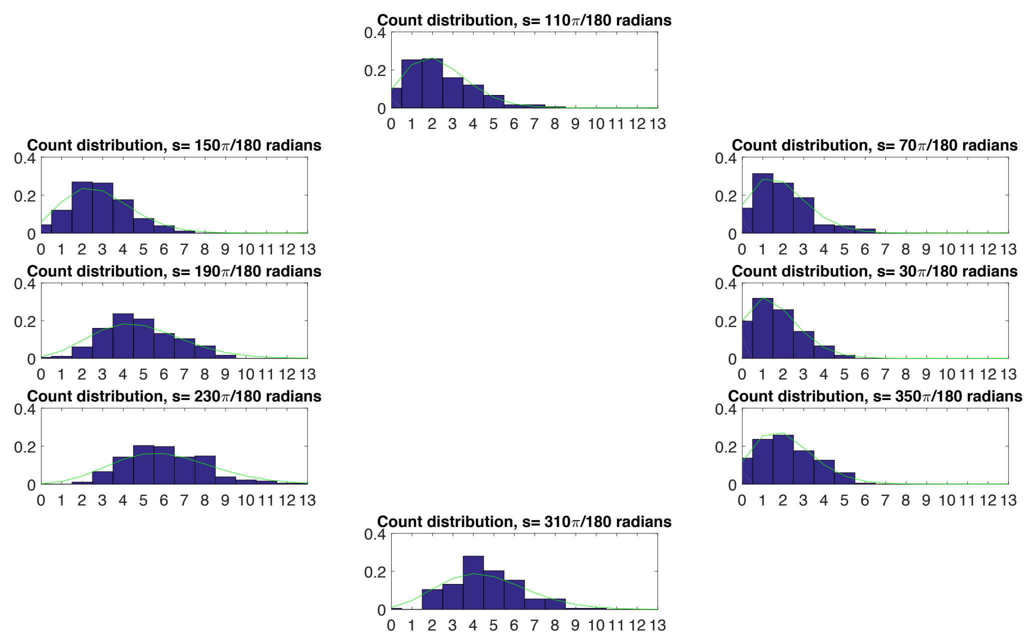
For each reaching angle, plot the normalized distribution of spike counts (using the same counts from part (c)). Plot the 8 distributions around a circle, as in part (a). Fit a Poisson distribution to each empirical distribution and plot it on top of the corresponding empirical distribution. Why might the empirical distributions differ from the idealized Poisson distributions?

Solution:

```

%% 4d

figure;
max_count = max(max(spike_counts));
spike_count_bin_centers = 0:max_count;
for con=1:num_cons
    % Maximum likelihood estimation for the Poisson
      distribution:
    lambda_hat_poisson = mean(spike_counts(con,:));
    Poisson_fit = exp(-lambda_hat_poisson)*(lambda_hat_poisson
      .^ ...
        spike_count_bin_centers)./factorial(
          spike_count_bin_centers);
```

```matlab
    subplot(num_plot_rows,num_plot_cols,subplot_indx(con));
    spike_count_histogram = hist(spike_counts(con,:), spike_count_bin_centers);
    bar(spike_count_bin_centers,spike_count_histogram/num_reps,1);
    hold on
    plot(spike_count_bin_centers,Poisson_fit,'g');
    title(['Count distribution, s= ', s_labels{con}, ' radians']);
    xlim([0, max_count])
end
% ****************************************
% The empirical distributions deviate from Poisson distributions due to
% neurons refractory periods. This may not be easily detected from the
% plots by eye.
% ****************************************
```

This produces the following output:



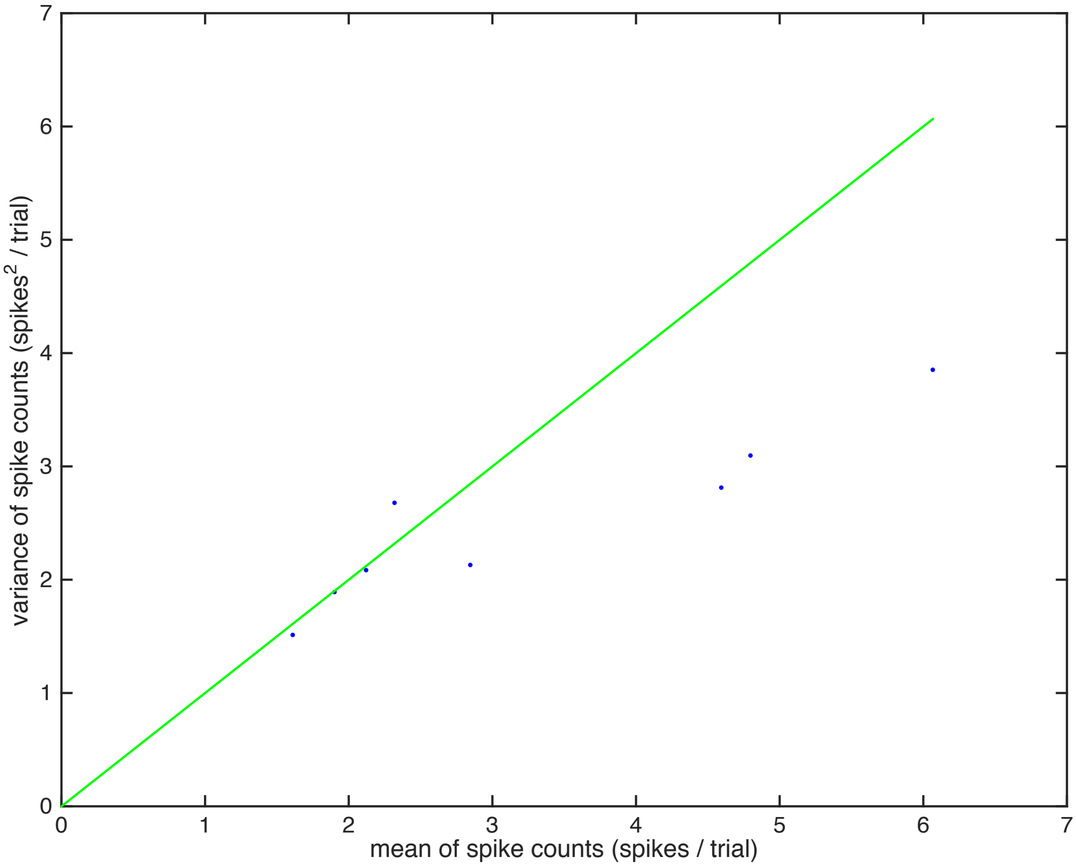
(e) (4 points) Fano factor

For each reaching angle, find the mean and variance of the spike counts across the 182 trials (using the same spike counts from part (c)). Plot the obtained mean and variance on the axes shown in Figure 1.14(A) in *TN*. There should be 8 points in this plot – one per reaching angle. Do these points lie near the 45 deg diagonal, as would be expected of a Poisson distribution?

Solution:

```

%% 4e

mean_spike_counts = mean(spike_counts,2);
var_spike_counts = var(spike_counts,1,2);
figure;
plot(mean_spike_counts,var_spike_counts,'b.')
hold on
max_plot_val = max([mean_spike_counts; var_spike_counts]);
plot([0 max_plot_val],[0 max_plot_val]','g-')
xlabel('mean of spike counts (spikes / trial)');
ylabel('variance of spike counts (spikes^2 / trial)');
% **********************************
% For low mean number of spikes, the points lie near the 45 degree
% diagonal, as is expected of a Poisson distribution. But as the mean
% number of spikes increases, the points fall below the 45 degree diagonal.
% This is because for high firing rates, the refractory period acts as a
% strong deterrant, causing the next spike to always occur within a narrow
% time window right after the refractory period ends. This causes lower
% variance in the spike counts for the entire trial.
% **********************************
```

This produces the following output:



(f) (5 points) Interspike interval (ISI) distribution
For each reaching angle, plot the normalized distribution of ISIs. Plot the 8 distributions around a circle, as in part (a). Fit an exponential distribution to each empirical distribution and plot it on top of the corresponding empirical distribution. Why might the empirical distributions differ from the idealized exponential distributions?

Solution:

```

%% 4f

figure;
num_ISI_bins = 200;
for con=1:num_cons
    subplot(num_plot_rows,num_plot_cols,subplot_indx(con));
    ISIs = InterspikeIntervalHistogram({spike_times{con,:}}, 5);
    hold on
    % Maximum likelihood estimation of the rate parameter from the
      ISIs:
    lambda_hat_exp = 1/mean(ISIs);
    t = linspace(0,max(ISIs),50);
    exp_fit = lambda_hat_exp*exp(-lambda_hat_exp*t);
    plot(t,exp_fit,'g');
```

```
    title(['ISI distribution, s= ', s_labels{con}, ' radians']);
    axis([0, T, 0, 0.04])
end
% *************************************************************************
% The empirical distributions deviate from exponential distributions due to
% neurons refractory periods and nonstationary firing rates.
% *************************************************************************
```

This produces the following output: