Table of Contents


```
% EE239.2 HW 3
```

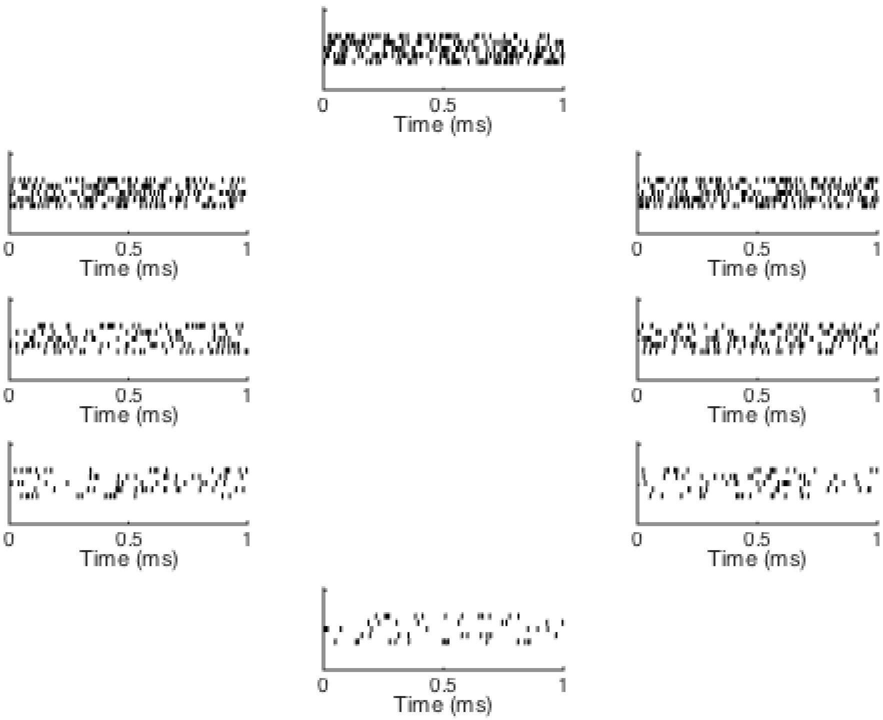
Problem 2: Homogeneous Poisson Process

```
% Collaborators: Vikranth Jeyakumar and Yusi Ou

clc
clear
close all
```

Part A: Spike Trains

```
s = [0 45 90 135 180 225 270 315];      % angles to calculate lambda

r_0 = 35;
r_max = 60;
s_max = 90;

lambda = r_0 + (r_max - r_0) * cosd(s - s_max);     % rates

n = 0;
T = 0;

time = 1;           % spike trains last 1 second
mu = 1./lambda;
numTrials = 100;    % number of trials for each lambda value

T_cell = {};        % matrix of 100 spike trains trials per lambda
T_vec = [];

for i = 1:length(mu)
    for k = 1:numTrials
        n = 0;      % number of spikes (reset for each trial)
        T = 0;      % spike time (reset for each trial)
        T_vec = [];         % spike vector (reset for each trial)

      while ( T < time )        % generate spikes while time is less than 1 sec
```

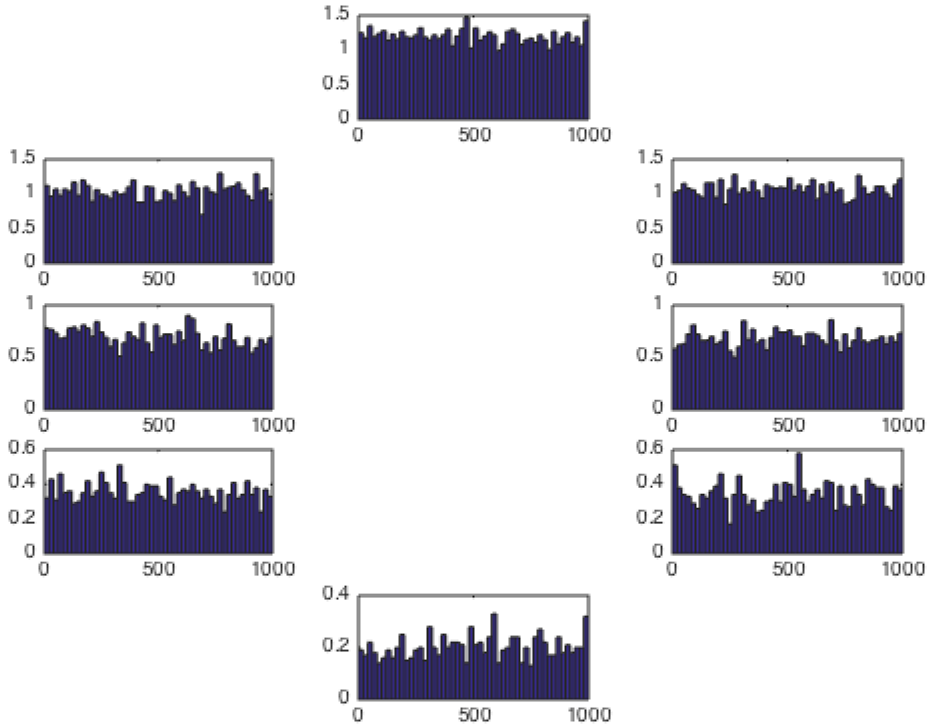
```
            dt = exprnd(mu(i));      % sample from exponential distribution

            T = T + dt;              % add spike time
            n = n + 1;               % iterate number of spikes
            T_vec(n) = T;            % add spike to vector

          end
            T_vec = T_vec(:,1:end-1);
            % delete last value because it will go over the set time

            T_cell{i,k} = T_vec;

        end
    end

    figure(1)
    subplotRaster(T_cell)
```



Part B: Spike Histogram

```
    bins = 0:0.020:1;                % 20 ms bins for 1 second

    counts_sum = zeros(8,length(bins));
    for j=1:length(mu)
        for i=1:numTrials
            counts = histc(T_cell{j,i}, bins);
```

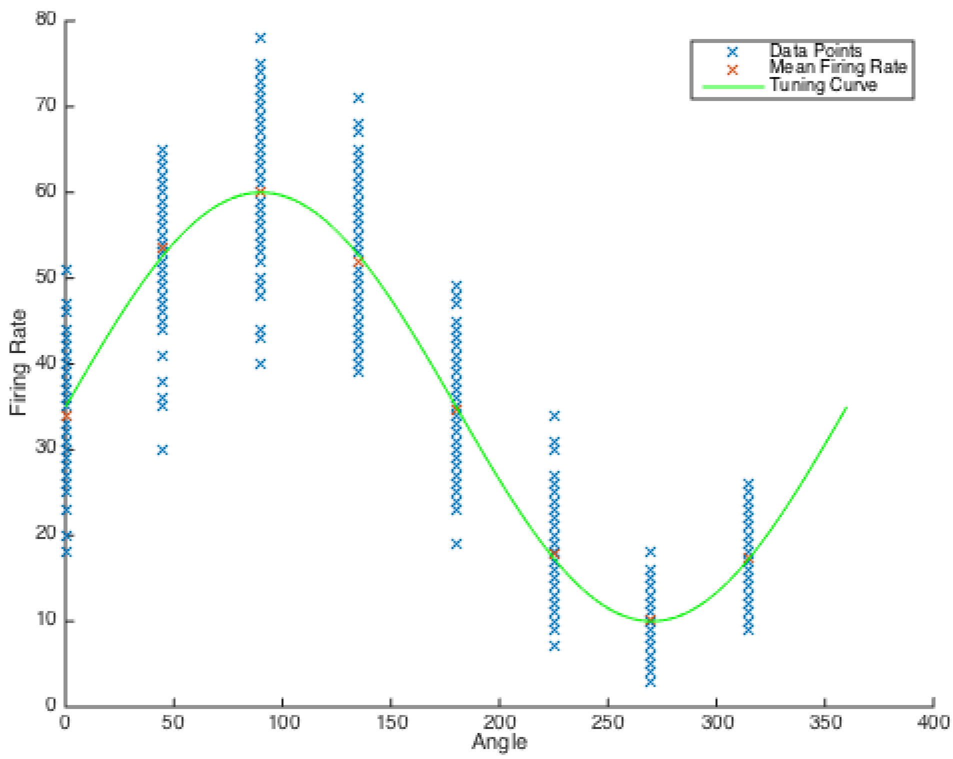
```
        counts_sum(j,:) = counts_sum(j,:) + counts;
        % number of spikes per 20 ms bin
    end
end

figure(2)
counts_sum = counts_sum(:,1:50);
counts_avg = counts_sum/100;    % average over 100 trials
subplotCounts(counts_avg,bins)
```



Part C: Tuning Curve

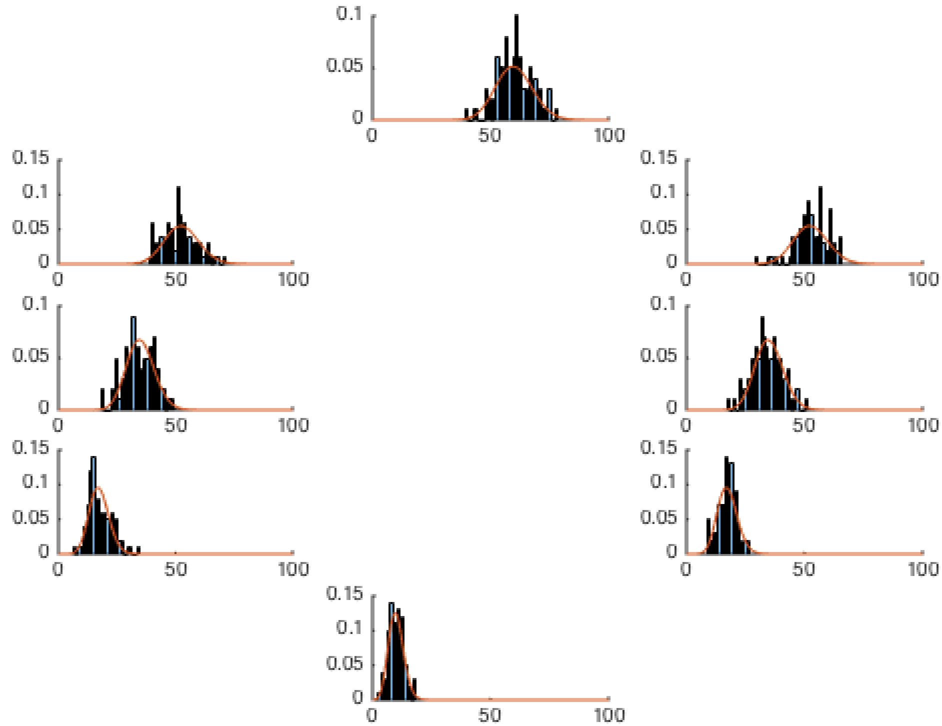
```
f_rate = zeros(1,800);
rate = zeros(8,100);
for i=1:length(mu)
    for j=1:numTrials
       rate(i,j) = length(T_cell{i,j});     % number of spikes per trial
    end
end

f_rate = reshape(rate',[1,800]);            % all 800 data points

s_rep = repmat(s,100,1);
s_rep = reshape(s_rep, 1, numel(s_rep));

figure(3)
```

```
title('Part C: Tuning Curve')
scatter(s_rep,f_rate,'x');
f_rate_mean = sum(rate,2)/numTrials;        % calculate mean firing rate
hold on;
scatter(s,f_rate_mean,'x')
s_2 = 0:360;
lambda_2 = r_0 + (r_max - r_0) * cosd(s_2 - s_max);      % plot actual tuning curve
plot(s_2,lambda_2,'g')
xlabel('Angle')
ylabel('Firing Rate')
legend('Data Points','Mean Firing Rate','Tuning Curve')
hold off

% The mean firing rates do follow a tuning curve. This makes sense, because
% we generated the data as a homogeneous Poisson process, with the lambda
% values that lie on the cosine curve in Equation 1.
```



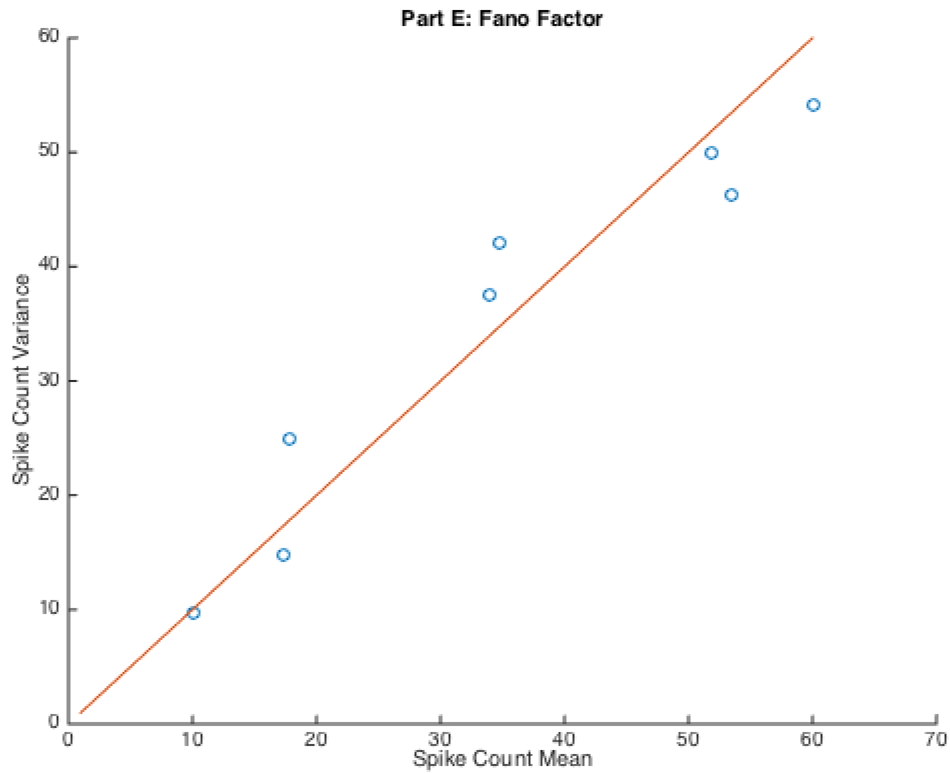
Part D: Count Distribution

```
figure(4)
subplotHist(rate, lambda)       % plot the rate with respect to firing rate

% The Poisson distribution does fit the empirical observations well, which
% is expected as we generated the ISIs according to an exponential
% distribution.
```



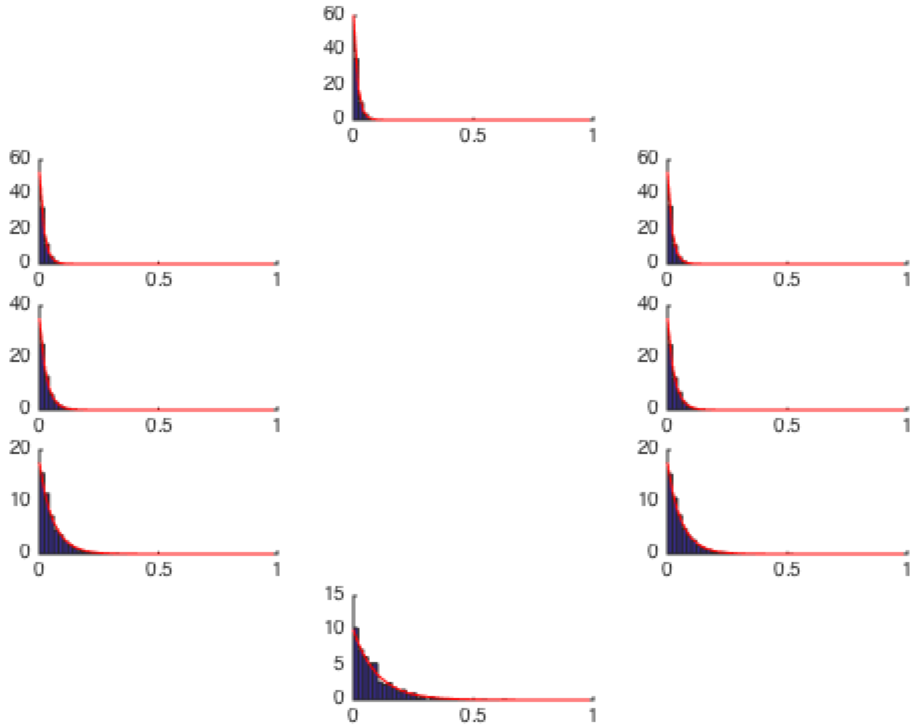
Part E: Fano Factor

```
counts_mean = mean(rate,2);
counts_var = var(rate,1,2);
figure(5)
hold on
scatter(counts_mean, counts_var)
plot(1:max(counts_mean),1:max(counts_mean))
title('Part E: Fano Factor')
xlabel('Spike Count Mean')
ylabel('Spike Count Variance')
hold off

% These points lie near the 45 degree diagonal, as would be expected of a
% Poisson distribution, since the mean and variance are the same.
```



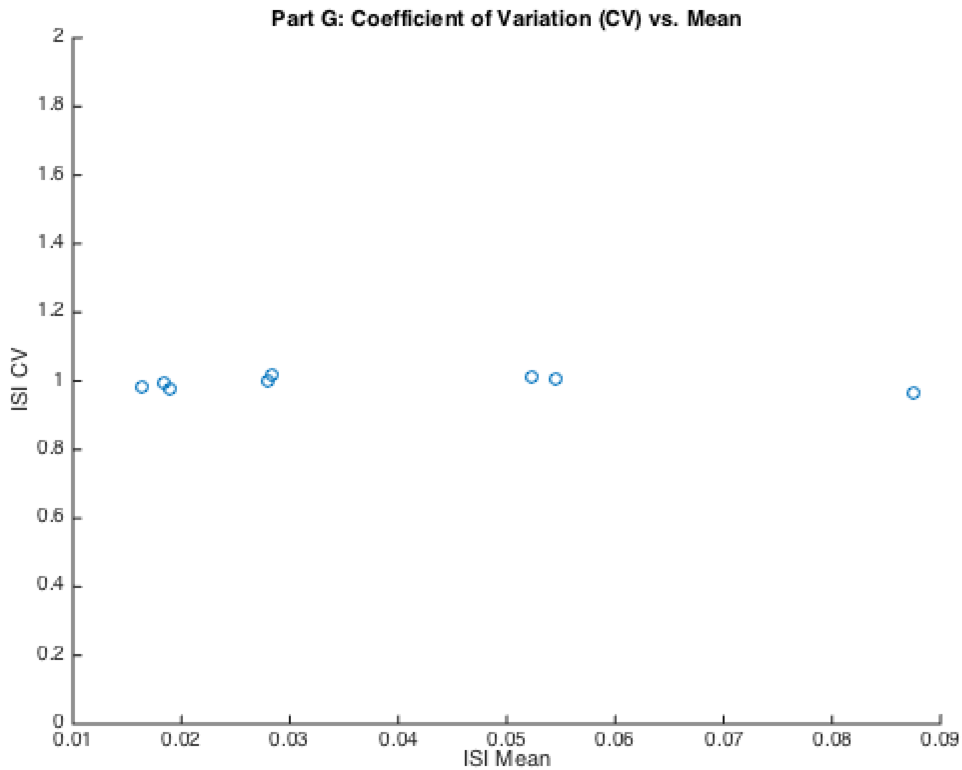
Part F: ISI Distribution

```
ISI = cell(8,1);
ISI_dist = cell(8,1);

for i = 1:length(mu)
    for k = 1:numTrials
        ISI{i} = [ISI{i}, diff(T_cell{i,k})];
    end
    ISI_hist = histcounts(ISI{i},bins,'Normalization','pdf');
    ISI_dist{i} = ISI_hist;
end

figure(6)
subplotISI(ISI_dist,bins(1:end-1),mu)

% The exponential distributons do fit the empirical ISI distributions well,
% as expected since we sampled from an exponential distribution to
% calculate the spike times.
```



Part G: Coefficient of Variation

```
for i = 1:length(mu)
    ISI_mean(i) = mean(ISI{i});
    ISI_CV(i) = std(ISI{i})/mean(ISI{i});
end

figure(7)
scatter(ISI_mean,ISI_CV)
ylim([0,2])
title('Part G: Coefficient of Variation (CV) vs. Mean')
ylabel('ISI CV')
xlabel('ISI Mean')

% The CV values lie near unity, as would be expected of a Poisson process
% since the mean and variance are equal. CV is calculated as standard
% deviation over mean, and so plotted against mean the values lie near one.
```



Published with MATLAB® R2014b