Table of Contents

EE239AS HW #6

```
% Collaborators: Vikranth, Yusi
clc
clear all
close all
```

Problem 2

```
load('/Users/Yusi/Documents/EE239AS/HW6/JR_2015-12-04_truncated2.mat');

% from part I of problem 1 (calculating spike counts for each direction)

n_trials = length(R);
n_electrodes = 96;

targets = zeros(2,n_trials);

for i = 1:n_trials
    targets(:,i) = R(i).target(1:2);
end

u_targets = unique(targets', 'rows');
n_targets = length(u_targets);

reach_raster = {};
reach_target = u_targets';

iter = zeros(1,n_targets);
reach_raster = cell(1,n_targets);

for i = 1:n_trials
    if (isequal(R(i).target(1:2),reach_target(:,1)))
        % remove if target is in the center
        iter(1) = iter(1)+1;
        % iterate if correct target
        full_reach_raster = full(R(i).spikeRaster);
        reach_raster{1}(:,:,iter(1)) = full_reach_raster(:,1:500);
        % convert spike train to full matrix and save first 500 ms
    end
```

```
if (isequal(R(i).target(1:2),reach_target(:,2)))
    % remove if target is in the center
    iter(2) = iter(2)+1;
    % iterate if correct target
    full_reach_raster = full(R(i).spikeRaster);
    reach_raster{2}(:,:,iter(2)) = full_reach_raster(:,1:500);
    % convert spike train to full matrix and save first 500 ms
end
if (isequal(R(i).target(1:2),reach_target(:,3)))
    % remove if target is in the center
    iter(3) = iter(3)+1;
    % iterate if correct target
    full_reach_raster = full(R(i).spikeRaster);
    reach_raster{3}(:,:,iter(3)) = full_reach_raster(:,1:500);
    % convert spike train to full matrix and save first 500 ms
end
if (isequal(R(i).target(1:2),reach_target(:,4)))
    % remove if target is in the center
    iter(4) = iter(4)+1;
    % iterate if correct target
    full_reach_raster = full(R(i).spikeRaster);
    reach_raster{4}(:,:,iter(4)) = full_reach_raster(:,1:500);
    % convert spike train to full matrix and save first 500 ms
end
if (isequal(R(i).target(1:2),reach_target(:,5)))
    % remove if target is in the center
    iter(5) = iter(5)+1;
    % iterate if correct target
    full_reach_raster = full(R(i).spikeRaster);
    reach_raster{5}(:,:,iter(5)) = full_reach_raster(:,1:500);
    % convert spike train to full matrix and save first 500 ms
end
if (isequal(R(i).target(1:2),reach_target(:,6)))
    % remove if target is in the center
    iter(6) = iter(6)+1;
    % iterate if correct target
    full_reach_raster = full(R(i).spikeRaster);
    reach_raster{6}(:,:,iter(6)) = full_reach_raster(:,1:500);
    % convert spike train to full matrix and save first 500 ms
end
if (isequal(R(i).target(1:2),reach_target(:,7)))
    % remove if target is in the center
    iter(7) = iter(7)+1;
    % iterate if correct target
    full_reach_raster = full(R(i).spikeRaster);
    reach_raster{7}(:,:,iter(7)) = full_reach_raster(:,1:500);
    % convert spike train to full matrix and save first 500 ms
end
if (isequal(R(i).target(1:2),reach_target(:,8)))
    % remove if target is in the center
    iter(8) = iter(8)+1;
    % iterate if correct target
    full_reach_raster = full(R(i).spikeRaster);
    reach_raster{8}(:,:,iter(8)) = full_reach_raster(:,1:500);
```

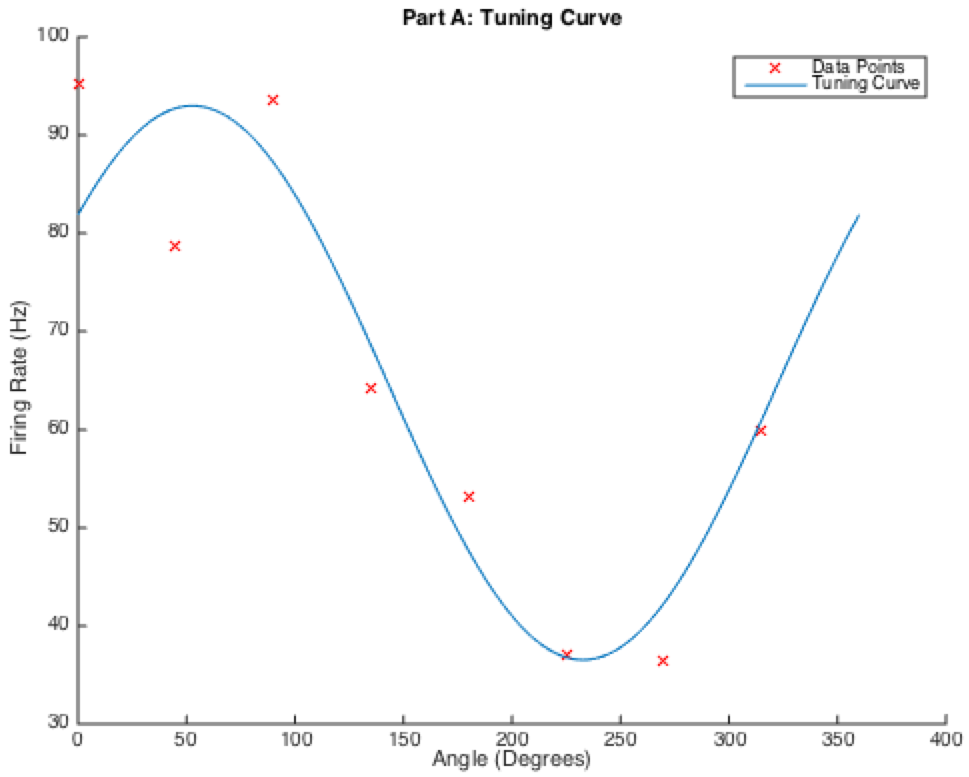
```
        % convert spike train to full matrix and save first 500 ms
    end
    if (isequal(R(i).target(1:2),reach_target(:,9)))
        % remove if target is in the center
        iter(9) = iter(9)+1;
        % iterate if correct target
        full_reach_raster = full(R(i).spikeRaster);
        reach_raster{9}(:,:,iter(9)) = full_reach_raster(:,1:500);
        % convert spike train to full matrix and save first 500 ms
    end
end

dt = 25;
spike_count = cell(1,n_targets);
spike_avg = cell(1,n_targets);

for i = 1:n_targets
    spike_count{i} = binFunc(reach_raster{i},dt);
    spike_avg{i} = sum(spike_count{i}(17,:,:),3)/iter(i);
    % taking the average across all trials for each bin
    % iter(i) is the number of trials for each direction
end
```

Part A: Tuning Curve Fit

```
% averaging firing rates from 250 to 500 ms for each reach direction
% starting at index 10 until end of vector
start_bin = length(1:dt:250) + 1;
mean_rates = zeros(1, n_targets);

for i = 1:n_targets
    mean_rates(i) = mean(spike_avg{i}(start_bin:end))*1000/25;
    % multiply by 1000 to get units of Hz for firing rate, then divide by
    % 25 due to 25 ms bins
end

% target locations (angles)

theta = [180 225 135 270 90 315 45 0];
y = [mean_rates(1:4) mean_rates(6:end)];
% take out the center 0, 0 target

figure(1)
[c0, c1, theta0] = tcFit(theta, y, 1);
fprintf('Tuning Curve Parameters for Part A: \n c0 = %2.2f, c1 = %2.2f, theta0 = %
    c0, c1, theta0)

% We see that the least squares fit for the cosine gives a good tuning
% curve approximation, but there are errors due to the fact that Electrode
% 17 represents multiple neurons, and tuning curves are usually used to
% represent the preferred direction for a single neuron.
```

```
Tuning Curve Parameters for Part A:
```

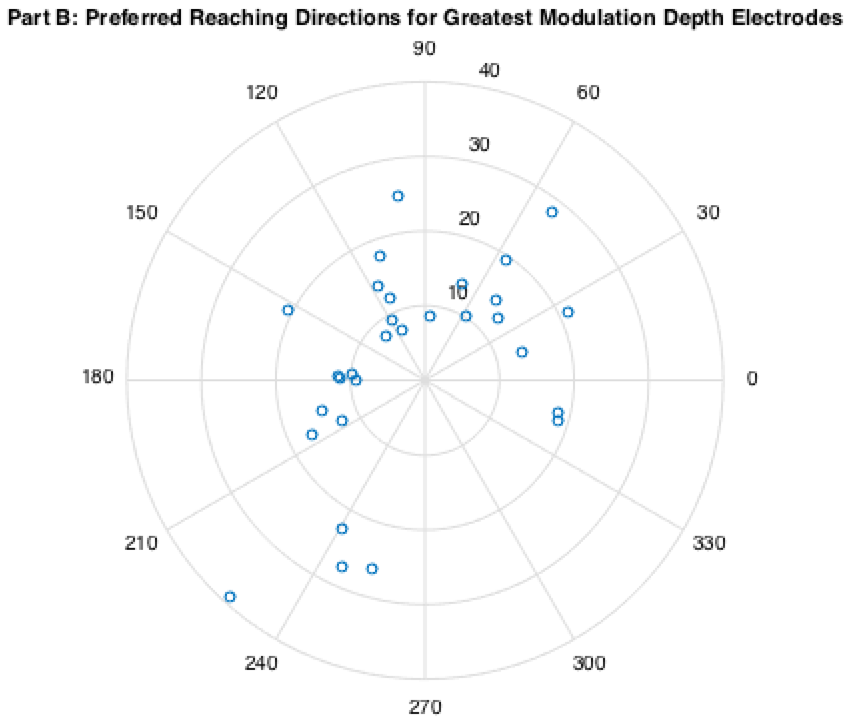
```
c0 = 64.76, c1 = 28.23, theta0 = 52.72
```



Part B: Top 30 Electrodes

```
spike_avg_all = cell(1,n_targets);

for i = 1:n_targets
    spike_avg_all{i} = sum(spike_count{i},3)/iter(i);
    % taking the average across all trials for each bin
    % iter(i) is the number of trials for each direction
end

mean_rates_all = zeros(n_electrodes, n_targets);
for i = 1:n_targets
    mean_rates_all(:,i) = mean(spike_avg_all{i}(:,start_bin:end),2)*1000/25;
    % multiply by 1000 to get units of Hz for firing rate, then divide by
    % 25 due to 25 ms bins
end

y_all = [mean_rates_all(:,1:4) mean_rates_all(:,6:end)];
c0_all = zeros(n_electrodes, 1);
c1_all = zeros(n_electrodes, 1);
theta0_all = zeros(n_electrodes, 1);

for i = 1:n_electrodes
    [c0_all(i), c1_all(i), theta0_all(i)] = tcFit(theta, y_all(i,:), 0);
```

```
end

[sorted_c1,sorted_ind] = sort(abs(c1_all),'descend');
top_c1_ind = sorted_ind(1:30);
for i = 1:length(top_c1_ind)
    if c1_all(top_c1_ind(i))>0
        theta_top(i) = theta0_all(top_c1_ind(i));
        c1_top(i) = c1_all(top_c1_ind(i));
    else
        theta_top(i) = theta0_all(top_c1_ind(i))+180;
        c1_top(i) = -c1_all(top_c1_ind(i));
    end
end

figure(2)
h = polar(theta_top*pi/180,c1_top,'o');
set(h,'markersize',5)
title('Part B: Preferred Reaching Directions for Greatest Modulation Depth Electro

% This does not span the reaching space well; there is a 60 degree gap in
% the bottom right (270-330 degrees) where there is no preferred reaching
% direction represented by the top 30 modulation depth electrodes.
```



Part C: Optimal Linear Estimator (Velocity)

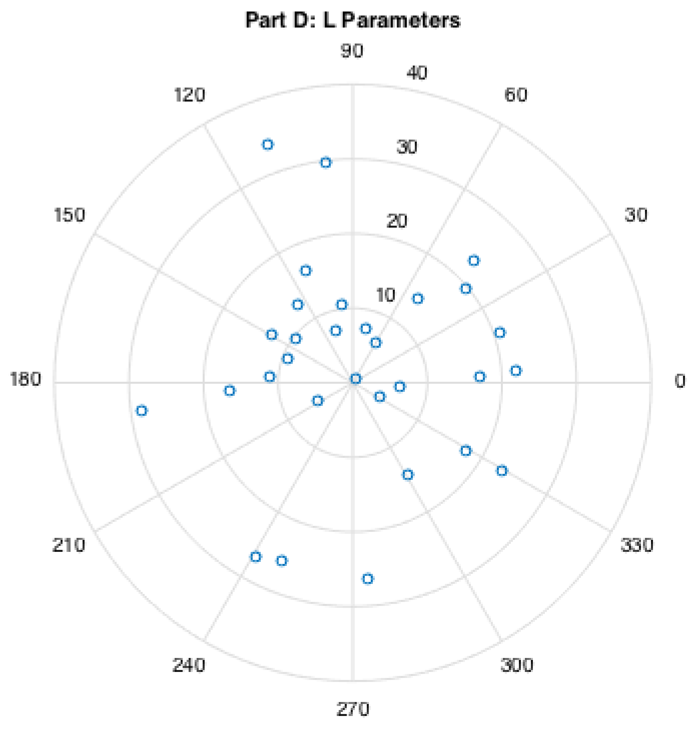
```
Y = [R(1:400).spikeRaster];
```

```
Y_reach_raster = full(Y);
Y_bin = binFunc(Y_reach_raster, dt);
Y_bin = Y_bin(:, 1:end-1);
% get rid of the last bin which does not have 25 ms worth of data
Y_bin = [Y_bin; ones(1, size(Y_bin, 2))];

X = [R(1:400).cursorPos];
sample_ind = 1:25:length(X);

pos_bin = X(1:2,sample_ind);

X_bin = diff(pos_bin,1,2)/0.025;

fprintf('\nY_bin Size:')
disp(size(Y_bin))

fprintf('\nX_bin Size:')
disp(size(X_bin))

% The dimensions of X_bin and Y_bin make sense, because they are the
% concatenated bins for all 400 training trials for all 96 neurons. We are
% also deleting the last bin from each trial if it does not have 25 ms worth
% of data.
```

```
Y_bin Size:          97       16465


X_bin Size:           2       16465
```

Part D: Optimal Linear Estimator (Parameters)

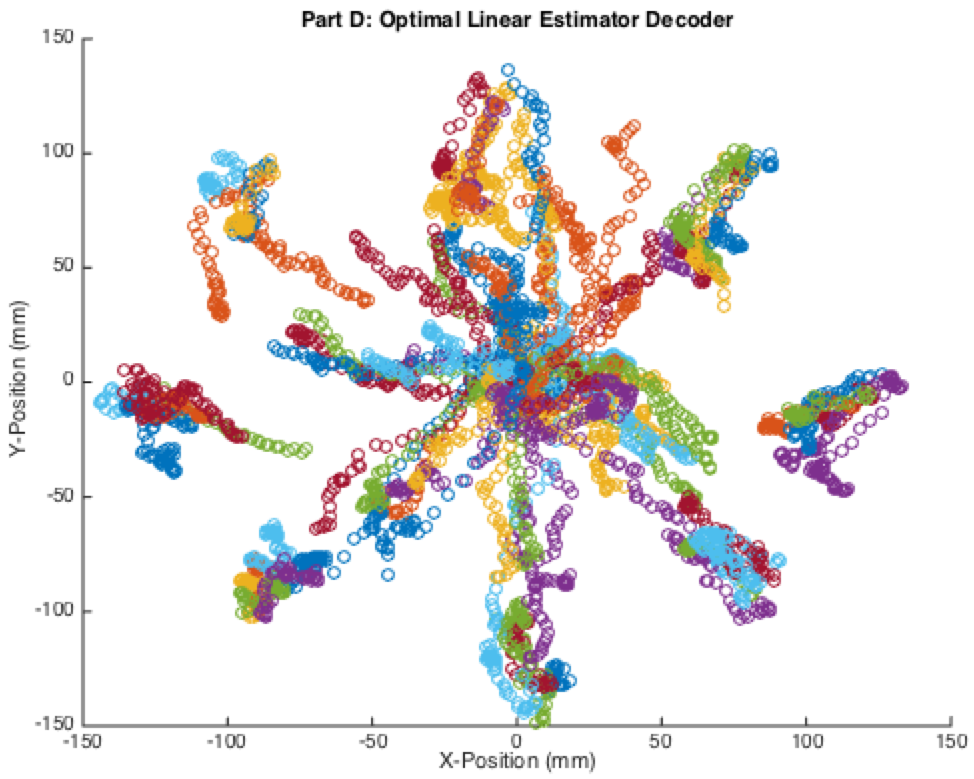
```
L = X_bin * pinv(Y_bin);
L_top = L(:,top_c1_ind);

figure(3)
[L_theta, L_rho] = cart2pol(L_top(1,:),L_top(2,:));

h = polar(L_theta,L_rho,'o');
set(h,'markersize',5)
hold on
title('Part D: L Parameters')
hold off

% The plot does not look similar to the plot in Part C, as it covers the
% reaching space better. The OLE fits the data, and therefore it doesn't
% have the gap at the bottom right reaching angles.
```



Part E: Optimal Linear Estimator (Decoding)

```
X_decode = cell(1, 106);
start_pos = zeros(2, 106);
X_test = cell(1, 106);
X_test_pos = cell(1,106);

for i = 1:106
    start_pos(:,i) = R(400+i).cursorPos(1:2,1);
    Y_test = full(R(400+i).spikeRaster);
    Y_test_bin = binFunc(Y_test, dt);
    Y_test_bin = [Y_test_bin; ones(1, size(Y_test_bin,2))];
    X_decode{i} = L * Y_test_bin;
    X_test{i} = X_decode{i}*0.025;
    %X_test_pos(:,1,i) = start_pos(:,i);
    X_test_pos{i}(:,1) = start_pos(:,i);
    for j = 2:length(X_test{i})+1
        X_test_pos{i}(:,j) = X_test_pos{i}(:,j-1) + X_test{i}(:,j-1);
    end
    % 2 x time point x trial
end

figure(4)
hold on
for i = 1:106
```

```
    scatter(X_test_pos{i}(1,:), X_test_pos{i}(2,:))
end

hold off
title('Part D: Optimal Linear Estimator Decoder')
xlabel('X-Position (mm)')
ylabel('Y-Position (mm)')

% There are idiosyncracies in the decoding. For example, when the monkey
% reaches from the center target to the left target vs. right target to
% center target, the positions and velocities are similar. However, the
% neural data is not the same for these two movements. The OLE cannot
% differentiate between the two because it only takes into account the
% preferred direction.
```



Part F: Mean-Square Error

```
errors = cell(1,106);
mean_errors = zeros(2,106);

for i = 1:106
    pos_vec = R(400+i).cursorPos(1:2,:);
    sample_ind_pos = [1:25:length(pos_vec), length(pos_vec)];
    pos_vec = pos_vec(:, sample_ind_pos);
    errors{i} = (pos_vec - X_test_pos{i}).^2;
    mean_errors(:,i) = mean(errors{i},2);
```

```
end

mean_error_all = sum(mean(mean_errors,2));

fprintf('\nOptimal Linear Estimator Mean Square Error: %4.2f\n', mean_error_all)
```

```
Optimal Linear Estimator Mean Square Error: 5391.29
```

Published with MATLAB® R2014b