Table of Contents

EE239AS HW #6

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

Problem 4

```
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;
dt = 25;
```

Part A: A Matrix

```
v_xx = 0.7;
v_yy = 0.7;
v_yx = 0;
v_xy = 0;

A = [1 0 dt 0 0;
    0 1 0 dt 0;
    0 0 v_xx v_xy 0;
    0 0 v_yx v_yy 0;
    0 0 0 0 1];

% extra column and row of zeros accounts for the bias term

fprintf('\nPart A -- Kalman Filter A Matrix: \n')
disp(A)
```

```
Part A -- Kalman Filter A Matrix:
    1.0000         0   25.0000         0         0
         0    1.0000         0   25.0000         0
         0         0    0.7000         0         0
         0         0         0    0.7000         0
         0         0         0         0    1.0000
```

Part B: A Matrix

```
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)/25;
% Euler approximation of velocities
X_k = [X_bin; ones(1, size(X_bin,2))];

A_s = X_k(:,2:end)*pinv(X_k(:,1:end-1));
% use ML to estimate v_xx, v_yy, v_yx, v_xy
A(3:4,3:4) = A_s(1:2, 1:2);

fprintf('\nPart B -- Kalman Filter A Matrix: \n')
disp(A)
```

```
Part B -- Kalman Filter A Matrix:
    1.0000         0   25.0000         0         0
         0    1.0000         0   25.0000         0
         0         0    0.7798   -0.0074         0
         0         0    0.0096    0.7808         0
         0         0         0         0    1.0000
```

Part C: C Matrix

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

Y_reach_raster = full(Y);
Y_bin = binFunc(Y_reach_raster, dt);
Y_bin = Y_bin(:, 1:end-1);

Y_k = Y_bin;
C = zeros(n_electrodes, 5);
C_s = Y_k * pinv(X_k);

C(:,end-2:end) = C_s;
C_sum = sum(C,1);

fprintf('\nPart C -- Sum of C Across Electrodes:\n')
disp(C_sum)
```

```
Part C -- Sum of C Across Electrodes:
         0         0   -4.4994    1.7053   40.1927
```

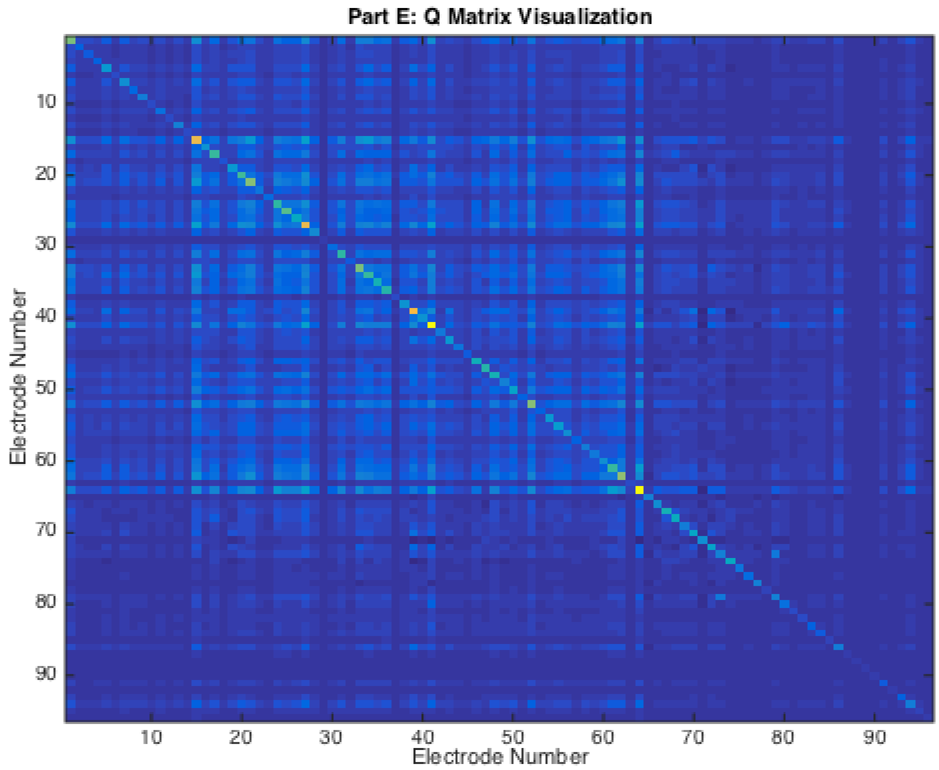
Part D: W Matrix

```
W_s = 1/(size(X_k,2)-1)*(X_k(:,2:end) - A_s*X_k(:,1:end-1))*(X_k(:,2:end)-A_s*X_k(
W = zeros(size(A));
W(3:4,3:4) = W_s(1:2,1:2);

fprintf('\nPart D -- Kalman Filter W Matrix: \n')
disp(W)
```

```
Part D -- Kalman Filter W Matrix:
         0         0         0         0         0
         0         0         0         0         0
         0         0    0.0115   -0.0008         0
         0         0   -0.0008    0.0121         0
         0         0         0         0         0
```

Part E: Q Matrix

```
Q = 1/size(X_k,2)*(Y_k-C_s*X_k)*(Y_k-C_s*X_k)';

figure(1)
imagesc(Q)
title('Part E: Q Matrix Visualization')
xlabel('Electrode Number')
ylabel('Electrode Number')

% tells you how well y_k approximates Cx_k
% diagonal is for each electrode
% off diagonals are co-varied in residuals
% dark neurons are close to 0 firing rate, and C is close to 0
```



Part F: Steady State Kalman Filter

```
S = zeros(size(W));

tol = 10^-13;
iterativeDiff = inf;
M_1_prev = 0;
M_2_prev = 0;
iter = 0;

while any(abs(iterativeDiff)>tol)
    iter = iter+1;
    S = A*S*A' + W;
    % introduce uncertainty
    S(1:2,:) = 0;
    S(:, 1:2) = 0;
    % zero out uncertainty so it does not propagate
    % calculate S (k_k-1) using previous S (k-1_k-1) value
    K_k = S*C'*inv(C*S*C' + Q);
    % calculate Kalman gain from updated S value
    S = S - S*C'*inv(C*S*C' + Q)*C*S;
    % calculate S (k_k) using previous S (k_k-1) value

    M_1 = A-K_k*C*A;
    M_2 = K_k;
```

```
    iterativeDiff = [reshape(M_1, 1, []) - reshape(M_1_prev,1,[])...
    reshape(M_2, 1, []) - reshape(M_2_prev,1,[])];
    M_1_prev = M_1;
    M_2_prev = M_2;
end

fprintf('\nPart F -- Kalman Filter M_1 Matrix:\n')
disp(M_1)

fprintf('\nPart F -- Kalman Gain (All Electrodes):\n')
disp(sum(M_2,2))
```

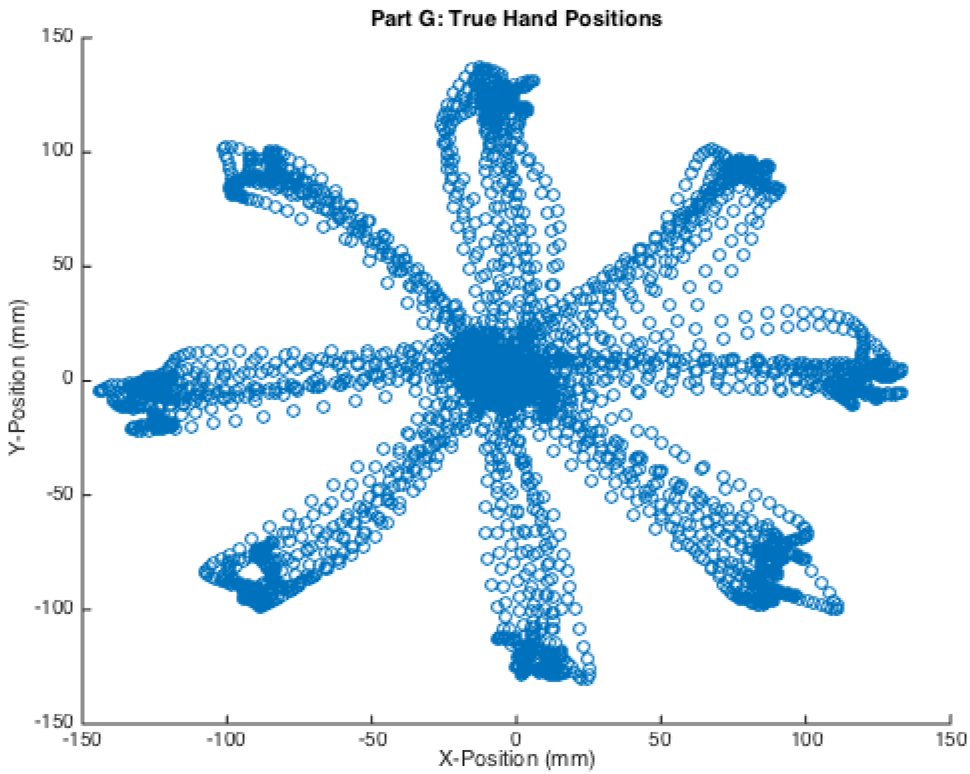
```
Part F -- Kalman Filter M_1 Matrix:
    1.0000         0   25.0000         0         0
         0    1.0000         0   25.0000         0
         0         0    0.6776   -0.0254    0.0123
         0         0   -0.0154    0.6276   -0.0586
         0         0         0         0    1.0000


Part F -- Kalman Gain (All Electrodes):
         0
         0
   -0.0583
    0.1267
         0
```

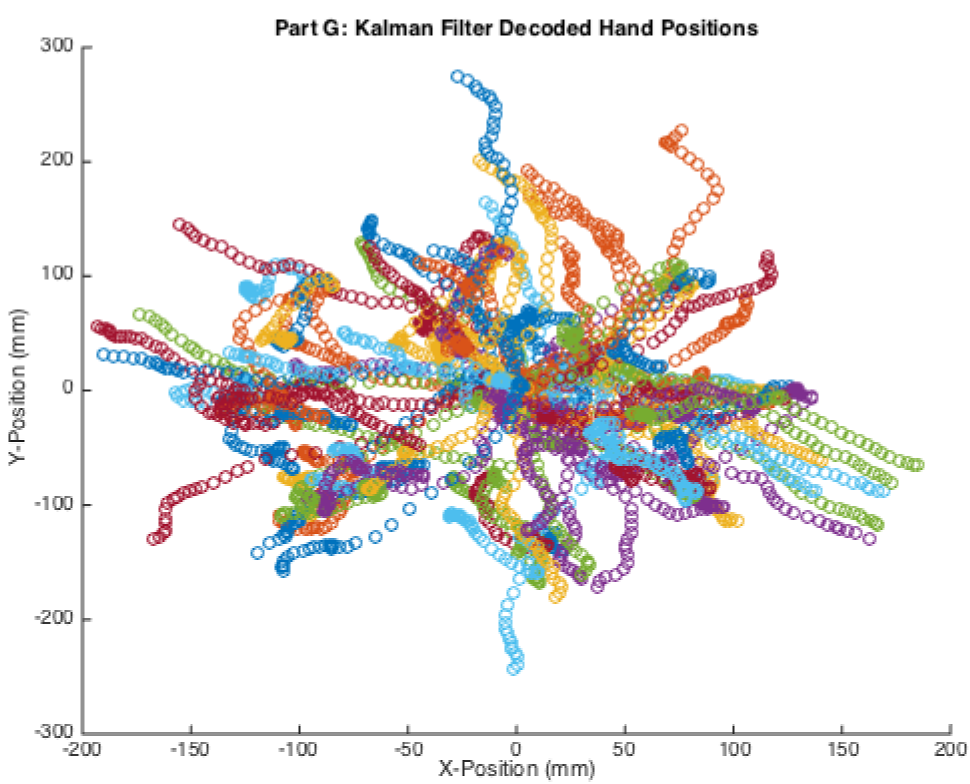
Part G: Kalman Filter Decoding

```
start_v = [0; 0];
% starting velocities in x and y
X_decode = {};

for i = 1:106
    start_pos(:,i) = R(400+i).cursorPos(1:2,1);
    % starting position in x and y
    Y_test = full(R(400+i).spikeRaster);
    Y_test_bin = binFunc(Y_test, dt);

    X_k_test = [start_pos(:,i); start_v; 1];
    n_bins = size(Y_test_bin,2);

    for j = 1:n_bins
        X_decode{i}(:,j) = M_1 * X_k_test + M_2 * Y_test_bin(:,j);
        X_k_test = X_decode{i}(:,j);
    end

    % 5x1 vector of velocities, positions, and bias term for each bin in
    % each trial
```

```
end

X_true = [R(401:end).cursorPos];

figure(2)
scatter(X_true(1,:), X_true(2,:))
title('Part G: True Hand Positions')
xlabel('X-Position (mm)')
ylabel('Y-Position (mm)')

figure(3)
hold on
for i = 1:106
    scatter(X_decode{i}(1,:), X_decode{i}(2,:))
end
hold off

title('Part G: Kalman Filter Decoded Hand Positions')
xlabel('X-Position (mm)')
ylabel('Y-Position (mm)')

% Part G(b): 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);
    pos_vec = pos_vec(:, sample_ind_pos);
    errors{i} = (pos_vec - X_decode{i}(1:2,:)).^2;
    mean_errors(:,i) = mean(errors{i},2);
end
mean_error_all = sum(mean(mean_errors,2));

fprintf('\nKalman Filter Mean Square Error: %4.2f\n', mean_error_all)
```

Kalman Filter Mean Square Error: 3716.53


Part G: True Hand Positions
Y-Position (mm)
X-Position (mm)
150
100
50
0
-50
-100
-150
-150
-100
-50
0
50
100
150


Part G: Kalman Filter Decoded Hand Positions
Y-Position (mm)
X-Position (mm)
300
200
100
0
-100
-200
-300
-200
-150
-100
-50
0
50
100
150
200

Published with MATLAB® R2014b