**Problem 1:**

1. X-position:

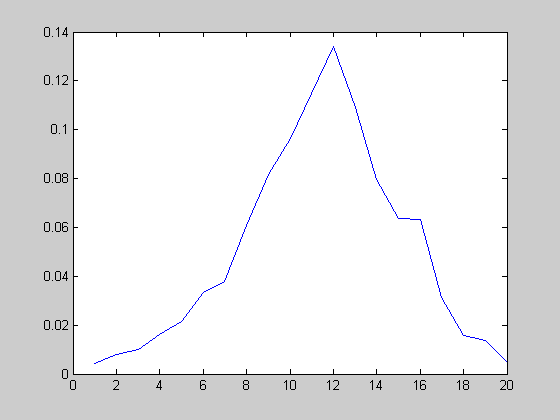


Figure 1: Plot for the prior

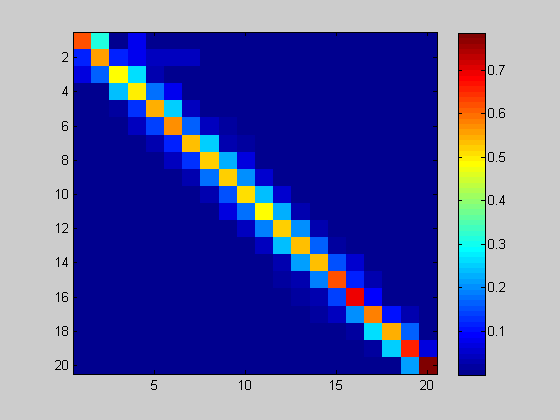
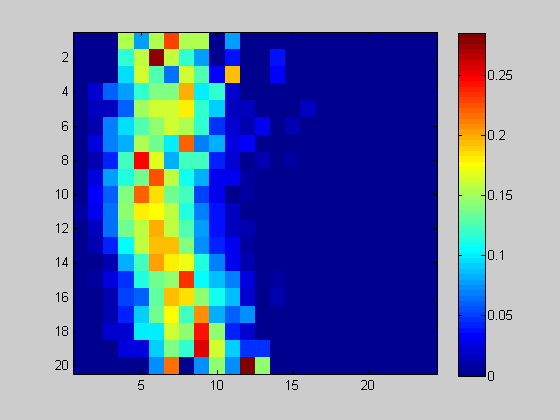


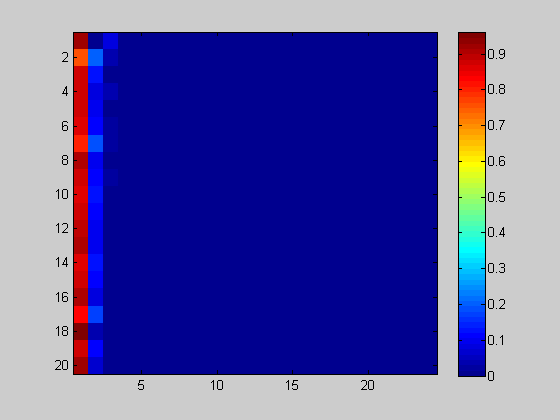
Figure 2: Plot for the transition probability {}

Plot the observation probability :

For this Cell 1, no matter where it is, the fire rate is always in a range:



For this cell 6, the fire rate is always very concentrated.



(b)

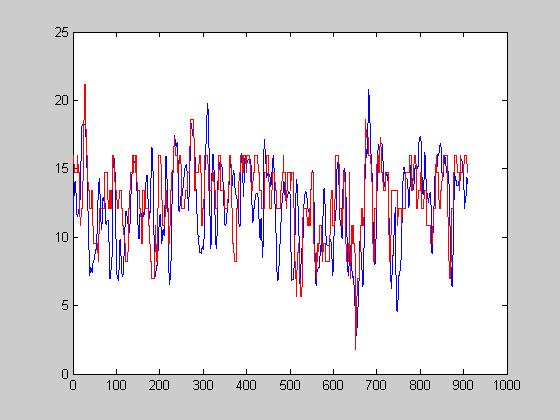


Figure 4: Plot for the estimated x-position and true x-position

R\_square = 0.2186

(c) Y-position:

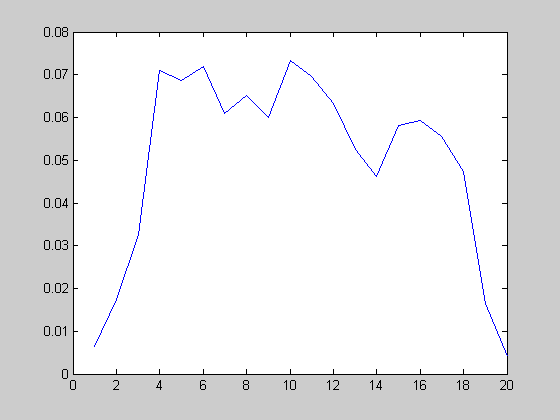


Figure 1: Plot for the prior

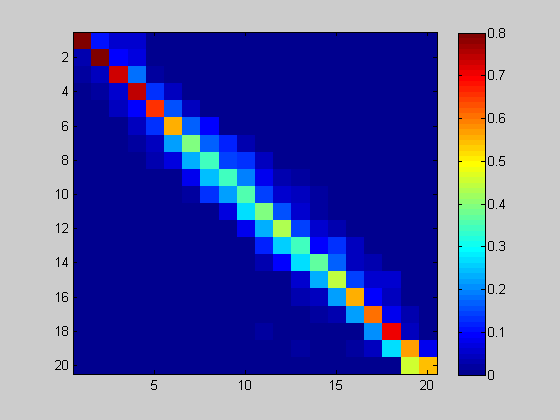
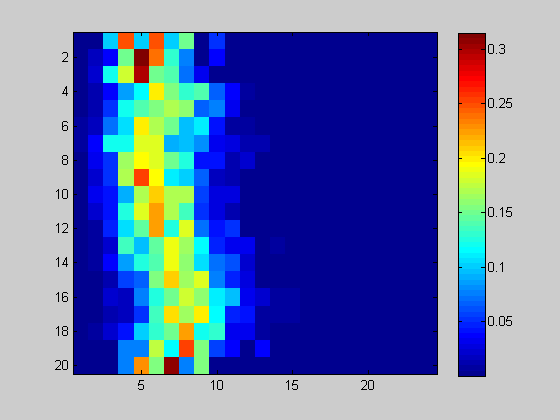


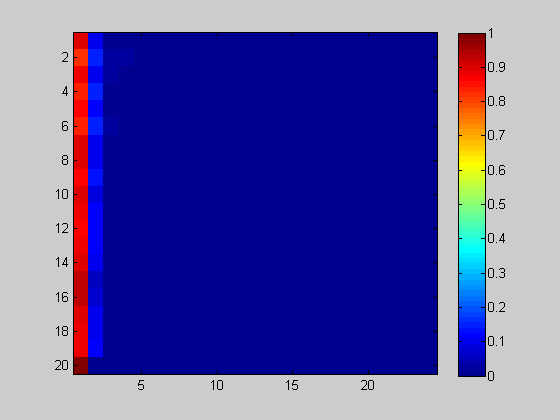
Figure 2: Plot for the transition probability {}

Plot the observation probability :

For this Cell 1, no matter where it is, the fire rate is always in a range.



For this cell 6, the fire rate is always very concentrated.



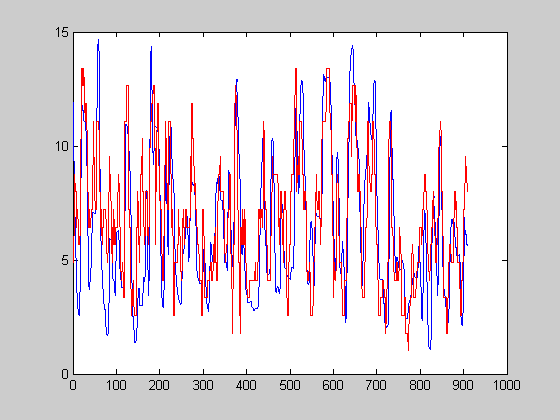


Figure 4: Plot for the estimated y-position and true y-position

R\_square = 0.5228

**Code:**

%% fit the model using training data

clear;

load 'C:\Documents and Settings\goober\Desktop\hw10\_1\_train';

[T, C] = size(rate);

% choose either x or y position

d = 2; % 1: x-pos, 2: y-pos

kin = kin(:,d);

% discretize the kinematics

N = 20; % discretization number

max\_kin = max(kin);

min\_kin = min(kin);

nkin = (kin-min\_kin)/(max\_kin-min\_kin); % scale to [0 1]

dkin = round((N-1)\*nkin) + 1; % discrete state: 1, 2,..., N

% the prior of kinematics p(x\_k)

for i = 1:N

Px(i) = sum(dkin == i);

end

Px = Px./sum(Px);

% the transition probability p(x\_{k+1} | x\_k)

for i = 1:N

for j = 1:N

Ptx(i,j) = sum(dkin(1:T-1) == i & dkin(2:T) == j);

end

end

Ptx = Ptx./(sum(Ptx,2)\*ones(1,N));

R = max(rate(:))+1; % firing rate is in {0, 1, 2, ..., R-1}

% the prior of firing rate p(z\_k)

for j = 1:R

Pz(j,:) = sum(rate == j-1);

end

Pz = Pz./(ones(R,1)\*sum(Pz));

% the observation probability p(z\_k | x\_k) for each neuron

for k = 1:C

for i = 1:N

for j = 1:R

Pzx(i,j,k) = sum(dkin == i & rate(:,k) == j-1);

end

end

Pzx(:,:,k) = Pzx(:,:,k)./(sum(Pzx(:,:,k),2)\*ones(1,R)+eps);

end

figure(1);

plot(Px);

figure(2);

Imagesc(Ptx);

colorbar;

figure (3);

Imagesc(Pzx(:,:,6));

colorbar;

%save hmm\_1d Px Ptx Pz Pzx max\_kin min\_kin d N;

% decode the hmm in the test data

% use log of probability in the calculation

load 'C:\Documents and Settings\goober\Desktop\hw10\_1\_test';

kin = kin(:,d);

[T, C] = size(rate);

% joint probability of firing rates of all neurons conditioned on kineamtics

% assuming independence over neurons

lpz\_jx = zeros(T,N);

for i = 1:T

for k = 1:C

lpzx(:,k) = log(Pzx(:,rate(i,k)+1,k)+eps);

end

lpz\_jx(i,:) = sum(lpzx,2)';

end

%%% Dynamic programming to find the optimal kinematics %%%

joint\_prob = zeros(T,N); % delta\_t(n) in the notes

joint\_prob(1,:) = log(Px+eps)+lpz\_jx(1,:); % delta\_1(n)

for k = 2:T

for m = 1:N

temp = joint\_prob(k-1,:)+log(Ptx(:,m)'+eps)+lpz\_jx(k,m);

[joint\_prob(k,m), ind(k,m)] = max(temp); % ind save optimal pre-index

end

end

[ignore, x\_h(T)] = max(joint\_prob(T,:)); % identify last time step

for k = T-1:-1:1 % recursively identification

x\_h(k) = ind(k+1,x\_h(k+1));

end

kin\_h = (max\_kin-min\_kin)\*(x\_h'-1)/(N-1)+min\_kin; % return to the original scale

figure(4);

plot(1:T, kin, 1:T, kin\_h, 'r','LineWidth',1.2);

R2 = 1 - sum((kin-kin\_h).^2)/(sum((kin-mean(kin)).^2));