18-699 / 42-590
Neural Signal Processing
TA: John Kelly Fall 2010

Problem Set 9: Solutions

Table 1: Point Breakdown

Problem	Points
1a	10
	10
1b	20
1c	10
	10
	5
1d	10
	10
	5
1e	10

Please see John with any questions regarding the grading of this problem set.

Problem 1

a) see MATLAB code

b)

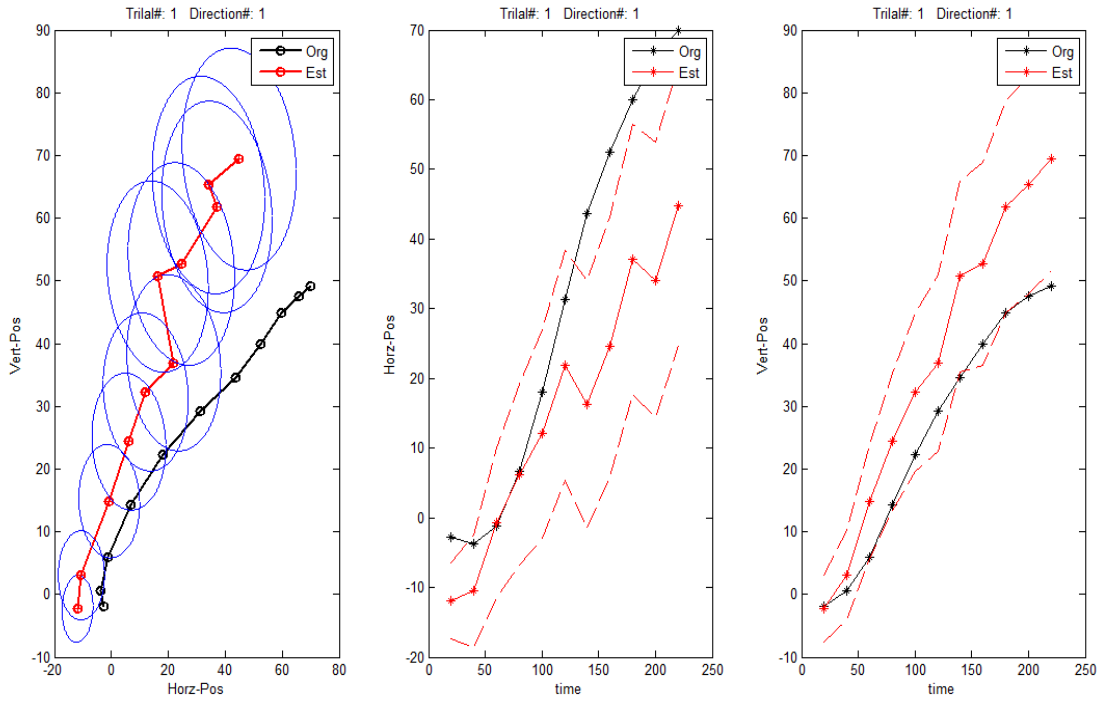
A =

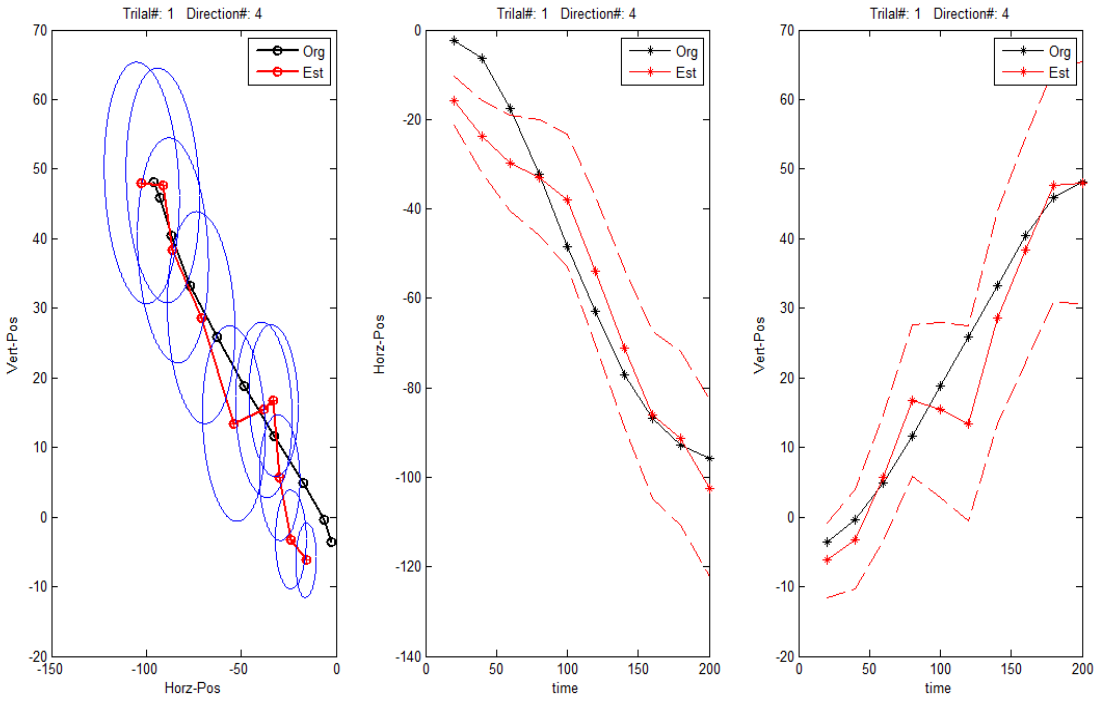
1.0000	0.0000	20.0000	0.0000
0.0000	1.0000	-0.0000	20.0000
-0.0019	-0.0008	1.0461	0.0677
0.0001	-0.0017	-0.0329	1.0404

Q =

0.0000	0.0000	0.0000	-0.0000
0.0000	0.0000	0.0000	-0.0000
0.0000	0.0000	0.0166	-0.0007
-0.0000	-0.0000	-0.0007	0.0139

c, d)





e)

dist =

18.8558

MATLAB Code

```
clear;
load 'ps9_data.mat';

[NTrain NDirect]=size(train_trial);
NTest=size(test_trial,1);

%% %%%%%%%%%%%%%%%%%(a) Data preprocessing%%%%%%%%%%%%%%%%%
timeStep=20;

for trialIX=1:NTrain
    for directIX=1:NDirect
        curSpikes=train_trial(trialIX,directIX).spikes;
        curPos=train_trial(trialIX,directIX).handPos;

        timeWinE=timeStep:timeStep:size(curSpikes,2);
        curSpikes=curSpikes(:,1:timeWinE(end-1))';
        curPos=curPos(:,1:timeWinE(end));

        % Take spike counts in each bin
        spikeMtr=reshape(curSpikes,timeStep,[],size(curSpikes,2));
        train(trialIX,directIX).spikeCount=squeeze(sum(spikeMtr,1))';

        % Compute a 4-dimensional arm state
        pos=curPos(1:2,timeWinE);
        vel=(pos(:,2:end)-pos(:,1:end-1))/timeStep;
        train(trialIX,directIX).state=[pos(:,1:end-1); vel];
    end
end

for trialIX=1:NTest
    for directIX=1:NDirect
        curSpikes=test_trial(trialIX,directIX).spikes;
        curPos=test_trial(trialIX,directIX).handPos;

        timeWinE=timeStep:timeStep:size(curSpikes,2);
        curSpikes=curSpikes(:,1:timeWinE(end-1))';
        curPos=curPos(:,1:timeWinE(end));

        % Take spike counts in each bin
        spikeMtr=reshape(curSpikes,timeStep,[],size(curSpikes,2));
        test(trialIX,directIX).spikeCount=squeeze(sum(spikeMtr,1))';

        % Compute a 4-dimensional arm state
        pos=curPos(1:2,timeWinE);
        vel=(pos(:,2:end)-pos(:,1:end-1))/timeStep;
        test(trialIX,directIX).state=[pos(:,1:end-1); vel];
    end
end
clear train_trial;
clear test_trial;

%% %%%%%%%%%%%%%%%%%(b) Training Phase%%%%%%%%%%%%%%%%%
% Fit A, Pi, V and C
interStateMtr=0;
intraStateMtr=0;
obsStateMtr=0;
```

```
obsIntraStateMtr=0;
count=0;
countPool=0;
for trialIX=1:NTrain
    for directIX=1:NDirect
        % for parameter A
        stateZ1=train(trialIX,directIX).state(:,2:end);
        stateZ2=train(trialIX,directIX).state(:,1:end-1);
        tmp=stateZ1*stateZ2';
        interStateMtr=interStateMtr+tmp;
        tmp=stateZ2*stateZ2';
        intraStateMtr=intraStateMtr+tmp;
        % for parameter C
        stateZ=train(trialIX,directIX).state;
        obserX=train(trialIX,directIX).spikeCount;
        tmp=obserX*stateZ';
        obsStateMtr=obsStateMtr+tmp;
        tmp=stateZ*stateZ';
        obsIntraStateMtr=obsIntraStateMtr+tmp;
        % for parameter Pi and V
        count=count+1;
        poolZStart(:,count)=train(trialIX,directIX).state(:,1);
    end
end
A=(interStateMtr/count)*inv(intraStateMtr/count);
C=(obsStateMtr/count)*inv(obsIntraStateMtr/count);
Pi=mean(poolZStart,2);
V=cov(poolZStart');

% Fit Q and R
sumQMtr=0;
sumRMtr=0;
count=0;
for trialIX=1:NTrain
    for directIX=1:NDirect
        % for parameter Q
        stateZ1=train(trialIX,directIX).state(:,2:end);
        stateZ2=train(trialIX,directIX).state(:,1:end-1);
        tmp=stateZ1-A*stateZ2;
        tmp=tmp*tmp';
        count=count+size(stateZ1,2);
        sumQMtr=sumQMtr+tmp;
        % for parameter R
        stateZ=train(trialIX,directIX).state;
        obserX=train(trialIX,directIX).spikeCount;
        tmp=obserX-C*stateZ;
        tmp=tmp*tmp';
        sumRMtr=sumRMtr+tmp;
    end
end
Q=sumQMtr/(count);
R=sumRMtr/(count+NTrain*NDirect);

%% %%%%%%%%%%%%%%%%%(c) Test phase%%%%%%%%%%%%%%%%%
for trialIX=1:NTest
    for directIX=1:NDirect
        curTrialLen=size(test(trialIX,directIX).state,2);
        % Initializaton
        mu=Pi;
```

```
        sigma=V;
        clear curEstStateMean;
        clear curEstStateCov;
        for timeIX=1:curTrialLen
            % Prediction
            mu=A*mu;
            sigma=A*sigma*A'+Q;
            % compute the Kalman gain
            K=sigma*C'*inv(C*sigma*C'+R);
            % update the state
            curObservX=test(trialIX,directIX).spikeCount(:,timeIX);
            mu=mu+K*(curObservX-C*mu);
            sigma=sigma-K*C*sigma;

            curEstStateMean(:,timeIX)=mu;
            curEstStateCov(:,:,timeIX)=sigma;
        end
        test(trialIX,directIX).estStateMean=curEstStateMean;
        test(trialIX,directIX).estStateCov=curEstStateCov;
    end
end

% Plot the selected trials;
% selTrialList={{1,1};{1,2};{1,3};{1,4};{1,5};{1,6};{1,7};{1,8}};
selTrialList={{1,1};{1,4}};
for trialIX=1:size(selTrialList,1)
    selTrial=selTrialList{trialIX};
    curTrial=test(selTrial{1},selTrial{2});
    figure(trialIX);
    titleStr=sprintf('Trilal#: %d   Direction#: %d',selTrial{1},selTrial{2});

    subplot(1,3,1);
    plot(curTrial.state(1,:),curTrial.state(2,:),'-ok','LineWidth',2);
    hold on; plot(curTrial.estStateMean(1,:),curTrial.estStateMean(2,:),...
'-or','LineWidth',2);
    for timeIX=1:length(curTrial.estStateMean(1,:))
        hold on; func_plotEllipse(curTrial.estStateMean(1:2,timeIX),...
curTrial.estStateCov(1:2,1:2,timeIX));
    end
    xlabel('Horz-Pos'); ylabel('Vert-Pos'); legend('Org','Est');
    title(titleStr);

    subplot(1,3,2);
    selState=1;
    curStd=(squeeze(sqrt(curTrial.estStateCov(selState,selState,:))))';
    curTimeIX=20*(1:length(curStd));
    plot(curTimeIX,curTrial.state(selState,:),'-*k');
    hold on; plot(curTimeIX,curTrial.estStateMean(selState,:),'-*r');
    hold on; plot(curTimeIX,curTrial.estStateMean(selState,:)+curStd,'--r');
    hold on; plot(curTimeIX,curTrial.estStateMean(selState,:)-curStd,'--r');
    xlabel('time'); ylabel('Horz-Pos'); legend('Org','Est');
    title(titleStr);

    subplot(1,3,3);
    selState=2;
    curStd=(squeeze(sqrt(curTrial.estStateCov(selState,selState,:))))';
    curTimeIX=timeStep*(1:length(curStd));
    plot(curTimeIX,curTrial.state(selState,:),'-*k');
    hold on; plot(curTimeIX,curTrial.estStateMean(selState,:),'-*r');
```

```
    hold on; plot(curTimeIX,curTrial.estStateMean(selState,:)+curStd,'--r');
    hold on; plot(curTimeIX,curTrial.estStateMean(selState,:)-curStd,'--r');
    xlabel('time'); ylabel('Vert-Pos'); legend('Org','Est');
    title(titleStr);
end

% Performance evaluation
dist=0;
for trialIX=1:NTest
    for directIX=1:NDirect
        currDiff=test(trialIX,directIX).estStateMean(1:2,:)-
test(trialIX,directIX).state(1:2,:);
        currDiff=sqrt(sum(currDiff.^2));
        dist=dist+mean(currDiff);
    end
end
dist=dist/(NTest*NDirect);

return;
```