HA2 ANALYSIS:

```
clear all
clc

%% Scenario 1: A first Kalman filter and its properties
A= 1;
H= 1;
Q= 1.5;
R= 3;
x_0= 2;
P_0= 8;
N= 35;

% Problem 1a
X= genLinearStateSequence(x_0, P_0, A, Q, N);
Y = genLinearMeasurementSequence(X, H, R);
figure;
plot(X(2:end),'-b')
hold on
plot(Y,'*r')
xlabel('Sequence length')
title('State Sequence and Measurement Sequence')
legend('State Sequence', 'Measurement
Sequence','location','best')
hold off

% Problem 1b
[X, P] = kalmanFilter(Y, x_0, P_0, A, Q, H, R);
figure;
hold on;
plot([1:N],X,'r');
plot([1:N],Y,'.b');
plot([0:N],[x_0 X],'g');
plot([0:N],[x_0 X]+3*sqrt([P_0 P(:)']),'--r');
plot([0:N],[x_0 X]-3*sqrt([P_0 P(:)']),'--r');
xlabel('time step(k)');
ylabel('x');
legend('true state','measurements', 'state estimate', '+3-
sigma level', '-3-sigma level','Location','best');
hold off

time_k_vals = [1,2,4,30];

for i = 1:length(time_k_vals)
```

```
    k_vals = -15:0.1:35;
    uncertainity= normpdf(k_vals, X(:,i), P(:,:,i));
    true_state= normpdf(k_vals, X(:,i+1), Q);
    figure;
    hold on
    plot(k_vals,uncertainity,'r')
    plot([X(:,i+1) X(:,i+1)], [0 max(uncertainity)],'--b')
    xlabel('k')
    ylabel('x')
    title('Uncertainity and True State')
    legend('Uncertainity','True State','Location','best')
    hold off
end

% Problem 1c
X = genLinearStateSequence(x_0, P_0, A, Q, N);
Y = genLinearMeasurementSequence(X, H, R);
[x_correct, P_correct] = kalmanFilter(Y, x_0, P_0, A, Q, H,
R);
x_0_wrong = 12;
P_0_wrong = 8;
[x_wrong, P_wrong] = kalmanFilter(Y, x_0_wrong, P_0_wrong,
A, Q, H, R);
figure;
hold on
plot([1:N],X(2:end),'g')
plot([1:N],Y,'ok')
plot([0:N],[x_0 x_correct],'b')
plot([0:N],[x_0 x_correct]+3*sqrt([P_0 P_correct(:)']),'--
b')
plot([0:N],[x_0 x_correct]-3*sqrt([P_0 P_correct(:)']),'--
b')
plot([0:N],[x_0_wrong x_wrong],'r')
plot([0:N],[x_0_wrong x_wrong]+3*sqrt([P_0_wrong
P_wrong(:)']),'--r')
plot([0:N],[x_0_wrong x_wrong]-3*sqrt([P_0_wrong
P_wrong(:)']),'--r')
xlabel('k')
ylabel('x')
legend('True State','Measurements', 'State estimate', '+3-
sigma level', '-3-sigma level', 'State estimate', '+3-sigma
level', '-3-sigma level', 'Location','best')
hold off
```

```
% problem 1d
k_vals= linspace(-12,12,1200);
%prior= normpdf(k_vals, X(:,10),Q);
%Prior - p(x_k-1|y_1:k-1):
sd = sqrtm(P_0);
y_pdf = normpdf(k_vals, x_0, sd);
figure;
hold on
[x_predicted,P_predicted]=
linearPrediction(X(:,10),P(:,:,10),A,Q);
prediction= normpdf(k_vals,x_predicted,P_predicted);
measurement= normpdf(k_vals,Y(:,10),R);
uncertainity= normpdf(k_vals,X(:,10),P(:,:,10));
figure;
hold on
% plot(k_vals,prior,'-b','Linewidth',3);
plot(k_vals, y_pdf,'-b')
plot(k_vals,prediction,'or');
plot([Y(:,10) Y(:,10)],[0 max(uncertainity)],'--k');
plot([X(:,11) X(:,11)],[0 max(uncertainity)],'--g');
plot(k_vals,uncertainity,'-c','Linewidth',1.5);
xlabel('k')
ylabel('x')
legend('$P(x_{9}|y_{1:9})$', '$P(x_{10}|y_{1:9})$',
'$y_{10}$', '$x_{10}$',
'$P(x_{10}|y_{1:10})$','Interpreter','latex','Location','no
rtheast');
hold off


% Problem 1e
X = genLinearStateSequence(x_0, P_0, A, Q, N);
Y = genLinearMeasurementSequence(X, H, R);
[x, P] = linearPrediction(x_0, P_0, A, Q);
[x, P] = linearUpdate(x, P, Y, H, R);
error= (X(2:end)-x);
error_pdf= normpdf(k_vals,0,sqrt(P(:,:,end)));
figure;
hold on
histogram(error,'Normalization','pdf')
plot(k_vals,error_pdf,'-r','linewidth',2)
xlabel('k')
ylabel('x')
title('Histogram of error')
```

```
legend('$histogram(x_{k}-
\hat{x_{k}})$','$\mathcal{N}(x_{0},0,P_{N|N})$','Interprete
r','latex','location','Northeast')
hold off
figure;
hold on
%autocorr(v);
xlabel('Lag')
ylabel('Autocorrelation')
title('Autocorrelation of innovation')
%legend('Autocorrelation','location','best')
hold off

%% scenario 2: Kalman filter and its tuning
T= 0.01;
A= [1, T; 0, 1];
H= [1, 0];
Q= [0,0;0,1.5];
R= 2;
x_0= [1; 3];
P_0= 4*eye(2);
N= 50;

% Problem 2a
X= genLinearStateSequence(x_0, P_0, A, Q, N);
Y= genLinearMeasurementSequence(X, H, R);
figure;
plot(X(1,2:end),'-b')
hold on
plot(Y(1,:),'*r')
xlabel('Time')
ylabel('Position')
legend('Position State Sequence', 'Position Measurement
Sequence','location','best');
hold off
figure;
plot(X(2,2:end),'-g')
xlabel('Time')
ylabel('Velocity')
legend('Velocity State','location','best')
hold off

% Problem 2b
[x, P] = kalmanFilter(Y, x_0, P_0, A, Q, H, R);
```

```
figure;
hold on
plot([1:N],X(1,2:end),'g')
plot([1:N],Y(1,:),'*b')
plot([0:N],[x_0(1) x(1,:)],'b')
plot([0:N],[x_0(1) x(1,:)] + 3*sqrt([P_0(1)
squeeze(P(1,1,:))']),'--r')
plot([0:N],[x_0(1) x(1,:)] - 3*sqrt([P_0(1)
squeeze(P(1,1,:))']),'--r')
xlabel('k')
ylabel('x')
legend('True State','Measurement', 'State estimate', '+3-
sigma level', '-3-sigma level','Location','southeast');
hold off
figure;
hold on
plot([1:N],X(2,2:end), 'g')
plot([0:N],[x_0(2) x(2,:)], 'b')
plot([0:N],[x_0(2) x(2,:)] + 3*sqrt([P_0(2)
squeeze(P(2,2,:))']), '--b')
plot([0:N],[x_0(2) x(2,:)] - 3*sqrt([P_0(2)
squeeze(P(2,2,:))']), '--b')
xlabel('k')
ylabel('v')
legend('True State','State estimate', '+3-sigma level', '-
3-sigma level','Location','southeast');
hold off

% Problem 2.c
Q= [0,0;0,2];
R= 2;
X= genLinearStateSequence(x_0, P_0, A, Q, N);
Y= genLinearMeasurementSequence(X, H, R);
Q_vals= [0.1,10,1.5];
for i= 1:length(Q_vals)
    Q(2,2)= Q_vals(i);
    [x, P]= kalmanFilter(Y, x_0, P_0, A, Q, H, R);
    figure;
    hold on
    plot([1:N],X(1,2:end),'g')
    plot([1:N],Y(1,:),'*r')
    plot([0:N],[x_0(1) x(1,:)],'b')
    plot([0:N],[x_0(1) x(1,:)]+3*sqrt([P_0(1)
squeeze(P(1,1,:))']),'--b')
```

```
    plot([0:N],[x_0(1) x(1,:)]-3*sqrt([P_0(1)
squeeze(P(1,1,:))']),'--b')
    xlabel('k')
    ylabel('x')
    legend('True State','Measurement', 'State estimate',
'+3-sigma level', '-3-sigma level','Location','southeast');
    hold off
    figure;
end
```

GEN LINEAR STATE SEQUENCE:

```
function X = genLinearStateSequence(x_0, P_0, A, Q, N)
%GENLINEARSTATESEQUENCE generates an N-long sequence of
states using a
%    Gaussian prior and a linear Gaussian process model
%
%Input:
%   x_0         [n x 1] Prior mean
%   P_0         [n x n] Prior covariance
%   A           [n x n] State transition matrix
%   Q           [n x n] Process noise covariance
%   N           [1 x 1] Number of states to generate
%
%Output:
%   X           [n x N+1] State vector sequence
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Your code here
X = zeros(size(x_0,1),N);
X(:,1) = (mvnrnd(x_0,P_0))';
a = (mvnrnd(zeros(size(Q,1),1),Q,N))';
for i=1:N
    X(:,i+1) = A*X(:,i) + a(:,i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end
```

GEN LINEAR MEASUREMENT SEQUENCE:

```
function Y = genLinearMeasurementSequence(X, H, R)
%GENLINEARMEASUREMENTSEQUENCE generates a sequence of
observations of the state
% sequence X using a linear measurement model. Measurement
noise is assumed to be
% zero mean and Gaussian.
%
%Input:
%   X           [n x N+1] State vector sequence. The k:th
state vector is X(:,k+1)
%   H           [m x n] Measurement matrix
%   R           [m x m] Measurement noise covariance
%
%Output:
%   Y           [m x N] Measurement sequence
%

% your code here
Y = zeros(size(H,1),size(X,2)-1);
r = (mvnrnd(zeros(size(R,1),1),R,size(X,2)-1))';
for i = 1:(size(X,2)-1)
    Y(:,i) = H*X(:,i+1)+r(:,i);
end
end
```

LINEAR PREDICTION:

```
function [x, P] = linearPrediction(x, P, A, Q)
%LINEARPREDICTION calculates mean and covariance of
predicted state
%   density using a liear Gaussian model.
%
%Input:
%   x           [n x 1] Prior mean
%   P           [n x n] Prior covariance
%   A           [n x n] State transition matrix
%   Q           [n x n] Process noise covariance
%
%Output:
%   x           [n x 1] predicted state mean
%   P           [n x n] predicted state covariance
%
```

```
% Your code here
x= A * x;
P= A * P * A' + Q;
end
```

LINEAR UPDATE:

```
function [x, P] = linearUpdate(x, P, y, H, R)
%LINEARPREDICTION calculates mean and covariance of
predicted state
%   density using a linear Gaussian model.
%
%Input:
%   x           [n x 1] Prior mean
%   P           [n x n] Prior covariance
%   y           [m x 1] Measurement
%   H           [m x n] Measurement model matrix
%   R           [m x m] Measurement noise covariance
%
%Output:
%   x           [n x 1] updated state mean
%   P           [n x n] updated state covariance
%

% Your code here
v = y -(H*x);
S = (H*P*H')+R;
K = P*H'/S;

x = x+(K*v);
P = P-(K*S*K');
end
```

KALMAN FILTER:

```
function [X, P] = kalmanFilter(Y, x_0, P_0, A, Q, H, R)
%KALMANFILTER Filters measurements sequence Y using a
Kalman filter.
%
%Input:
%   Y           [m x N] Measurement sequence
%   x_0         [n x 1] Prior mean
%   P_0         [n x n] Prior covariance
```

```matlab
%   A           [n x n] State transition matrix
%   Q           [n x n] Process noise covariance
%   H           [m x n] Measurement model matrix
%   R           [m x m] Measurement noise covariance
%
%Output:
%   x           [n x N] Estimated state vector sequence
%   P           [n x n x N] Filter error convariance
%

%% Parameters
N = size(Y,2);
n = length(x_0);
m = size(Y,1);

%% Data allocation
x = zeros(n,N);
P = zeros(n,n,N);

x(:,1)   = x_0;
P(:,:,1) = P_0;
for i= 1:N
    [x(:,i+1), P(:,:,i+1)] =
linearPrediction(x(:,i),P(:,:,i),A,Q);
    [x(:,i+1), P(:,:,i+1)] =
linearUpdate(x(:,i+1),P(:,:,i+1),Y(:,i),H,R);
end
X= x(:,2:end);
P= P(:,:,2:end);
end
```