```
%% ----- Exercise 3 - Frequency and Kalman filters (part B) -----
% Version: 2022
% Course: TME 192 Active Safety
%         Chalmers
% Author: Alberto Morando (morando@chalmers.se)
%         Alexander Rasch (arasch@chalmers.se)
%         Marco Dozza (dozza@chalmers.se)
%
% Group: [PLEASE FILL OUT]
%

clear all
close all
clc

rng(23) % for reproducibility

%% SETTINGS
f = 25; % sampling frequency [Hz]
dt = 1/f; % sampling period [s]
duration = 60; % Simulation duration [s]
T = duration/dt; % Time series

%% SYSTEM DEFINITION
% The system we want to estimate consist of the kinematic
% equations of motion of a vehicle moving in one dimension:
%
% pos_x_t = pos_x_t_minus_1 + v_x_t_minus_1 * dt + 1/2 a_x_t_minus_1 * dt^2
% vel_x_t = v_x_t_minus_1 + a_x_t_minus_1 * dt
%
% The states we want to estimate are position (pos_x) and velocity (v_x), given an
acceleration input (a_x).
% Thus, the state space representation is
%
% x_t = A * x_t_minus_1 + B * u
% y_t = H * x_t_minus_1
%
% The equations above represent an ideal model. We can add process noise
% (w) and measurement noise (z):
% x_t = A * x_t_minus_1 + B * u + w
% z_t = H * x_t_minus_1 + v

% State transition matrix
A = [1 dt;
     0 1];

% Input matrix
B = [dt^2/2;
     dt];

% Measurement matrix (we measure only position)
H = [1, 0];

% Measurement error covariance (how much variance there is in each measurement
beacuse of the sensor noise)
sensor_noise = 5; % Position sensor noise (standard deviation) [m]
R = sensor_noise^2; % Sensor noise covariance [m2]

% Q : process noise covariance (assumed constant over time)
```

```
% The process variance is how much error there is in the process model.
% Our model is imperfect. Road conditions and vehicle parameters are random
% variables
u_noise = 0.05; % Acceleration noise (standard deviation) [m/s2]

Q = [B(1)^2 * u_noise^2,  (B(1) * u_noise) * (B(2) * u_noise);
    (B(1) * u_noise) * (B(2) * u_noise), B(2)^2 * u_noise^2];  % see paper from Dan
Simon.

%% SIMULATE THE VEHICLE MOTION
n_states = size(A, 1); % We are estimating two states: position_x and velocity_x

x = nan(n_states, T);
x(:, 1) = 0; % initial conditions: we start from pos_x = 0 m and v_x = 0 m/s

u = ones(1, T) * 0.3; % commanded acceleration without noise, constant across time

% Initialize measurement timeseries vector (which are noisy because of the quality
of the sensor)
z = zeros(1, T);

% Simulate the system across time
for k = 2:T

    % STATE EQUATION
    x(:, k) = A * x(:, k-1) + B * u(k-1);
    % We add a process noise. The car motion is perturbed by the road bumps and
gust of winds
    w = mvnrnd([0, 0], Q)';
    x(:, k) = x(:, k) + w;

    % OUTPUT EQUATION
    % Teh measurement are corrupted by the sensor noise
    v = mvnrnd(0, R)';
    z(:, k) = H * x(:, k) + v;

end

%% KALMAN FILTER (This section needs to be completed by the students)

% Initialize the vector to store the state estimate timeseries
x_hat = nan(n_states, T);

% Guess for the initial conditions
% we start from pos_x = 0 m and v_x = 0 m/s ...
x_hat(:, 1) = 0;

% ... but we are not very confident on this guess. Thus, we initialize the
% error covariance matrix P with a wide standard deviation.
P = eye(2).*10^2;

% Apply the filter iteratively
for k = 2 : T
    % ======== YOUR CODE HERE ========

    % *** PREDICT ***
    % (1) Project the state ahead
    x_hat(:, k) = A*x_hat(:, k-1) + B*u( k-1);
```

```
    % (2) Project the error covariance matrix ahead.
    % The command to transpose a matrix is the single quotation mark `'`
    P = (A*P*A') + Q;

    % *** CORRECT ***
    % (1) Compute the Kalman gain.
    K = P*H'*inv((H*P*H') + R);

    % (2) Update estimate with the measurement z
    x_hat(:, k) = x_hat(:, k) + (K*(z(:,k) - (H*x_hat(: , k))));

    % (3) Updated the error covariance matrix
    % (the identity matrix I can be calculated with `eye`)
    P = (eye(n_states)-K*H)*P;

end

%% PLOT THE RESULTS
n_subplots = 4;

figure('Name', 'Kalman', 'NumberTitle', 'off', 'position', [50, 50, 600, 800])
subplot(n_subplots, 1, 1)
plot([1:T] * dt, x(1,:), [1:T] * dt, z, [1:T] * dt, x_hat(1,:));
ylabel('Position (m)');
title('Vehicle Position');
legend('True', 'Measured', 'Estimated');

subplot(n_subplots, 1, 2)
plot([1:T] * dt, x(1,:)-z, [1:T] * dt, x(1,:) - x_hat(1,:));
ylabel('Error (m)');
title('Position error');
legend('Measurement error', 'Estimation error');

subplot(n_subplots, 1, 3)
plot([1:T] * dt, x(2, :), [1:T] * dt, x_hat(2, :));
ylabel('Velocity (m/s)');
title('Velocity');
legend('True', 'Estimated');

subplot(n_subplots, 1, 4)
plot([1:T] * dt, x(2, :) - x_hat(2, :));
xlabel('Time (s)');
ylabel('Error (m/s)');
title('Velocity error');
```