[F21] Sensing, Perception and Actuation
Assignment 3

Walid Shaker

w.shaker@innopolis.university

Task: Multidimensional Kalman filter with sensor fusion

The idea of this task is to get data from two sensors which are GPS and Accelerometer using mobile phone, then fuse measurements of the sensors in order to apply Kalman Filter and predict the trajectories.

The implementation of this task goes as follows:

1. Get the data from 2 sensors

In this section, GPS and accelerometer data are spilt into x and y. In addition, time step and initial GPS position values for x, y are determined.

```
% read the data
acc_data = xlsread('acc_data.xlsx');
GPS_data = xlsread('GPS_data.xlsx');

% time vector
time = acc_data(:,1);
% get timestamp
dt = (acc_data(2,1)-acc_data(1,1)) * 1e-3;

% get intial position values from GPS data
xo = GPS_data(1,1);
yo = GPS_data(1,2);

% get gps data in two seprate vectors one for X and one for Y
X_GPS = GPS_data(:,1);
Y_GPS = GPS_data(:,2);

% get linear acc data in two seprate vectors one for X and one for Y
X_acc = acc_data(:,2);
Y_acc = acc_data(:,3);
```

2. Design constant velocity model

The process state vector selected for this task is $x = [x, \quad \dot{x}, \quad y, \quad \dot{y}]^T$

The equations that describe the constant velocity are

$$x = x_0 + \dot{x}dt$$

$$y = y_0 + \dot{y}dt$$

How to find initial $\dot{x}$ and $\dot{y}$?

First GPS data are linearly fit with time in order to get the slope and intercept, which we will use later to estimate x_model and y_model. The slope for each direction obtained is the actually the velocity for this direction.

```
% fit line to data using polyfit
c_x = polyfit(time,X_GPS,1);
c_y = polyfit(time,Y_GPS,1);

% fitting x and y models from obtained slope and intercept
x_model = c_x(1).*time + c_x(2);
y_model = c_y(1).*time + c_y(2);
```

The equations that can describe this fitting are

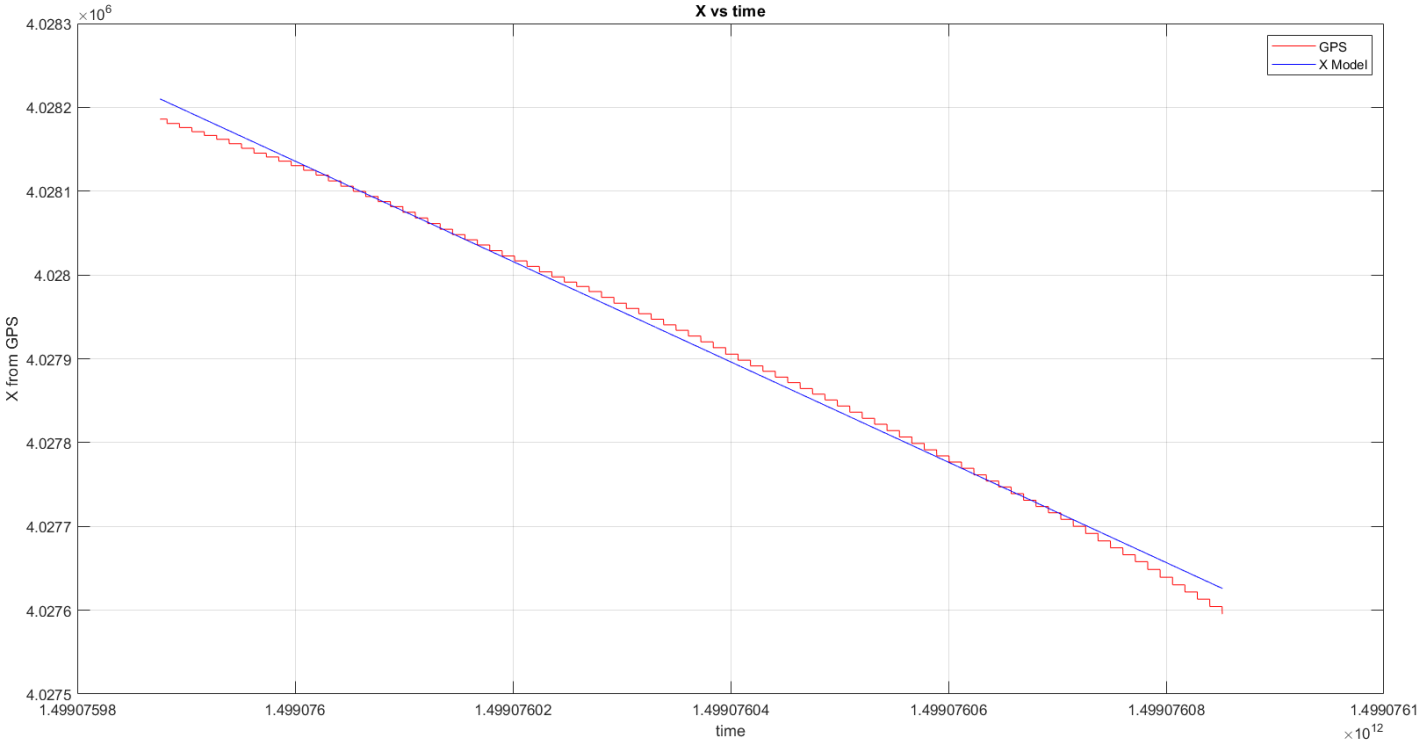
```
Evaluated equation x_model = x_slope * t + x_intercept
 x = -0.005987*t + 8979057060.2673
Evaluated equation y_model = y_slope * t + y_intercept
 y = -0.017342*t + 25997435947.506
```

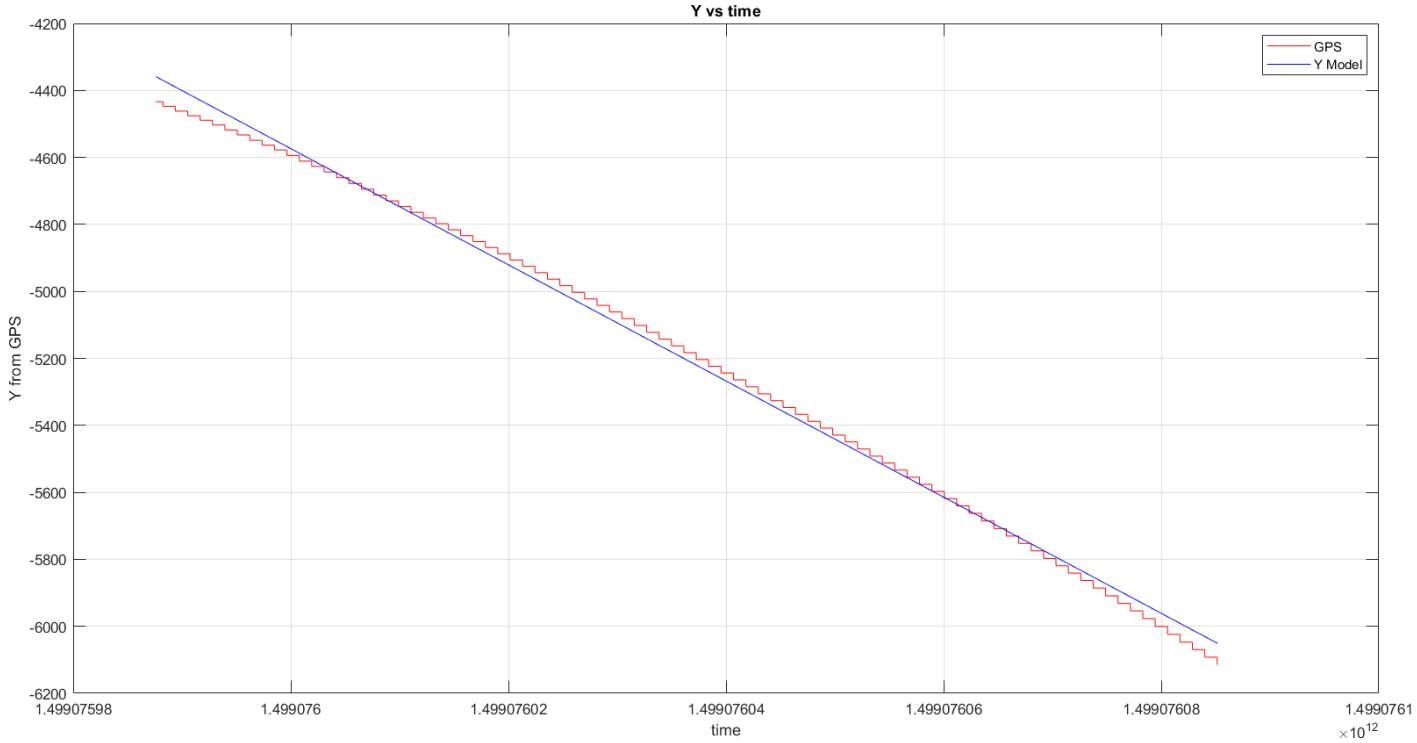
From equations, the slopes are v_x and v_y.

```
%x and y velocities for model are
v_x = c_x(1);
v_y = c_y(1);
```

Plotting GPS data and the fitting model for x and y:

```
figure;
plot(time,X_GPS,'r',time,x_model,'b');
title('X vs time');
xlabel('time');
ylabel('X from GPS');
legend('GPS','X Model');
grid;

figure;
plot(time,Y_GPS,'r',time,y_model,'b');
title('Y vs time');
xlabel('time');
ylabel('Y from GPS');
legend('GPS','Y Model');
grid;
```


X vs time
GPS
X Model
X from GPS
time
×10^6
4.0283
4.0282
4.0281
4.028
4.0279
4.0278
4.0277
4.0276
4.0275
1.49907598
1.499076
1.49907602
1.49907604
1.49907606
1.49907608
1.4990761
×10^12


Y vs time
GPS
Y Model
Y from GPS
time
-4200
-4400
-4600
-4800
-5000
-5200
-5400
-5600
-5800
-6000
-6200
1.49907598
1.499076
1.49907602
1.49907604
1.49907606
1.49907608
1.4990761
×10^12

After that, data from accelerometer are integrated into position as follows:

```
% inital values for velocity and position in X
vel_x(1) = v_x + X_acc(1)*dt;
X_pos(1) = xo + vel_x(1)*dt;

% inital values for velocity and position Y
vel_y(1) = v_y + Y_acc(1)*dt;
Y_pos(1) = yo + vel_y(1)*dt;

for i = 2:length(X_acc)
    vel_x(i) = vel_x(i-1) + X_acc(i-1)*dt;
    X_pos(i) = X_pos(i-1) + vel_x(i)*dt;

    vel_y(i) = vel_y(i-1) + Y_acc(i-1)*dt;
    Y_pos(i) = Y_pos(i-1) + vel_y(i)*dt;
end
```

Now, our measurement vector is $z = [xgps, \quad xpos, \quad ygps, \quad ypos]^T$

where $xpos$ and $ypos$ are the result of the integration of $\ddot{x}$ and $\ddot{y}$ obtained from the accelerometer.

3. Design matrices $\emptyset$, H, Q, R, P

$$x_{k+1} = \Phi_k x_k + w_k$$

$$\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + \omega_k$$

So, transition matrix $\emptyset = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$z_k = H_k x_k + v_k$$

$$\begin{bmatrix} xgps \\ xpos \\ ygps \\ ypos \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + v_k$$

So, design matrix $H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

To design process noise covariance matrix Q , it is assumed to contain a very low noise since it represents white noise.

$$Q = \begin{bmatrix} {\sigma^2}_x & 0 & 0 & 0 \\ 0 & {\sigma^2}_{\dot{x}} & 0 & 0 \\ 0 & 0 & {\sigma^2}_y & 0 \\ 0 & 0 & 01 & {\sigma^2}_{\dot{y}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

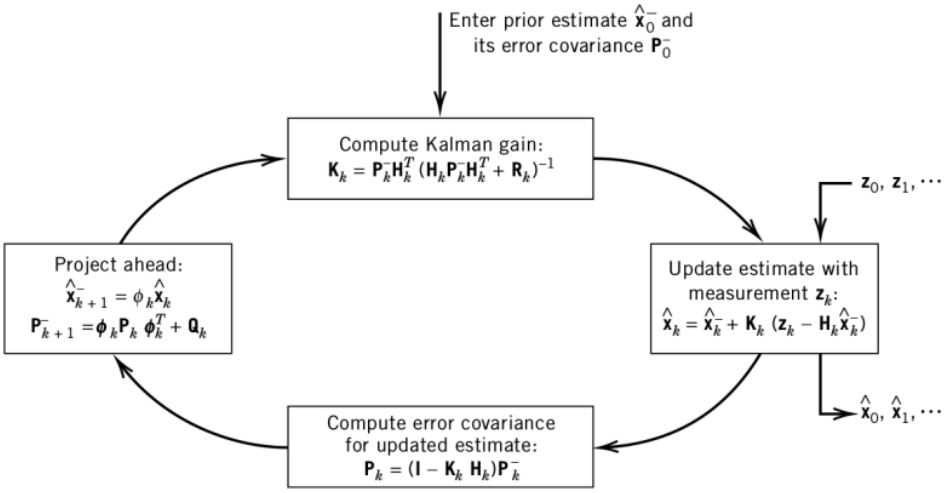
For measurement noise covariance matrix R , GPS is considered to have a std of $5m$ so, the variance is 25 but for the accelerometer, it is assumed to have a higher variance (400) since the data come from the accelerometer are usually very noisy.

$$R = \begin{bmatrix} {\sigma^2}_{xgps} & 0 & 0 & 0 \\ 0 & {\sigma^2}_{xacc} & 0 & 0 \\ 0 & 0 & {\sigma^2}_{ygps} & 0 \\ 0 & 0 & 01 & {\sigma^2}_{yacc} \end{bmatrix} = \begin{bmatrix} 25 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 400 \end{bmatrix}$$

Finally, error covariance matrix P is initialized with high values and it is noticed that it will diverge eventually.

$$P = \begin{bmatrix} 200 & 0 & 0 & 0 \\ 0 & 200 & 0 & 0 \\ 0 & 0 & 200 & 0 \\ 0 & 0 & 0 & 200 \end{bmatrix}$$

4. Kalman Filter loop



Using Kalman Filter loop notation presented above, MATLAB algorithm is implemented as follows:

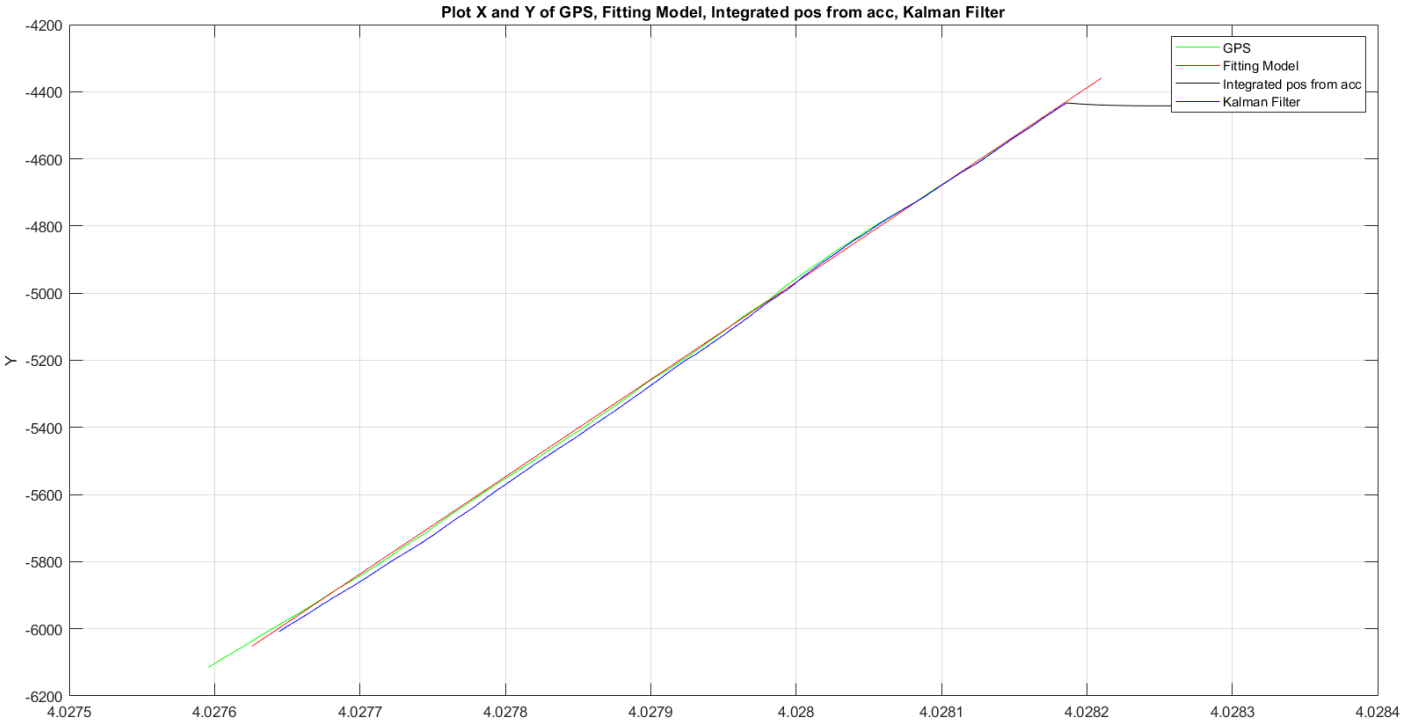
```
% inital value for X state vector
Xo = [xo; v_x; yo; v_y];
X(:,1) = Xo;

for i=1:length(Y_acc)
   Zk = [X_GPS(i); X_pos(i); Y_GPS(i); Y_pos(i)];

   % project ahead
   Xp = phi* X(:,i);
   Pp = phi * P * phi' + Q;

   % compute kalman gain
   K = Pp * H' * inv(H * Pp * H' + R);

   % update step
   X(:,i+1) = Xp + K * (Zk - H * Xp);

   % compute error covariance for updated estimate
   P = (eye(4) - K*H) * Pp * (eye(4) - K*H)' + K* R *K';

end
```

As expected, error covariance matrix P is diverged at the end of the loop.

```
Final error covariance matrix P
    4.5626    4.3551         0         0
    4.3551  104.7654         0         0
         0         0    4.5626    4.3551
         0         0    4.3551  104.7654
```

5. Graph plotting

In this section, GPS data, accelerometer data after integration, and Kalman Filter result for estimated trajectory are compared by plotting. Also, the linearly fit model implemented in section (3.2) is added to the graph.



6. Discussion

It has been proven that Multidimensional Kalman Filter could handle the noisy data from the accelerometer by adjusting proper variances in R matrix. When trying to reduce the variance of the accelerometer data from 400 to 100, a dramatic decrease in the Kalman Filter fitting has been noticed. The results obtained are quite good as the error covariance matrix P diverged finally.