Sensing, Perception and Actuation

HW4

Submitted by: **Mohamed Moustafa Elsayed Ahmed**

Email: **o.ahmed@innopolis.university**

Program: **Robotics**

Date:

21/10/2020

1. Calculating u*

a) Our SISO system & Parameters:

Consider the following SISO system,

$$x_{k+1} = \begin{bmatrix} 0.7 & 0.5 & 0 \\ -0.5 & 0.7 & 0 \\ 0 & 0 & 0.9 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} u_k, \quad x_o = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \end{bmatrix}$$

$$y_k = [0 \quad -1 \quad 1]x_k + 0.5u_k$$

So our Parameters are:

```
% A matrix
A=[0.7 0.5 0;
   -0.5 0.7 0;
   0 0 0.9];

% B vector
B=[1;1;1];

% C vector
C=[0 -1 1];

%D=0.5
D=0.5;

% N
N=0:19;

% y ref
y_ref=[1 0 0 4 4 1 0 2 0 1 3 4 4 2 1 2 4 3 2 2]';

% xo
xo=[0.1;0.2;0.3];
```

b) Calculating u*

To calculate u* we can use this formula:

$$u^* = (Q^T Q)^{-1} Q^T (y^{ref} - \phi x_0)$$

But first we need to calculate Q & phi in order to calculate u*:

Where Q can be computed as following:

N is our horizon

$$Q_k = CA^{k-1}B.$$

$$\underbrace{\begin{bmatrix} Q_0 & \dots & \dots & \dots & 0 \\ Q_1 & Q_0 & \dots & \dots & 0 \\ Q_2 & Q_1 & Q_0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ Q_{N-1} & Q_{N-2} & \dots & \dots & Q_0 \end{bmatrix}}_{Q}.$$

Phi can be computed as following:

N is our horizon

$$\underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{t+N-1} \end{bmatrix}}_{\phi}$$

c) Plot our data:

Now we can get our y and compare it to y_ref through the following:

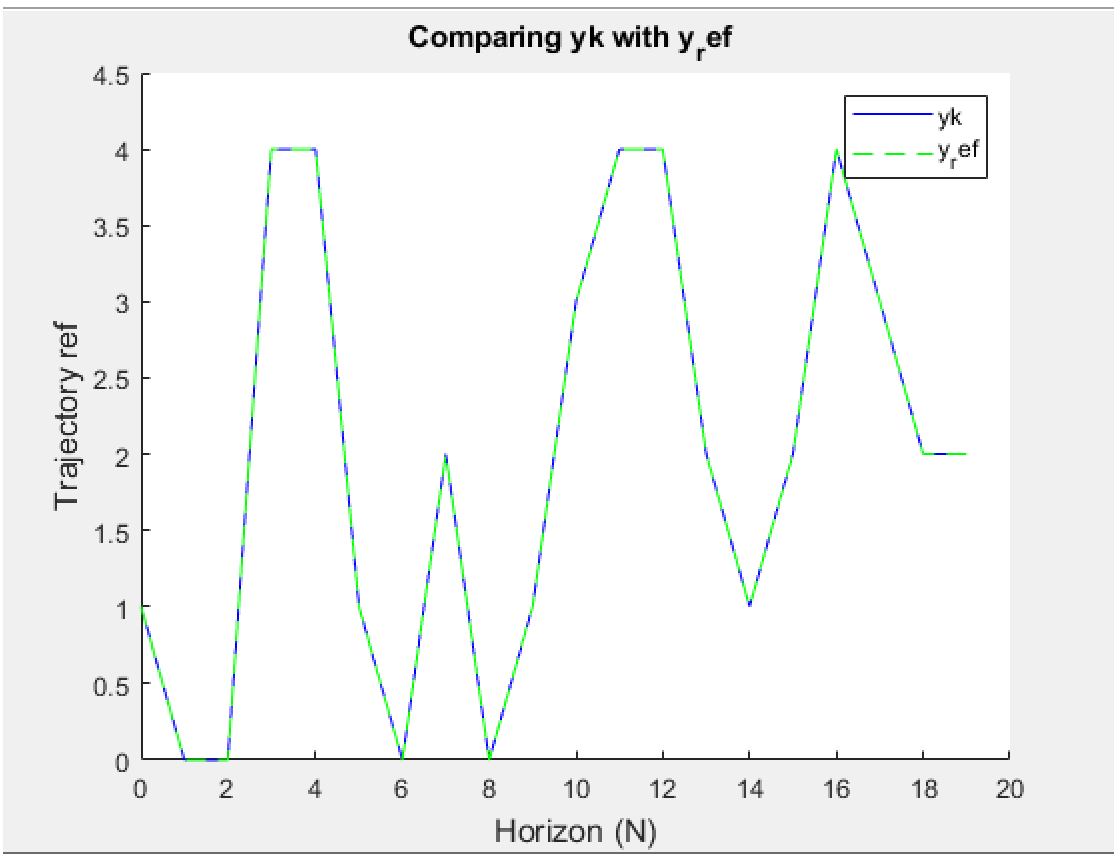
$$\underbrace{\begin{bmatrix} y_t \\ y_{t+1} \\ \vdots \\ y_{t+N-1} \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{t+N-1} \end{bmatrix}}_{\phi} x_0 + \underbrace{\begin{bmatrix} Q_0 & \dots & \dots & \dots & 0 \\ Q_1 & Q_0 & \dots & \dots & 0 \\ Q_2 & Q_1 & Q_0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ Q_{N-1} & Q_{N-2} & \dots & \dots & Q_0 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} u_t \\ u_{t+1} \\ \vdots \\ u_{N-1} \end{bmatrix}}_{u}$$

Now we have computed phi , Q & u* and we are ready to use them to get y

But for u* I will use to approaches , first to take u* as we get exactly from equation

Second to limit the u* and see what happens on the plot

First approach u* as it is:



As we can see yk got same result as y_ref as expected from u*

Second u* after limit its values:

- u* is our control input so it is computed without taking into consideration control and physical world actuation limits maybe it is so now I will limits the value of u* to not exceed 100 and don't be less than -100
- so : -100 < uk < 100
  this will be our graph in case we limit the u* system input which will be more realistic in real life situation:

u_opt

Editor - hw4.m

u_opt

20x1 double

	1
1	1.8000
2	-0.3600
3	-3.0500
4	3.2998
5	7.0750
6	0.5478
7	-12.2814
8	-17.9149
9	-9.2422
10	33.1229
11	82.2286
12	62.2680
13	-91.5095
14	-313.8993
15	-323.5513
16	201.9898
17	1.1423e+03
18	1.4992e+03
19	-160.0734
20	-3.9266e+03

u_limit

Editor - hw4.m

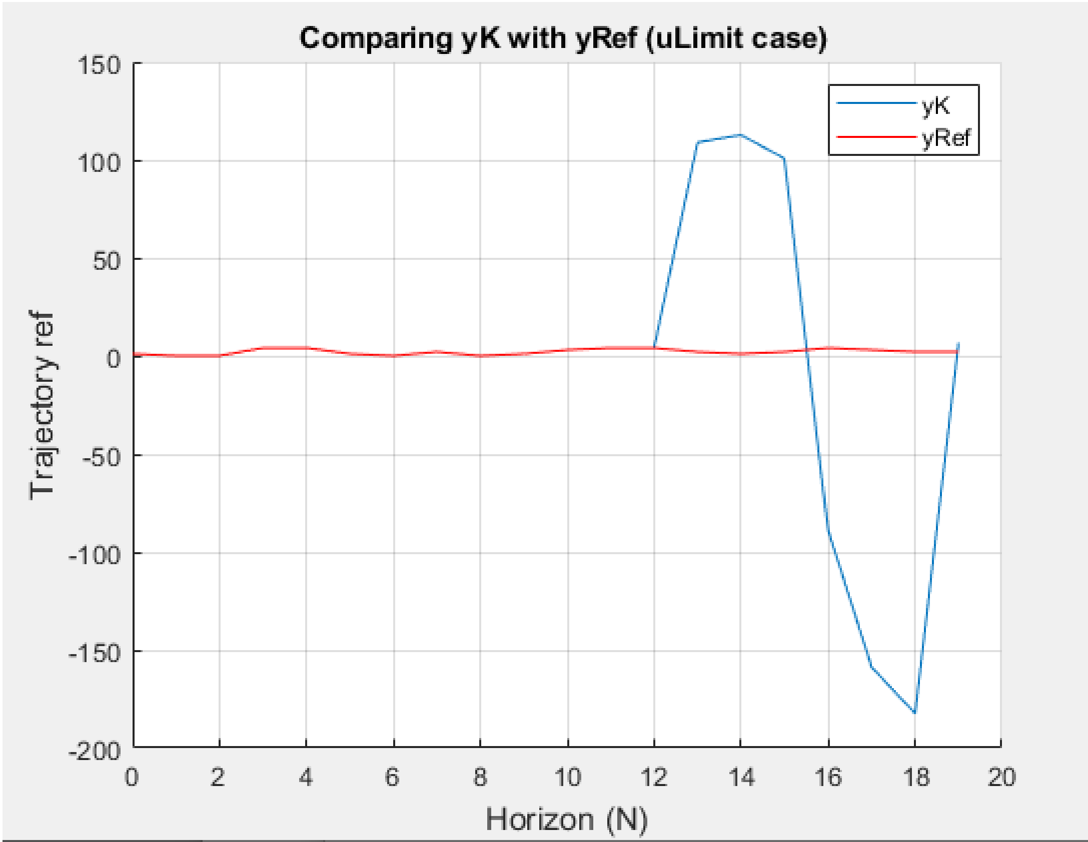
u_limit

20x1 double

	1
1	1.8000
2	-0.3600
3	-3.0500
4	3.2998
5	7.0750
6	0.5478
7	-12.2814
8	-17.9149
9	-9.2422
10	33.1229
11	82.2286
12	62.2680
13	-91.5095
14	-100
15	-100
16	100
17	100
18	100
19	-100
20	-100

we limited our control to not exceed 100 and not be less -100
But for lase steps we can see that u* change hugely from u_limit
Which indicates that our yk will get distrurbaed much at the end and wouldnn't be equal to y_ref

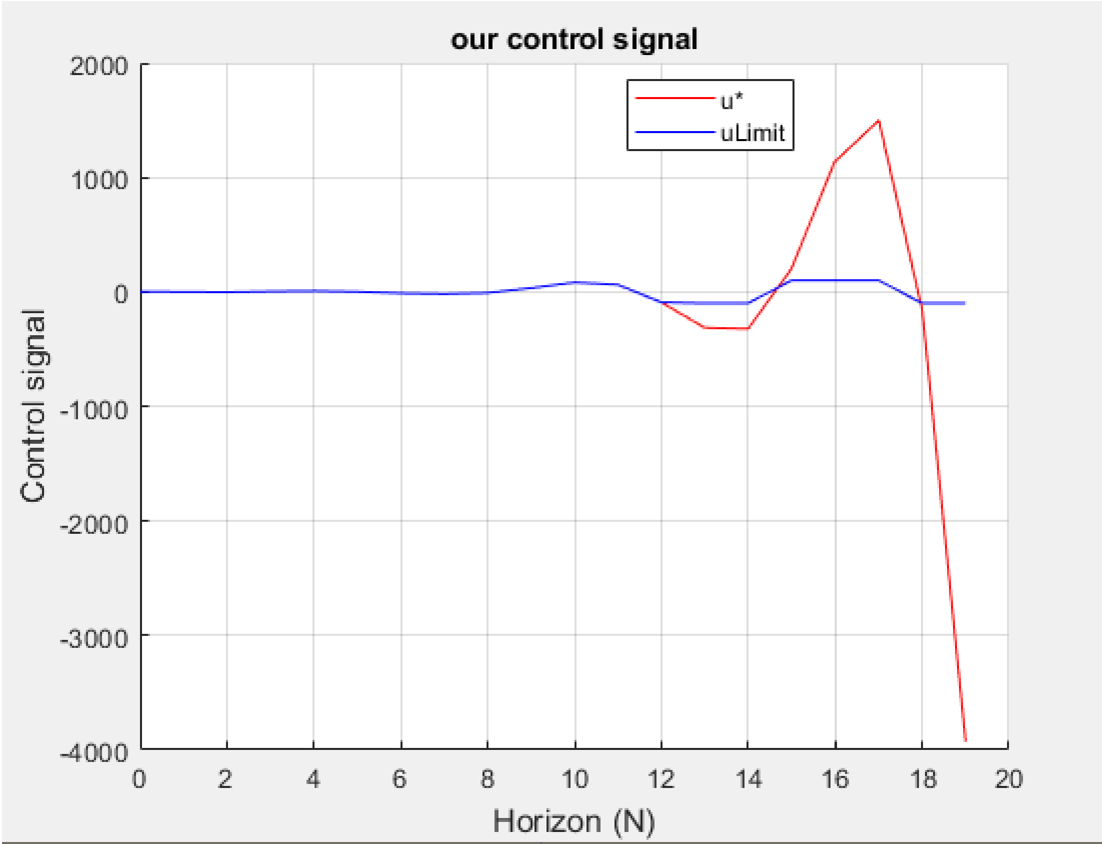
Our plot:



Exactly as we expect

Which means u* may not be the best solution for real life application and some other techniques may be used to achieve more robust and soft controller

Our control signal:



As we can see u* may be very hard to achieve in real life

And uLimit didn't achieve our desired yRef values as we see in last figure

Which may lead us to use another controlling technique like PID control for example

2. (Multidimensional Kalman Filter)

a) Our Measurements:

$$z_k = x_k + [\mathcal{N}(0,\,1.7),\ \mathcal{N}(0,\,1.0),\ \mathcal{N}(0,\,1.8)]^T$$

b) Our Process model:

$$x_{k+1} = \begin{bmatrix} 0.7 & 0.5 & 0 \\ -0.5 & 0.7 & 0 \\ 0 & 0 & 0.9 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} u_k, \quad x_o = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \end{bmatrix}$$

c) Our Design Parameters:

```
% phi matrix (Transition State Matrix)
phi=[0.7 0.5 0;
   -0.5 0.7 0;
   0 0 0.9];

% Q matrix (White Noise Covariance matrix)
q=    [0.5 0 0;
       0 0.5 0;
       0 0 0.5];
% H matrix
H=    [1 0 0;
       0 1 0;
       0 0 1];
% R matrix (Sensor Covariance matrix)
R=   [1.7 0 0;
      0 1 0;
      0 0 1.8];

% P matrix
P=    [1 0 0;
       0 1 0;
       0 0 1];
```

Phi = A; both act as Transition state matrix

Q: is the covariance of model will assume it as 0.5

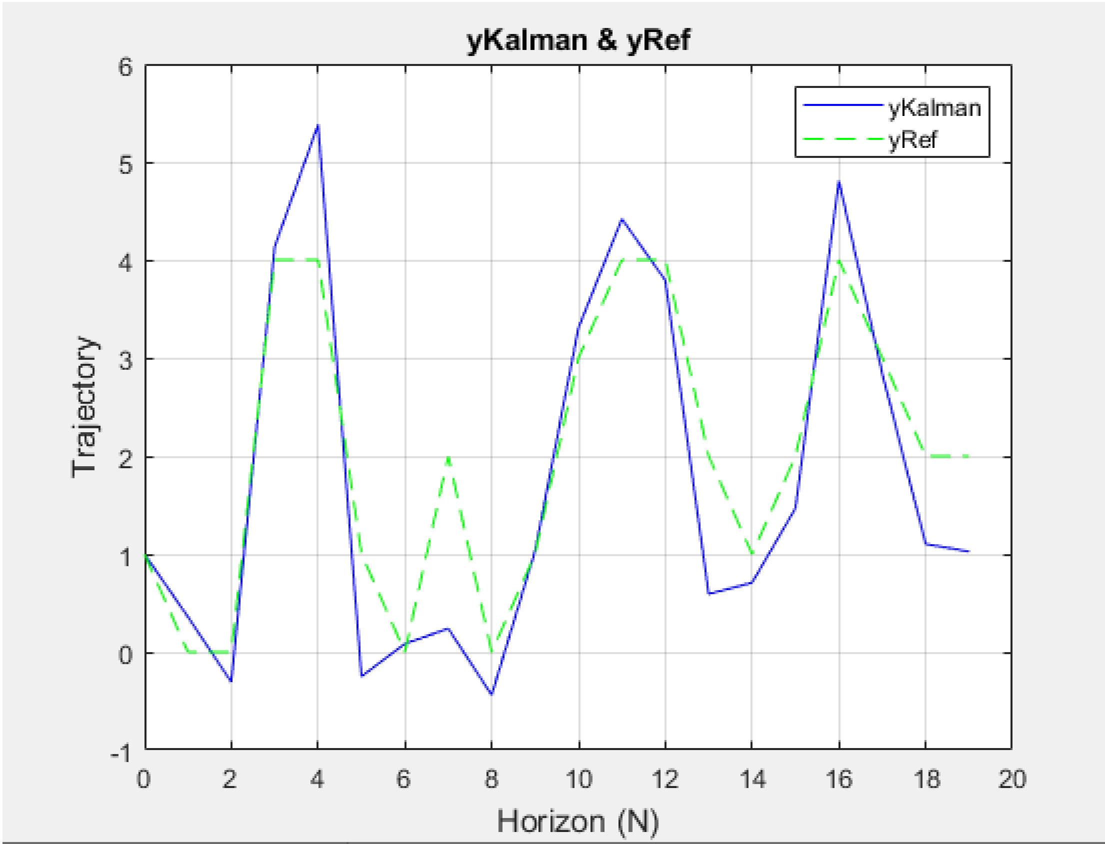
H: Conversion matrix (I need sensor measurements as they are)

R: is the Covariance of sensor reading will make it as the given variances for sensor

P: we can initialize P with any value it will converge to a specific value eventually

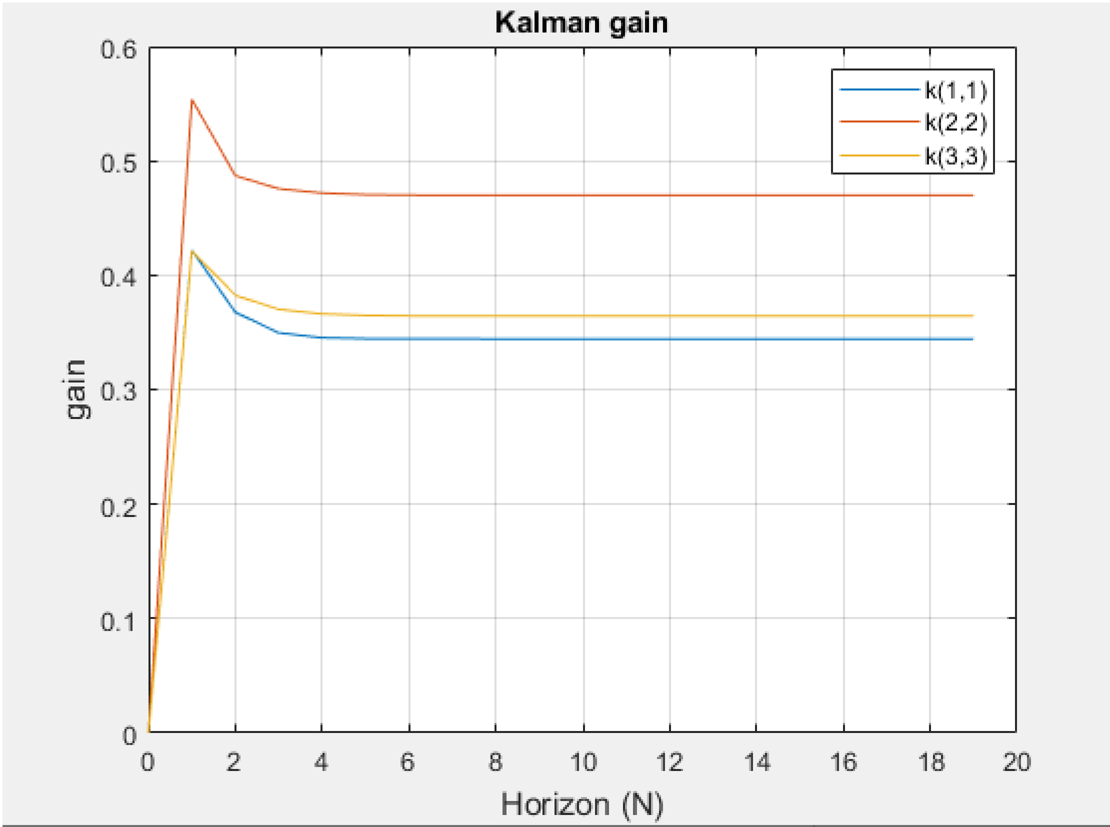
d) Plot our data:

After we the loop of kalman filter here's the data we obtained:



As we can see our kalman filter performs very well and close to yRef

3. (Plot Kalman gain)



As we can see K reaches 0.35 & 0.48 so our kalman filter depends on both model and sensors with closely equal weights … but depends on model more as it has less Covariance than sensor

github link:

https://github.com/Mohamed-Moustafa/SPA_HW4.git