Sladka Karla

25.4.2018

Process Control

Compulsory exercise 6

1.      **Kalman Filter**:

a.       Implement, in Matlab, a simulator of the liquid tank (including random process disturbances and measurement noise) and the Kalman Filter for the (simulated) tank presented in example 18.2 in [the text-book](http://techteach.no/publications/books/advanced_dynamics_and_control/advanced_dynamics_control_textbook.pdf). ([Here](http://techteach.no/simview/kalmanfilter/index.php) is a SimView simulator of this system.) Implement the native algorithm of the Kalman gain (i.e. do not use any inbuilt Matlab-function for calculating K). Run some illustrative simulations (e.g. try various values of the Q matrix, and observe the effect on the estimates).

Mathematical model of mass balance

the density can be cancelled than the model is

Outflow we assume as constant

Definition of the state space model

Then

The discretization of continue state space model with Ts

The process disturbance auto-covariance , where

And R is measurement variance.

Output measurement is level

The predicate level measurement

Innovation variable

The corrected state estimation

The predicate state estimate

The information to Calculate the steady state Kalman Filter Gain Ks are

Identity matrix

Initially adjusted Q and R.

The matab code for calculation the steady state Kalman Filter gain

%tank parameters

Ts=0.1;

A\_tank=0.1;

Kp=0.002;

R=0.000001;

Q=[0.01,0;0,0.0001];

G=[1,0;0,1];

I=[1,0;0,1];

%setting parameters

L= 10000;

Pp = [0.25,0;0,1];

t= 0:0.1:999.9;

u= 3\*ones(L,1);

e=ones(L,1);

x1=zeros(L,1);

x2=zeros(L,1);

x1p=zeros(L,1);

x2p=zeros(L,1);

x1c=zeros(L,1);

x2c=zeros(L,1);

K=zeros(2,1);

v=0.000001+0.000001.\*randn(L,1);

w = 0.000001+0.000001.\*randn(L,2);

ik=1;

ek=1;

%State space model

A\_con=[0,-1/A\_tank;0,0];

B\_con=[Kp/A\_tank;0];

C\_con=[1,0];

D\_con=[0];

sys\_con=ss(A\_con,B\_con,C\_con,D\_con);

%discretization

sys\_dis=c2d(sys\_con, Ts);

A= sys\_dis.a;

B=sys\_dis.b;

C=sys\_dis.c;

D=sys\_dis.d;

%Kalman filter gain manually

%cycle while is run until the error will be less than 0.0005

while abs(e)>0.00005

x1(1)=0.5;

%kalman gain

K= Pp\*C'\*inv(C\*Pp\*C'+R);

Pc = (I-K\*C)\*Pp;

Pp=A\*Pc\*A'+G\*Q\*G';

%determine of error

y(ik)=x1(ik)+v(ik,1);

yp(ik)= x1p(ik);

e=y(ik)-yp(ik)

%state correction

x1c(ik)=x1p(ik)+K(1)\*e;

x2c(ik)=x2p(ik)+K(2)\*e;

%state estimaton

x1p(ik+1)=x1c(ik)+(Ts/A\_tank)\*(Kp\*u(ik)-x2c(ik));

x2p(ik+1)=x2c(ik);

%step +1

ik=ik+1;

end

The result of the Kalman filter gain is

Kalman filter estimation

%Kalman filter

%setting parameters

e=1;

x1=zeros(L,1);

x2=0.0006\*ones(L,1);

x1p=zeros(L,1);

x2p=zeros(L,1);

x1c=zeros(L,1);

x2c=zeros(L,1);

y=zeros(L,1);

yp=zeros(L,1);

for i=1:10000

x1(1)=0.5;

x2(1)=0.006;

x1p(1)=x1(1);

x2p(1)=0;

%determine of error

y(i)=x1(i);

yp(i)= x1p(i);

e(i)=y(i)-yp(i);

%state correction

x1c(i)=x1p(i)+K(1)\*e(i);

x2c(i)=x2p(i)+K(2)\*e(i);

%state prediction estimaton

x1p(i+1)=x1c(i)+(Ts/A\_tank)\*(Kp\*u(i)-x2c(i));

x2p(i+1)=x2c(i);

%state space model

x1(i+1)= x1(i)+(Ts/A\_tank)\*(Kp\*u(i)-x2(i))+w(i,1);

x2(i+1)= x2(i)+w(i,2);

Result are different what I expect. I also created simulation (implementationKF.mdl) and the results are the same.

A close up of a map

Description generated with high confidence

b.      Assume that the outflow is through a valve which is placed h0 [m] below the outlet of the tank, with the following relation between Fout and the liquid level:  
  
Fout = Kv\*f(z)\*sqrt[p/G]  
  
where Kv is valve constant, f is valve function (known), z is valve opening [%] (known), p [Pa] is pressure drop across the valve which assumed being equal to the hydrostatic pressure due to liquid level above the valve, rho is liquid density, g [m/s^2] is the gravity, and G [1] is specific density. Fout is not measured (and shall not be estimated either). Formulate the mathematical model (a continuous-time state-space model) which can be used to estimate Kv with a Kalman filter. Is the state-space model linear of nonlinear? (You are not required to implement this Kalman Filter.)

The valve relation where

Then the mass balance is

Definition of the state space model

Then

The function of the valve is usually nonlinear.

2.      **Moving Horizon Estimation (MHE):** In Example 1.10 in the lecture notes about optimization, the process disturbance *d* is estimated. Modify the Matlab script presented in the example (the script is also available from the course home page) so that the gain *K* is estimated in stead of the *d*. (You can give *d* any fixed value you want.) Is your implementation successful?

The implementation wasn’t successful. I change x3 as K instead d. The d was a fixed value. There is multiply K\*u both are not fix values and I think that makes problems. It stops on step5.

A close up of a piece of paper

Description generated with very high confidence

3.      **Model-predictive Control (MPC)**: [Here](http://techteach.no/fag/process_control_nmbu_2018/mpc/script_mpc_airheater_160418.m) is a Matlab script implementing MPC for a simulated [air heater](http://home.usn.no/finnh/air_heater/). (The model of the air heater, from control signal acting on the heater to measured tube outlet temperature, is basically time constant with time delay.) The script includes an Observer for estimating an “input disturbance”, which is a disturbance added to the control signal. One purpose of including such an estimated disturbance in MPC is to ensure zero steady-state control error despite model errors in the underlying model of the MPC. Note that in the present script, the Observer uses the control signal *after* the time-delay as control signal (or model input signal). In this way, the time-delay is not a part of the Oberver, which simplifies the implementation of the Observer.

a.       Draw a block diagram which shows how the various parts (simulator; Observer; MPC) of the system implemented in the script, are interconnected.)

b.      Run the script. Does it seem to work?

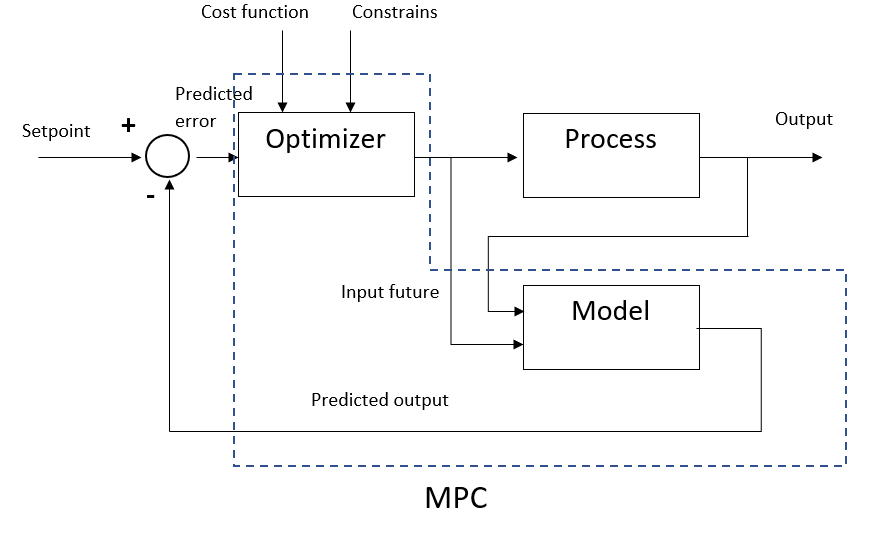
c.       Open the script in Matlab, and try to understand it.

d.      Play with some of the settings (you should decide which), and observe the impact of the changes on the behaviour of the MPC control system.

e.       Implement a model error in the gain of the model. Check by simulations if using the Observer ensures steady state error despite this model error.

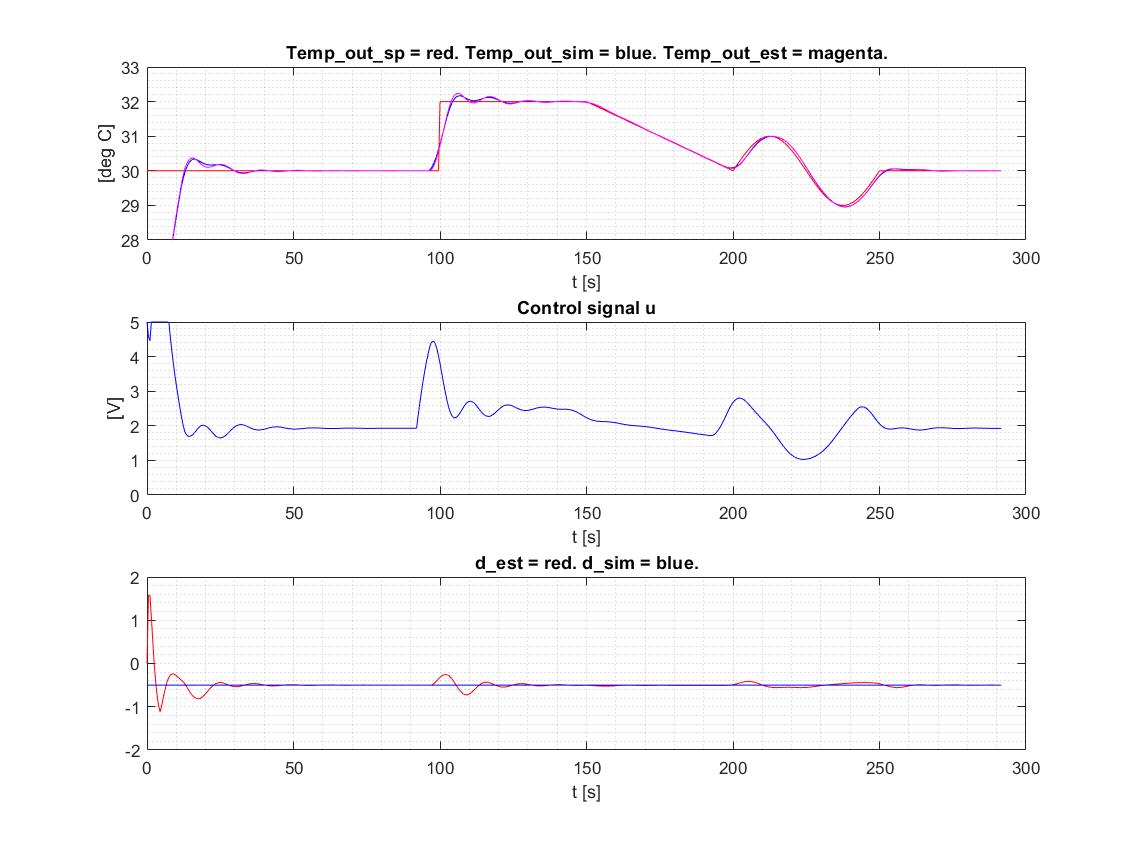
f.       Voluntary: Replace the Observer in the script by a Kalman Filter which you should implement from scratch (that is, without using an inbuilt function in Matlab to calculate the Kalman gain). Is the system still working?

1. **MPC scheme**



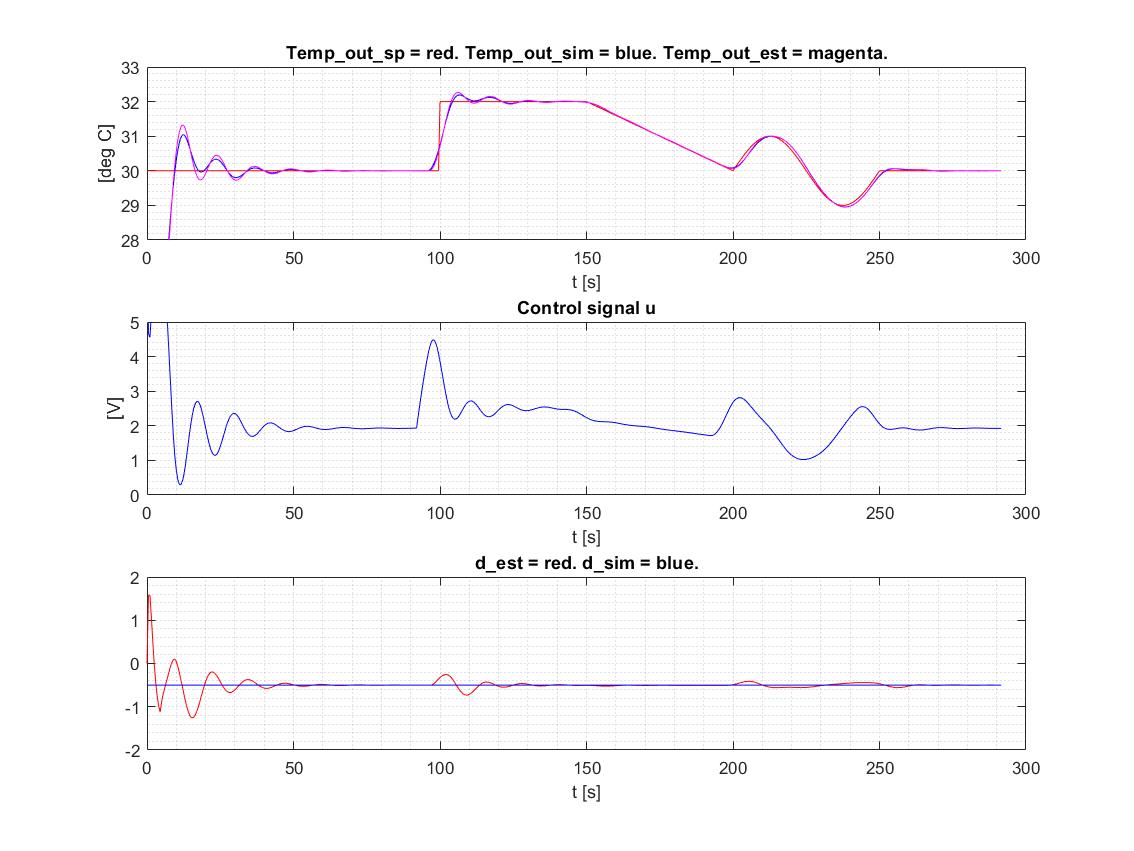
1. **The working script**

The script seems to work. The temperature is following the setpoint with the MPC control very precisely. The result is on the following picture.

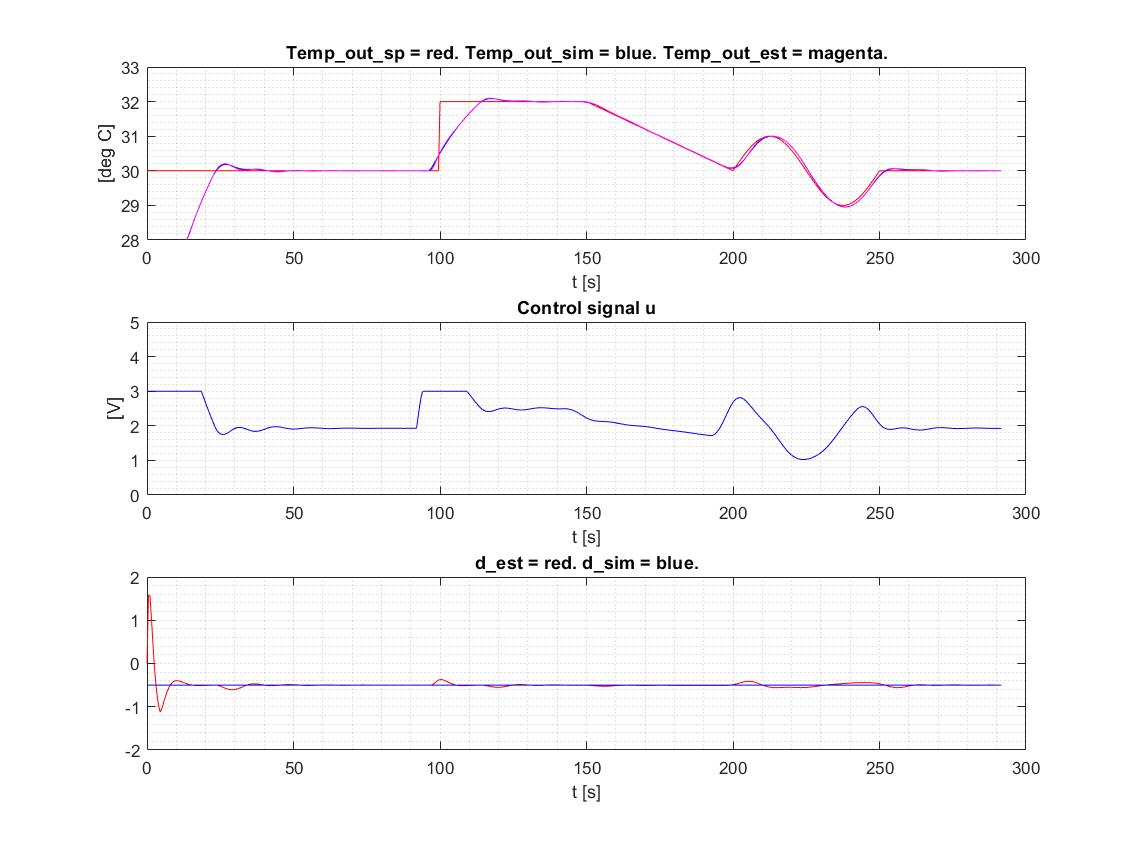
****

1. **Understand the code**
2. **Play with some of the settings**

I change initial setting of the predicted error C\_e from 1 to 10 and predicted input from 20 to 30 and upper limit of optim variable for use in fmincon of the fmincon u\_max from 5 to 10.



I change initial setting of the predicted error C\_e from 1 to 10 and predicted input from 20 to 30 and upper limit of optim variable for use in fmincon of the fmincon u\_max from 5 to 3.



I change initial guess of fmincon

%Initial guessed optimal control sequence:

Temp\_heat\_sim\_k = 0; %[C]

Temp\_out\_sim\_k = 20; %[C]

d\_sim\_k = -0.5;

%----------------------------------

%Initial values for estimator:

Temp\_heat\_est\_k = 0; %[C]

Temp\_out\_est\_k = 16; %[C]

d\_est\_k = 0; %[V]

