Endothermic Reactor Modelling

By Maria Garcia- Zarandieta Martinez,

Spring 2020

~ FINAL DRAFT ~

Illinois Institute of Technology

ChE 530 Advanced Process Control

Dr. Donald Chmielewski

Table of Contents

Table of Figures

Abstract

The objective of this paper is to develop a partial state stochastic LQOC of an Endothermic Reactor. For which project development, the chemical engineering process has been studied, analysing the equations that determine the process and modelling the dynamic system. Moreover, disturbances in the form of a shaping filter where introduced and the outputs monitored using the EDOR in the shape of an ellipse, which was compared to the simulated data.

1. Endothermic CSTR process

This reactor is defined by 6 equations of which 5 are differential equations each corresponding to one for the state variables: T1, Ca, Cb, T3 and T4. The manipulated variable being Q, which is the amount of heat added to stream 1. As for the disturbance, it has been determined that it would be the initial concentration of component CA0 which can be characterized as if it was coloured noise with mean of 1 kmol/m3, standard deviation of 0.2 kmol/m3 and a 0.25 h correlation time. These equations from the function corresponding to f(s, m, p) have been transformed into the following notation which determines the physical aspects of the process. S corresponds to the state variables, m to the manipulated variables and p to the inputs. In this case, the vector of performance outputs q= [Q T3] has no function h(s, m, p) as such, but it still can be used as the project is developed. The values for the rest of the constants of the process may be found in the Annex that corresponds to the "Main code for Modelling of Dynamic systems"

```
dsdt=[ds1dt;ds2dt;ds3dt;ds4dt;ds5dt];

ds1dt = ( (v_0*rho*C_p*(T_0-s(1))) + (U*a*(s(5)-s(1))) ) / (rho*C_p*V_1);
ds2dt = ( (v_0*(C_A0-s(2))) - (k_0*exp(-((E)/(R*s(4))))*s(2)*V_3) ) / (V_3);
ds3dt = ( -(v_0*s(3)) + (k_0*exp(-((E)/(R*s(4))))*s(2)*V_3 )) / (V_3);
ds4dt = ( (v_0*rho*C_p*((s(1)+Q/(v_0*rho*C_p))-s(4))) + (neg_del_H*k_0*exp(-
((E)/(R*s(4)))))*s(2)*V_3 ) / (rho*C_p*V_3);
ds5dt = ( (v_0*rho*C_p*(s(4)-s(5))) - (U*a*(s(5)-s(1))) ) / (rho*C_p*V_4);
```

2. Modelling of the Dynamic System

In order to determine the Steady State Operating Point (SSOP), the derivates in the above equations need to be set to 0, this meaning that there is constant time. These values for SS are s=[s1, s2, s3, s4, s5]= [T1, Ca, Cb, T3, T4] = [362.22; 0.0154; 0.9846; 449.79; 385.57]; Moreover, the linearization of this process around the SSOP has been developed. Which has been solved using MATLAB can be checked in the *function lin1.m* of the codes that can be found in the Annexes. In this way, the process will be shown as ẋ =Ax+Bu+Gw which corresponds to the continuous time process. The results for the matrixes A, B and G that correspond to State variables, manipulated variables and disturbance inputs respectively are:

```
A =

   1.0e+04 *

   -0.0009         0         0         0    0.0007
         0   -0.0163         0   -0.0000         0
         0    0.0160   -0.0003    0.0000         0
    0.0003   -3.2008         0   -0.0005         0
    0.0007         0         0    0.0003   -0.0009
```

```
B =

   1.0e-03 *

         0
         0
         0
    0.2500
         0
```

```
G =

         0
    2.5000
         0
         0
         0
```

Once the linear model has been determined, the following real scenario was implemented in the process and copared the nonlinear model with the linearized model. This code may be found in the *simpleode45.m*.

$$t < 0 \rightarrow C_{A0} = 1.0\frac{kmole}{m^3} and\ Q = 2.845x10^6\ kcal/h$$
$$t > 0\ and\ t < 10 \rightarrow C_{A0} = 1.2\frac{kmole}{m^3} and\ Q = 2.7x10^6\ kcal/h$$
$$t > 10\ and\ t < 20 \rightarrow C_{A0} = 1.0\frac{kmole}{m^3} and\ Q = 3.1x10^6\ kcal/h$$

The following sets of plots compare the nonlinear model to the linearized model:

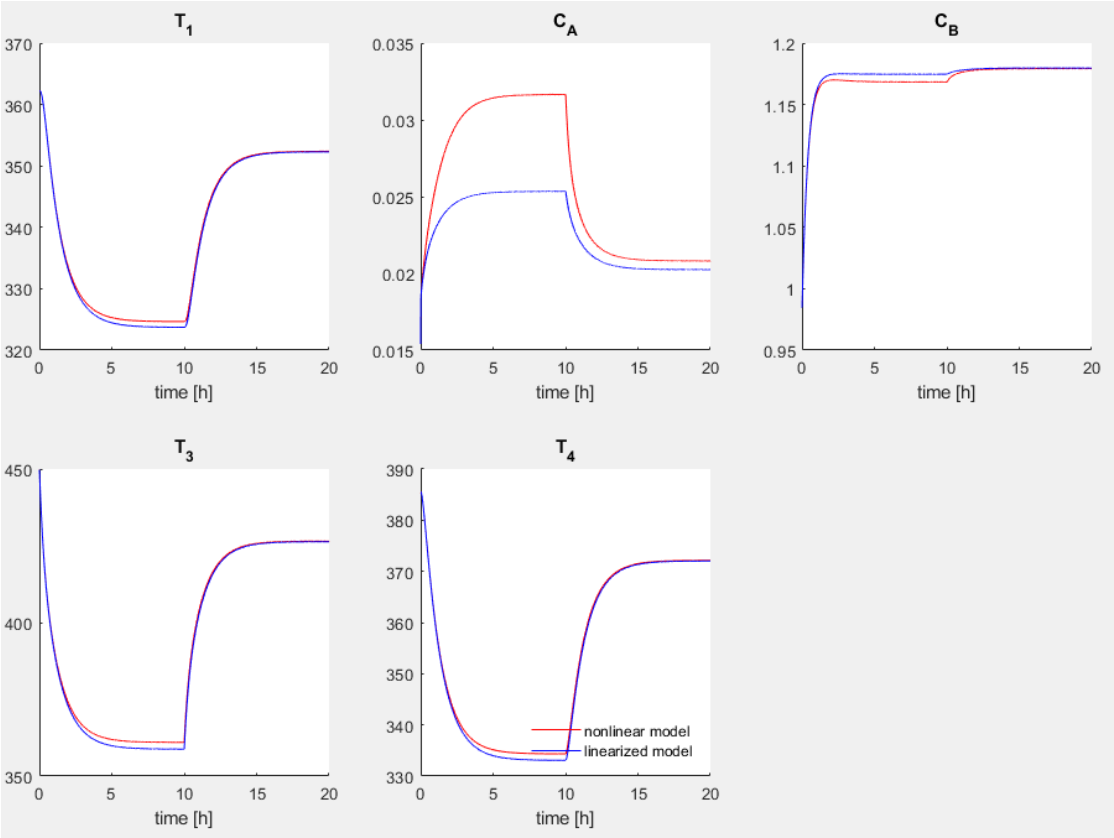

Figure 1: Comparison of nonlinear model vs. linearized model (CODE: simpleode45)

In order to simplify the code and the ODE solutions and to acquire a more accurate representation, the model has been transformed into discrete time and a sample time of dt=0.1 has been used. In this case the model would be in the form of $\dot{x}_{k+1} = A_d x_k + B_d u_k + G_d w_k$

```
Ad =

    0.4913   -7.9066         0    0.0439    0.2915
   -0.0000    0.0118         0   -0.0001   -0.0000
    0.0000    0.7670    0.7788    0.0001    0.0000
    0.1483 -154.0843         0    0.7710    0.0439
    0.3075  -28.1279         0    0.1483    0.4913
```

```
Bd =

   1.0e-04 *

    0.0043
   -0.0000
    0.0000
    0.2184
    0.0219
```

```
Gd =

   -0.7292
    0.0182
    0.2030
  -40.6416
   -3.8706
```

The same scenario as before has set and the comparaison now was made between the continous time model (linearized model) and the discrete time model. The reason being that the discrete tme model should be more accureate than the first. Although,, as it can be seen from the charts on figure 2, both are

very similar. Something that can also be seen on the following charts in the project. Such as the Expected Dynamic Operating Region (EDOR).

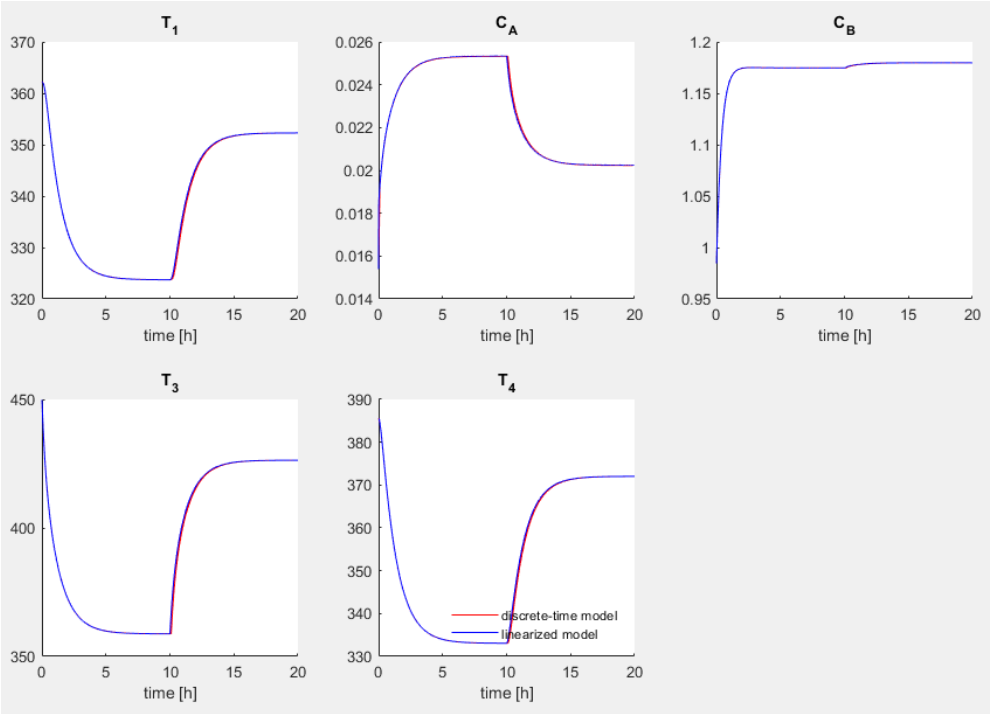

Figure 2: Comparison of continuous time (linearized) model vs. discrete time model (CODE: simpleode45_discrete)

3. Modelling with disturbances and performance outputs

There is a new equation being introduced that corresponds to the output variables which is: $Z = D_x x + D_u u + D_w w$, where each of the matrixes correspond to the performance outputs derivates with respect to the steady state values of the state variables. By this stage, the disturbance has been added to the process using a shaping filter, thus obtaining new values for the matrixes A, B and G. Which are now An, Bn and Gn.

```
An =

   1.0e+04 *

   -0.0009         0         0         0    0.0007         0
         0   -0.0163         0   -0.0000         0    0.0003
         0    0.0160   -0.0003    0.0000         0         0
    0.0003   -3.2008         0   -0.0005         0         0
    0.0007         0         0    0.0003   -0.0009         0
         0         0         0         0         0   -0.0004
```

```
Bn =

   1.0e-03 *

         0
         0
         0
    0.2500
         0
         0
```

```
Gn =

     0
     0
     0
     0
     0
     4
```

Covariance matrixes between steady state variables and performance outputs have been calculated for Open Loop configuration. All the values for the different configurations in continuous and discrete time can be found on *Table 1*.

In order to create the Closed Loop configuration, a controller was introduced into the process in the shape of u(t)=-L*x(t) where `L = (10^4)*[0.0047 -1.6407 0 0.0084 0.0036 -0.7817].`

Using this controller, the plots for the EDOR ellipses corresponding to the performance outputs Q and T3 (which means that ∑z covariances in closed loop where used) have been plotted as can be seen on Figure 3. It is perceivable that there is a small difference between these plots, since there is also a small difference between the discrete time and the continuous time models. Moreover, the process with such characteristics was plotted and it was verified that the scatter plot corresponds to the discrete-time ellipse

which corresponds to Figure 4. The coding can be found in the annexes as “MAIN_Ch5” which employs the functions “disc_time_EDOR” and “cont_time_EDOR”.

Continuous time

	∑x	∑z
Open Loop	126.4936 -0.0281 -0.8848 287.8339 172.4913 -0.4003 -0.0281 0.0000 0.0005 -0.1156 -0.0443 0.0008 -0.8848 0.0005 0.0144 -3.3935 -1.3869 0.0146 287.8339 -0.1156 -3.3935 892.5330 422.9125 -3.0657 172.4913 -0.0443 -1.3869 422.9125 239.2703 -0.7788 -0.4003 0.0008 0.0146 -3.0657 -0.7788 0.0400	1.0e+03 * 0.8925 -1.5967 -1.5967 3.0047
Closed Loop	122.1303 -0.0273 -0.8755 277.7042 166.5413 -0.3970 -0.0273 0.0000 0.0005 -0.1132 -0.0431 0.0008 -0.8755 0.0005 0.0144 -3.3532 -1.3720 0.0146 277.7042 -0.1132 -3.3532 865.1605 408.5985 -3.0406 166.5413 -0.0431 -1.3720 408.5985 231.0899 -0.7724 -0.3970 0.0008 0.0146 -3.0406 -0.7724 0.0400	1.0e+07 * 0.0001 -0.0126 -0.0126 1.9687

Discrete time

	∑x	∑z
Open Loop	126.2839 -0.0282 -0.8839 287.4807 172.2056 -0.4057 -0.0282 0.0000 0.0005 -0.1154 -0.0444 0.0008 -0.8839 0.0005 0.0143 -3.3698 -1.3835 0.0148 287.4807 -0.1154 -3.3698 887.6186 421.9968 -3.1072 172.2056 -0.0444 -1.3835 421.9968 238.8273 -0.7894 -0.4057 0.0008 0.0148 -3.1072 -0.7894 0.0394	1.0e+03 * 0.8876 -1.5905 -1.5905 2.9959
Closed Loop	122.0587 -0.0274 -0.8756 277.6533 166.4440 -0.4030 -0.0274 0.0000 0.0005 -0.1131 -0.0432 0.0008 -0.8756 0.0005 0.0143 -3.3322 -1.3698 0.0148 277.6533 -0.1131 -3.3322 861.3704 408.1505 -3.0871 166.4440 -0.0432 -1.3698 408.1505 230.9110 -0.7842 -0.4030 0.0008 0.0148 -3.0871 -0.7842 0.0394	1.0e+07 * 0.0001 -0.0126 -0.0126 1.9692

Table 1: Covariance Matrixes for Discrete and Continuous time Open and Close Loop Configurations.

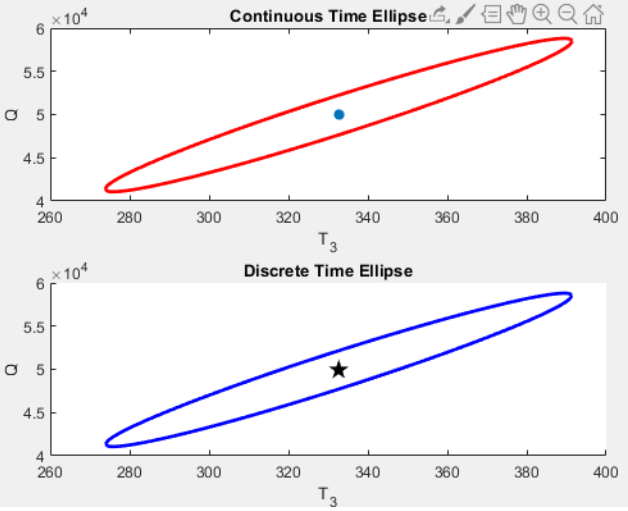

Figure 3: Continuous time and Discrete time Closed Loop EDOR

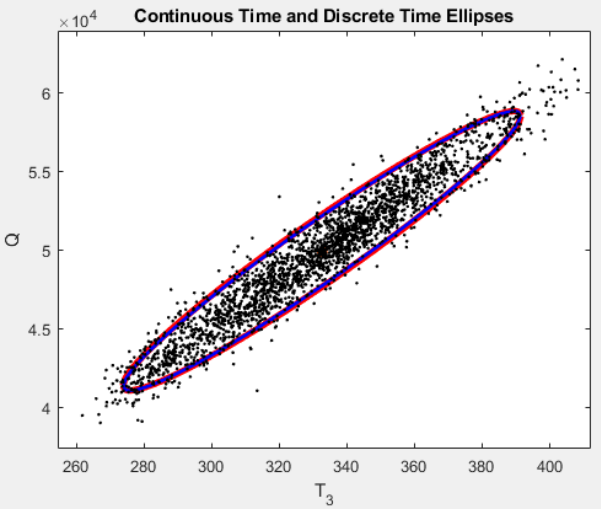

Figure 4: EDOR ellipses for continupus time and discrete time with Scatter plot simulated data.

4. State Estimation of the Process

In this step of the development of the project, a measurement information structure has been implemented in order to develop a state estimator. Therefore, the concept of error in the process is introduced. It has been assumed that the Partial State Information x(t) is unknown to the user because it is unavailable due to technical issues, furthermore, there are stochastic disturbances which means that the state will change in stochastic fashion over time. Therefore, the objective is to track the changes of x(t) using the measurement data. In this case, the disturbance CA0 is characterized as being coloured noise, which influences the output measurements that are T1 and T3, which are also corrupted with white noise.

In order to control and measure the system, the feedback controller first introduced was `L = (10^4)*[0.0047 -1.6407 0 0.0084 0.0036 -0.7817];` The system used has already been linearized around steady state variables and with the new controller u(x=-L*x(t), the Kalman filter has been implemented in discrete-time and thus the following simulations in a 10h period have been produced that compare the actual state with the state estimate. Moreover, one-step prediction from Kalman filter has been coded in order to simulate the process and can also be observed on *Figure 5*.

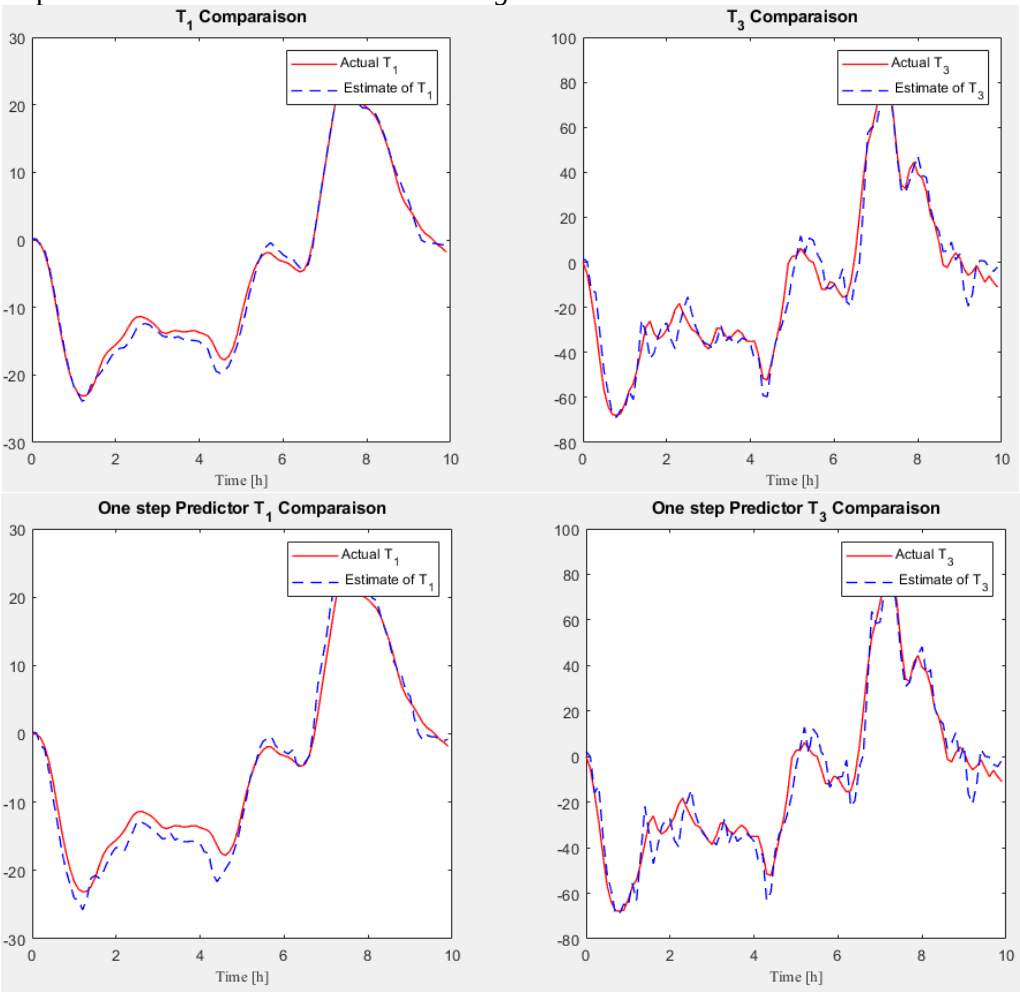

Figure 5:Coparaison actual state and state estimate with Kalman filter implementation.

On order to address the issue of checking for outliers in the process simulations, the following plots have been created both with and without one step prediction. In those, it can be observed that the error e(t)=x_hat(t)-x(t) stays in the great majority of the cases within the limit of the red dotted lines. These red

dotted lines act as Upper and Lower limits of control, thus, if one of the points its outside these limits it should be flagged as potential error in the system. The Limits have been calculated as Limit = E(x)±2*SD. Where E(x) is the mean of the sample process, and SD the standard deviation. The limits can be modified to be "harder" limits if the constant 2 is decreased or modified to be "softer" if it is increased. In a normal distribution, which is the distribution that a stochastic process follows, ±2*SD means that 95.45% of the values lie within that range from the mean.

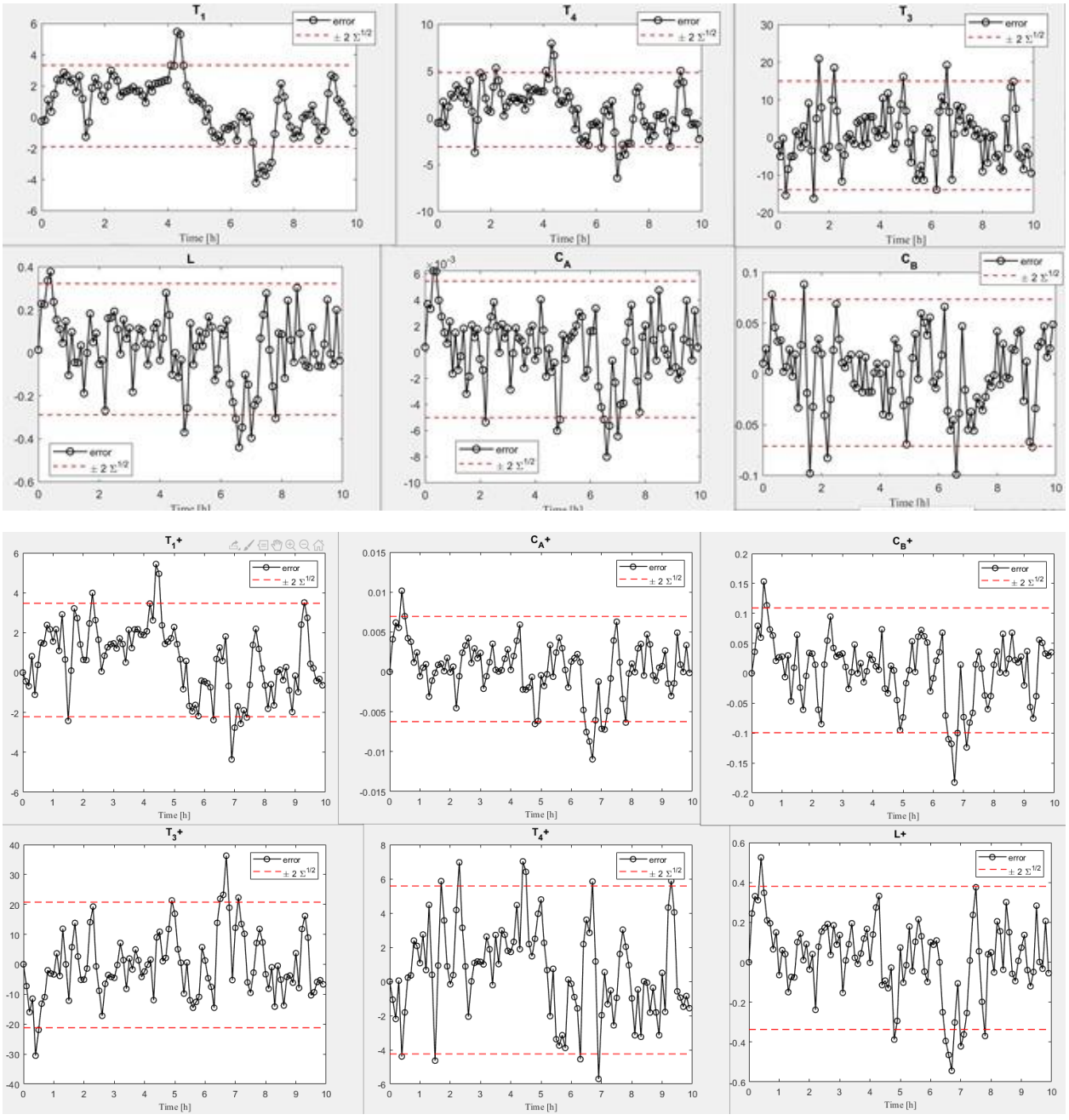

Figure 6: Standard Errors Plots with Upper and Lower control limits

5. Linear Quadratic Optimal Control

At this point, a better controller that fits the process needs to be applied to further create a realistic process. This has been achieved by applying Linear Quadratic Optimal Control LQOC by applying the batch solution method. The solution obtained is in the form of lineal feedback thus the new controller for discrete-time is: `Liid = 1.0e+04 *[ 0.0048 -1.5478 0.0000 0.0078 0.0036]`, different than the first applied.

Thus with this new feedback control, the process was simulated over 10 hours considering that the initial conditions where: `x0=[0; 0.5; -0.5; -30; -10];` These plots represent the actual state variables in natural form, and are divided into Temperatures and Concentrations as can be observed in Figure 7.

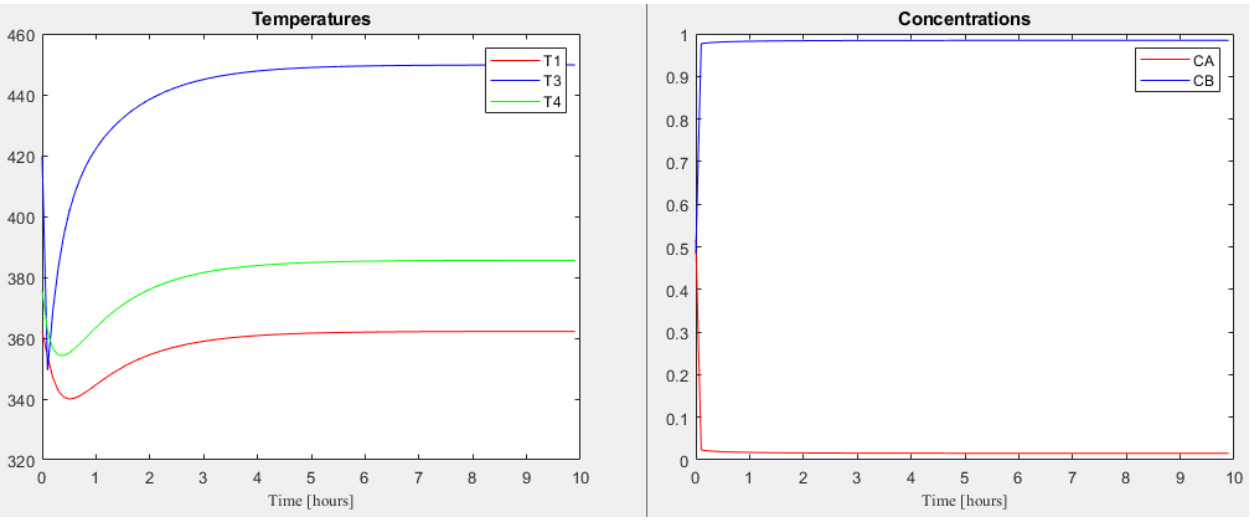

Figure 7:LQOC actual state variables in natural form simulation.

6. Actual State Modelling-Stochastic LQOC

In the previous step of the project, the process was not considered stochastic to make it easier to understand at first. However, it is necessary to introduce into the model, the noises that influence the measurements. Thus, to the same problem stated in the previous point of the project, the disturbance of Ca0 as a coloured noise with mean 1 kmol/m3, standard deviation of 0.2 kmol/m3 and correlation time of 0.25 hours has been added back to the system. The linearization around the steady state values and conversiton to discreate time, while assuming a Q=1000I and R=0.001, lleads to the comparaison of the actual state variables in natural form to be as the observed on figure 8.

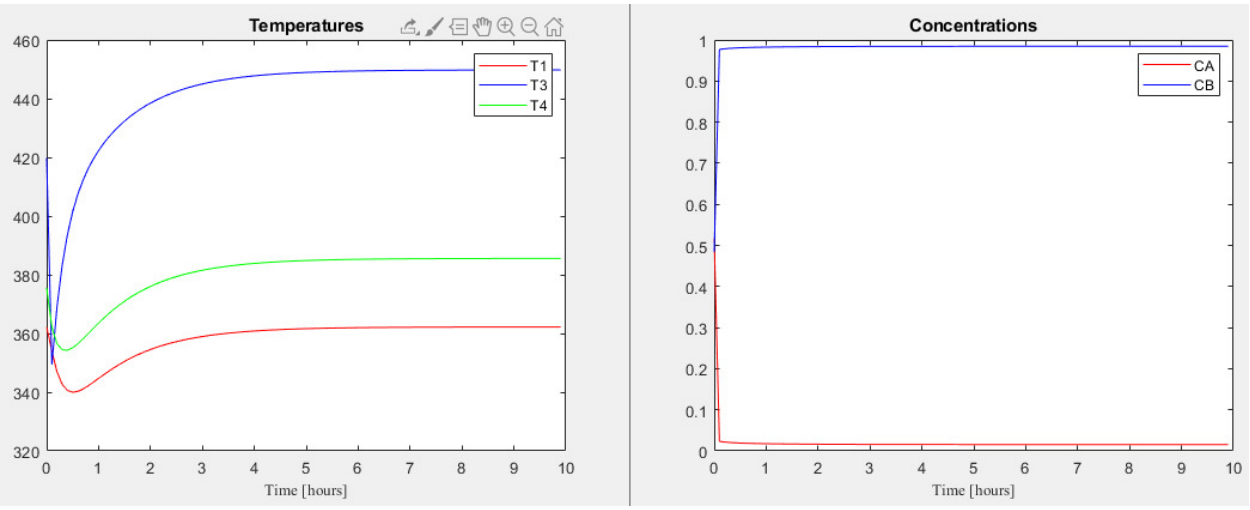

Figure 8; LQOC actual state variables in natural form simulation considered stochastic process

Moreover, a simulation period of 400 hours has been created to plot the EDOR and the ellipse that fits the operating region of the outputs T3 and Q. The model has been coded such that the values for temperature and flowrate where physically possible.

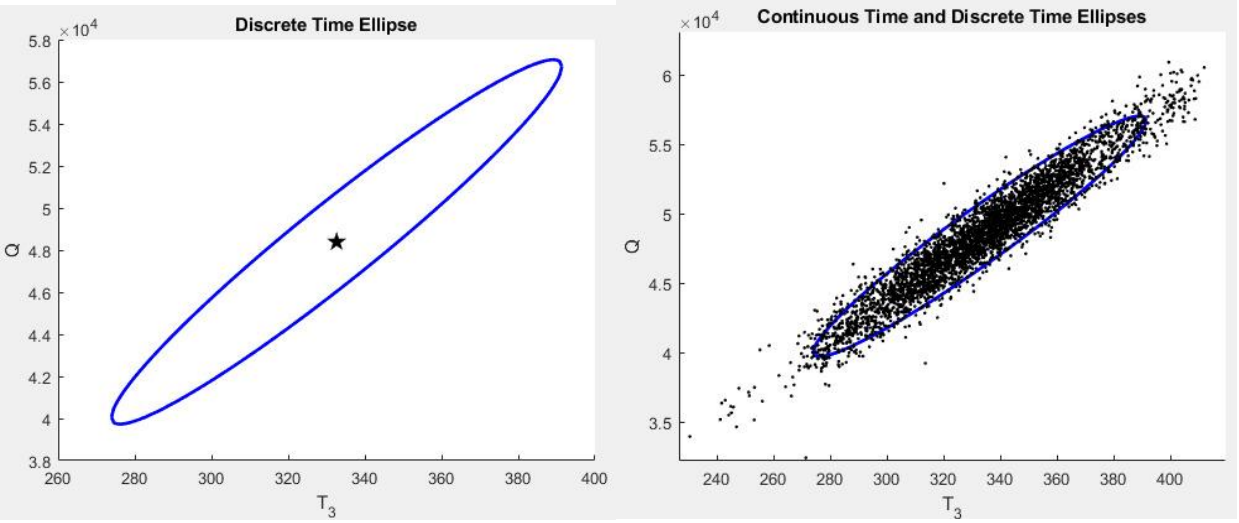

Figure 9: Discrete time ellipse and scatter plot of Stochastic LQOC

Lastly, the process has been modelled to be even more realistic and to follow a stochastic manner by adding to the system the measurements of variables T0 (belonging to the disturbance CA0 and its shaping filter) and T3, which are simulated as zero mean white noise with spectral densities Sv1=40 and Sv2=10, which corrupt the outputs q(Q and T3) with noise.

This way, using a simulation period of 200 hours, the phase plane scatter plot has been graphed and the EDOR (with two standard deviations) has been plotted over it, which leads to observe that it fits the scatter plot of the simulated process.

7. Partial State Information (PSI) LQOC

A second form of stochastic LQOC was determined and PSI (partial State nforation) was used. PSI assumes that for feedback their measurements available are corrupted by noise. These measurements in this case where T0 and T1 corrupted by zero mean white noise. As can be seen in figure 10, where the red ellipse corresponds to the simulation with delay and the blue corresponds to without delay.

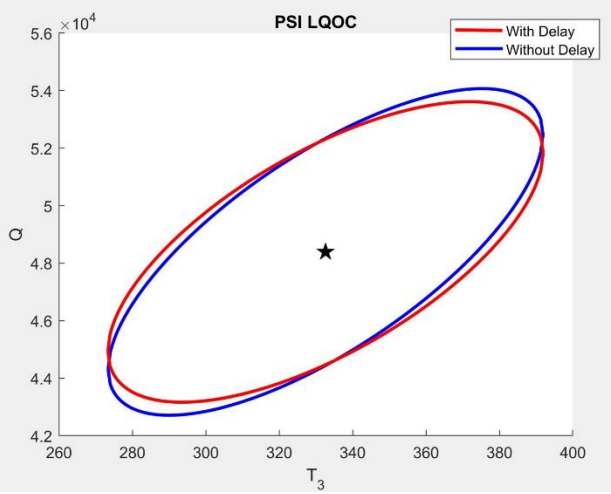

Figure 10: EDOR ellipse without delay (blue) and with delay (red)

8. Conclusion

In order to create a model for a endothermic CSTR reactor, it was first necessary to understand the process for which the equations where analysed and the steady state operation points where found. These points are considered as the stabilizable values to which the system should converge. After that, a comparison between non-linearized model has been created and the results show that they are very similar which is good in terms of the model output. Because this is a continuous model and discretizing a continuous model makes anything the data simpler, this was also done. Because a small dt, the models were also very similar to each other.

To continue with making the simulated model as close as reality as possible, a colored noise disturbance was introduced, which was introduced into the model via a shaping filter. Moreover, a discrete time Kalman Filter and one-step predictor Kalman filter where coded into the simulation such that there could be a comparison made between the estimated values and the real values. A vector that contained the errors between the estimated and the real values was also created and consequently plotted. These values where plotted with limits of ±2 Standard deviations, and most of the values where consistently between the limits but for a few outliers. Two standard deviations from the mean in a normalized set of data, means that there is a confidence of 95% that the values would be within, and a 5% chance that they are off limits. It is important to determine whether there are false negative when faults are introduced.

Finally, the model was processed such that the Linear Quadratic Optimal Control could be applied. Actual state variables in natural form were plotted and there seams to be an error in the concentration units as they abruptly increase in value form the initial points, should be something to take into consideration in future works. In conclusion, modelling a CSTR reactor is fairly simple compared to other systems and can be understood and simulated in order to increase knowledge in this matter as a good example.

Annexes

1. Main code for Modelling of Dynamic systems

```
clear; close all force; clc;

%%%constants
V_1 = 4;                %volume [m^3]
V_2 = V_1;
V_3 = V_1;
V_4 = V_1;

v_0 = 10;               % volumetric flow ate [m^3/h]
k_0 = 1.5*10^3;         % reaction rate constant[1/h]
E = 2*10^3;             % activation energy for rxn [kcal/kmol]
R = 1.987;              % mass-specific gas constant [kcal/kmole/K]
rho = 1000;             % density [kg/m^3]
C_p = 1;                 % heat capacity with const. P [kcal/kg/K]
T_0 = 298;              % initial temp. [K]
neg_del_H = -2*10^5;    % negative delta enthalpy - realeased heat [kcal/kmole]
U = 550;                % internal energy [kcal/hr/K/m^2]
a = 50;                 % area [m^2]

dt = 0.1;               % sample time [h]

%%%SSOP constants
Q_ss = 2.845*10^6;                  %heat (SSOP) [kcal/h]
C_A0_ss = 1;                        % inlet conc. (SSOP) [kmole/m^3]

%%from part ii --> myfun.m
T_2_ss = 646.72;                                    % [K]
s_ss = [362.22; 0.0154; 0.9846; 449.79; 385.57];  % s_ss = [T1 c_a c_b T3 T4]
q_ss = [449.79; 2.845*10^6];                        % qout = [s(4); Q_ss]
T_3= s_ss(4);
C_A= s_ss(2);



A =[ -(U*a + C_p*rho*v_0)/(C_p*V_1*rho),
0,        0,
0,                (U*a)/(C_p*V_1*rho);
                                  0,      -(v_0 + V_3*k_0*exp(-
E/(R*T_3)))/V_3,        0,                                                -
(C_A*E*k_0*exp(-E/(R*T_3)))/(R*T_3^2),                                       0;
                                  0,                        k_0*exp(-
E/(R*T_3)), -v_0/V_3,
(C_A*E*k_0*exp(-E/(R*T_3)))/(R*T_3^2),                                       0;
                             v_0/V_3, (k_0*neg_del_H*exp(-
E/(R*T_3)))/(C_p*rho),        0, -(C_p*rho*v_0 -
(C_A*E*V_3*k_0*neg_del_H*exp(-E/(R*T_3)))/(R*T_3^2))/(C_p*V_3*rho),
0;
                 (U*a)/(C_p*V_4*rho),
0,        0,
v_0/V_4, -(U*a + C_p*rho*v_0)/(C_p*V_4*rho)];
```

```
B = [0; 0; 0; 1/(rho*C_p*V_3); 0];

G = [0; v_0/V_3; 0; 0; 0];

%%%compare the linearized model with the original nonlinear model
%Call function simpleode45()

%%%Report the values of Ad , Bd, Gd
Ad=expm(A*dt)
sum=zeros(5);
Ndt=2000;
ddt=dt/Ndt;
for ii=1:Ndt
    sum=sum+expm(A*ii*ddt);
end
Bd=sum*B*ddt
Gd=sum*G*ddt
%
% %%%compare the discrete-time simulation with the linear continuous-time
simulation (from part iv)
%       %I GIVE UP%
```

2. Linealization

```
%Linearization of model around SSOP, Report A, B and G
%dx/dt= A*X+ B*u+ G*w

function [A,B,G] = lin1()

syms v_0 rho C_p T_0 T_1 T_3 T_4 U a V_1 V_3 V_4 C_A0 C_A C_B k_0 E R Q
neg_del_H

f1 = ( (v_0*rho*C_p*(T_0-T_1)) + (U*a*(T_4-T_1)) ) / (rho*C_p*V_1);
f2 = ( (v_0*(C_A0-C_A)) - (k_0*exp(-((E)/(R*T_3)))*C_A*V_3) ) / (V_3);
f3 = ( -(v_0*C_B) + (k_0*exp(-((E)/(R*T_3)))*C_A*V_3  )) / (V_3);
f4 = ( (v_0*rho*C_p*((T_1+Q/(v_0*rho*C_p))-T_3)) + (neg_del_H*k_0*exp(-
((E)/(R*T_3))))*C_A*V_3 ) / (rho*C_p*V_3);
f5 = ( (v_0*rho*C_p*(T_3-T_4)) - (U*a*(T_4-T_1)) ) / (rho*C_p*V_4);

A = jacobian ([f1; f2; f3; f4; f5], [T_1 C_A C_B T_3 T_4]);

B = jacobian ([f1; f2; f3; f4; f5], [Q]);

G = jacobian ([f1; f2; f3; f4; f5], [C_A0]);

%%%constants
V_1 = 4;            %volume [m^3]
V_3 = V_1;
```

```
V_4 = V_1;

v_0 = 10;            % volumetric flow ate [m^3/h]
k_0 = 1.5e3;         % reaction rate constant[1/h]
E = 2e3;             % activation energy for rxn [kcal/kmol]
R = 1.987;           % mass-specific gas constant [kcal/kmole/K]
rho = 1000;          % density [kg/m^3]
C_p = 1;             % heat capacity with const. P [kcal/kg/K]
T_0 = 298;           % initial temp. [K]
neg_del_H = -2e5;    % negative delta enthalpy - realeased heat [kcal/kmole]
U = 550;             % internal energy [kcal/hr/K/m^2]
a = 50;              % area [m^2]

s_ss = [362.22; 0.0154; 0.9846; 449.79; 385.57];  % s_ss = [T_1 C_A C_B T_3
T_4]
C_A = s_ss(2);
T_3 = s_ss(4);


A = eval(A)

B = eval(B)

G = eval(G)

end
```

3. Comparaison between Nonlinear and linear models

```
close all

tspan = [0 20];

%NON LINEAR
%Non Linear Initial value = s_ss

s_ss = [362.22; 0.0154; 0.9846; 449.79; 385.57];
[t1,y1] = ode45('rxns_ode', tspan, s_ss);


%LINEAR
[t2,y2] = ode45('rxns_ode_lin', tspan, zeros(1,5));
titles=["T_1";"C_A";"C_B";"T_3";"T_4"];

%Plot all variables

for n=1:5
    y2 (:,n) = y2(:,n) +s_ss(n);
%     figure()
  subplot(2,3,n);
    title(titles(n))
```

```
    hold on
    plot(t1,y1(:,n),'r')
    plot(t2,y2(:,n),'b')
%     legend('nonlinear model','linearized model')
    xlabel('time [h]')
end
legend('nonlinear model','linearized model')
legend('Location','southeast')
legend('boxoff')
```

4. Comparaison between discrete time and continupus time models

```
close all

tspan = [0 20];

%NON LINEAR
%Discrete Time
s_ss = [362.22; 0.0154; 0.9846; 449.79; 385.57];
Ad =[0.491325149210661,-
7.90655159101652,0,0.0438761361021651,0.291511365549888;
    -1.07755691972936e-05,0.0117974188021300,0,-5.90284905365730e-05,-
3.02893791379800e-06;
    1.07755691972713e-
05,0.767003364269276,0.778800783071405,5.90284905365708e-05,3.02893791387334e-
06;
    0.148330942772703,-
154.084309104058,0,0.771030816653235,0.0438761361021508;
    0.307466324132495,-
28.1278772314054,0,0.148330942772699,0.491325149210660];

Bd = [4.32082172648495e-07;-1.55694729737992e-09;1.55694729737991e-
09;2.18354097417844e-05;2.18525181527072e-06];

Gd = [-0.729216424576047;0.0182031047226589;0.202982287542898;-
40.6415861980369;-3.87057984837156];

dt=0.1;

nx=5;

N=20/dt; td=zeros(1,N); X=zeros(nx, N);
x0=zeros(nx,1); X(:,1)=x0;
for k=1:N-1
td(k+1)=dt*k;
uk=[2.7e6-2.845e6];
wk=[0.2];
if (td(k)>10)
    uk=[3.1e6-2.845e6];
    wk=[0.2];
end
    X(:,k+1)=Ad*X(:,k)+Bd*uk+Gd*wk;
```

```
end
y1=X';
%Continuous time
[t2,y2] = ode45('rxns_ode_lin', tspan, zeros(1,5));
titles=["T_1";"C_A";"C_B";"T_3";"T_4"];

%Plot all variables


for n=1:5
    y2 (:,n) = y2(:,n) +s_ss(n);
    y1 (:,n) = y1(:,n) +s_ss(n);
%     figure()
    subplot(2,3,n);
    title(titles(n))
    hold on
    plot(td',y1(:,n),'r')
    plot(t2,y2(:,n),'b')

    xlabel('time [h]')
end
legend('discrete-time model','linearized model')
legend('Location','southeast')
legend('boxoff')
```

5. Main_Ch5

```
%%Main function
clc
close all
clear all

%%% A. CONTINUOUS TIME

% Linearize system around SS values And calulate OPEN LOOP cov Matrix
    [SigxOL,SigzOL, SigxCL, SigzCL, An, Bn, Gn, Dx_new, Dun, Dwn,L]=
cont_time_EDOR();
    disp("CONTINUOUS TIME PROCESS:")
    disp("Open Loop with Filter New Compound Matrixes:")
    disp("An*x+Bn*u+Gn*w")
    An
    Bn
    Gn
disp("z=Dx_new*x+ Dun*u+ Dwn*w")
    Dx_new
    Dun
    Dwn

disp("Open Loop Covariance Matrixes:")
    SigxOL
    SigzOL

% Linearize system around SS values And calulate CLOSED LOOP cov Matrix
```

```
disp("With controller u(t)=-L(x(t) and L=")
    L
disp("The values of the Covariance Matrixes in Closed Loop are:")
    SigxCL
    SigzCL

%Plot CLOSE LOOP continuous time EDOR
disp("Standard deviation parameter alpha=2")

%Solve for steady state
pbar=1;

%%SIGNS

sbarCC=inv(An)*Gn*pbar;
qbarCC=(Dx_new - Dun*L)*sbarCC;

%Define EDOR
zbar=qbarCC;
sigZ=SigzCL;
alpha = 2;
Ezinv=inv(sigZ);

%Determie range of independent variable

xind_max=sqrt(alpha^2*Ezinv(2,2)/det(Ezinv));
xind_min=-xind_max;
N=200;
xind2=zeros(1,N);
yup=zeros(1,N);
ydown=zeros(1,N);
step = (xind_max-xind_min)/(N-1);

%calculate upper and lower curve values

for i=1:N
xind=xind_min+step*(i-1);
xind2(i)=xind;
b=Ezinv(1,2)*xind/Ezinv(2,2);
c=(Ezinv(1,1)*xind*xind-alpha^2)/Ezinv(2,2);
yup(i)=-b+sqrt(b^2-c);
ydown(i)=-b-sqrt(b^2-c);
end

%Plot upper and lower curves

xind2=xind2+zbar(1); yup=abs(yup+zbar(2)); ydown=abs(ydown+zbar(2));
figure()
hold off
plot(abs(zbar(1)),abs(zbar(2)),'*',
xind2,yup,':',xind2,ydown,':','LineWidth',4)
xlabel('T_3')
ylabel('Q')
title('Continuous Time Ellipse')
```

```
%%% B. DISCRETE TIME

% Linearize system around SS values And calulate OPEN LOOP cov Matrix
    [SigxdOL,SigzdOL, SigxdCL, SigzdCL, Ad_new, Bd_new, Gd_new, Dx_new,
Du_new, Dwn]= disc_time_EDOR;
    disp("Discrete TIME PROCESS:")
    disp("Open Loop with Filter New Compound Matrixes:")
    disp("Ad*x+Bd*u+Gd*w")
    Ad_new
    Bd_new
    Gd_new
disp("z=Dx_new*x+ Dun*u+ Dwn*w")
    Dx_new
    Dun
    Dwn

disp("Open Loop Covariance Matrixes:")
    SigxdOL
    SigzdOL

% Linearize system around SS values And calulate CLOSED LOOP cov Matrix

disp("With controller u(t)=-L(x(t) and L=")
    L
disp("The values of the Covariance Matrixes in Closed Loop are:")
    SigxdCL
    SigzdCL

%Plot CLOSE LOOP continuous time EDOR
disp("Standard deviation parameter alpha=2")

%Solve for steady state

sbarDD=-inv(eye(size(Ad_new))-Ad_new)*Gd_new*pbar;
avdd=(Dx_new - Dun*L)*sbarDD;


%Define EDOR
zbar=avdd;
dt=0.1;
Sw=0.02;

% Change this too

sigZd=SigzdCL;
Ezinvd=inv(sigZd);
%Determine ranges of independent variable

xind_maxd=sqrt(alpha^2*Ezinvd(2,2)/det(Ezinvd)); xind_mind=-xind_maxd;
N=200; xind2d=zeros(1,N); yupd=zeros(1,N); yd=zeros(1,N);
step = (xind_maxd-xind_mind)/(N-1);

%Calculate upper and lowe curve values
```

```
for ii=1:N
xindd=xind_mind+step*(ii-1);
xind2d(ii)=xindd;
b=Ezinvd(1,2)*xindd/Ezinvd(2,2);
c=(Ezinvd(1,1)*xindd*xindd-alpha^2)/Ezinvd(2,2);
yupd(ii)=-b+sqrt(b^2-c);
yd(ii)=-b-sqrt(b^2-c);
end

%Plot upper and lower curves
xind2d=xind2d+zbar(1);
yupd=abs(yupd+zbar(2));
yd=abs(yd+zbar(2));
figure()
hold on
plot(abs(zbar(1)),abs(zbar(2)),'pk',
xind2d,yupd,'b',xind2d,yd,'b','LineWidth',2)
xlabel('T_3')
ylabel('Q')
title('Discrete Time Ellipse')

% Simulate process
randn('state',2^6-1);
Ld = (10^4)*[0.0047 -1.6407 0 0.0084 0.0036 -0.7817];
NN=round(500/dt);
Sig_p = Sw/dt;
tt=zeros(1,NN);
ss=zeros(6,NN);
qq=zeros(2,NN);
ss0=sbarDD;
ss(:,1)=ss0;

for kk=1:NN-1
tt(kk+1)=dt*kk;
pk=sqrt(Sig_p)*randn+pbar;
ss(:,kk+1)=(Ad_new)*ss(:,kk)+Gd_new*pk;
%ss(:,kk+1)=(Ad_new-Bd_new*Ld)*ss(:,kk)+Gd_new*pk;
qq(:,kk)=(Dx_new - Dun*Ld)*ss(:,kk);
%qq(:,kk)=(Dx_new)*ss(:,kk);
end
qq(:,NN)=qq(:,NN-1);

figure()
hold off
plot(
xind2,yup,':',abs(zbar(1)),abs(zbar(2)),'*',xind2,ydown,':','LineWidth',4)
% legend({'Continuous time'},'Location','southwest')
xlabel('T_3')
ylabel('Q')
hold on
plot(abs(zbar(1)),abs(zbar(2)),'pk',
xind2d,yupd,'b',xind2d,yd,'b','LineWidth',2)
hold on
plot(abs(qq(1,:)), abs(qq(2,:)),'k.');
title('Continuous Time and Discrete Time Ellipses')
```

```
% legend({'~','Continuous time','Discrete time', 'Simulated discrete time
process'},'Location','southwest')
```

5.1. Function cont_time_EDOR

```
function [SigxOL,SigzOL, SigxCL, SigzCL, An, Bn, Gn, Dx_new, Dun, Dwn,L]=
cont_time_EDOR()

%%%%%%constants
Q_ss = 2.845*10^6;       % heat (SSOP) [kcal/h]
C_A0_ss = 1;             % inlet conc. (SSOP) [kmole/m^3]
dt=0.1;
s_ss = [362.2 0.0154 0.9846 449.8 385.6];
q_ss = [s_ss(4); Q_ss];  % assume no limitations on performance output q
%m = [Q];
%p = [C_A0];

pbar = 1;                % Assume the disturbance,CA0, is characterized as
being colored noise with a mean of 1 kmol e/m
st_div = 0.2;            % standard diviation [kmole/m^3]
tau = 0.25;              % correlation time [h]
Sw = 2*tau*st_div^2;     % not 100% sure about this eq

%matrices
A = [-9.3750,0,0,0,6.875;
    0,-162.538579499525,0,-0.0122619049667084,0;
    0,160.038579499525,-2.5,0.0122619049667084,0;
    2.5,-32007.7158999051,0,-4.95238099334167,0;
    6.875,0,0,2.5,-9.375];

B = [0;0;0;0.00025;0];

G = [0;2.5;0;0;0];


%z=Dx*x+ Du*u+ Dw*w or  q=[T_3 ;Q]=Dx*[s]+ Du*[m]+ Dw*[0]  // where// z=q-qss
/ w=p-p22 disturbance inputs [Ca0]/ u=m-mss manipulated variables [Q]/
Dx=[0 0 0 1 0; 0 0 0 0 0];
Du=[0; 1];
Dw = [0; 0];

%define shaping filter
st_div = 0.2;                    % standard diviation [kmole/m^3]
tau = 0.25;                      % correlation time [h]
Sw = 2*tau*st_div^2;             % spectral density

Swf = [Sw];                      % spectral density with filter
Af = [-1/tau];
Gf = [1/tau];
Dxf = eye(1);                    % Dx with filter
Dwf = zeros(1);                  % Dw with filter

%Continuous time
```

```
An = [A G*Dxf;0 0 0 0 0 Af];     % new matrix A eq. 5.28
Bn = [B; 0];                     % new matrix B
Gn =[G*Dwf; Gf];                 % new matrix G
Dx_new=[Dx Dw*Dxf];              % new matrix Dx
Dun=Du;
Dwn=Dw*Dwf;

% sample and hold --> for discrete time only
nx=6; Ndt=200; ddt=dt/Ndt; sum=zeros(nx); Sigw=Swf./dt;
for jjj=1:Ndt
    sum=sum+expm(An*jjj*ddt);
end
Ad_new = expm(An*dt);                % new matrix Ad
Bd_new = sum*Bn*ddt;                 % new matrix Bd
Gd_new = sum*Gn*ddt;                 % new matrix Gd


%%%(i) linearizing around ss (open-loop) + ss covariance matrix (x) %u = 0;
SigxOL = lyap(An,Gn*Sw*Gn');
SigzOL = (Dx_new - Dun)*SigxOL*(Dx_new - Dun)'  ;


%%%(ii) repeat part (i) with closed-loop --> u = -Lx(t)
L = (10^4)*[0.0047 -1.6407 0 0.0084 0.0036 -0.7817];

SigxCL = lyap(An-Bn*L, Gn*Sw*Gn');

SigzCL = (Dx_new - Dun*L)*SigxCL*(Dx_new - Dun*L)' ;
```

5.2. Function disc_time_EDOR

```
function [SigxdOL,SigzdOL, SigxdCL, SigzdCL, Ad_new, Bd_new, Gd_new, Dx_new,
Du_new, Dwn]= disc_time_EDOR()
%%%%%constants
Q_ss = 2.845*10^6;       % heat (SSOP) [kcal/h]
C_A0_ss = 1;             % inlet conc. (SSOP) [kmole/m^3]
dt=0.1;
s_ss = [362.2 0.0154 0.9846 449.8 385.6];
q_ss = [s_ss(4); Q_ss];  % assume no limitations on performance output q
%m = [Q];
%p = [C_A0];

%matrices
A = [-9.37500000000000,0,0,0,6.87500000000000;
    0,-162.538579499525,0,-0.0122619049667084,0;
    0,160.038579499525,-2.50000000000000,0.0122619049667084,0;
    2.50000000000000,-32007.7158999051,0,-4.95238099334167,0;
    6.87500000000000,0,0,2.50000000000000,-9.37500000000000];

B = [0;0;0;0.000250000000000000;0];

G = [0;2.50000000000000;0;0;0];
```

```
Dx = [0 0 0 1 0 ; 0 0 0 0 0 ];

Du=[0;1];

Dw = [0; 0];

%define shaping filter
st_div = 0.2;                   % standard diviation [kmole/m^3]
tau = 0.25;                     % correlation time [h]
Sw = 2*tau*st_div^2;            % spectral density

Swf = [Sw];                     % spectral density with filter
Af = [-1/tau];                  % A matrix with filter
Gf=[1/tau];                     % G matrix with filter
Dxf = eye(1);                   % Dx with filter
Dwf = zeros(1);                 % Dw with filter

%Continuous time
An = [A G*Dxf;0 0 0 0 0 Af];    % new matrix A eq. 5.28
Bn = [B; 0];                    % new matrix B
Gn =[G*Dwf; Gf];                % new matrix G
Dx_new=[Dx Dw*Dxf];             % new matrix Dx
Du_new=Du;
Dwn=Dw*Dwf;

% sample and hold --> (can be done either before or after addinf shaping
filter) Eq 5.136
nx=6;
Ndt=200;
ddt=dt/Ndt;
sum=zeros(nx);
Sigw=Swf./dt;
for jjj=1:Ndt
    sum=sum+expm(An*jjj*ddt);
end
Ad_new = expm(An*dt);                   % new matrix Ad
Bd_new = sum*Bn*ddt;                    % new matrix Bd
Gd_new = sum*Gn*ddt;                    % new matrix Gd

%%%(iv)open-loop (comapred to cont-time = super close yay)
SigxdOL=dlyap(Ad_new-Bd_new,Gd_new*Sigw*Gd_new');
SigzdOL=(Dx_new-Du_new)*SigxdOL*(Dx_new-Du_new)';


%%%(iv) closed-loop (comapred to cont-time = super close yay)
Ld = (10^4)*[0.0047 -1.6407 0 0.0084 0.0036 -0.7817];
SigxdCL=dlyap(Ad_new-Bd_new*Ld,Gd_new*Sigw*Gd_new');
SigzdCL=(Dx_new-Du_new*Ld)*SigxdCL*(Dx_new-Du_new*Ld)';

%%%(v) plot closed-loop discrete-time EDOR ellipse T3 and Q + use 'hold on' to
compare cont. with disc. times plots
```

```
%%%(vi) simulate closed-loop discrete time; add appropriate outputs to EDOR +
verify scatter plot corresponds to discrete-time ellipse

% %I took this from "help_past"; I fixed xx0 from (5,1) to (6,1) and NN bc we
have in h and they had dt in sec
% NN=1000/dt;
% tt=zeros(1,NN);
% xx=zeros(nx,NN);
% zz=zeros(2,NN);
% xx0=zeros(6,1);
% xx(:,1)=xx0;
% for kk=1:NN-1
%     tt(kk+1)=dt*kk
%     wk=sqrt(Sigw)*randn(1,6);
%     xx(:,kk+1)=(Ad_new-Bd_new*Ld)*xx(:,kk)+Gd_new*wk;
%     zz(:,kk+1)=(Dx_new-Du_new*Ld)*xx(:,kk)+Dwn*wk;
% end
% zz(:,NN)=zz(:,NN-1);
%
% %not sure how helpful it, there are also "plots" this is at the very end
% of the help code "%Simulate Process"
```