Chemical and Energy Engineering

PC Project

Control of multivariable process-Newell and Lee evaporator

Submitted by

Name	Matriculation No.
Rakesh Battula	236792
Sasikanth Vadde	236711

Under Supervision of

Dr. Ilknur Disli-Kienle

Institute for Automation Engineering

Submission date: 13.05.2023

Table of Contents

Chapter 1 Task 1

1. Process Description:

Evaporation is a common unit operation in the chemical industry, where a liquid solution is concentrated by removing some of the solvents as vapor. There are different types of evaporators, such as natural circulation, falling film, and forced circulation. The forced circulation evaporator was developed by Newell and Lee in 1989 and has become a benchmark process in the evaporator industry.

Forced circulation evaporators are widely used in chemical, food, and other industries to concentrate solutions by removing water through evaporation. A pump is used to recirculate the solution through the heating element at a high velocity in forced circulation evaporators. This ensures good heat transfer between the solution and the heating element, allowing the evaporator to handle solutions with higher viscosities and concentrations, and to operate at higher temperatures and with higher heat fluxes. This design helps to prevent fouling and scaling of the heating surface and has since been modified and improved upon to become a widely used technology in various industries.

The evaporation chamber functions as a heat exchanger, utilizing latent heat from steam condensation to evaporate a portion of the circulating process stream. After separation, the gas phase is condensed prior to exiting the system, while a portion of the liquid is extracted as a product before the remaining fraction is blended with fresh feed and recirculated back into the evaporator.

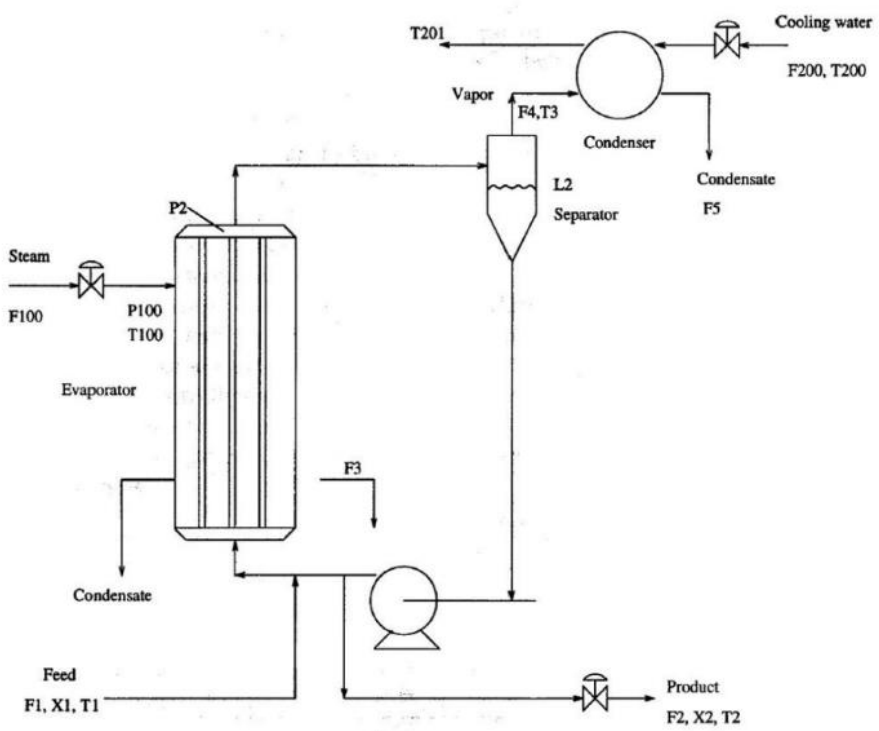

Figure 1-1 Evaporator system

1.1 Design aspects of a process control system:

1.1.1 Control Objective:

To achieve optimal operation of the Newell and Lee evaporator, the control system must maintain the desired operating conditions, which include the constant level (L2) in the separator, desirable product quality (X2), permissible operating pressure (P2), and compliance with safety, environmental, and economic considerations.

1.1.2 Input Variables:

Manipulated variables

- Product Flow Rate (F2)
- Steam Pressure (P100)
- Cooling Water Flow Rate (F200)

Disturbance variables

- Feed Flow Rate (F1)
- Feed Composition (X1)
- Feed Temperature (T1)
- Circulating flow rate (F3)
- Cooling Water Inlet Temperature (T200)

1.1.3 Output Variables:

- Product Composition (X2)
- Operating Pressure (P2)
- Separator Level (L2)
- Product temperature (T2)
- Condensate Flow Rate (F5)
- Cooling Water Outlet Temperature (T201)

1.1.4 Constraints:

- Operating Pressure (P2), Separator Level (L2), and flow rates are to be maintained at an operable limit and obtain the desired product composition.

1.1.5 Operating Characteristics:

- The Process is continuous as there is a continuous Inlet and Outlet feed flow rate from the system.

1.1.6 Control Structure:

- The MPC controller, which is constructed using the state space model, can be considered a form of feedforward control as it uses a prediction model to anticipate the effect of disturbances and optimize the control inputs

- The PI controller, on the other hand, is a classic feedback control structure that measures the system output and adjusts the control inputs accordingly to maintain the desired set point. We are using both feedback & feedforward control structure in this.

1.2 The model equations:

The separator

A total mass balance around the separator gives

$$\rho A \frac{dL2}{dt} = F1 - F4 - F2$$

Where ρ is the liquid density and A is the cross-sectional area of the separator and ρA=20 kg/m.

The evaporator

The evaporator itself is modeled by the following 5 equations:

$$M \frac{dX2}{dt} = F1 \times X1 - F2 \times X2$$

$$C \frac{dP2}{dt} = F4 - F5$$

$$T2 = 0.5616 \times P2 + 0.3126 \times X2 + 48.43$$

$$T3 = 0.507 \times P2 + 55.0$$

$$F4 = \frac{Q100 - F1 \times Cp(T2 - T1)}{\lambda}$$

Where M is a constant liquid held up in the evaporator of 20 kg. Cp and λ are the heat capacity and the latent heat of evaporation of the process liquid which is assumed constant at 0.07 kW/K (kg/min) and 38.5 kW/(kg/min) respectively. The constant C = 4 kg/kPa is used to convert a mass of vapor into a pressure in the vessel.

The steam jacked

The steam side of the evaporator is modeled with three algebraic equations as the dynamics are assumed to be very fast

$$T100 = 0.1538 \times P100 + 90.0$$

$$Q100 = 0.16(F1 + F3)(T100 - T2)$$

$$F100 = \frac{Q100}{\lambda s}$$

Where λs = 36.6 kW/(kg/min) is the latent heat for steam

The condenser

The condenser is also modeled as a set of algebraic equations

$$Q200 = \frac{UA2(T3 - T200)}{1 + \frac{UA2}{2Cp \times F200}}$$

$$T201 = T200 + \frac{Q200}{F200 \times Cp}$$

$$F5 = \frac{Q200}{\lambda}$$

Where UA2 = 6.84 kW/K is the overall heat transfer coefficient times the area and the heat of evaporation λ =38.5 kW/(kg/min).

1.3 Degrees of freedom:

Number of unknown variables=20
Number of independent equations=12
DOF=20-12=8
Therefore 8 variables must be defined. These variables are F2, P100, F200 (manipulated variables) and F1, T1, X1, F3, T200 (Disturbance variables)

Chapter 2 Task 2

2.0 Steady state solution:

The method of using ODE solvers in MATLAB is an effective way to simulate the behavior of the Newell and Lee Evaporator based on its mathematical model. This involves setting arbitrary initial conditions for the differential variables and integrating the system of ODEs over a long enough time period to observe the system's evolution and eventual attainment of a steady state.

MATLAB code for evaporation equations:

```
function dxdt = evp(t,x,p)
% states
X2 = x(1);
P2= x(2);
L2 =x(3);
% inputs
F1=p.F1;
F2=p.F2;
F3=p.F3;
X1=p.X1;
T1=p.T1;
P100=p.P100;
F200=p.F200;
T200=p.T200;


% constant parameters
M=20;
C=4;
Cp=0.07;
lambda=38.5;
lambdas=36.6;
rhoA =20;
lambdas=36.6;
UA2=7.1;
```

```
%model equations

T2=0.5616*P2+0.3126*X2+48.43;
T3=0.507*P2+55.0;
T100=0.1538*P100+90.0;
Q100=0.16*(F1+F3)*(T100-T2);
F100=Q100/lambdas;
F4=(Q100-F1*Cp*(T2-T1))/lambda;
Q200=(UA2*(T3-T200))/(1+(UA2/(2*Cp*F200)));
F5=Q200/lambda;
T201=T200+(Q200/(F200*Cp));

% Differential equations
dX2dt=(F1*X1-F2*X2)*(1/M);
dP2dt=(F4-F5)*(1/C);
dL2dt= (F1-F4-F2)*(1/rhoA);

% output
dxdt = [dX2dt dP2dt dL2dt]';
end
```

MATLAB code for steady state solution:

```
% Input parameters
p.F1=9.7045;   % Feed flow rate(kg/min)
p.F2=2.4;      % Product flow rate(kg/min)
p.F3=35;       % Circulating flow rate(kg/min)
p.X1=5;        % Feed composition(%)
p.T1=40;       % Feed temperature(°C)
p.P100=194.7;  % Steam pressure(kPa)
p.F200=190;    % Cooling water flow rate(kg/min)
p.T200=25;     % Cooling water inlet temperature(°C)



tspan=[0 500];
x0=[1 1 10];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t,x]=ode45(@evp,tspan,x0,options,p)

xss = x(end,:)
save int_xss xss;

plot(t,x), grid on
 xlabel('time')
 ylabel('state variable')
 legend('X_2','P_2','L_2')
```

```
 title('dynamics of the solution')

tspan=[0 500];
load int_xss;
x0=xss;
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t,x]=ode45(@evp,tspan,x0,options,p)

plot(t,x,'LineWidth',2), grid on
 xlabel('time')
 ylabel('state variable')
 legend('X_2','P_2','L_2')
 title('steady state solution')
```

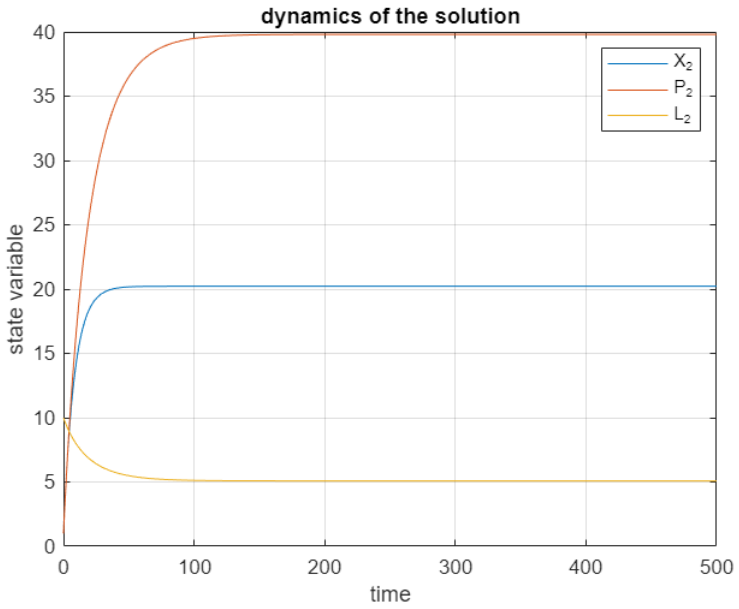

Figure 2-1 Dynamics of the solution

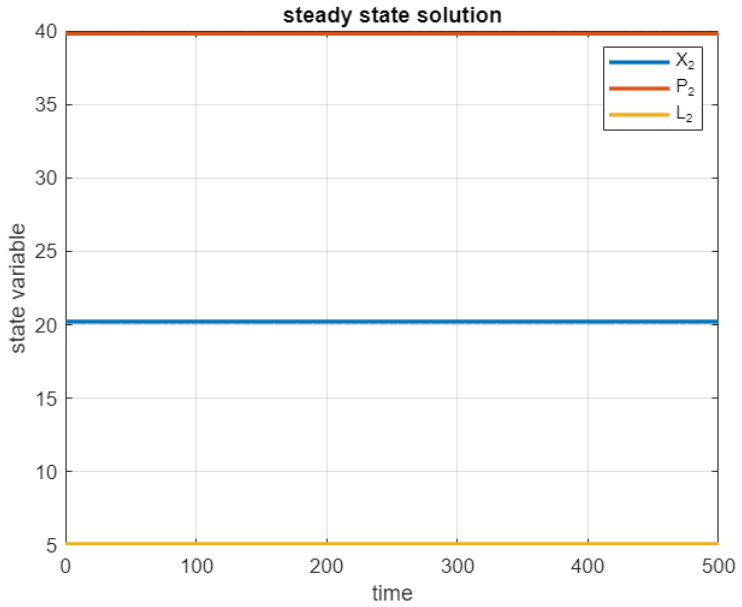

Figure 2-2 Steady state solution

2.1 Dynamic responses of the system states:

1. Simulation scenarios: Step change in some disturbance variables

(a) A step change of + 10 in the feed flow rate , F1
(b) A step change of - 10 % in, F1
(c) A step change of +5 % in the feed composition, X1
(d) A step change of - 5 % in the feed composition, X1
(e) A step change of +10 % in T200 (Cooling water inlet temperature)
(f) A step change of - 10 % in T200 (Cooling water inlet temperature)

MATLAB codes for the above step changes:
Simulation scenarios a & b:

```
% Input parameters
p.F1=9.7045;    % Feed flow rate(kg/min)
p.F2=2.4;       % Product flow rate(kg/min)
p.F3=35;        % Circulating flow rate(kg/min)
p.X1=5;         % Feed composition(%)
p.T1=40;        % Feed temperature(°C)
p.P100=194.7;   % Steam pressure(kPa)
p.F200=190;     % Cooling water flow rate(kg/min)
p.T200=25;      % Cooling water inlet temperature(°C)
```

Disturbance scenario 1(a) (F1+10%)

```
p.F1 = 10.67495;
load int_xss
x0=xss;
tspan=[0 500];
```

```
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t1,x1]=ode45(@evp,tspan,x0,options,p)

save step_F1_inc10 t1 x1
```

Disturbance scenario 1(b) (F1-10%)

```
p.F1 = 8.73405;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t2,x2]=ode45(@evp,tspan,x0,options,p)

save step_F1_dec10 t2 x2

subplot(3,1,1);
plot(t1,x1(:,1),t2,x2(:,1));
grid on
xlabel('time')
ylabel('X_2')
legend('10% increase','10% decrease')
title('dynamic response of X2 to disturbance of F1')

subplot(3,1,2);
plot(t1,x1(:,2),t2,x2(:,2));
grid on
xlabel('time')
ylabel('P_2')
legend('10% increase','10% decrease')
title('dynamic response of P2 to disturbance of F1')

subplot(3,1,3);
plot(t1,x1(:,3),t2,x2(:,3));
grid on
xlabel('time')
ylabel('L_2')
legend('10% increase','10% decrease')
title('dynamic response of L2 to disturbance of F1')
```

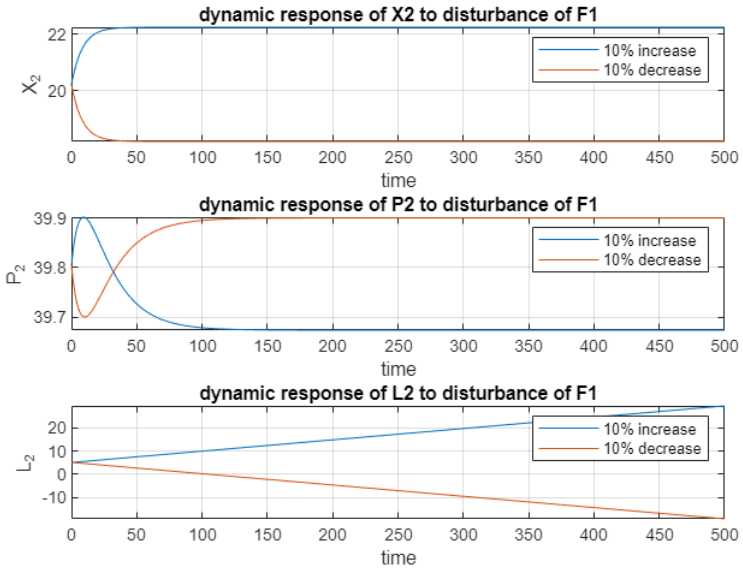

Figure 2-3 Dynamic responses of state variables

Simulation scenarios c & d:

```
% Input parameters
p.F1=9.7045;   % Feed flow rate(kg/min)
p.F2=2.4;      % Product flow rate(kg/min)
p.F3=35;       % Circulating flow rate(kg/min)
p.X1=5;        % Feed composition(%)
p.T1=40;       % Feed temperature(°C)
p.P100=194.7;  % Steam pressure(kPa)
p.F200=190;    % Cooling water flow rate(kg/min)
p.T200=25;     % Cooling water inlet temperature(°C)
```

Disturbance scenario 1(c) (X1+5%)

```
p.X1 = 5.25;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t3,x3]=ode45(@evp,tspan,x0,options,p)

save step_X1_inc5 t3 x3
```

Disturbance scenario 1(d) (X1-5%)

```
p.X1 = 4.75;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
```

```
[t4,x4]=ode45(@evp,tspan,x0,options,p)
save step_X1_dec5 t4 x4

subplot(3,1,1);
plot(t3,x3(:,1),t4,x4(:,1));
grid on
xlabel('time')
ylabel('X_2')
legend('5% increase','5% decrease')
title('dynamic response of X2 to disturbance of X1')

subplot(3,1,2);
plot(t3,x3(:,2),t4,x4(:,2));
grid on
xlabel('time')
ylabel('P_2')
legend('5% increase','5% decrease')
title('dynamic response of P2 to disturbance of X1')

subplot(3,1,3);
plot(t3,x3(:,3),t4,x4(:,3));
grid on
xlabel('time')
ylabel('L_2')
legend('5% increase','5% decrease')
title('dynamic response of L2 to disturbance of X1')
```

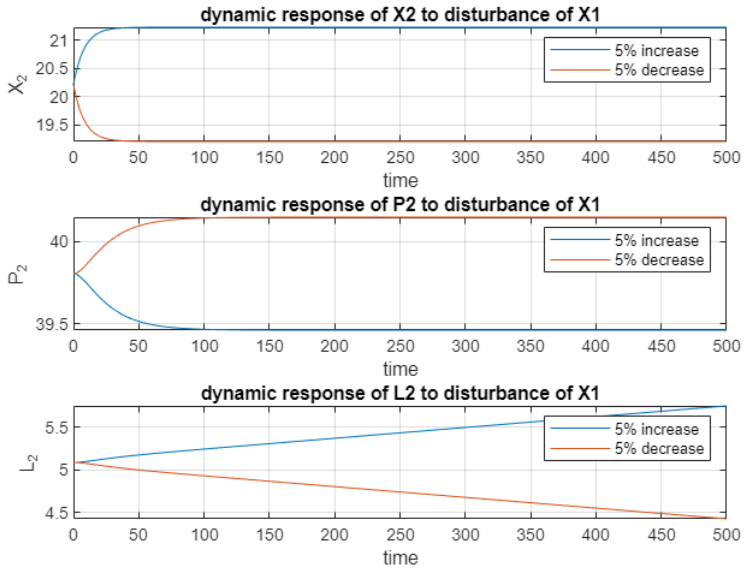

Figure 2-4 Dynamic responses of state variables

Simulation scenarios e & f:

```
% Input parameters
p.F1=9.7045;   % Feed flow rate(kg/min)
p.F2=2.4;      % Product flow rate(kg/min)
p.F3=35;       % Circulating flow rate(kg/min)
p.X1=5;        % Feed composition(%)
p.T1=40;       % Feed temperature(°C)
p.P100=194.7;  % Steam pressure(kPa)
p.F200=190;    % Cooling water flow rate(kg/min)
p.T200=25;     % Cooling water inlet temperature(°C)
```

Disturbance scenario 1(e) (T200+10%)

```
p.T200 = 27.5;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t5,x5]=ode45(@evp,tspan,x0,options,p)
save step_T200_inc10 t5 x5
```

Disturbance scenario 1(f) (T200-10%)

```
p.T200 = 22.5;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t6,x6]=ode45(@evp,tspan,x0,options,p)

save step_T200_dec10 t6 x6

subplot(3,1,1);
plot(t5,x5(:,1),t6,x6(:,1));
grid on
xlabel('time')
ylabel('X_2')
legend('10% increase','10% decrease')
title('dynamic response of X2 to disturbance of T200')

subplot(3,1,2);
plot(t5,x5(:,2),t6,x6(:,2));
grid on
xlabel('time')
ylabel('P_2')
legend('10% increase','10% decrease')
title('dynamic response of P2 to disturbance of T200')
```

```
subplot(3,1,3);
plot(t5,x5(:,3),t6,x6(:,3));
grid on
xlabel('time')
ylabel('L_2')
legend('10% increase','10% decrease')
title('dynamic response of L2 to disturbance of T200')
```

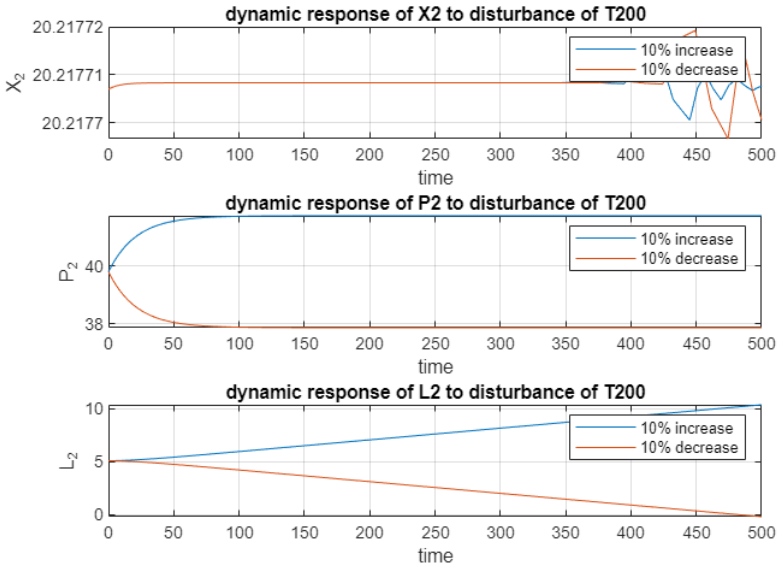

Figure 2-5 Dynamic response of system variables

2.2 Additional Simulation scenarios:

Step change in some manipulated variables

(a) A step change of + 10 in F200
(b) A step change of - 10 % in F200
(c) A step change of +10 % in P100
(d) A step change of - 10 % in P100
(e) A step change of +10 % in F2
(f) A step change of - 10 % in F2

Simulation scenarios a & b:

```
% Input parameters
p.F1=9.7045;   % Feed flow rate(kg/min)
p.F2=2.4;      % Product flow rate(kg/min)
p.F3=35;       % Circulating flow rate(kg/min)
p.X1=5;        % Feed composition(%)
p.T1=40;       % Feed temperature(°C)
p.P100=194.7;  % Steam pressure(kPa)
p.F200=190;    % Cooling water flow rate(kg/min)
```

```
p.T200=25;     % Cooling water inlet temperature(°C)
```

Disturbance scenario 2(a) (F200+10%)

```
p.F200 = 209;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t7,x7]=ode45(@evp,tspan,x0,options,p)
save step_F200_inc10 t7 x7
```

Disturbance scenario 3(b) (F200-10%)

```
p.F200 = 171;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t8,x8]=ode45(@evp,tspan,x0,options,p)
save step_F200_dec10 t8 x8
subplot(3,1,1);
plot(t7,x7(:,1),t8,x8(:,1));
grid on
xlabel('time')
ylabel('X_2')
legend('10% increase','10% decrease')
title('dynamic response of X2 to manipulation of F200')

subplot(3,1,2);
plot(t7,x7(:,2),t8,x8(:,2));
grid on
xlabel('time')
ylabel('P_2')
legend('10% increase','10% decrease')
title('dynamic response of P2 to manipulation of F200')

subplot(3,1,3);
plot(t7,x7(:,3),t8,x8(:,3));
grid on
xlabel('time')
ylabel('L_2')
legend('10% increase','10% decrease')
title('dynamic response of L2 to manipulation of F200')
```

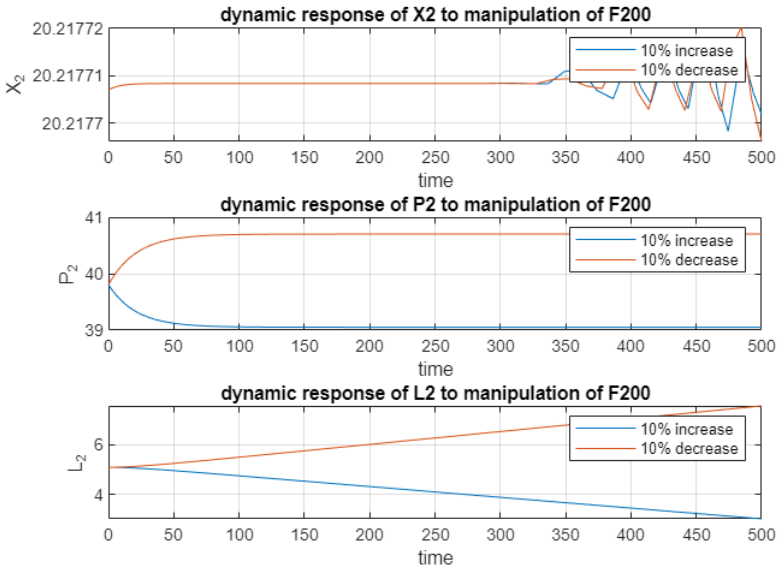

Figure 2-6 Dynamic responses of system variables

Simulation scenarios c & d:

```
% Input parameters
p.F1=9.7045;    % Feed flow rate(kg/min)
p.F2=2.4;       % Product flow rate(kg/min)
p.F3=35;        % Circulating flow rate(kg/min)
p.X1=5;         % Feed composition(%)
p.T1=40;        % Feed temperature(°C)
p.P100=194.7;   % Steam pressure(kPa)
p.F200=190;     % Cooling water flow rate(kg/min)
p.T200=25;      % Cooling water inlet temperature(°C)
```

Disturbance scenario 3(c) (P100+10%)

```
p.P100 = 214.17;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t9,x9]=ode45(@evp,tspan,x0,options,p)
save step_P100_inc10 t9 x9
```

Disturbance scenario 3(d) (P100-10%)

```
p.P100 = 175.23;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t10,x10]=ode45(@evp,tspan,x0,options,p)
```

```
save step_P100_dec10 t10 x10


subplot(3,1,1);
plot(t9,x9(:,1),t10,x10(:,1));
grid on
xlabel('time')
ylabel('X_2')
legend('10% increase','10% decrease')
title('dynamic response of X2 to manipulation of P100')

subplot(3,1,2);
plot(t9,x9(:,2),t10,x10(:,2));
grid on
xlabel('time')
ylabel('P_2')
legend('10% increase','10% decrease')
title('dynamic response of P2 to manipulation of P100')

subplot(3,1,3);
plot(t9,x9(:,3),t10,x10(:,3));
grid on
xlabel('time')
ylabel('L_2')
legend('10% increase','10% decrease')
title('dynamic response of L2 to manipulation of P100')
```

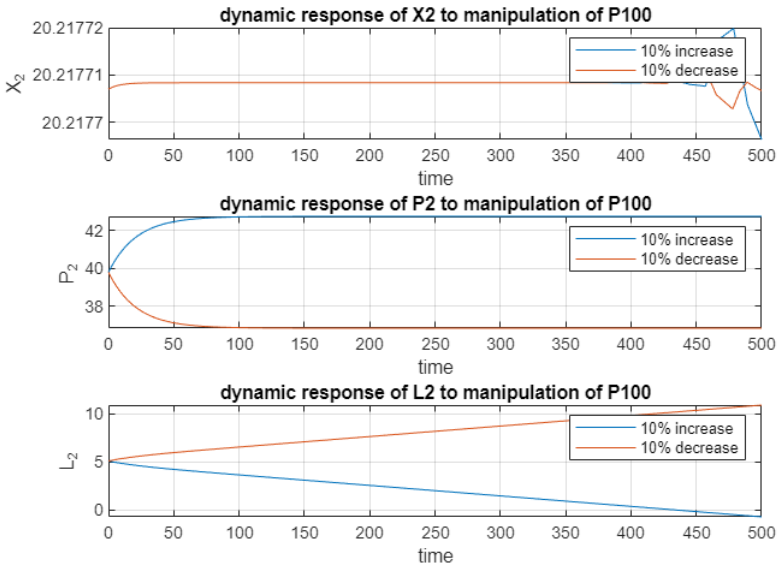

Figure 2-7 Dynamic response of system variables

Simulation scenarios e & f:

```
% Input parameters
p.F1=9.7045;   % Feed flow rate(kg/min)
p.F2=2.4;      % Product flow rate(kg/min)
p.F3=35;       % Circulating flow rate(kg/min)
p.X1=5;        % Feed composition(%)
p.T1=40;       % Feed temperature(°C)
p.P100=194.7;  % Steam pressure(kPa)
p.F200=190;    % Cooling water flow rate(kg/min)
p.T200=25;     % Cooling water inlet temperature(°C)
```

Disturbance scenario 3(e) (F2+10%)

```
p.F2 = 2.64;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t11,x11]=ode45(@evp,tspan,x0,options,p)
save step_F2_inc10 t11 x11
```

Disturbance scenario 3(f) (F2-10%)

```
p.F2 = 2.16;
load int_xss
x0=xss;
tspan=[0 500];
options=odeset('RelTol',1e-6,'AbsTol',[1.0e-6 1.e-06 1.e-06]);
[t12,x12]=ode45(@evp,tspan,x0,options,p)
save step_F2_dec10 t12 x12

subplot(3,1,1);
plot(t11,x11(:,1),t12,x12(:,1));
grid on
xlabel('time')
ylabel('X_2')
legend('10% increase','10% decrease')
title('dynamic response of X2 to manipulation of F2')

subplot(3,1,2);
plot(t11,x11(:,2),t12,x12(:,2));
grid on
xlabel('time')
ylabel('P_2')
legend('10% increase','10% decrease')
title('dynamic response of P2 to manipulation of F2')

subplot(3,1,3);
```

```
plot(t11,x11(:,3),t12,x12(:,3));
grid on
xlabel('time')
ylabel('L_2')
legend('10% increase','10% decrease')
title('dynamic response of L2 to manipulation of F2')
```

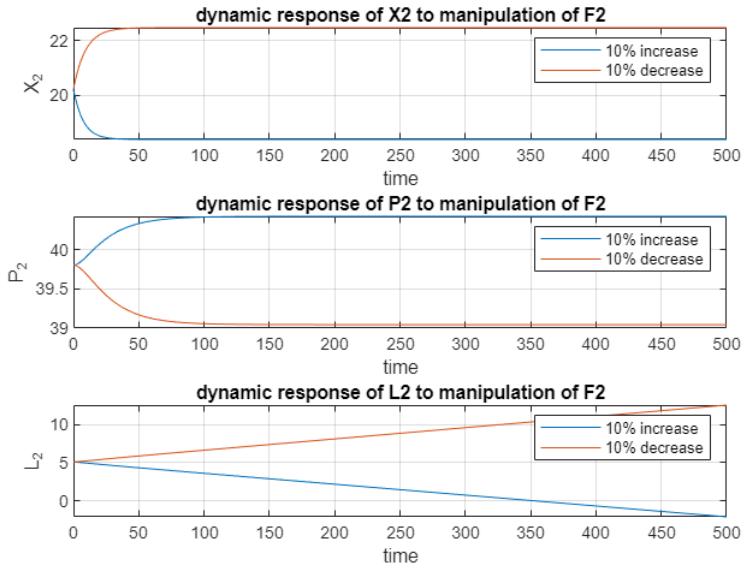

Figure 2-8 Dynamic responses of system variables

Comment: The L2 norms of the output for step inputs are not bounded. This indicates that the process is indeed not BIBO stable. When the input to the process is a step function, the output should settle to a steady-state value after a certain amount of time. However, the L2 norm of the output is not bounded, it means that the output does not settle to a steady-state value but continues to grow without bound. This can lead to instability and unpredictable behavior in the process.

Chapter 3 Task -3

3.0 Simulink model of 3x3 MIMO system

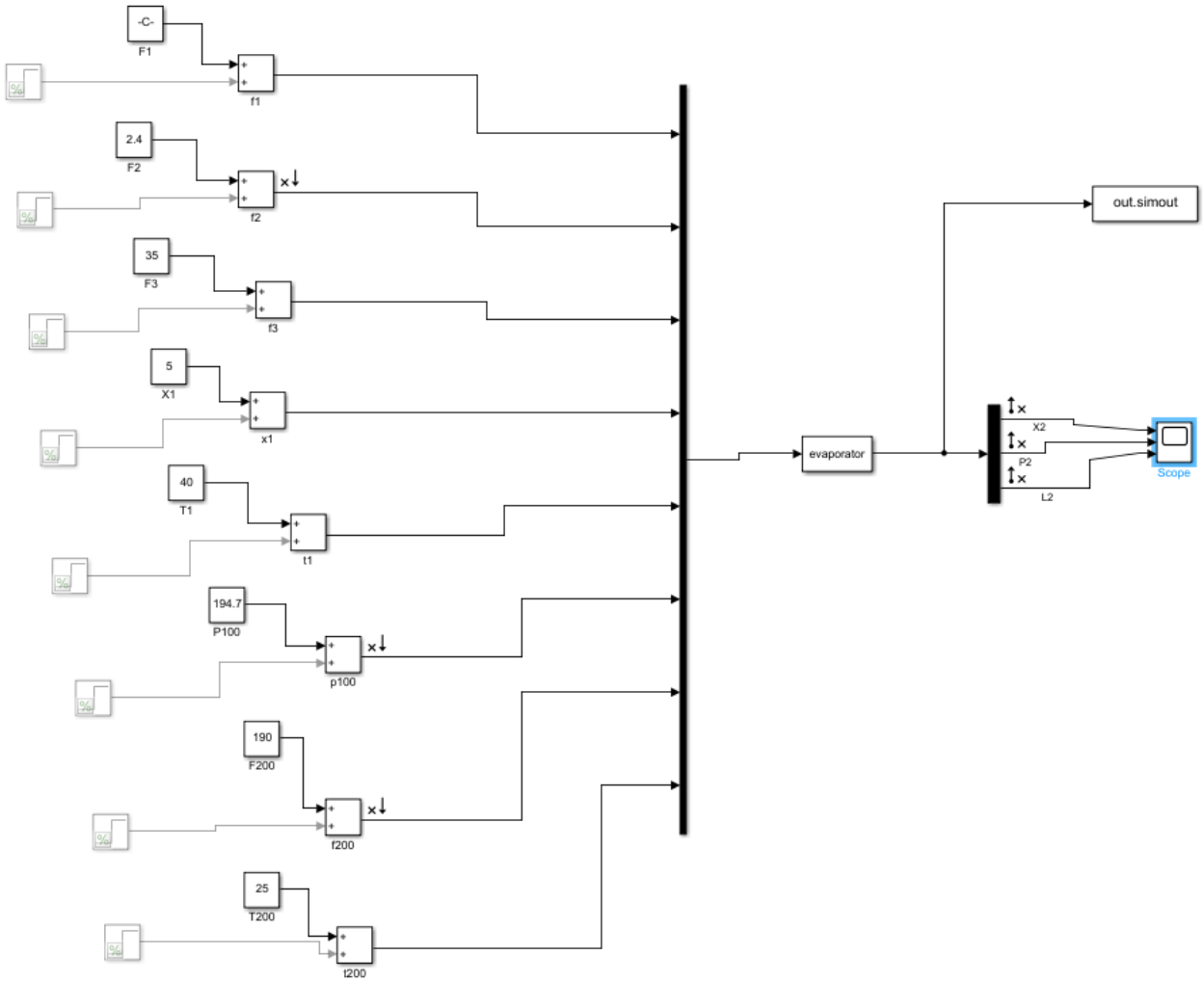

Figure 3-1 Simulink model of 3x3 MIMO system

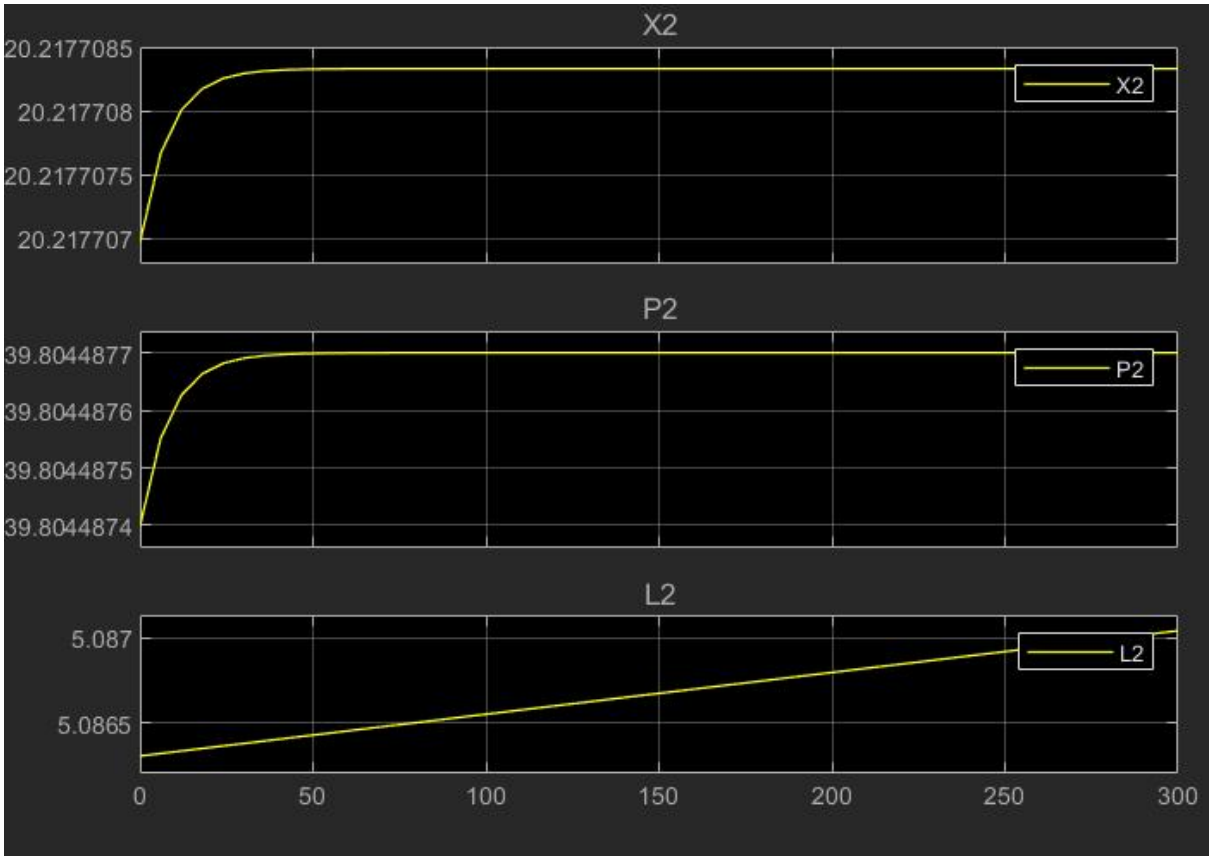

Figure 3-2 Graphical representation of 3x3 MIMO system

```
[A, B,C,D]= ssdata(linsys1)

Lambda=eig(A)

G0 = -C*pinv(A)*B+D %matrix A is singular

RGA0 = G0.*(pinv(G0'))

NI = det(G0)/(G0(1,1)*G0(2,2))
```

The determinant of matrix A is zero, indicating that the matrix is singular and does not have an inverse. In such cases, finding the inverse using the **inv** function is not possible. Instead, we can use the **pinv** function, which calculates the pseudo-inverse of the matrix. The pseudo-inverse is a generalized inverse that can be computed for any matrix, including singular matrices.

A =

-0.1200	0	0
-0.0159	-0.0470	0
0.0032	0.0057	0

B =

-1.0109	0	0
0	-0.0020	0.0071
-0.0500	0	-0.0014

C =

1.0000	0	0
0	1.0000	0
0	0	1.0000

D =

0	0	0
0	0	0
0	0	0

Lambda =

0
-0.0470
-0.1200

G0 =

```
   -8.4189    0.0000    0.0000
    3.0011   -0.0425    0.1534
         0         0         0
```

RGA0 =

```
    0.9999   -0.0000    0.0001
    0.0001    0.0711    0.9288
         0         0         0
```

NI =0

The computation of the Relative Gain Array (RGA) using **pinv(G0)** is successful. However, it is observed that the general property of the RGA is not fulfilled.

The stability of a system can be determined by analyzing the eigenvalues of its system matrix. If all eigenvalues have negative real parts, the system is stable, and if any eigenvalue has a positive real part, the system is unstable. Additionally, if an eigenvalue is equal to zero, it indicates that the corresponding variable exhibits integrating behavior. In the given scenario, the eigenvalues suggest that variable L2 exhibits integrating behavior.

To determine which manipulated input affects the output L2, we can examine the step response of the linearized system. By applying a step input to each manipulated input individually and observing the response of L2, we can identify the influence of each input on the output.

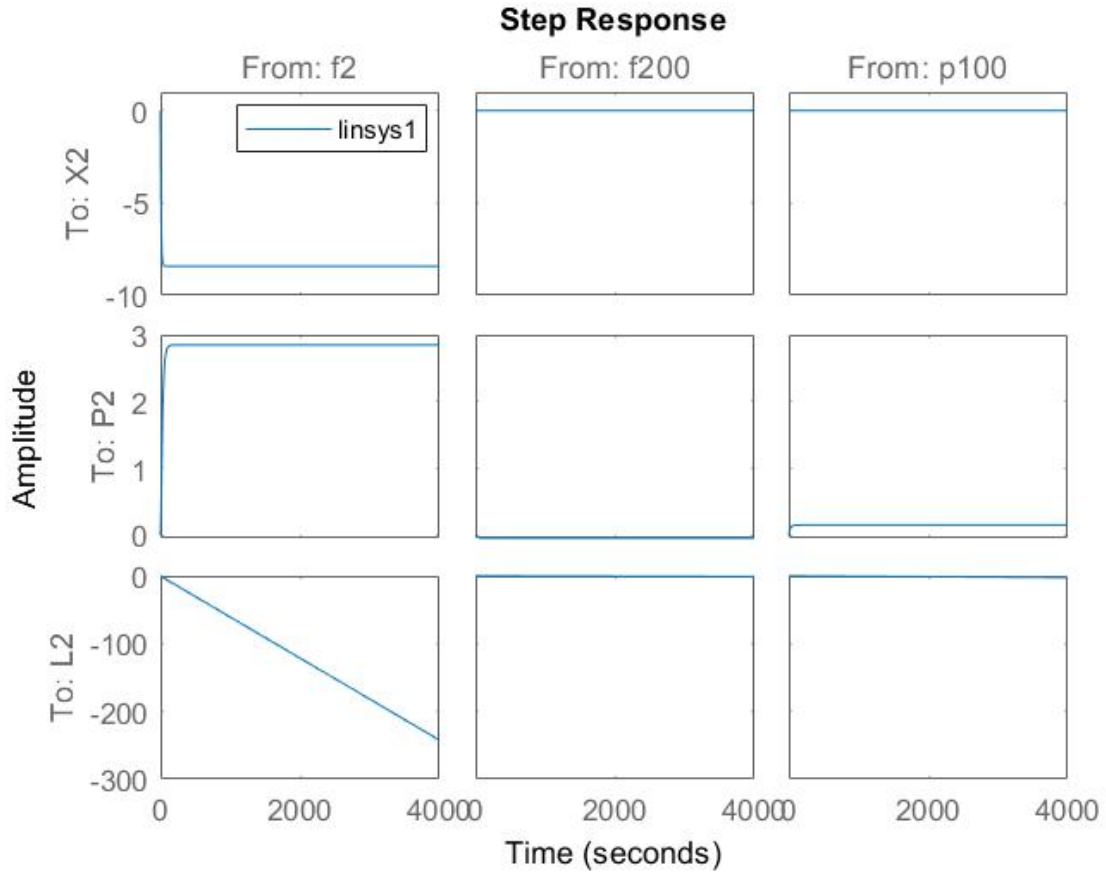

Figure 3-3 Linearization of 3x3 MIMO system

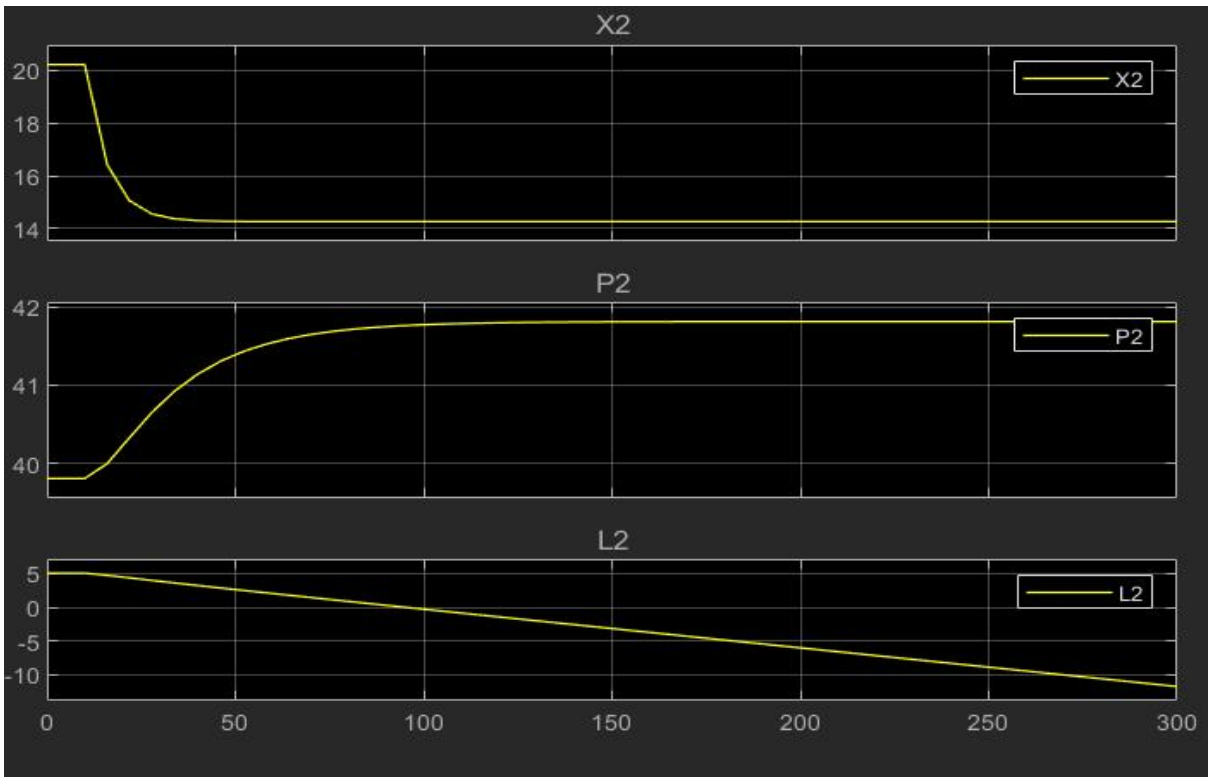

Figure 3-4 Graphical representation step change in F2 of 3x3 MIMO system

After analyzing the step response values, it was observed that L2 and F2 exhibit an inverse relationship. Therefore, it is appropriate to use a proportional controller to close the loop and control the system. In order to achieve negative feedback, a direct-acting process should be controlled using a reverse-acting controller. As a result, a negative gain was applied to the controller to ensure the desired negative feedback configuration.

After implementing the closed-loop configuration on the system, the new control degrees of freedom are reduced to 2. The stability of the system was assessed by performing linearization on this 2x2 MIMO (Multiple-Input Multiple-Output) system and calculating its eigenvalues.

3.1 Simulink model of 2x2 MIMO system

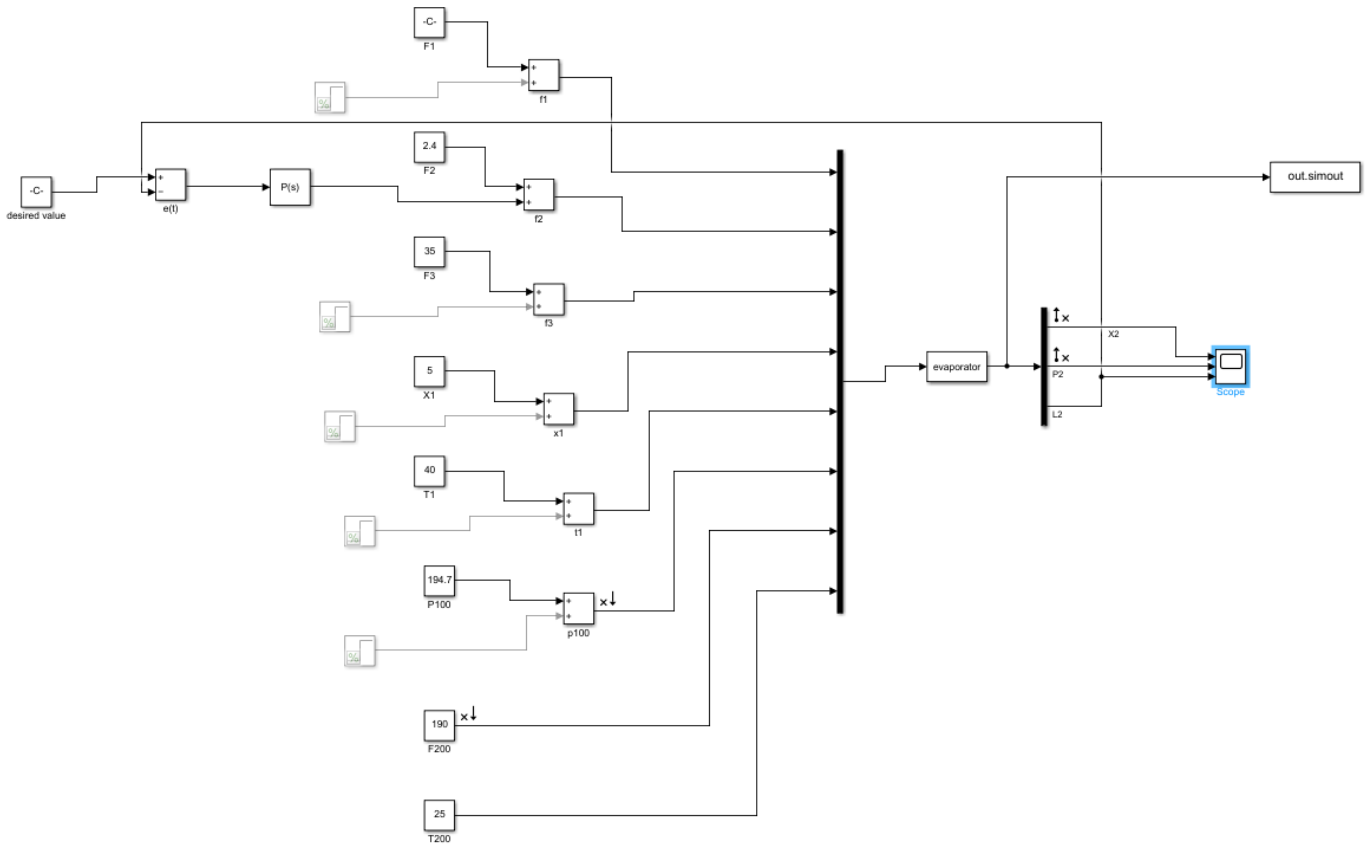

Figure 3-5 Simulink model of 2x2 MIMO system

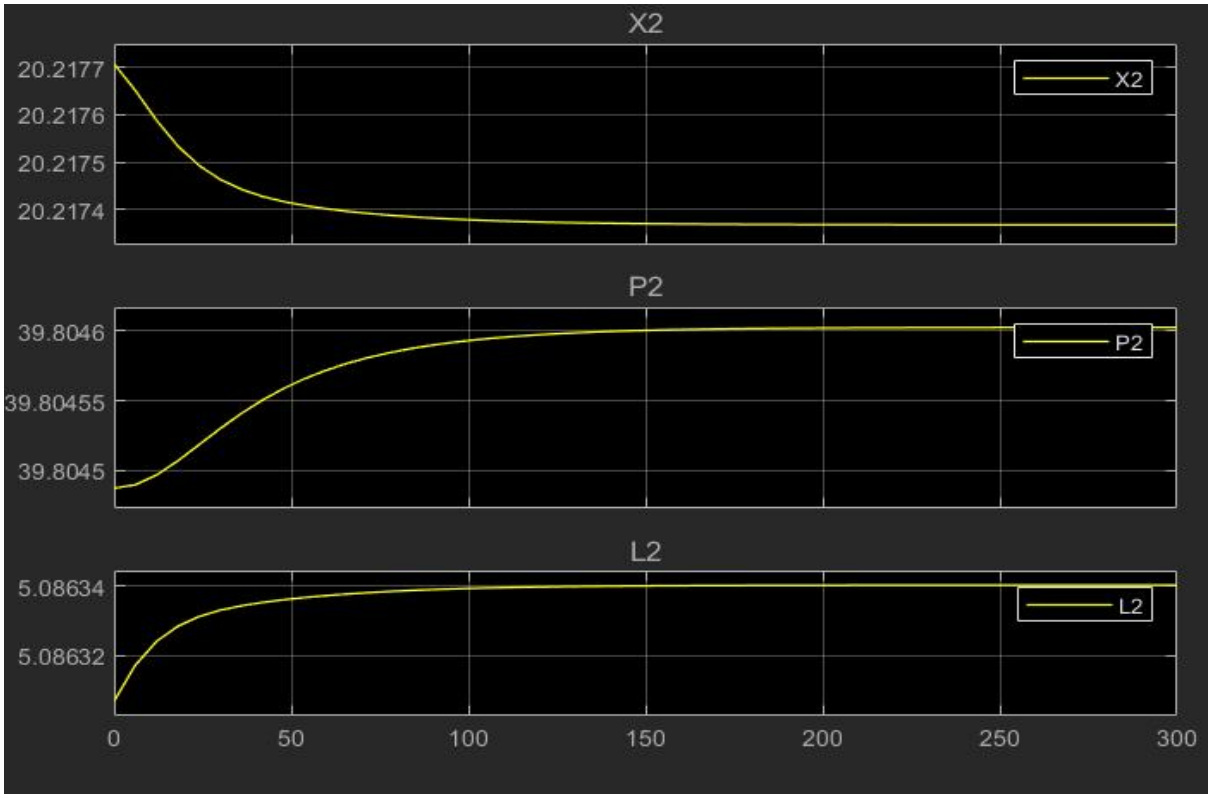

Figure 3-6 Steady state representation of 2x2 MIMO system

```
[A, B,C,D]= ssdata(linsys2)

Lambda=eig(A)

G0 = -C*pinv(A)*B+D

RGA0 = G0.*(pinv(G0'))

NI = det(G0)/(G0(1,1)*G0(2,2))
```

A =

-0.1200	0	-1.0109
-0.0159	-0.0470	0
0.0032	0.0057	-0.0500

B =

```
         0         0
   -0.0020    0.0071
         0   -0.0014
```

C =

```
    1.0000         0         0
         0    1.0000         0
```

D =

```
     0     0
     0     0
```

Lambda =

```
  -0.0940 + 0.0540i
  -0.0940 - 0.0540i
  -0.0291 + 0.0000i
```

G0 =

```
    0.0343    0.0781
   -0.0547    0.1256
```

RGA0 =

```
    0.5020    0.4980
    0.4980    0.5020
```

NI = 1.9920

All the real parts of the eigenvalues are negative, indicating that the system is stable. The Relative Gain Array (RGA0) was calculated to determine the best input/output pairs. The diagonal eigen values in the RGA0 are greater than 0.5 and near to 1, based on this F200-X2 and P100-P2 are the optimal I/O pairs.

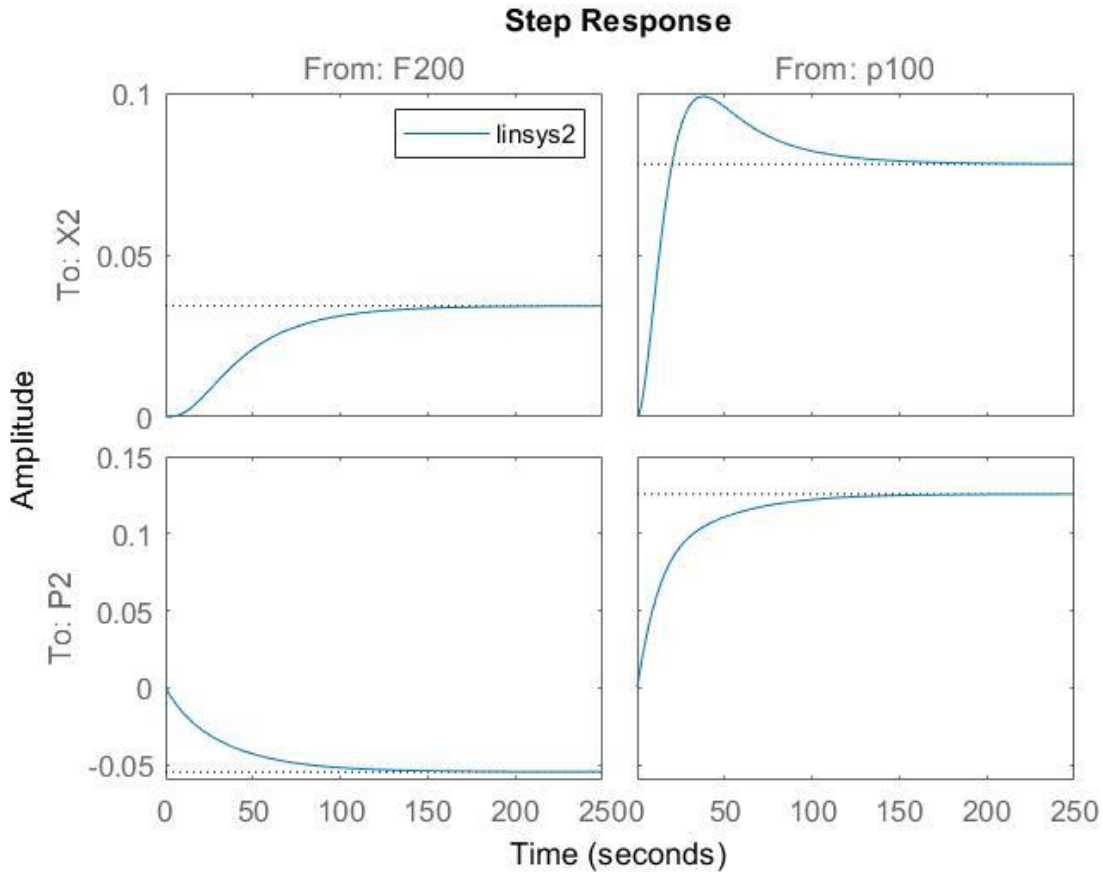

Figure 3-7 3 Linearization of 2x2 MIMO system

The Niederlinski Index is positive for the selected I/O pairs which indicates that the system is stable.

Chapter 4 Task-4

4.0 Controller design

We have converted the decentralized MIMO pairs into appropriate SISO pairs in the above task. The optimal pairs we have taken F200-X2 & P100-P2 as our input/output pairs.

a) Design SISO controllers
   SISO controllers for the selected I/O pairs for the 2x2 system can be designed by using the following methods.
   1) Ziegler-Nichols open loop method
   2) Relay Feedback as PID tuning method

4.1 Feedback Control Design for a SISO System based on Ziegler Nichols Method Open Loop Method:

```
load("linsys2.mat")
G1=tf(linsys2)
```

G1 =

From input "F200" to output...
1.169e-05
X2: ----------
s^3 + 0.217 s^2 + 0.01721 s + 0.0003414

-0.002025 s^2 - 0.0003442 s - 1.866e-05
P2: ----------
s^3 + 0.217 s^2 + 0.01721 s + 0.0003414

From input "p100" to output...
0.001444 s + 2.665e-05
X2: ----------
s^3 + 0.217 s^2 + 0.01721 s + 0.0003414

0.007143 s^2 + 0.001214 s + 4.286e-05
P2: ----------
s^3 + 0.217 s^2 + 0.01721 s + 0.0003414

Name: Linearization at model initial condition
Continuous-time transfer function.

```
figure(1)
step(G1)
[y,t] = step(G1);
```

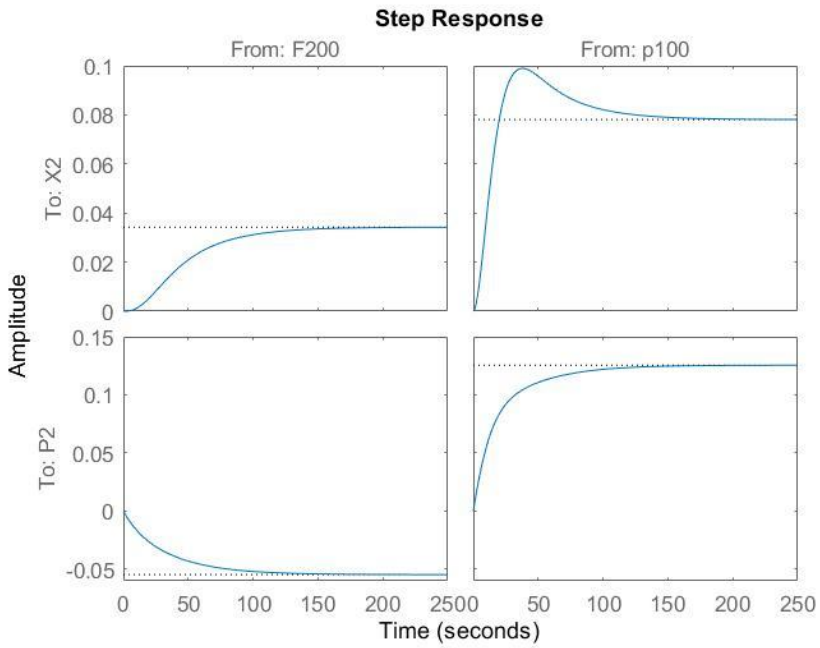

Figure 4-1 Step response of G1

```
figure(2)
plot(t,y(:,1,1),'r')
legend('X_2')
xlabel('time (min)')
ylabel('X2')
title('Step Change in F200 to Change in X2')
figure(3)
plot(t,y(:,2,2),'g')
legend('P_2')
xlabel('time (min)')
ylabel('P2')
title('Step Change in P100 to Change in P2')
```

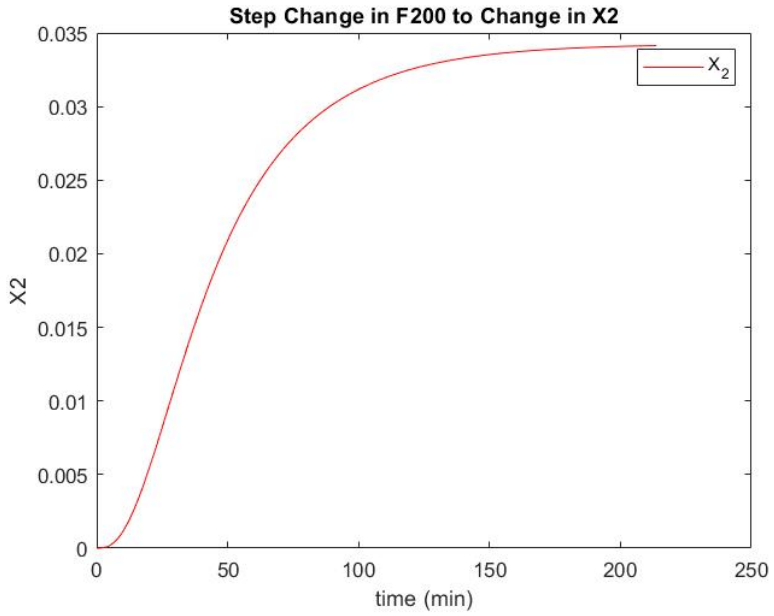

Figure 4-2 Step response of X2

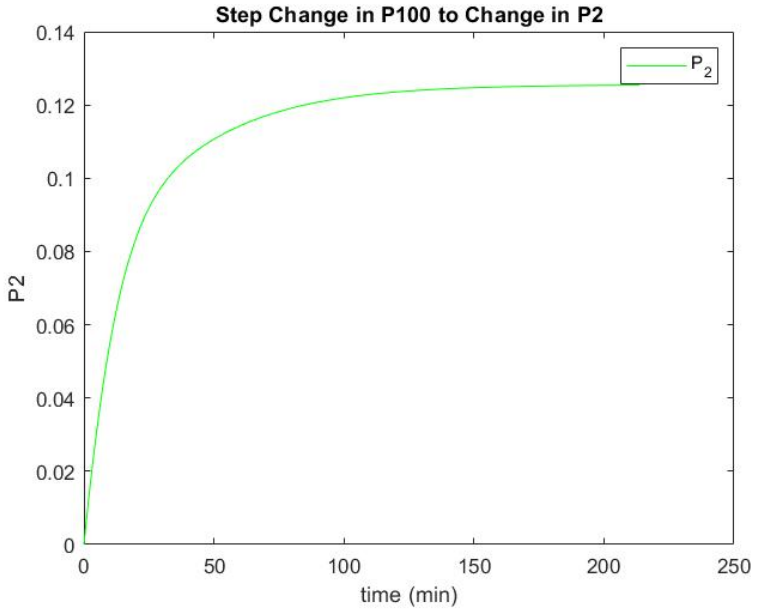

Figure 4-3 Step response of P2

```
slope1 = gradient(y(:,1,1),t)
slope2 = gradient(y(:,2,2),t)

[tslope1,idx1] = max(slope1);
```

```
tIP1 = t(idx1);
yIP1 = y(idx1,1,1);
[tslope2,idx2] = max(slope2);
tIP2 = t(idx2);
yIP2 = y(idx2,2,2);

figure(2)
plot(t,y(:,1,1),'r')
xlabel('time (min)')
ylabel('X2')
title('Step Change in F200 to Change in X2')
hold on
plot(tIP1,yIP1,'r*'),grid on
legend('X_2','Inflection Point')
hold off

figure(3)
plot(t,y(:,2,2),'g')
xlabel('time (min)')
ylabel('P2')
title('Step Change in P100 to Change in P2')
hold on
plot(tIP2,yIP2,'g*'),grid on
legend('P_2','Inflection Point')
hold off
```

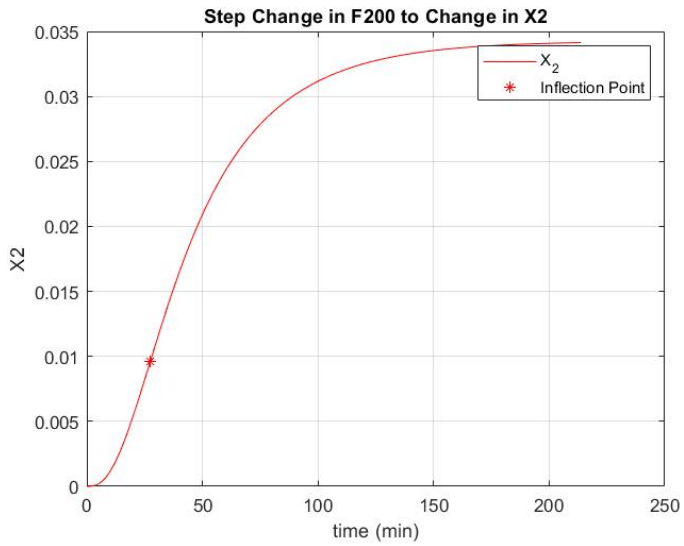

Figure 4-4 Inflection point in step response of X2

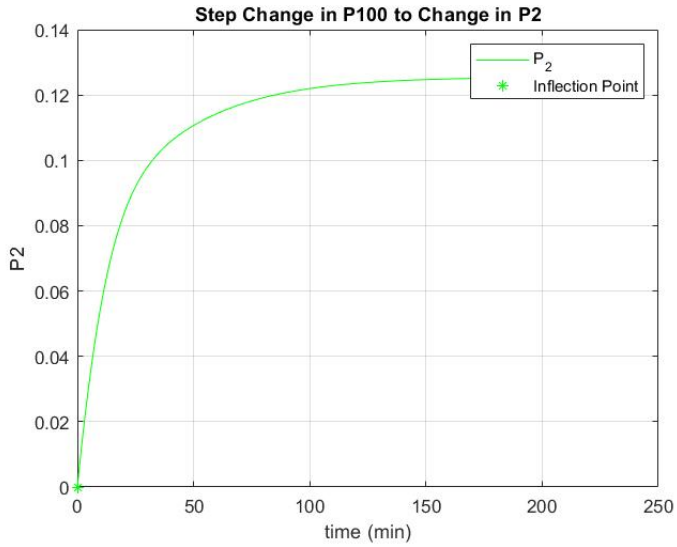

Figure 4-5 Inflection point in step response of P2

```
yTangentLine1 = tslope1*(t-tIP1) + yIP1;
yTangentLine2 = tslope2*(t-tIP2) + yIP2;
figure(2)
plot(t,y(:,1,1),'r')
xlabel('time (min)')
ylabel('X2')
title('Step Change in F200 to Change in X2')
hold on
plot(tIP1,yIP1,'r*'),grid on
plot(t,yTangentLine1,'b')
legend('X_2','Inflection Point','Tangent')
hold off
figure(3)
plot(t,y(:,2,2),'g')
xlabel('time (min)')
ylabel('P2')
title('Step Change in P100 to Change in P2')
hold on
plot(tIP2,yIP2,'g*'),grid on
plot(t,yTangentLine2,'b')
legend('P_2','Inflection Point','Tangent')
hold off
```

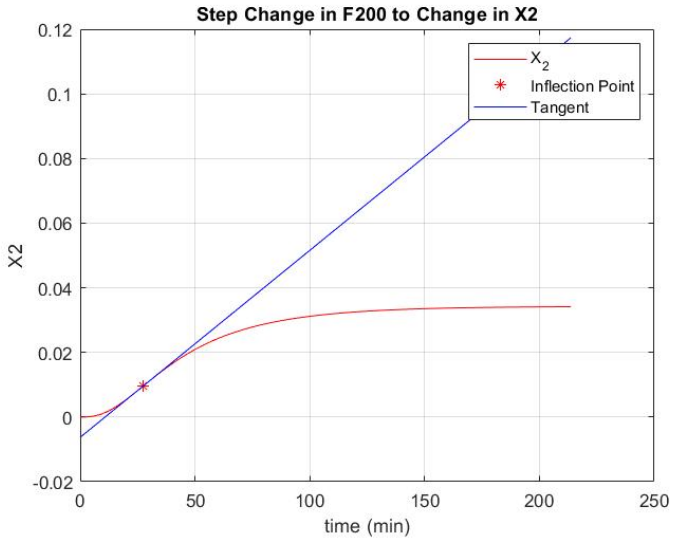

Figure 4-6 Tangent line to the step response curve of X2 at inflection point

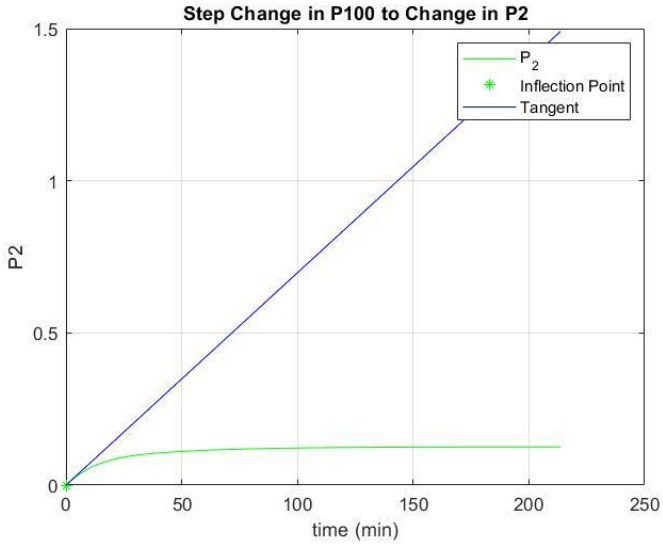

Figure 4-7 Tangent line to the step response curve of P2 at inflection point

Estimation of parameters of transfer function model
Time delay (Td):

```
Td1 = tIP1-(yIP1/tslope1)
Td2 = tIP2-(yIP2/tslope2)
```

Td1 = 10.8451
Td2 = 0
P100-P2 pair doesn't have any time delay. Therefore let's assume Td=1

```
%since Td2=0, lets assume a time delay
Td2=1;
figure(2)
plot(t,y(:,1,1),'r')
xlabel('time (min)')
ylabel('X2')
title('Step Change in F200 to Change in X2')
hold on
plot(tIP1,yIP1,'r*'),grid on
plot(t,yTangentLine1,'b')
plot(Td1,0,'k.','MarkerSize',15)
legend('X_2','Inflection Point','Tangent','Time delay')
hold off
figure(3)
plot(t,y(:,2,2),'g')
xlabel('time (min)')
ylabel('P2')
title('Step Change in P100 to Change in P2')
hold on
plot(tIP2,yIP2,'g*'),grid on
plot(t,yTangentLine2,'b')
plot(Td2,0,'k-','MarkerSize',15)
legend('P_2','Inflection Point','Tangent','Time delay')
hold off
```

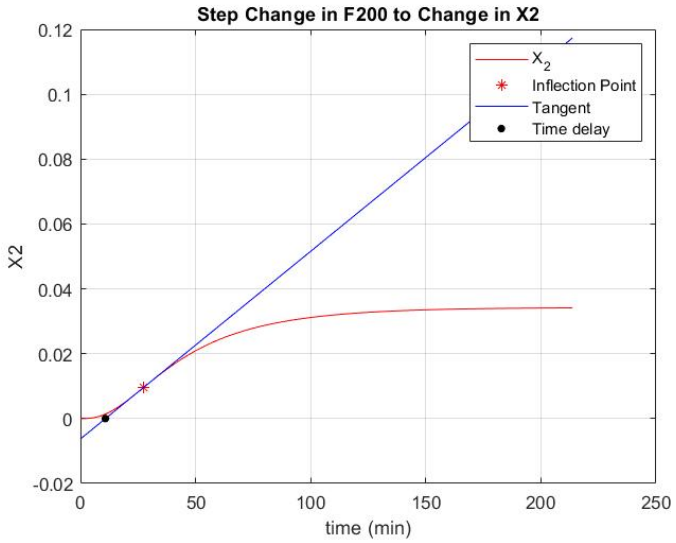

Figure 4-8 Step response of X2 with time delay

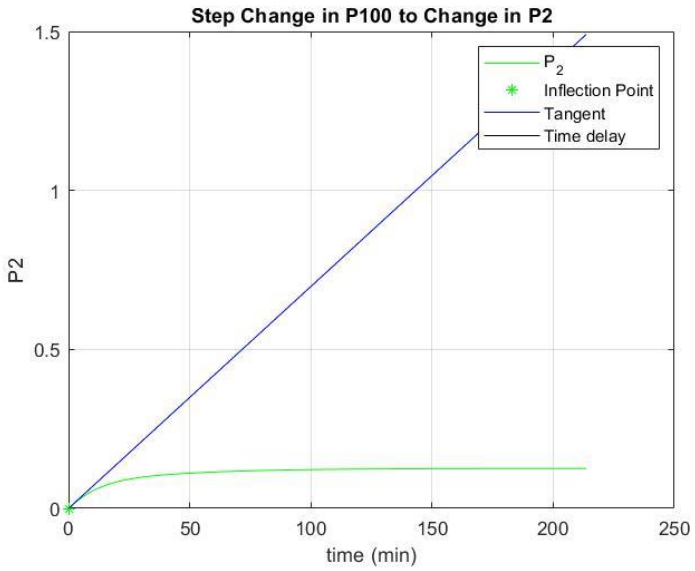

Figure 4-9 Step response of P2 with time delay

Time constant (Tau):

```
Tau1 = y(end,1,1)/tslope1
Tau2 = y(end,2,2)/tslope2
```

Tau1 = 59.0062
Tau2 = 17.9743

Process gain (K):

```
K1 =y(end,1,1)/1
K2 =y(end,2,2)/1
```

K1 = 0.0341
K2 = 0.1254

Defining approximate transfer function and plotting step response:

```
G11 = tf(K1, [Tau1 1],'InputDelay',Td1)
[y1,t1] = step(G11);

G22 = tf(K2, [Tau2 1],'InputDelay',Td2)
[y2,t2]= step(G22);

figure(4)
plot(t, y(:,1,1), 'r')
xlabel('time (min)')
ylabel('X2')
title('Step Change in F200 to Change in X2')
hold on
plot(tIP1, yIP1, 'r*')
grid on
```

```
 plot(t, yTangentLine1, 'b')
 plot(Td1, 0, 'k.', 'MarkerSize', 15)
 plot(t1, y1, 'c--')
 legend('X_2', 'Inflection Point', 'Tangent', 'Time delay', 'Approximated
FOPTD')
 hold off

 figure(5)
 plot(t, y(:,2,2), 'g')
 xlabel('time (min)')
 ylabel('P2')
 title('Step Change in P100 to Change in P2')
 hold on
 plot(tIP2, yIP2, 'g*')
 grid on
 plot(t, yTangentLine2, 'b')
 plot(Td2, 0, 'k-', 'MarkerSize', 15)
 plot(t2, y2, 'y--')
 legend('P_2', 'Inflection Point', 'Tangent', 'Time delay', 'Approximated
FOPTD')
 hold off
```

G11 =

$$\exp(-10.8s) * \frac{0.03414}{59.01\,s + 1}$$

Continuous-time transfer function.

G22 =

$$\exp(-1s) * \frac{0.1254}{17.97\,s + 1}$$

Continuous-time transfer function.

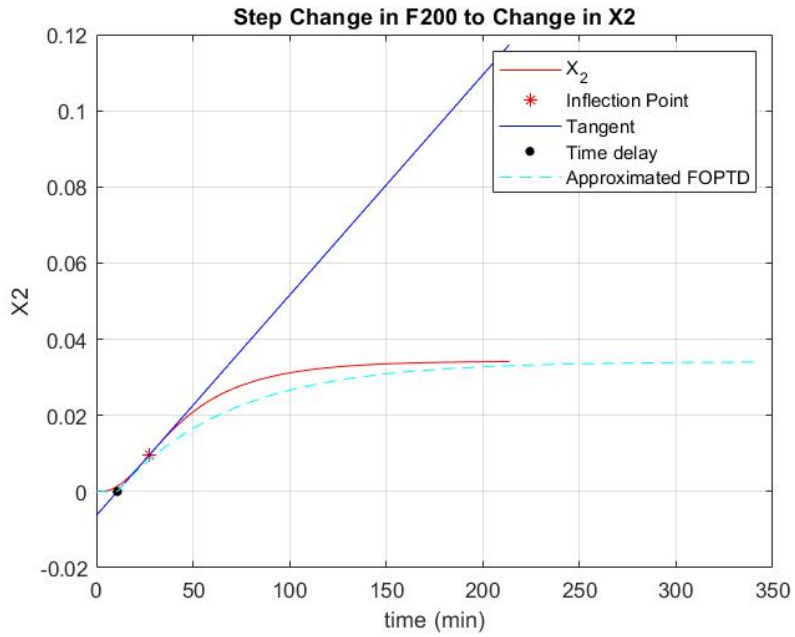

Figure 4-10 Step response of X2 (Approximated)

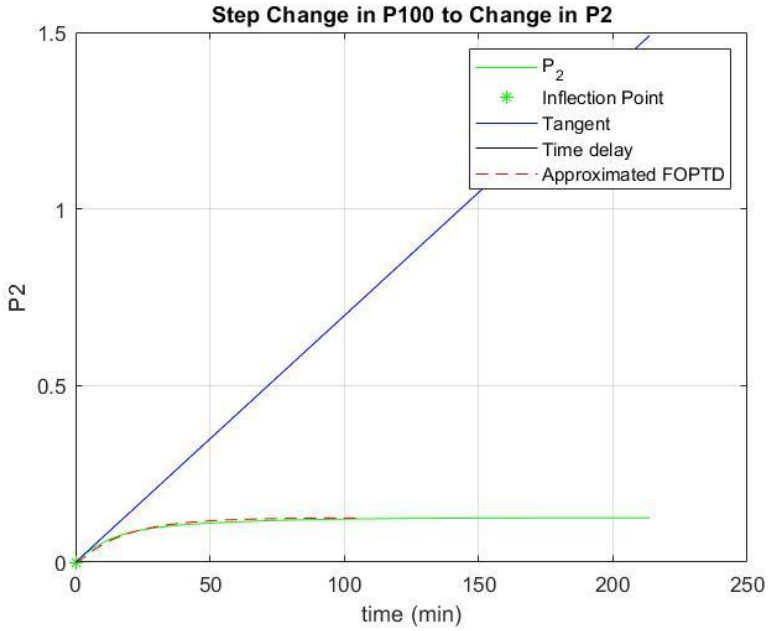

Figure 4-11 Step response of P2 (Approximated)

Calculating PID controller parameters (Time constant form) by following Ziegler-Nichols open loop method :

```
Kp1 = 1.2*Tau1/(K1*Td1)
Kp2 = 1.2*Tau2/(K2*Td2)

TI1=2*Td1
```

```
TI2=2*Td2

TD1= 0.5*Td1
TD2= 0.5*Td2
```

Converting from time constant form to parallel form:

```
% converting from time constant form to parallel form
KI1= Kp1/TI1
KI2= Kp2/TI2

KD1= Kp1*TD1
KD2= Kp2*TD2
```

Kp1 = 191.2477
Kp2 = 171.9717
TI1 = 21.6902
TI2 = 2
TD1 = 5.4226
TD2 = 0.5000
KI1 = 8.8172
KI2 = 85.9859
KD1 = 1.0371e+03
KD2 = 85.9859

These controller parameters are implemented in the Simulink PID block.

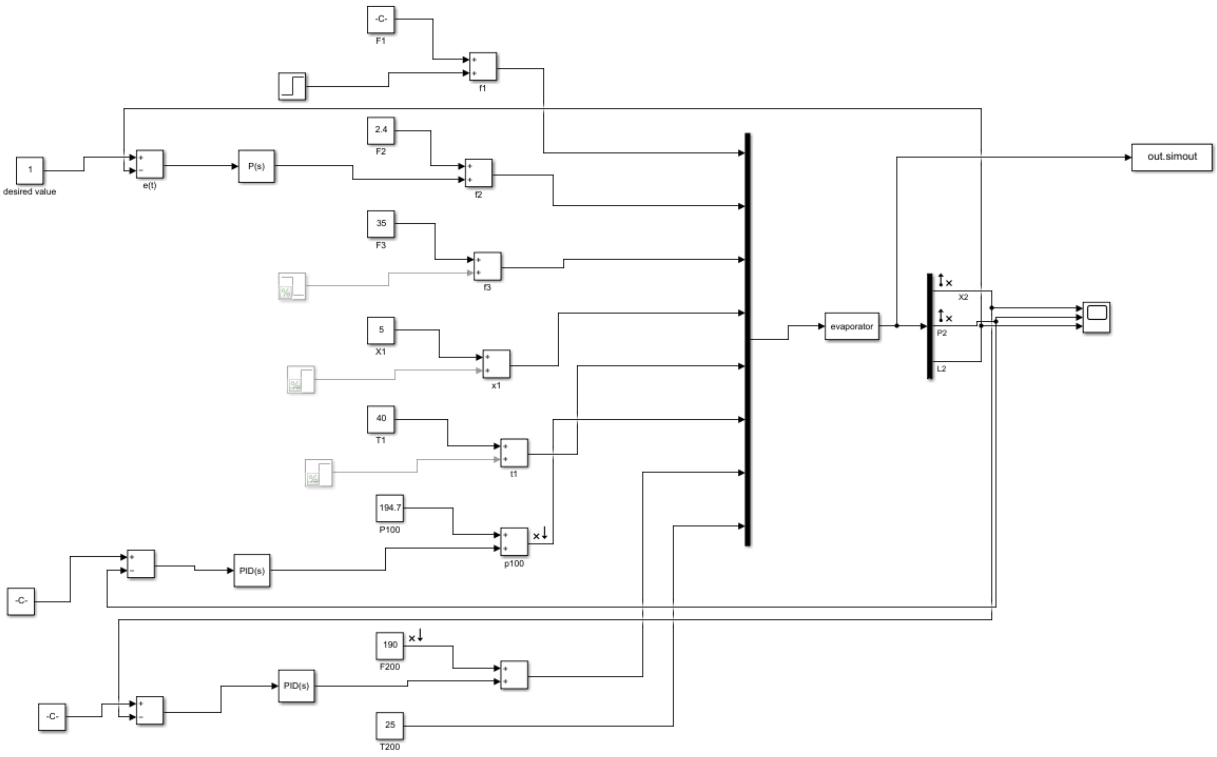

Figure 4-12 Simulink Model with PID controllers (Ziegler-Nichols open loop method)

4.2 Relay Feedback as PID tuning method in Simulink

This method involves utilizing a relay to provide an oscillatory input to the system and analyzing the resulting output. It is crucial for the "error signal" to surpass the measurement noise. To address this, we took into account the anticipated amplitude of the process output oscillations and adjusted the error signal accordingly to fall within that range. As part of our efforts to implement Relay Feedback, we made modifications to the default parameters of the Relay block. Notably, the "switch on/off" parameters directly influence the error. We proceeded with the assumption that it is feasible to obtain an error signal magnitude of 1& 0.5 for the pressure variables P2 and X2 respectively. These considerations and adjustments contribute to the overall implementation and effectiveness of Relay Feedback as a PID tuning method.

4.2.1 Relay implementation for P100 and P2 pair

For the Relay block:
Switch on point: 1
Switch off point: -1

Output on/off parameters related to the output signal of the Relay block should be manipulated until we get the required output. We start with a small value such as
Output when on: 1
Output when off: -1

We run the simulation and keep on increasing the Output values in a fixed proportions in order to get constant oscillation in the output (with constant amplitude and frequency)
Output when on: 16
Output when off: -16

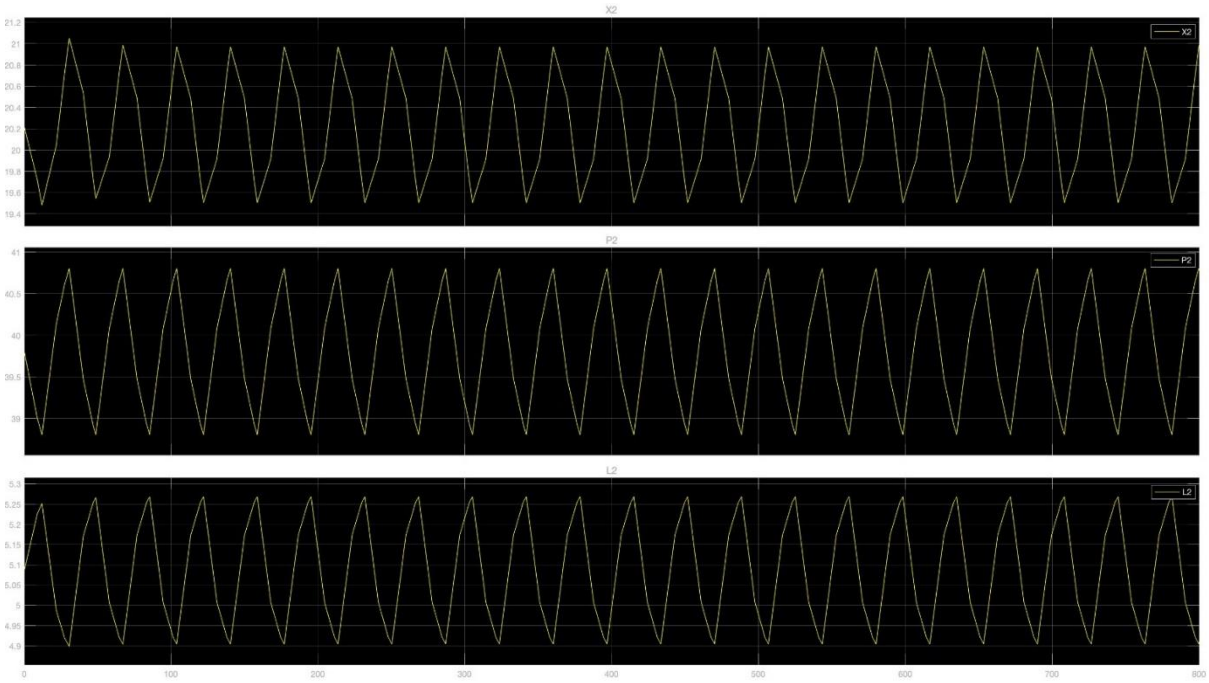

Figure 4-13 Output Oscillations for P100-P2

From the above graph we obtained the following values
Pu = 37.011 s
h = 16
PID Controller values:
Kp = 0.6*Ku = 12.378
TI = 0.5*Pu = 18.5055
Td = 0.125*Pu = 4.626

a = 0.9875
Ku= $4h/\pi a$ = 20.63

4.2.2 Relay implementation for F200 and X2 pair

For the Relay block, we get the following values
Switch on point: 0.5
Switch off point: -0.5
Output when on: 20
Output when off: -20

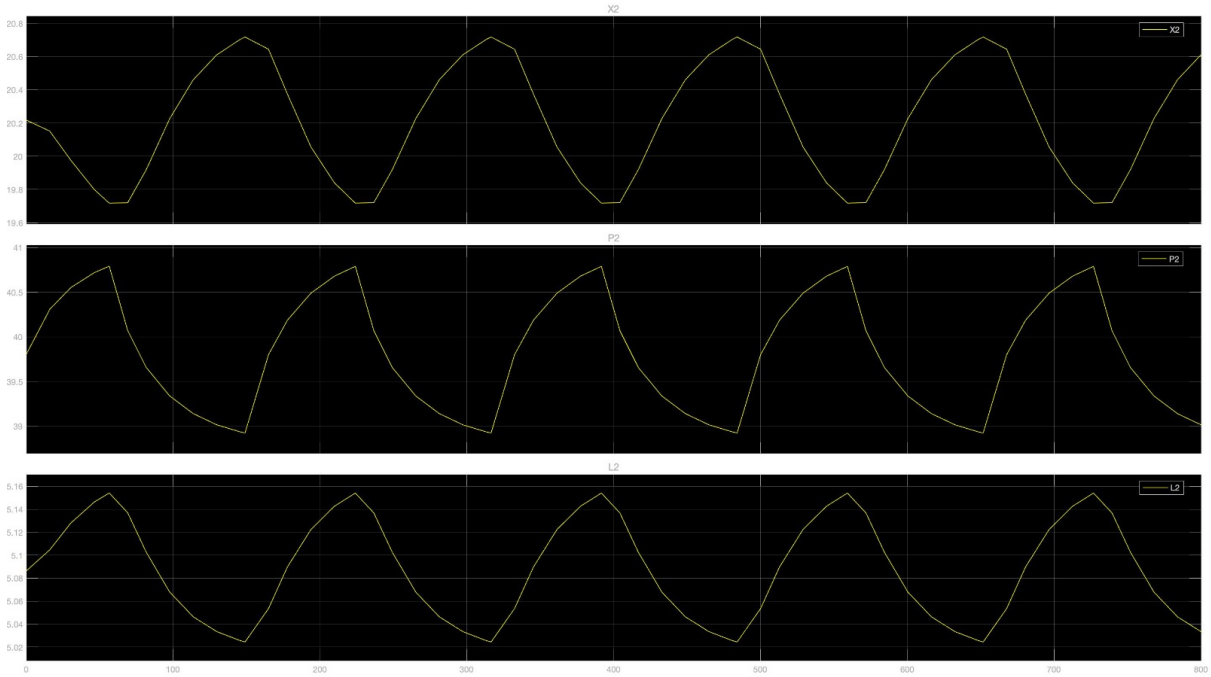

Figure 4-14 Output Oscillations for F200-X2

From the above graph we obtained the following values

Pu = 169.570 s

a = 0.4962

h = 20

Ku= $4h/\pi a$ = 51.32

PID Controller values:

Kp = 0.6*Ku = 30.792

TI = 0.5*Pu = 84.785

Td = 0.125*Pu = 21.20

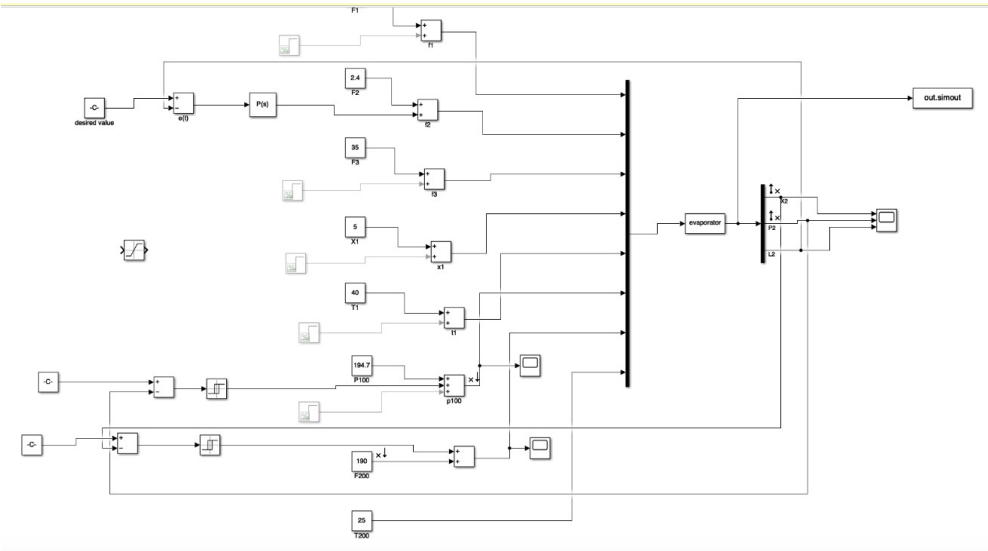

Figure 4-15 Simulink Model with Relay Blocks

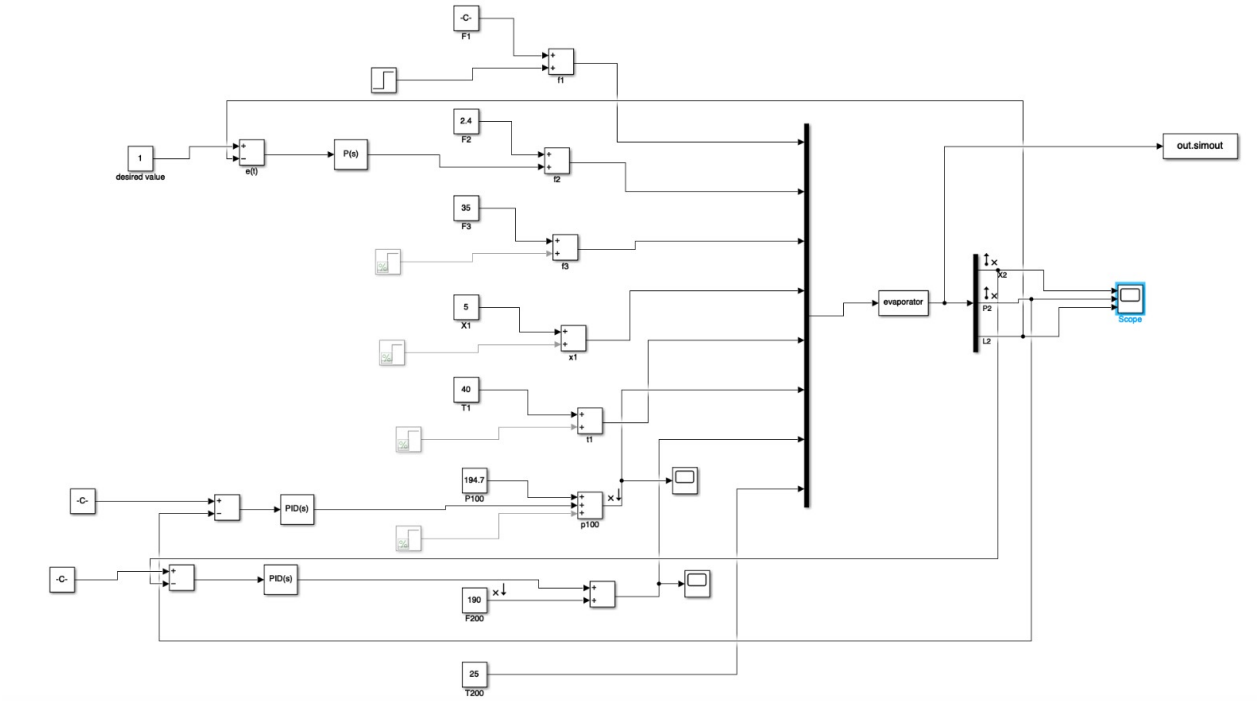

Figure 4-16 Simulink Model with PID controllers (Relay Feedback as PID tuning method)

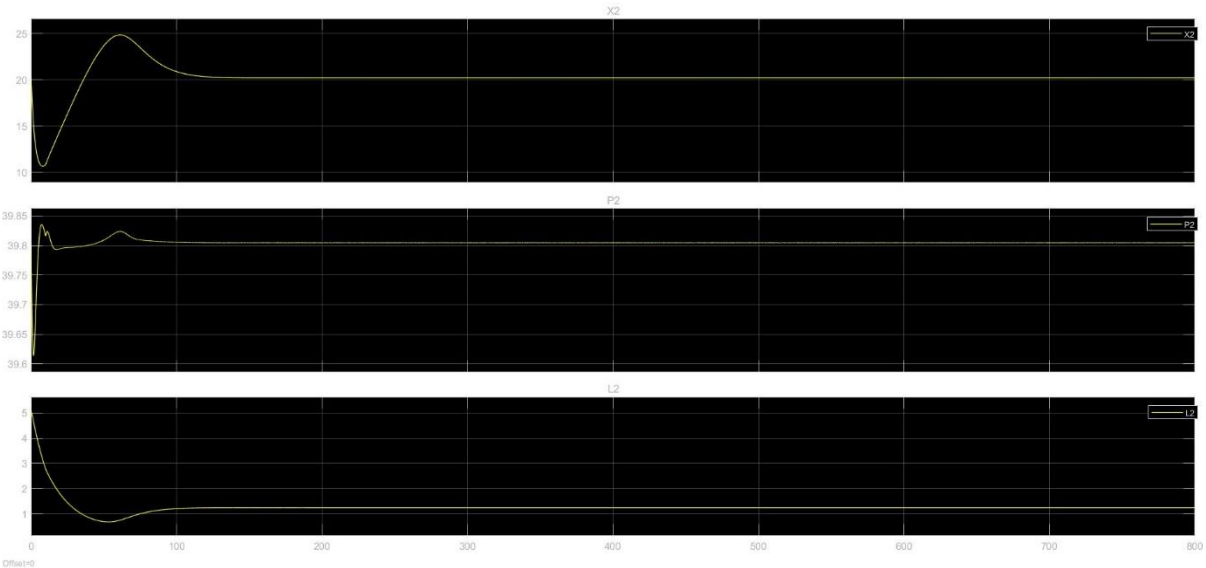

Figure 4-17 Output Graph with a step change in F1 (Ziegler-Nichols open loop method as PID tuning method)

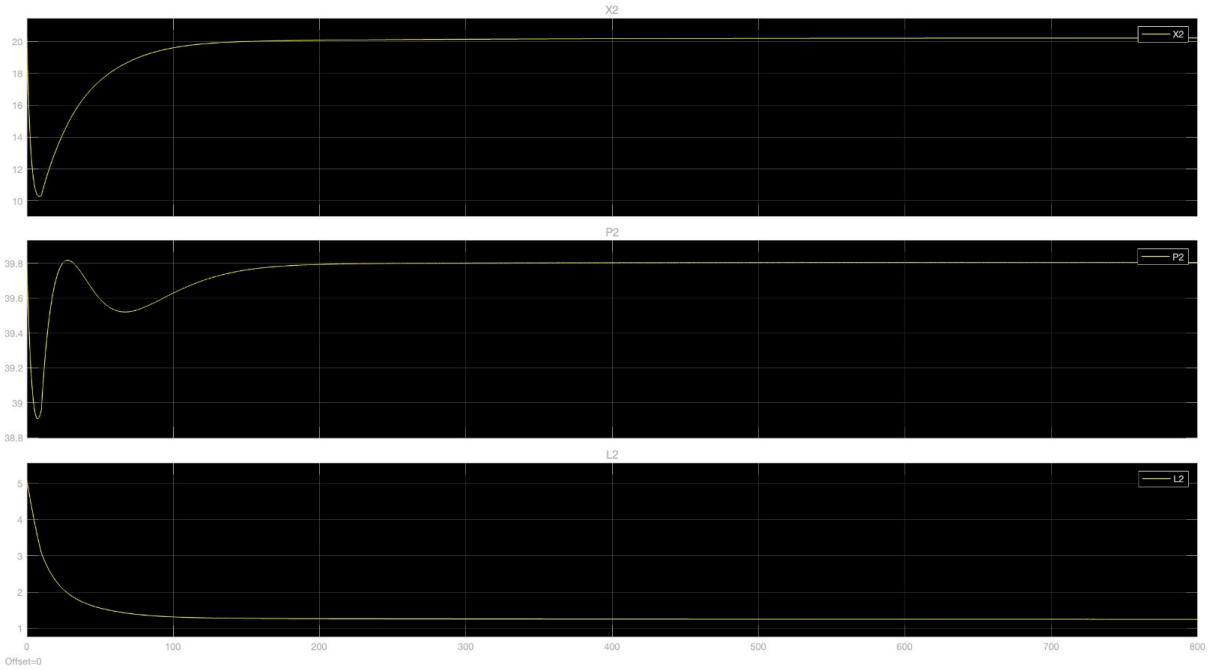

Figure 4-18 Output Graph with a step change in F1 (Relay Feedback as PID tuning method)

From the above graphs we can observe that there are more disturbances in Ziegler-Nichols open loop method compared to relay tuning method. As the relay tuning method directly observes the system's response to disturbances in a closed-loop configuration and adjusts the controller parameters accordingly, while Ziegler-Nichols open-loop relies on finding the ultimate gain and ultimate period from sustained oscillations in an open-loop system. While it provides a means to determine controller parameters, it may not be as responsive to disturbances as the relay tuning method.

With the relay tuning method, the sustained oscillations induced by the relay control action allow for a direct measurement of the system's response time and amplitude. This information can be used to calculate the controller parameters in a manner that provides a quick response to disturbances. By fine-tuning the parameters based on the observed oscillation characteristics, the controller can be optimized for faster disturbance rejection.

Scenario 1: +15 % step change in the feed flow rate F1 at t= 10 min

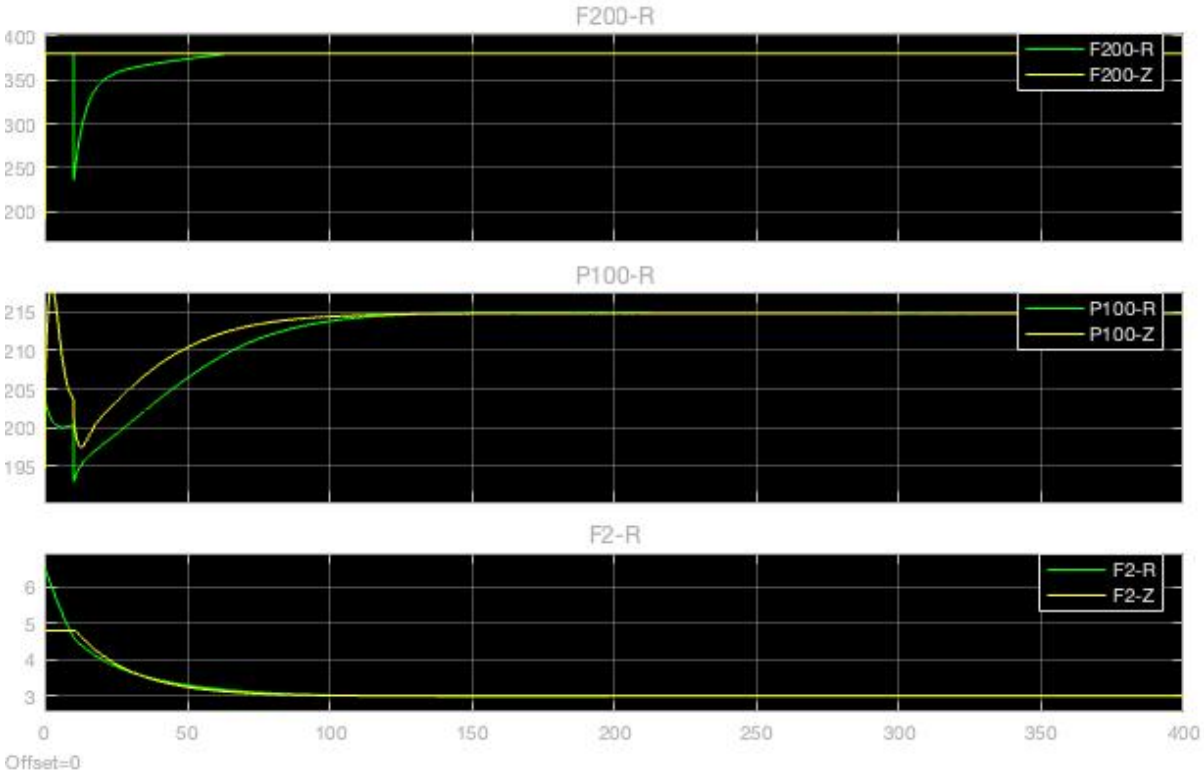

Figure 4-19 Disturbance in the Manipulated Variables due to step change in F1

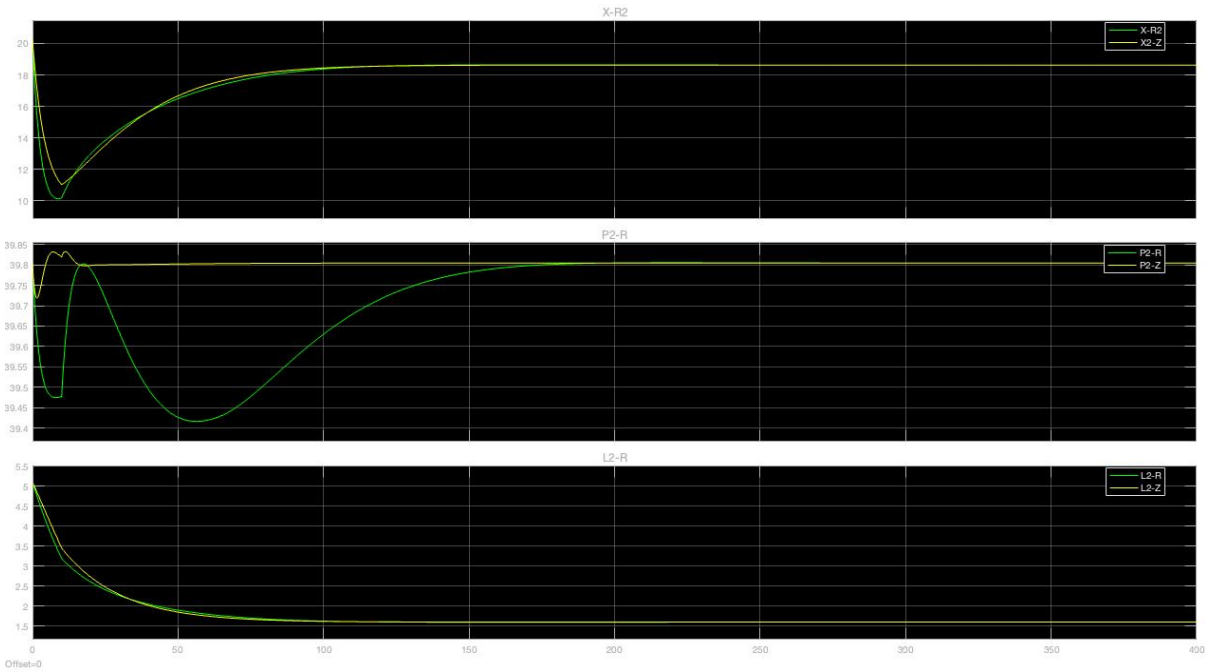

Figure 4-20 Output Graph for the step change in the F1

If we compare the rise time from the above graph, we can conclude that the Ziegler-Nichols open loop method demonstrates a better response than the relay block method for the X2 and P2 variables. However, for the L2 variable, the responses of both methods are almost similar. It is important to note that initially, the deviation from the steady state value is greater in the relay block method.

When considering the settling time, it is observed that the relay block method takes longer compared to the Ziegler-Nichols open loop method in case of P2. However, for the X2 and L2 variables, the settling times are nearly similar between the two methods. In terms of overshoot, the relay block method exhibits more oscillations than the Ziegler-Nichols open loop method for the P2 and X2 variables.

Scenario 2: +10 % step change in the feed flow rate F1 at t= 10 min & - 5 % step change in the feed composition X1 at t= 300 min.

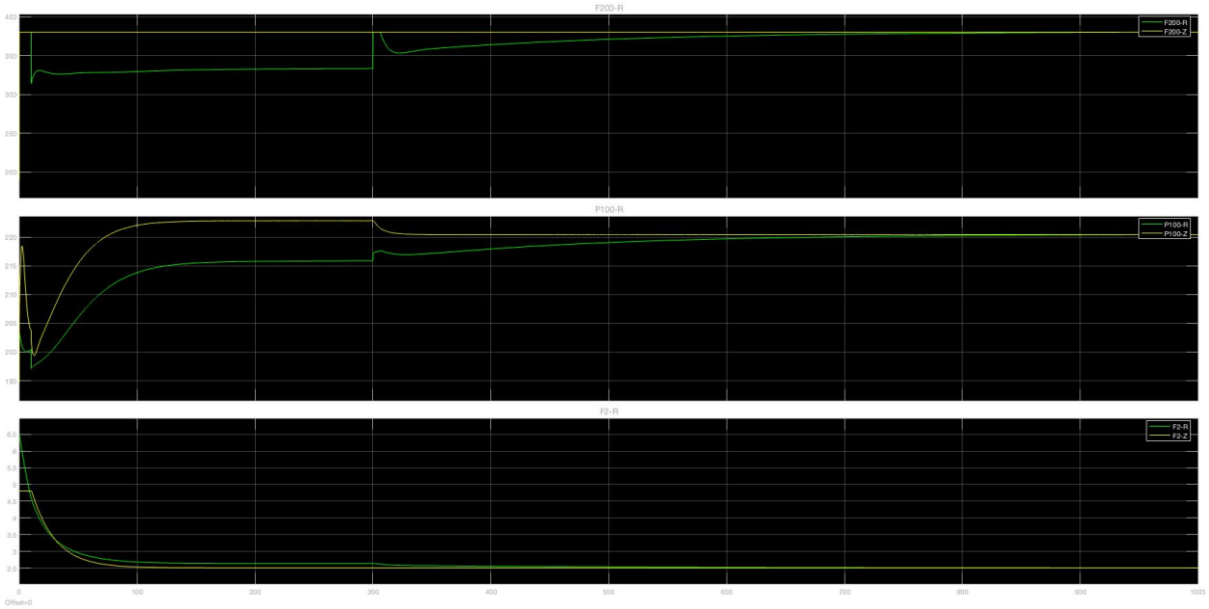

Figure 4-21 Disturbance in the Manipulated Variables due to step change in F1 & X1

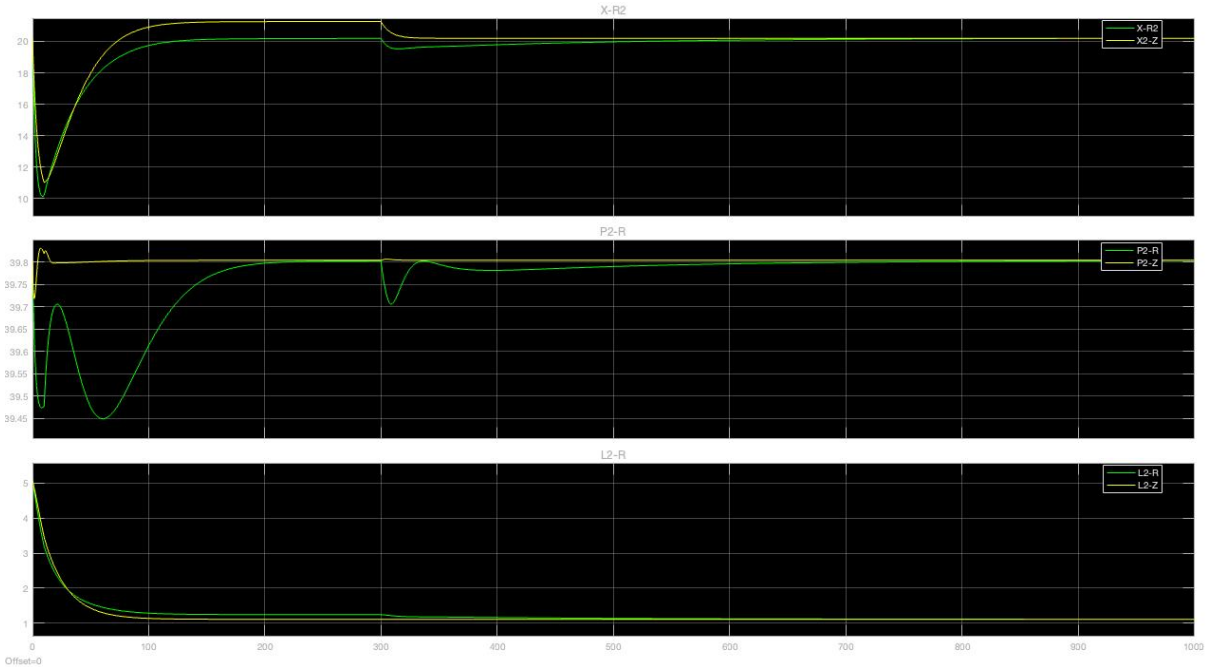

Figure 4-22 Output Graph for the step change in the F1 & X1

If we consider the rise time, it can be observed that the relay block method takes more time to reach a steady state compared to the Ziegler-Nichols open loop method. Additionally, the deviation from the desired value is greater in the relay block method.

The settling time is greater for the relay block method when considering the P2 variable. Moreover, the relay block method exhibits a higher degree of overshoot compared to the Ziegler-Nichols open loop method specifically for the P2 variable.

Scenario 3: + 10 % step change in the set point of X2 at t= 10 min.

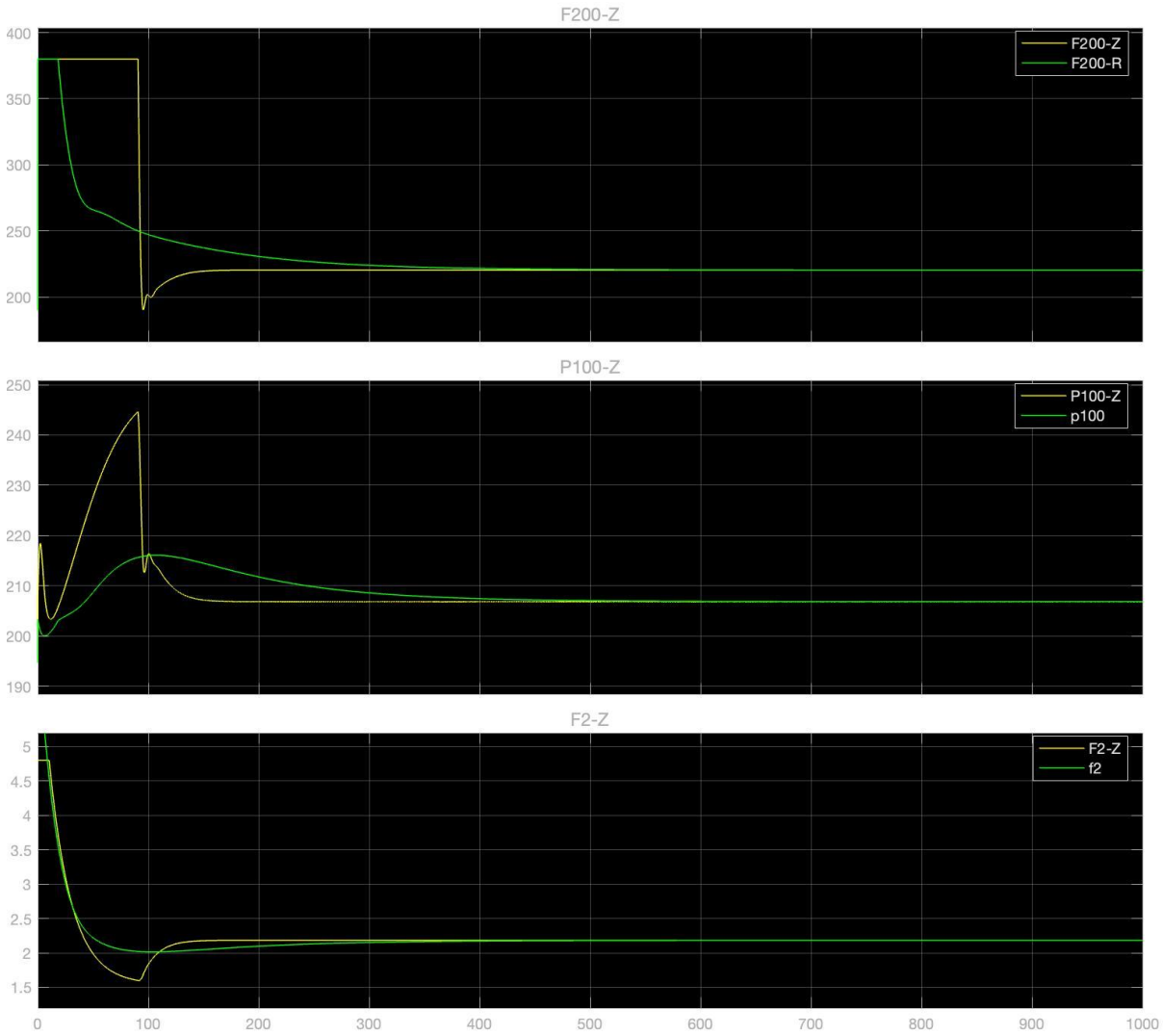

Figure 4-23 Disturbance in the Manipulated Variables due to step change in the set point X2

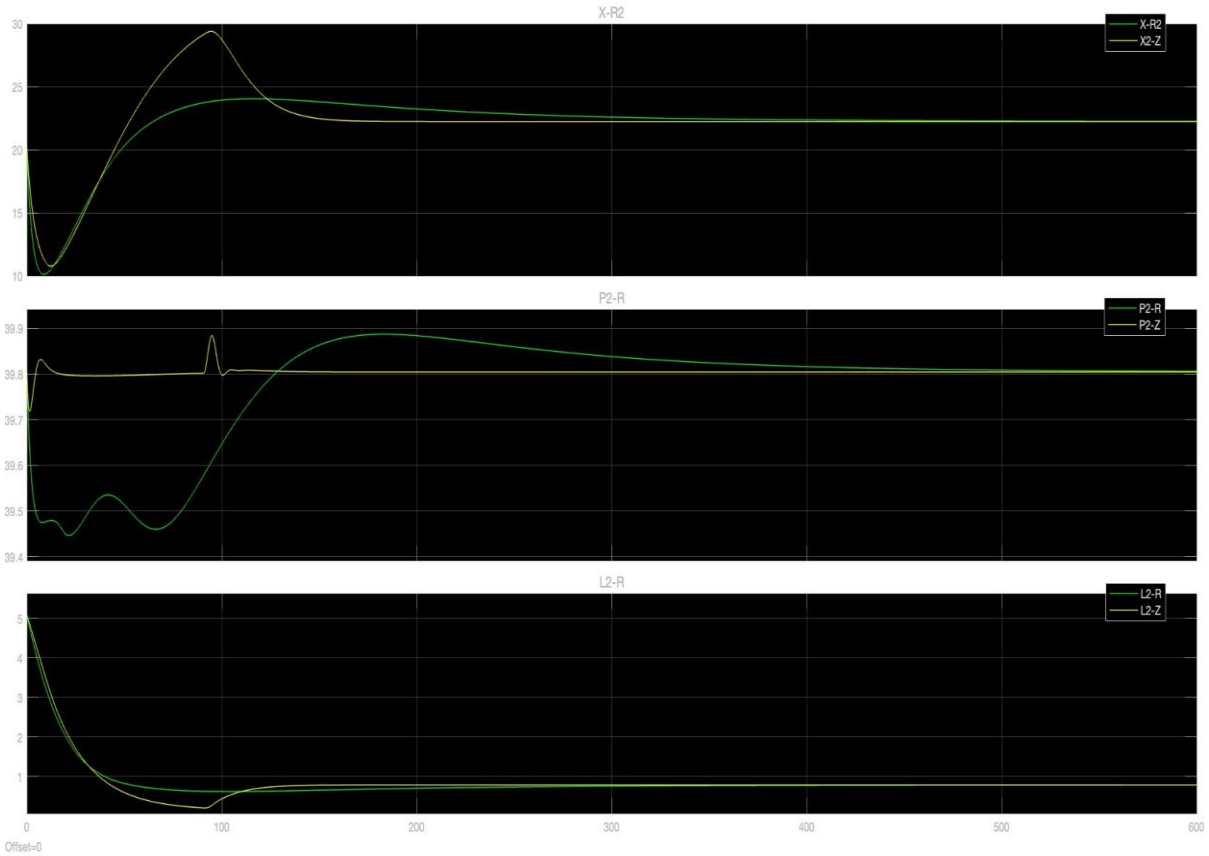

Figure 4-24 Output Graph for the step change in the set point X2

In the relay block method, the rise time is longer compared to the Ziegler-Nichols open loop method. In the case of the settling time, it is observed that the relay block method takes more time to reach a steady state for the P2 variable. On the other hand, the Ziegler-Nichols open loop method takes more settling time for the X2 variable. Similarly, concerning overshoot, the relay block method exhibits a higher overshoot for the P2 variable compared to the Ziegler-Nichols open loop method. However, the relay block method shows less overshoot for the X2 and L2 variables.

Scenario 4: + 5 % step change in the set point of X2 at t= 10 min & + 5 % step change in the set point of P2 at t= 300 min.

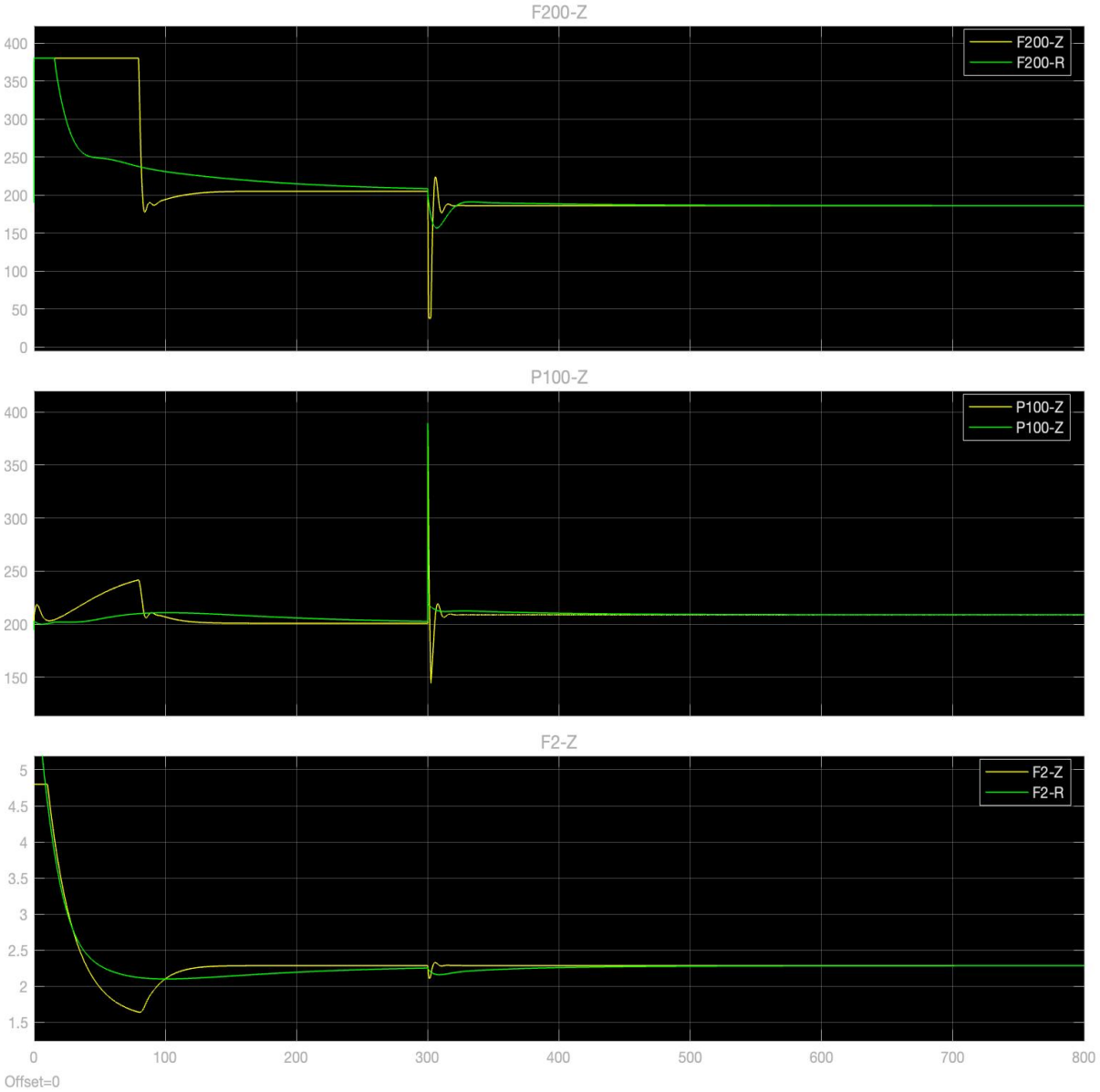

Figure 4-25 Disturbance in the Manipulated Variables due to step change in the set point of X2 & P2

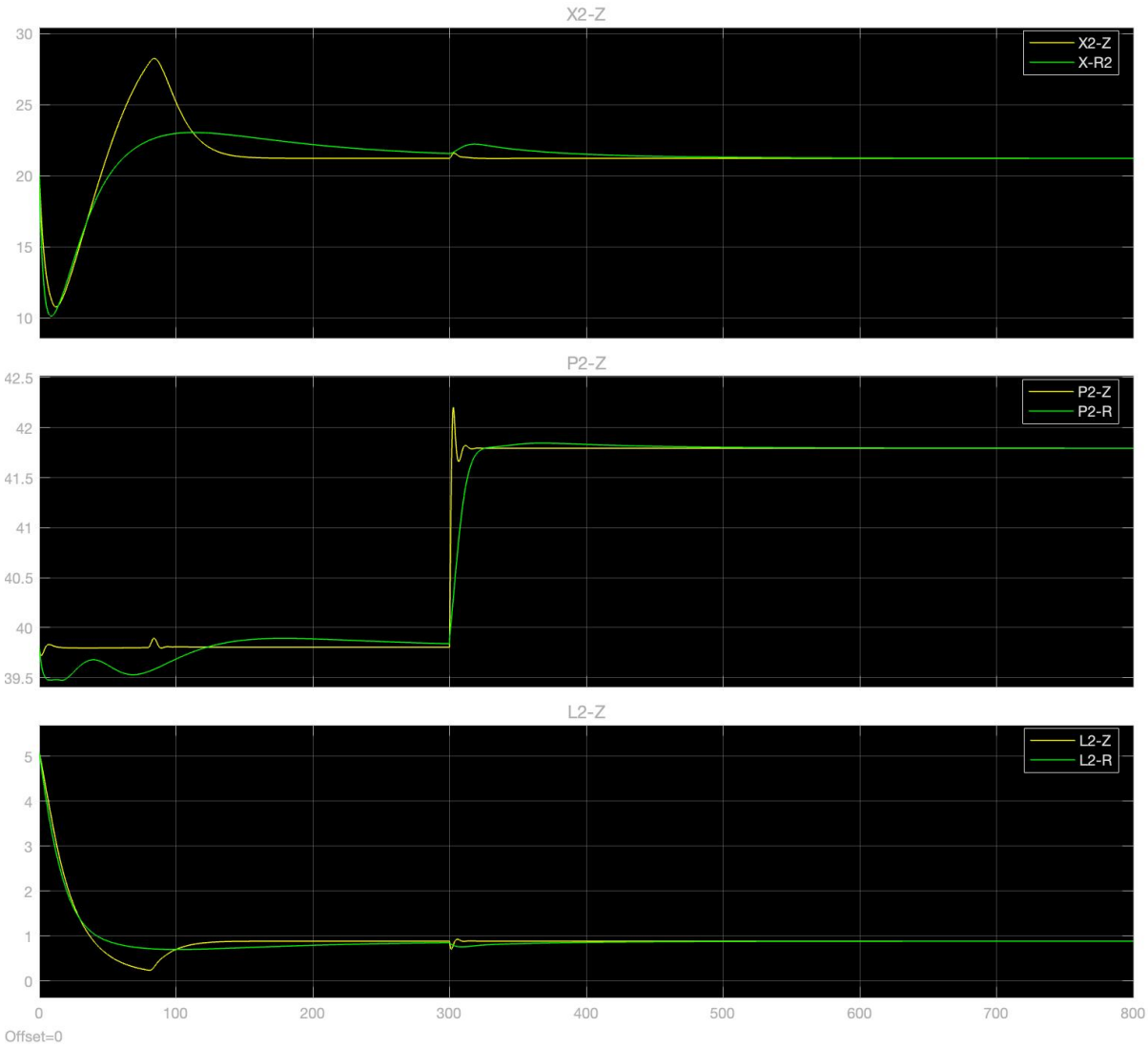

Figure 4-26 Output Graph for the step change in the set point of X2 & P2

In the relay block method, the rise time is longer compared to the Ziegler-Nichols open loop method. However, in the Ziegler-Nichols open loop method, there are more noticeable oscillations before reaching the steady state compared to the relay block method for P2.

The settling time is more for relay block method and the overshoot is more observed in Ziegler-Nichols open loop method.

5. Conclusion:

In conclusion, the control of the Newell and Lee evaporator has been successfully addressed through a systematic approach. The control objective was to achieve optimal operation by maintaining desired operating conditions, including constant level in the separator, desirable product quality, permissible operating pressure, and compliance with safety, environmental, and economic considerations.

ODE solvers are used in MATLAB to simulate the behaviour of the evaporator based on its mathematical model. Dynamic responses of the system states were analysed through step scenarios, revealing that the process is not BIBO stable. This instability can lead to unpredictable behaviour, emphasizing the need for effective control.

A Simulink model was developed, and RGA analysis was performed to determine the stability and influence of different inputs on the output. The eigenvalues indicated that variable L2 exhibits integrating behavior, highlighting its significance in the control strategy.

To determine the optimal input/output pairs, step responses of the linearized system were examined. It was observed that L2 and F2 exhibit an inverse relationship, leading to the selection of F200-X2 and P100-P2 as the optimal pairs. The control strategy involved implementing a proportional controller with negative gain to ensure the desired negative feedback configuration.

After implementing the closed-loop configuration, the stability of the system was confirmed by analyzing the eigenvalues of the 2x2 MIMO system. All eigenvalues had negative real parts, indicating stability. The Relative Gain Array (RGA) analysis helped convert the decentralized MIMO pairs into appropriate SISO pairs, further optimizing the control system.

SISO controllers were designed for the selected input/output pairs using two methods: Ziegler-Nichols open-loop and relay feedback.

The performance of each method can be assessed by considering their effectiveness in terms of rise time, settling time, and overshoot. Based on observations from the step change scenario, it can be concluded that the Ziegler-Nichols open-loop method generally exhibited better overall performance. In the case of set point change scenarios, although the rise time and settling time were similar for both methods, the relay block method demonstrated fewer oscillations from the steady state, indicating lower overshoot.

After evaluating all the scenarios for the F200-X2 I/O pair, it can be concluded that the relay block method is the most suitable choice for controller design due to its lower overshoot. Conversely, for the P2-P100 I/O pair, the Ziegler-Nichols open loop method is more favorable as it demonstrates lower overshoot and settling time compared to the relay block method.

It is important to note that the effectiveness of a controller design method can vary depending on factors such as system characteristics, control objectives, and constraints. Therefore, selecting an appropriate controller design method should involve a comprehensive evaluation of the system's dynamics, performance requirements, and practical considerations.

6. References

Jakob Kjøbsted Huusom, & John Bagterp Jørgensen. (2014). A Realistic Process Example for MIMO MPC based on Autoregressive Models. *IFAC Proceedings Volumes*, 3086-3091.

Two tanks in series exercise by Professor Ilknur Disli Kienl