# Process Control - Project (WiSe 2020/21)

## Control of a Multivariable Process

*Submitted By:*
*Shailesh Kumar XXX (229916),*
*Veeranjaneyulu Akula ( 229473),*
*Arman Ahmed Khan (230601)*
*Electrical Engineering and Information Technology*
*Otto-von-Guericke-Universitat Magdeburg*

## Newell and Lee Evaporator

**Obtain the process output step response curve in the case of a unit step change in the input :**

```
>> G = tf(linsys1)

G =

  From input "P100" to output...
             0.002397 s + 4.371e-05
  X2:  ------------------------------------
       s^3 + 0.2058 s^2 + 0.0186 s + 0.0003743

        0.009588 s^2 + 0.001438 s + 4.794e-05
  P2:  ------------------------------------
       s^3 + 0.2058 s^2 + 0.0186 s + 0.0003743

  From input "F200" to output...
                  1.717e-05
  X2:  ------------------------------------
       s^3 + 0.2058 s^2 + 0.0186 s + 0.0003743

       -0.001829 s^2 - 0.0002743 s - 1.87e-05
  P2:  ------------------------------------
       s^3 + 0.2058 s^2 + 0.0186 s + 0.0003743

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

**Obtain the process output step response curve in the case of a unit step change in the input :**

```
>> figure(1)
step(G)
```

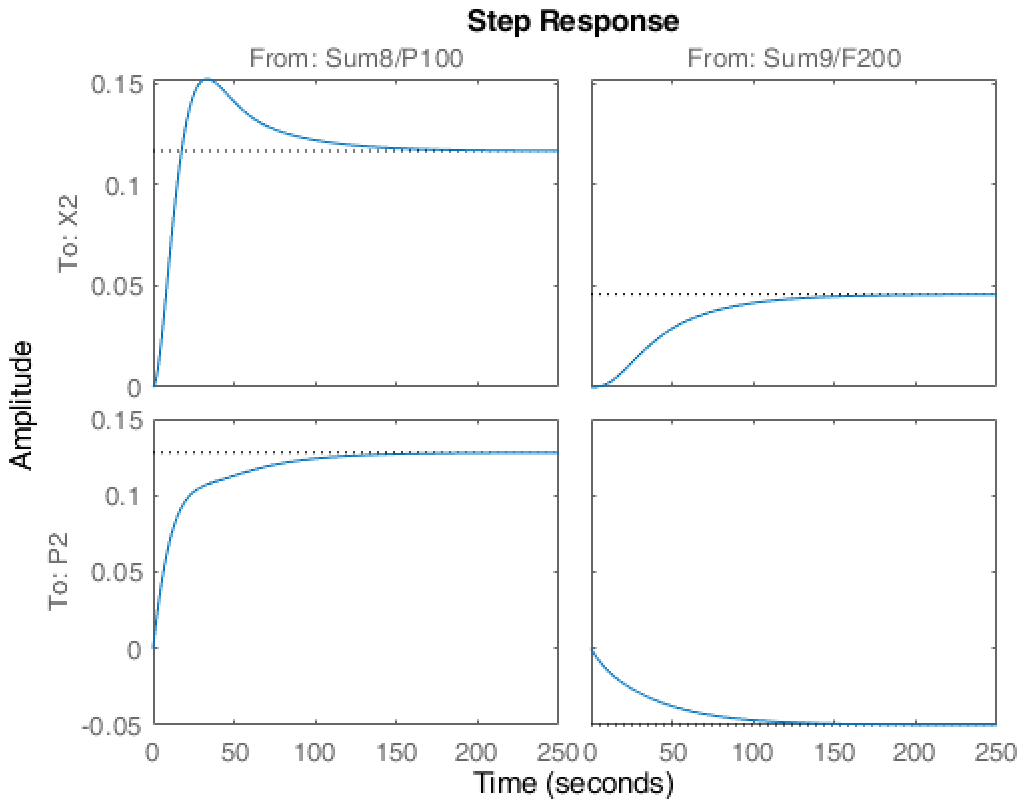


Fig 1 : Step response of F200 & P100 to P2 & X2

**Store the output response data under variable , y and time vector , t .**

```
>> [y,t] = step(G)
```

**Plot the output response data again :**

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

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

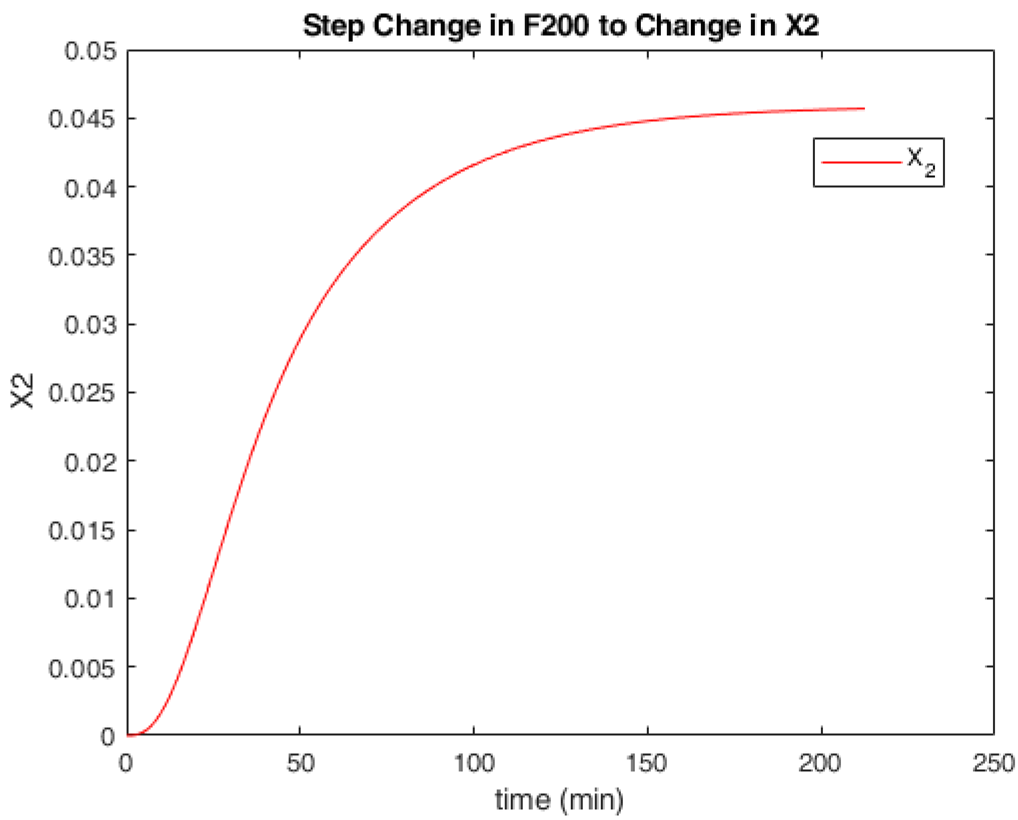


Fig 2 : Step Change in F200 to Change in X2

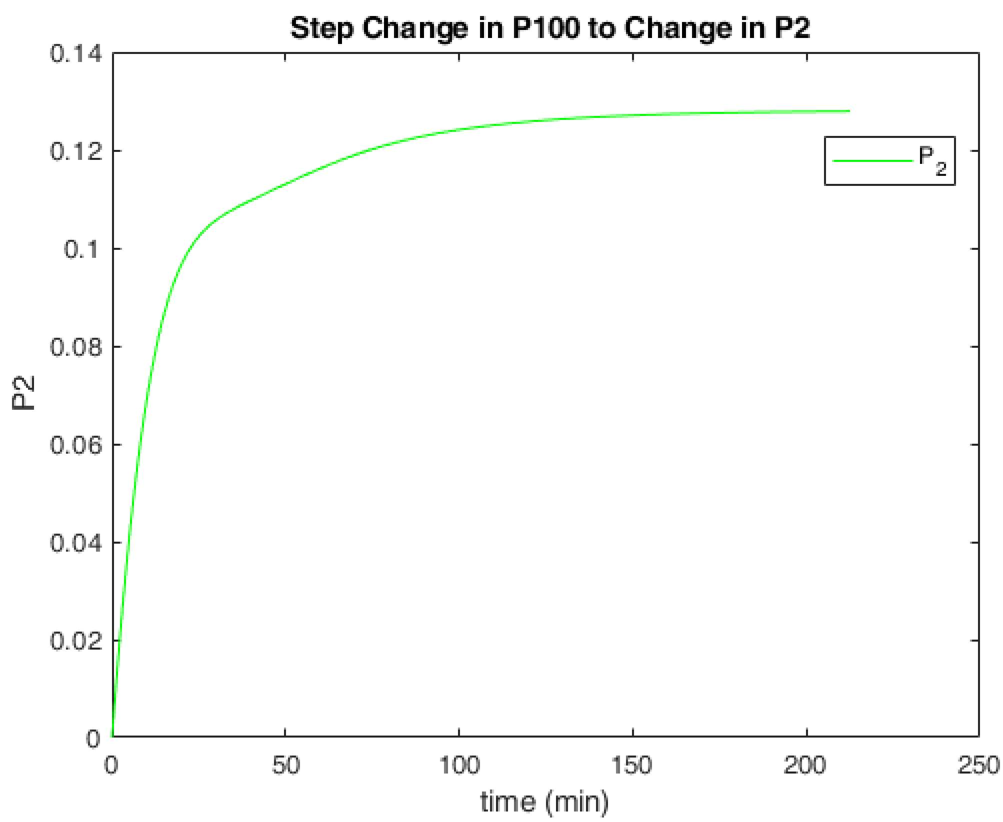


Fig 3 : Step Change in P100 to Change in P2

## Slope at each point :

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

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

## Coordinates of the point of inflection :

```
>>[tslope1,idx1] = max(slope1)
tIP1 = t(idx1)
yIP1 = y(idx1,1,2)

[tslope2,idx2] = max(slope2)
tIP2 = t(idx2)
yIP2 = y(idx2,2,1)

tslope1 = 8.1683e-04
idx1 = 26
tIP1 = 25.7944
yIP1 =  0.0126

tslope2 = 0.0093
idx2 = 1
tIP2 = 0
yIP2 = 0
```

## The tangent line of the step response curve at the inflection point :

```
yTangentLine1 = tslope1*(t-tIP1) + yIP1
yTangentLine2 = tslope2*(t-tIP2) + yIP2
```

## The inflection point and the tangent line of the output response curve :

```
>> figure(2)
plot(t,y(:,1,2),'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','Inflation Point','Tangent')
hold off

figure(3)
plot(t,y(:,2,1),'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','Inflation Point','Tangent')
hold off
```

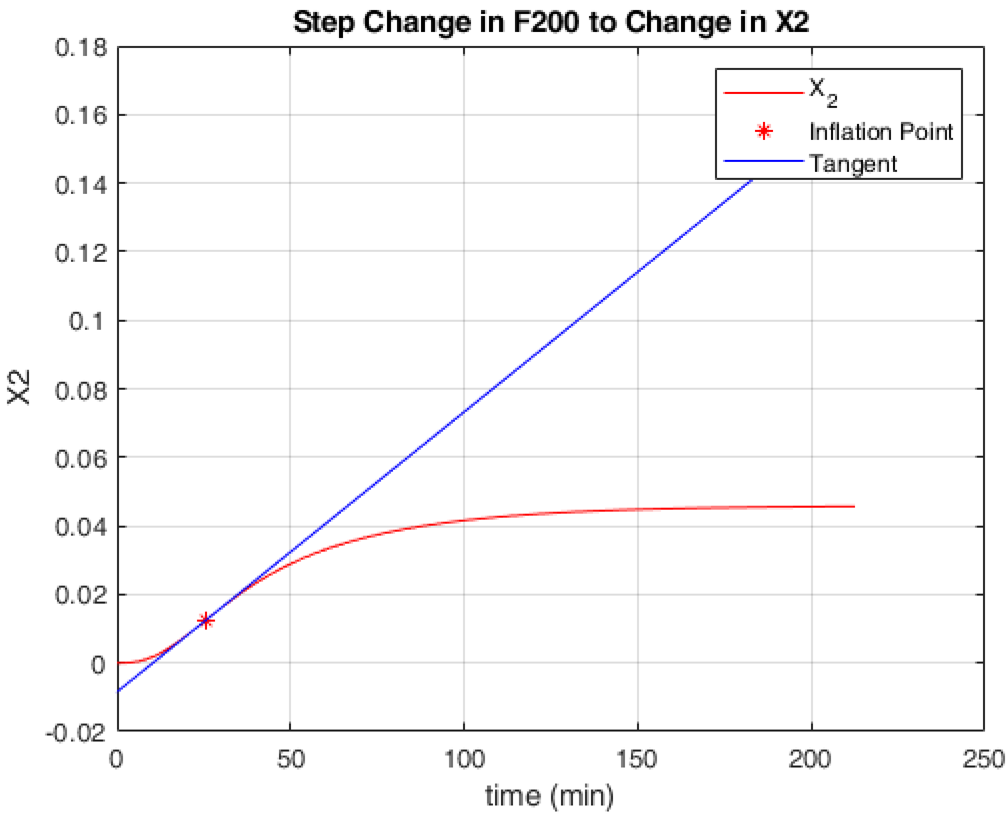


Fig 4 : Step Change in F200 to Change in X2 with Tangent at Inflation Point

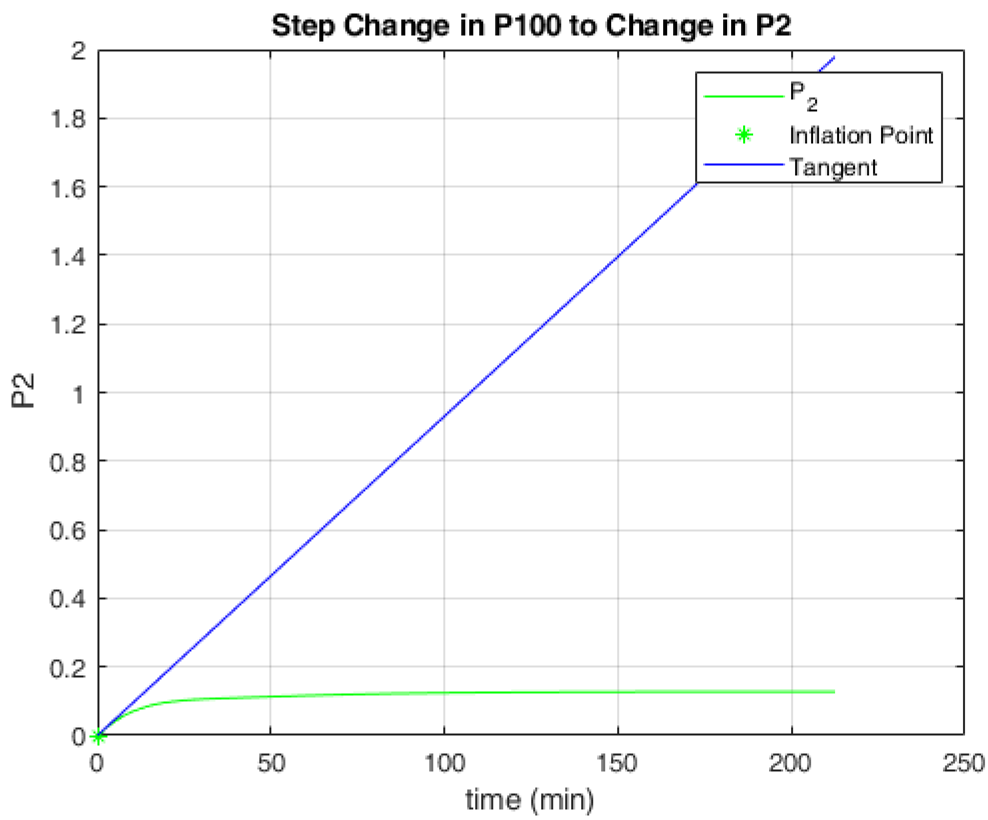


Fig 5 : Step Change in P100 to Change in P2 with Tangent at Inflation Point

## Estimate parameters of the transfer function ( FOPTD) model :

### Time delay (Dead time) , Td :

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

Td1 = 10.3182
Td2 = 0
```

The output step response of P100 to X2 curve doesn't have any inflection point , hence Td=0 . When we insert Td=0 in the formula for Kp , it is no surprise that in this case Kp= infinitive .
In order to avoid the inconvenient Kp value (infinitive) we are introducing an artificial Td which may be due to the measurement delay. That is Td2 =1 minute.

```
Td2 = 60.0
```

### Time constant , Tau :

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

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

Tau1 = 55.9295
Tau2 = 13.7390
```

**Process gain, K :**

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

K1 = 0.0457
K2 = 0.1279
```

**Show the delay time on the graph:**

```
>> figure(2)
plot(t,y(:,1,2),'r')
xlabel('time (min)')
ylabel('X2')
title('Step Change in F200 to Change in X2')
hold on
plot(Td1,0,'g.','MarkerSize',15)
legend('X_2','Time Delay')
hold off

figure(3)
plot(t,y(:,2,1),'g')
xlabel('time (min)')
ylabel('P2')
title('Step Change in P100 to Change in P2')
hold on
plot(Td2,0,'r.','MarkerSize',15)
legend('P_2','Time Delay')
hold off
```

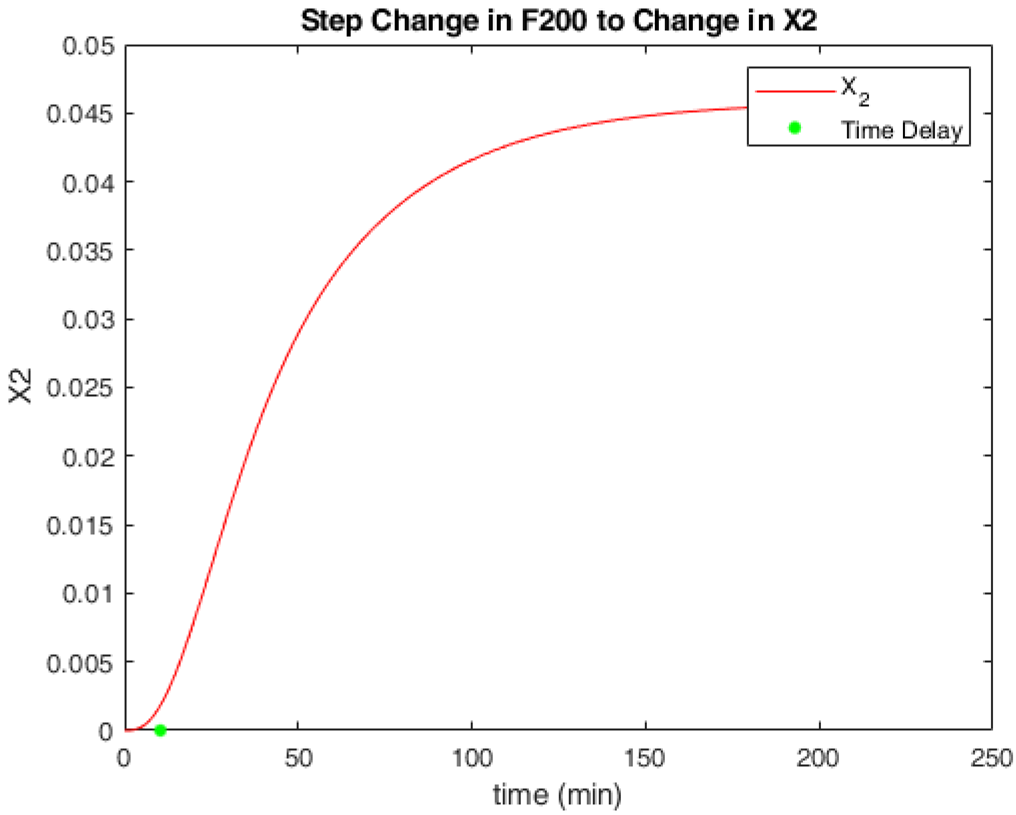


Fig 6 : Step Change in F200 to Change in X2 with Time Delay

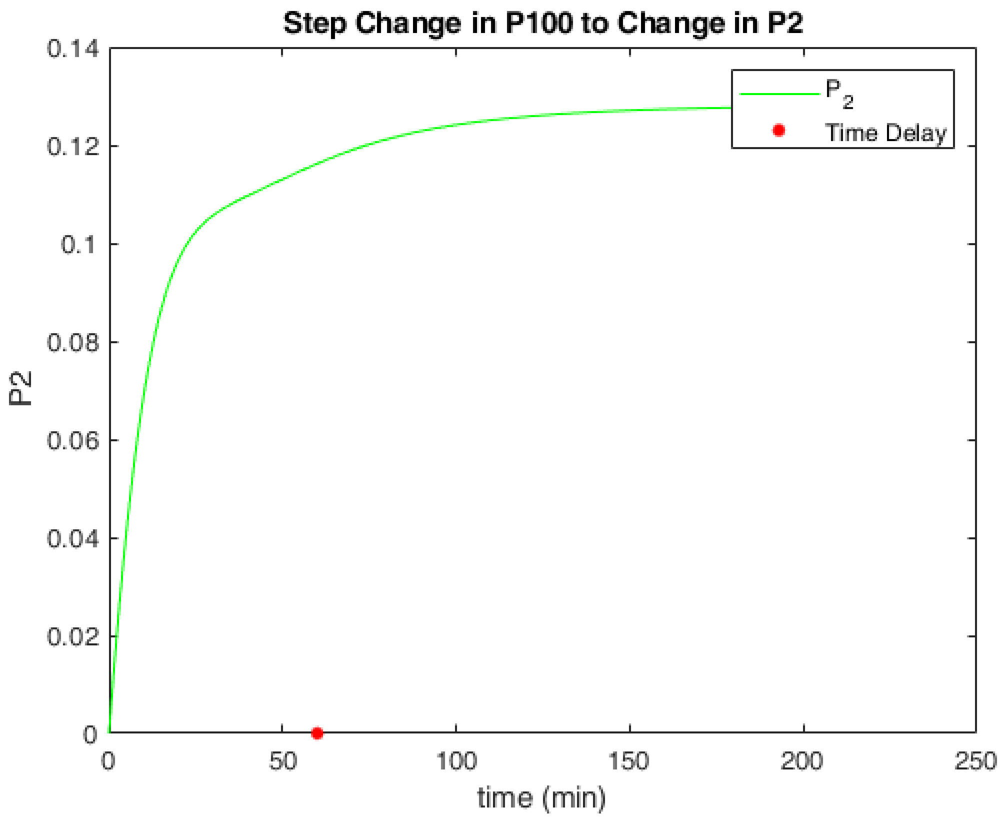


Fig 7 : Step Change in P100 to Change in P2 with Assigned Time delay

## Approximate transfer function and plot the step response :

```
G12 = tf(K1,[Tau1],'InputDelay',Td1)
[ynew1,tnew] = step(G12);
figure(4)
plot(t,y(:,1,2),'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,'g.','MarkerSize',15)
plot(tnew,ynew1,'r--')
legend('Actual Process','InflectionPoint','Tangent at IP','Dead Time','Approximated FOPTD')
hold off
```

```
G21 = tf(K2,[Tau2],'InputDelay',Td2)
[ynew2,tnew] = step(G21);
figure(5)
plot(t,y(:,2,1),'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,'r.','MarkerSize',15)
plot(tnew,ynew2,'g--')
legend('Actual Process','InflectionPoint','Tangent at IP','Dead Time','Approximated FOPTD')
hold off
```

G12 = exp(-10.3*s) * (0.0008168)

Continuous-time transfer function.

G21 = exp(-60*s) * (0.009308)

Continuous-time transfer function.

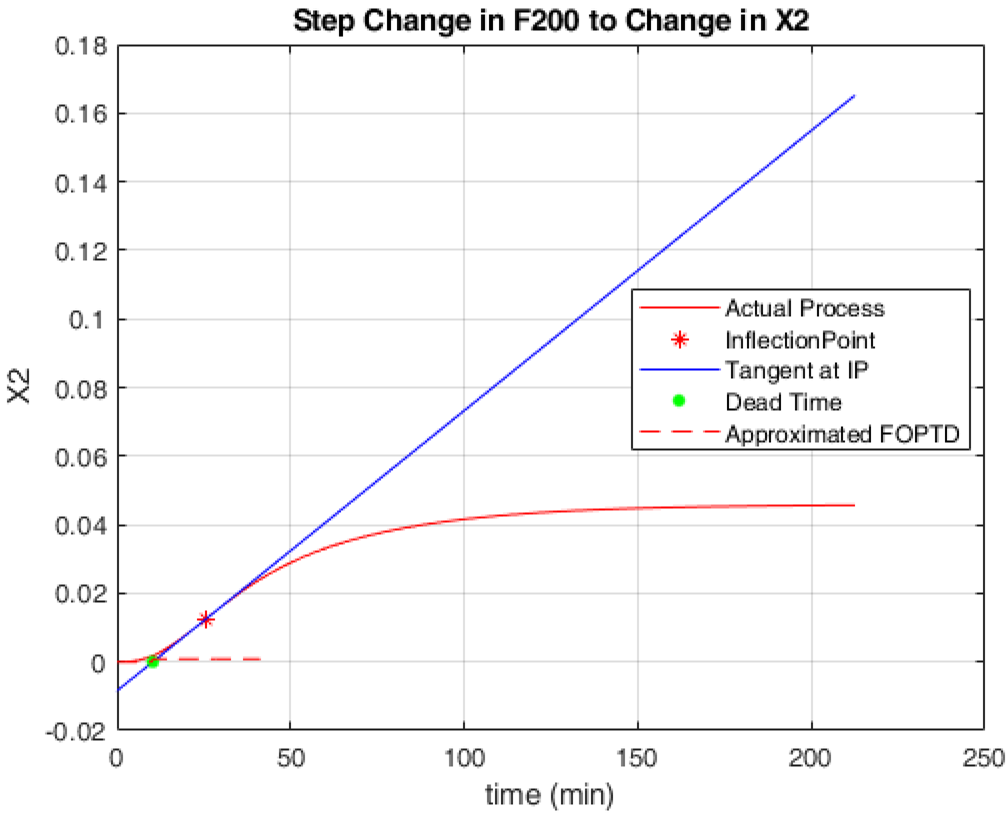


Fig 8 : Step Change in F200 to Change in X2 original and Approximated

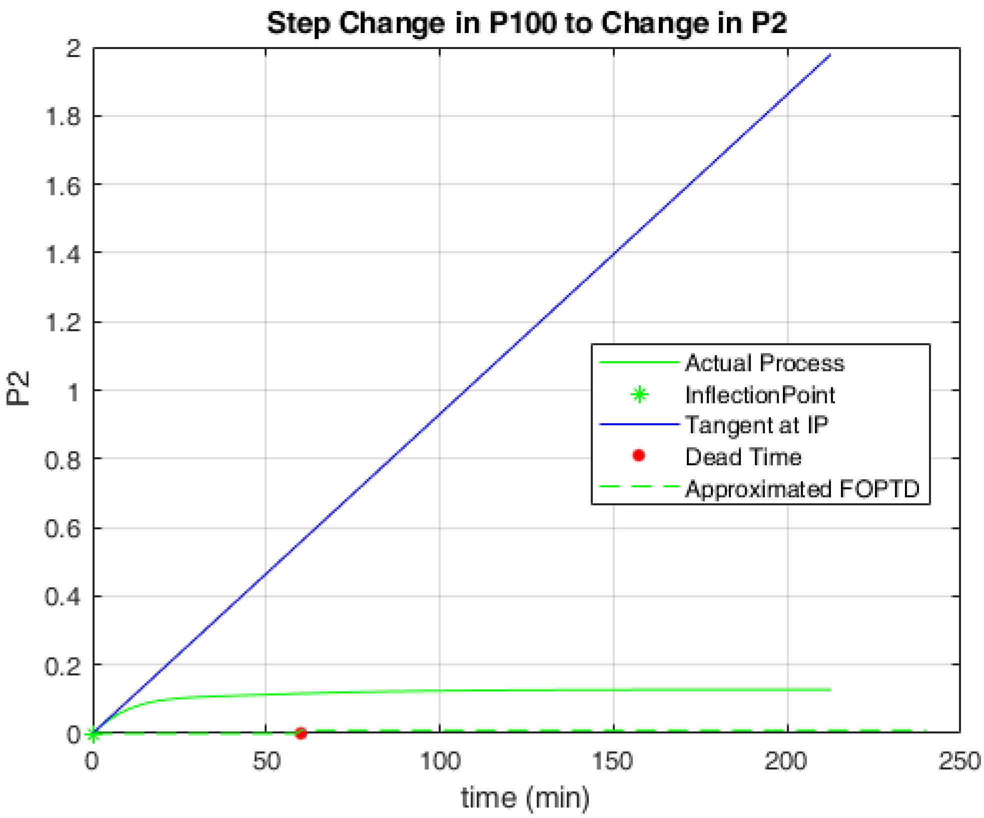


Fig 9 : Step Change in P100 to Change in P2 original and Approximated

**Calculate PI or PID controller parameters :**

Proportional gain , Kp1 & Kp2 :
Kp1 = 1.2 * Tau1/( K1*Td1)
Kp2 = 1.2 * Tau2/( K2*Td2)

Kp1 = 142.3780
Kp2 = 2.1486

Integral time TI1 & Ti2 :

TI1 = 2*Td1
TI2 = 2*Td2

TI1 = 20.6365
TI2 = 120

Derivative time :
TD1 = 0.5*Td1
TD2 = 0.5*Td2

TD1 = 5.1591
TD2 = 30

As we have calculated Kp, TI & Td for both o/p and i/p pairs which is used in time constant form of PID controller but Matlab uses Ideal form for that we have to calculate Kp, KI, kd.

$$\begin{aligned} u(t) &= K_P\left(e(t) + \frac{K_I}{K_P}\int e(t)dt + \frac{K_D}{K_P}\frac{de(t)}{dt}\right) \\ &= K_P\left(e(t) + \frac{1}{T_I}\int e(t)dt + T_D\frac{de(t)}{dt}\right) \\ \text{with:} \quad & T_I = K_P/K_I \text{ (integral time constant)}, \\ & T_D = K_D/K_P \text{ (derivative time constant)} \end{aligned}$$

So,

KI1=Kp1/TI1
KI2=Kp2/TI2
Kd1=Kp1*TD1
Kd2=Kp2*TD2

KI1 = 6.8993
KI2 = 0.0179

Kd1 = 734.5442
Kd2 = 64.4577

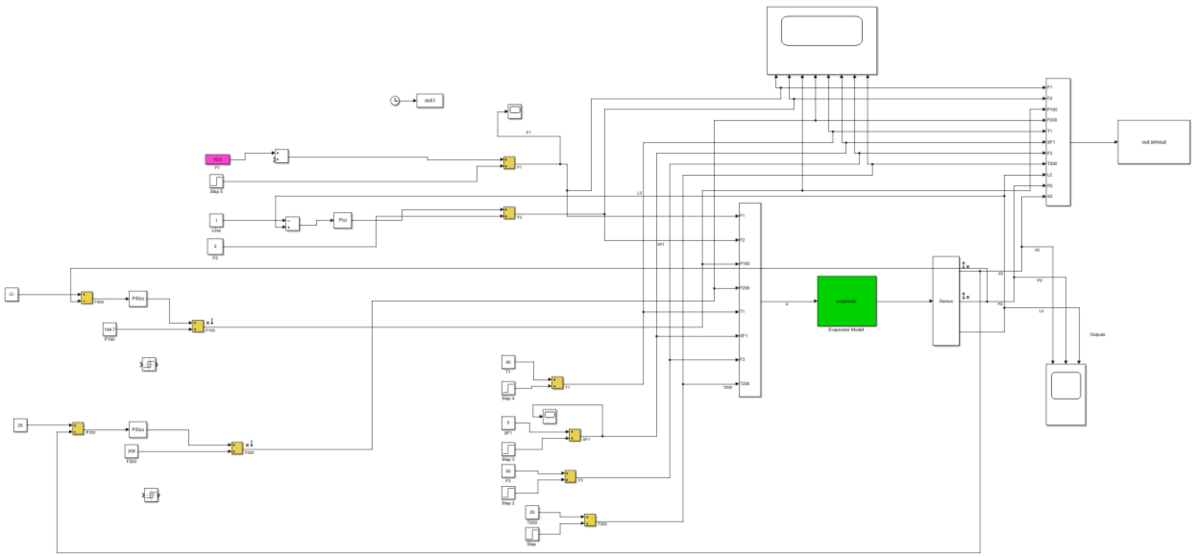

Fig 10 : Updated Simulink model of Evaporator model with PID Controller.

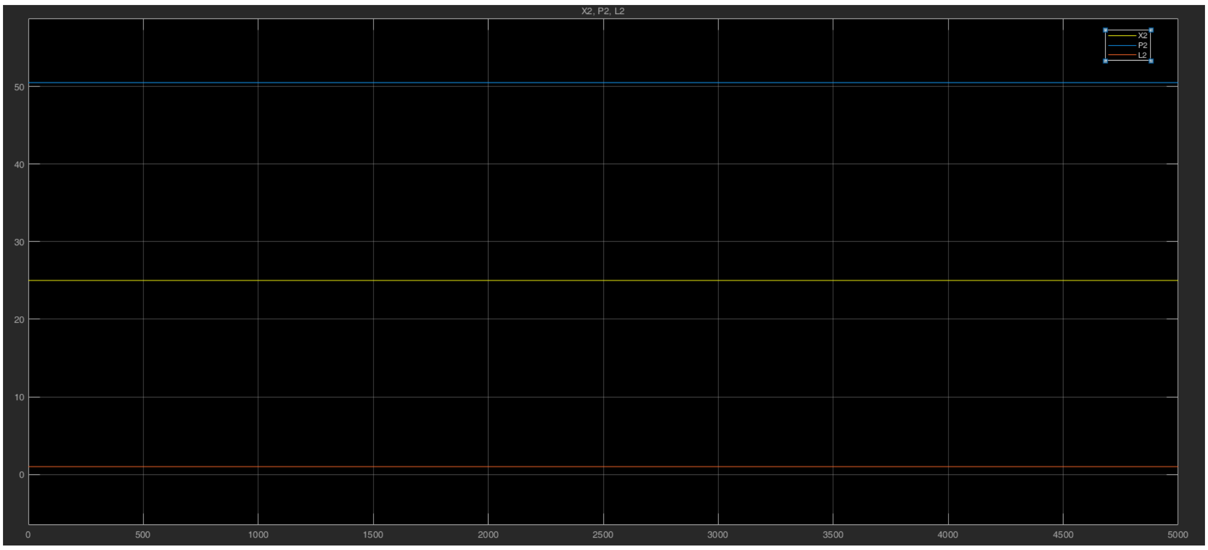


Fig 11 : Updated steady state output of Evaporator model with PID Controller.

## Using Relay Feedback as PID tuning method in MATLAB/Simulink :

*In general, the **relay feedback** tuning method is used on a real plant .* Typically the 'error signal' should be larger than the measurement noise, such a measurement noise is much larger then **'eps'** value . we assumed the amplitude of the expected process output oscillations and accordingly choose the error signal in the same range.
We changed default values of **Relay block** parameters. **'Switch on/off'** parameters are related to the **error**. Default value of **'eps'** is a very small number.
We assumed that it is possible to get an error signal with magnitude 1 for the pressure, P2 & X2

Hence for the Relay block :
Switch on point : 1
Switch off point : -1
*Output on/off* parameters are related to the output signal of the Relay block and consequently to the manipulated variable. we started with a small value such as :
Output when on : 1
Output when off : -1
And run the Simulink simulation , where we observe the process output of interest. There was no oscillation then we double the values of *Output on/off* for each run until we get constant oscillations in the process output at :
Output when on : 6
Output when off : -6

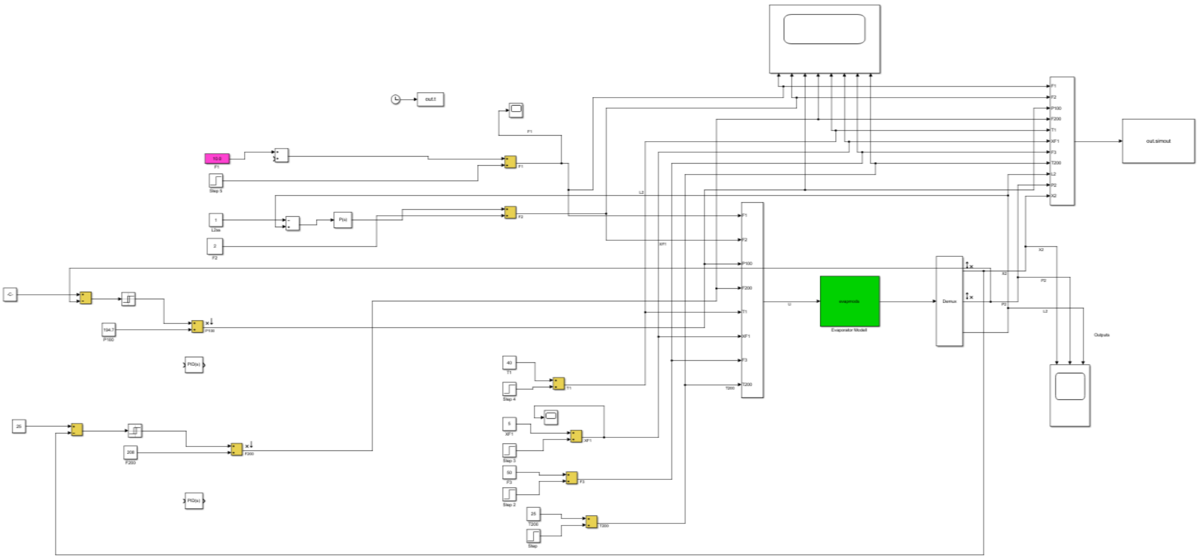

Fig 12 : Updated Simulink model of Evaporator with Relay Feedback.

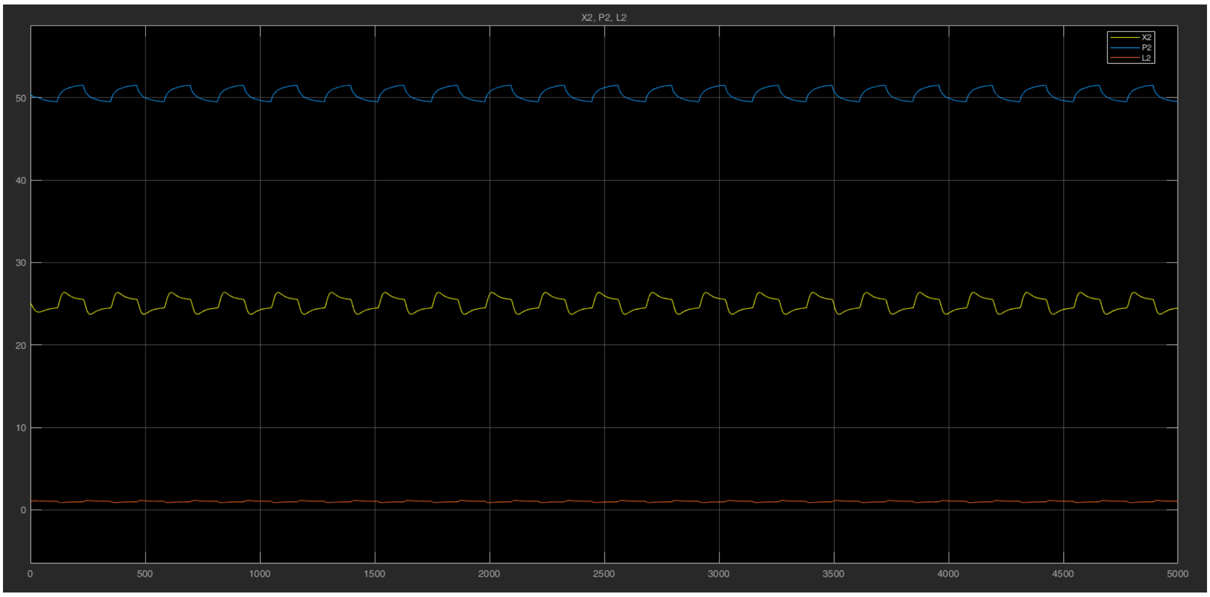


Fig 13 : Updated steady state output of Evaporator with Relay Feedback.

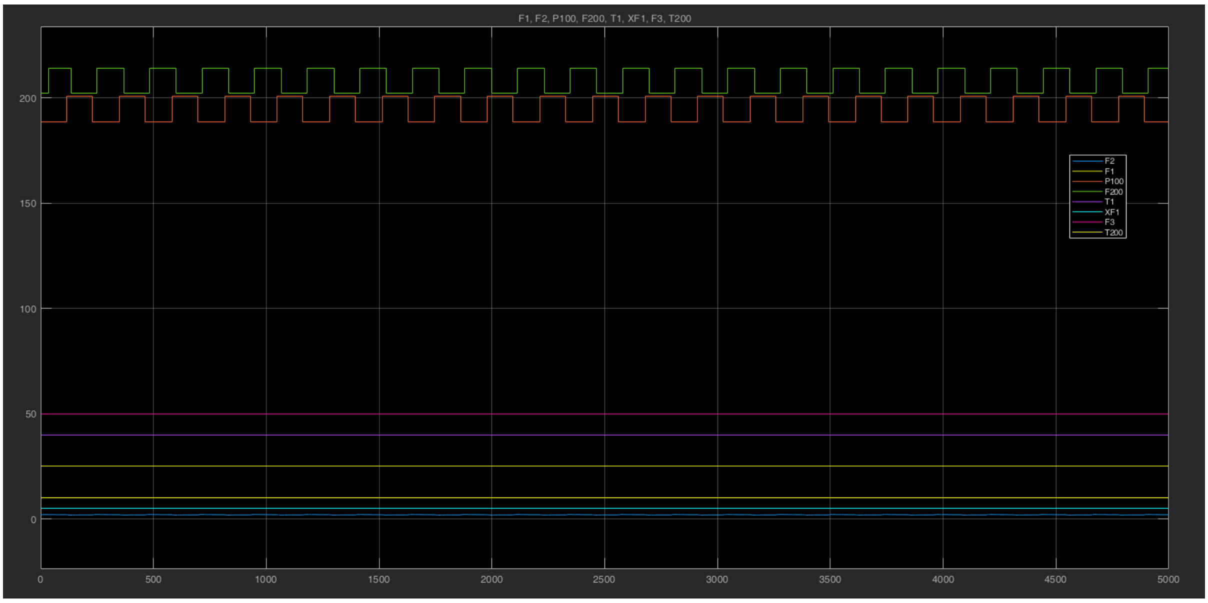


Fig 14 : Updated Input of Evaporator with Relay Feedback.

## Simulation scenarios : Step change in some disturbances .

**Scenario (i) : apply only +15 % step change in F1 a t=10 min :**

Step time =600
Initial value =0
Final value = 10*0.15

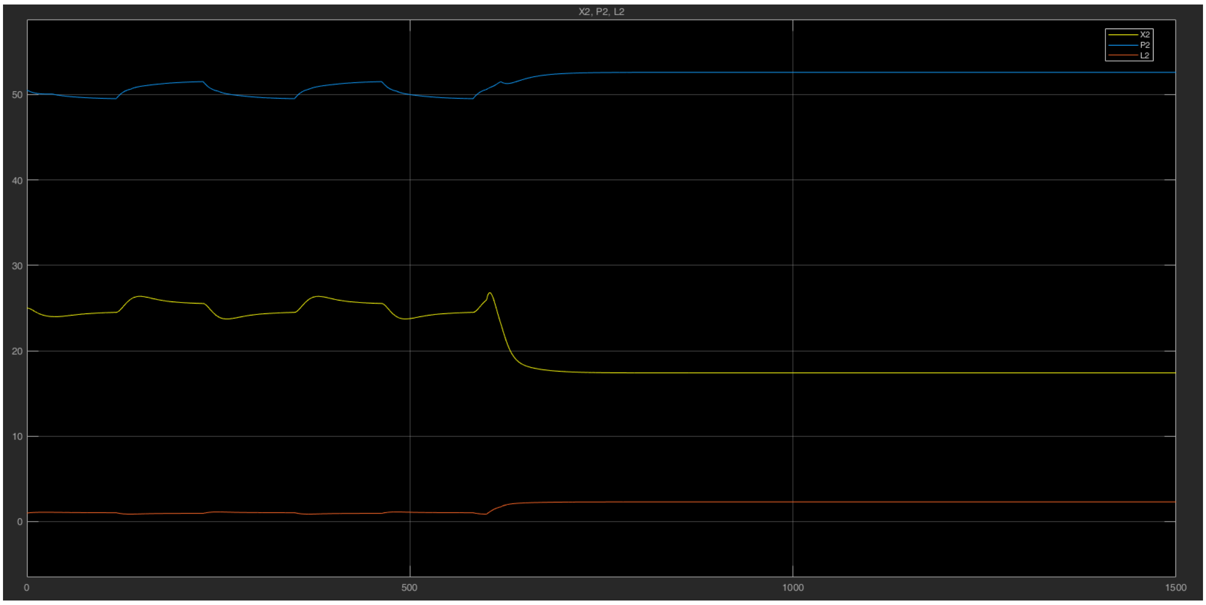


Fig 15 : A step change of + 15 % in the feed flow rate , F1

**Scenario (ii) : apply only +10 % step change in X1 at t=10 min**
**( In this case the step in F1 should be set as 0)**

Step time =600
Initial value =0
Final value = 5*0.10

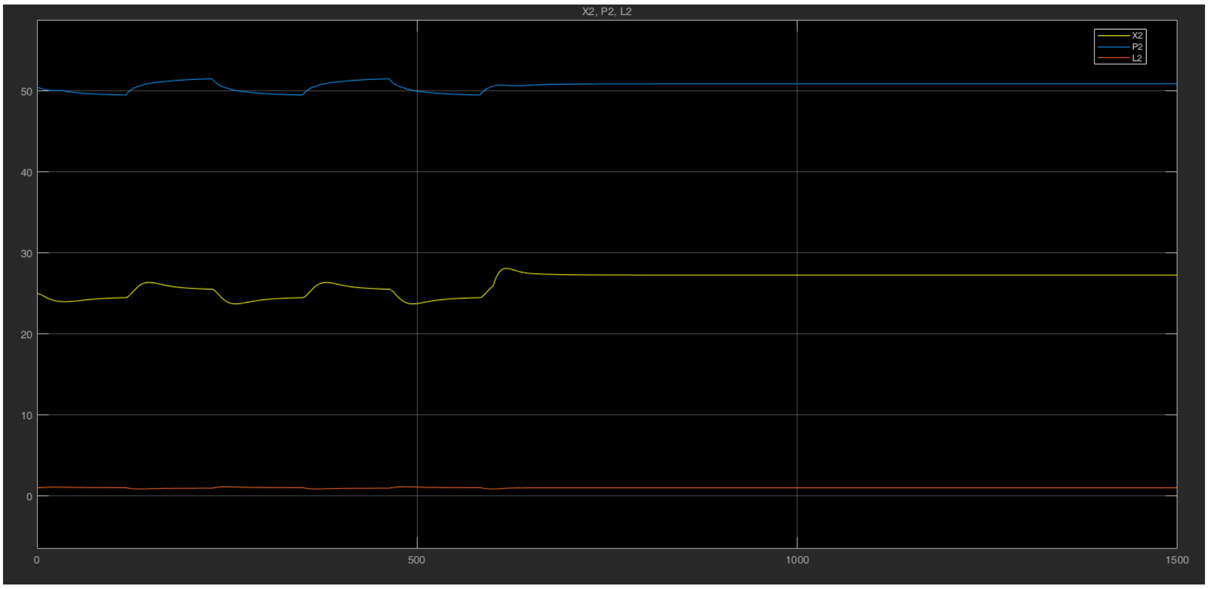


Fig 16 : A step change of + 10 % in X1

**Scenario (iii) : apply only +10 % step change in T1 at t=10 min**

Step time =600
Initial value =0
Final value = 40*.1

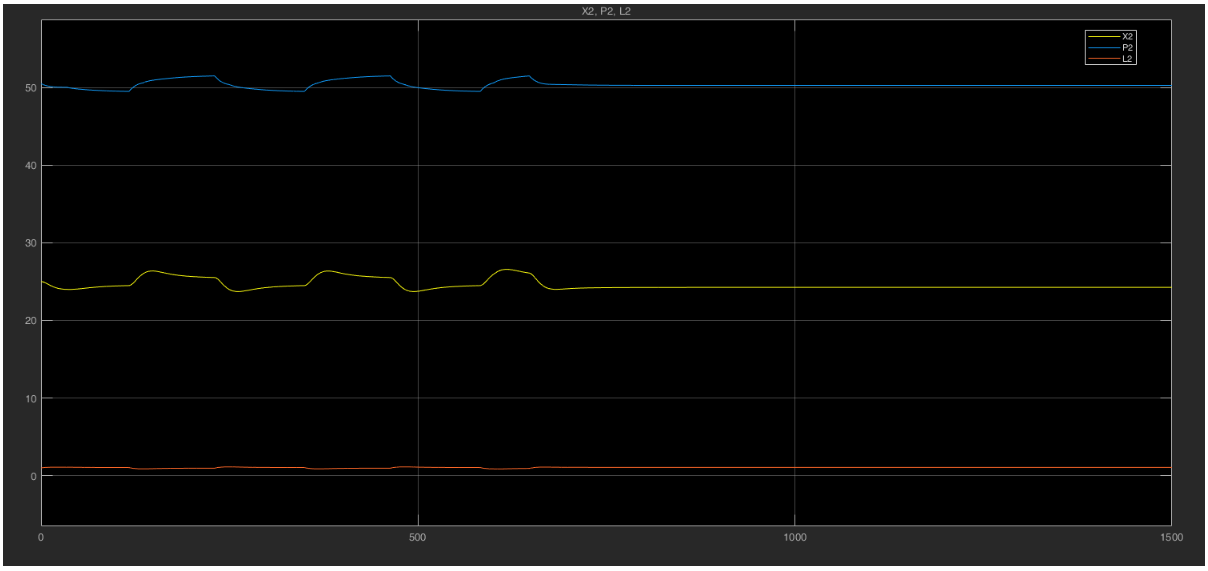


Fig 17 : A step change of + 10 % in T1

**Scenario (iv) : apply only +15 % step change in F3 at t=10 min**

Step time =600
Initial value =0

Final value = 50*0.15

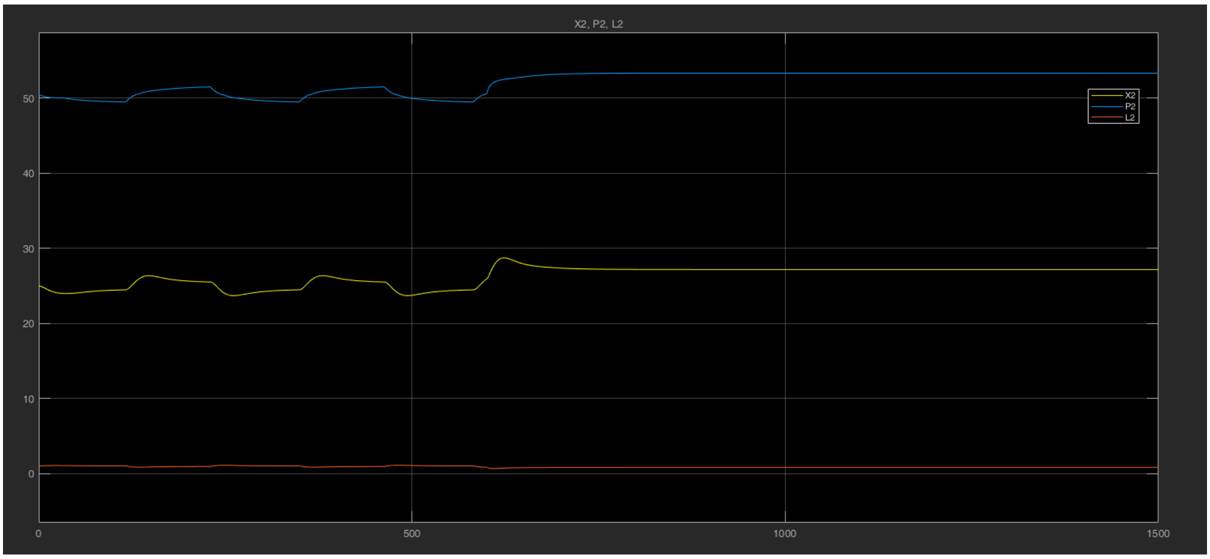


Fig 18 : A step change of + 15 % in F3

**Scenario (v) : apply only +10 % step change in T200 at t=10 min**

Step time =600
Initial value =0
Final value = 25*0.10

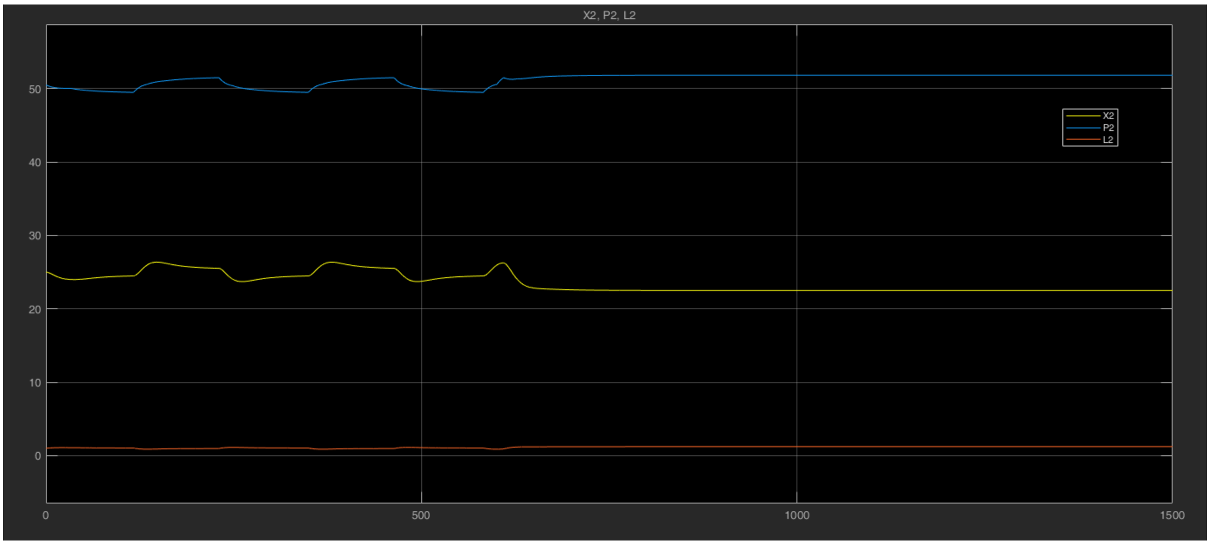


Fig 19 : A step change of + 10 % in T200

**Scenario (vi) : apply +10 % step change in T200 at t=5 min and +15 % step change in F1 a t=10 min**

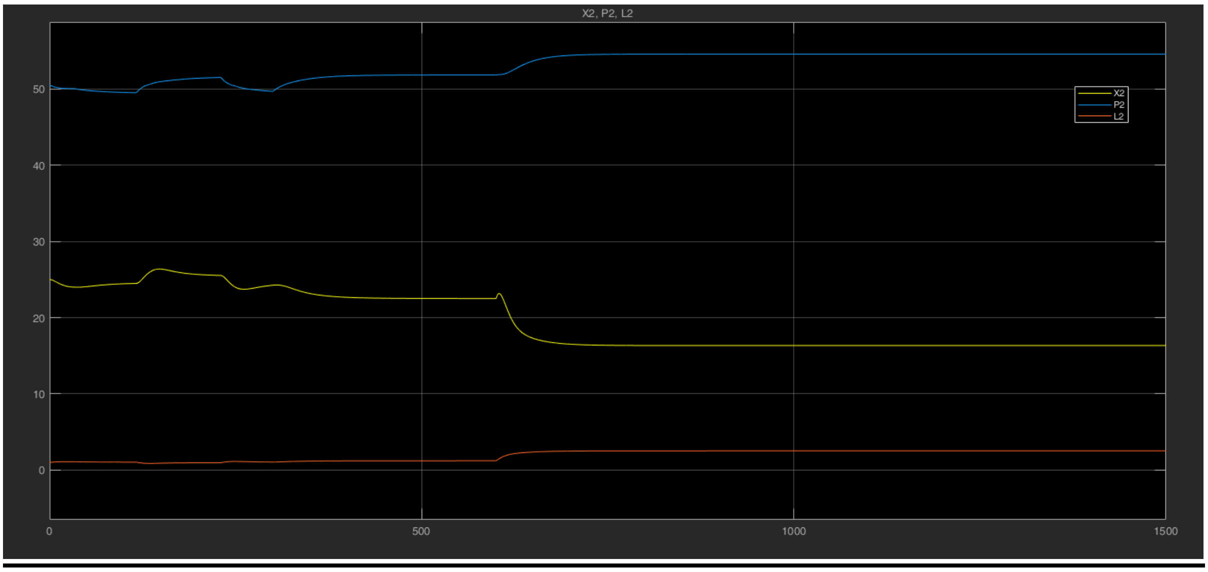


Fig 20 : +10 % step change in T200 at t=5 min and +15 % step change in F1 a t=10 min