Zewail City of science and technology
Applied Digital Control and Drives NANENG 512
FALL 2020

# Phase 2:

Water tank level controller.

Rami Wail - Ahmed Alaa - Mohamed Elshafey
IDs: 201600112 - 201600352 - 201700907

DATE
Jan 7th, 2021

# I. INTRODUCTION:

Our Water tank level controller consists of a dynamic model of coupled tank system [1]. Recently, the most common control problem for the process of an industrial site is to control various variables such as the temperature, water level, and internal pressure of a tank and a chemical reactor [2]. Ultimately, controlling the liquid level of a tank is the primary element for oil and chemical liquid processes. The purpose of level control is to make the water level follow a given set value or to quickly restore a constant water level in case of disturbance [2]. Generally, a PID controller is widely used for level control; and the gains of the controller should be tuned so that the required performance of a controlled system can be satisfied. This tuned PID controller can satisfy the required performance of the controlled system when the operating environment of the controlled system is unchanged. Unfortunately, this is mostly not the case as parameters of the controlled system change due to the change in the operating environment of the controlled system. Thus, satisfactory control cannot be guaranteed unless the gains of the controller are artificially changed again. Accordingly, this paper aims to provide an extensive overview of the control system and its parameters while considering the software (MATLAB) and hardware (components) perspectives, alternatively.

# II. Level Control System:

The schematic diagram of the coupled tank is shown in Figure 1. Two tanks are connected in an interactive valve. The inflow of tank 1 is qi and outflow of tank 2 is qo2. The control variable is level in tank 2. According to Figure 1 the input u is the input voltage which is taken to the pump, and the output h2 is the water level in tank 2 [1].

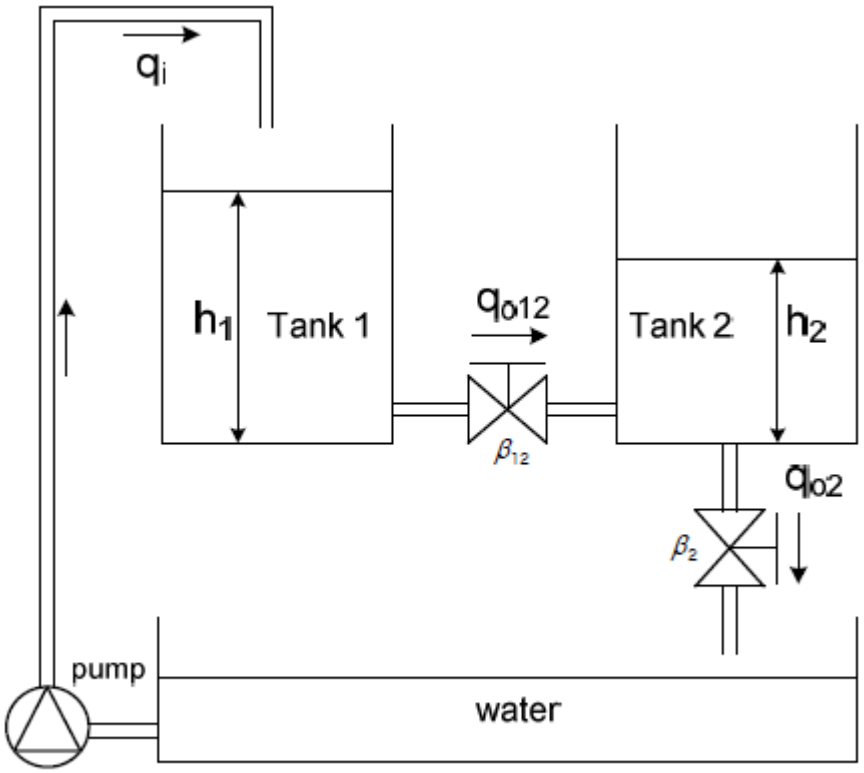



**Figure 1:** The coupled tank system.

The nonlinear equation can be obtained by mass equivalent equation and Bernury's law is given by.

$$\frac{dh_1(t)}{dt} = -\frac{\beta_{12}a_{12}}{A_1}\sqrt{2g\big(h_1(t)-h_2(t)\big)} + \frac{k}{A_1}u(t) \qquad (1a)$$

$$\frac{dh_2(t)}{dt} = -\frac{\beta_2 a_2}{A_2}\sqrt{2gh_2(t)} + \frac{\beta_{12}a_{12}}{A_2}\sqrt{2g(h_1(t)-h_2(t))} \qquad (1b)$$

where A1 and A2 are the cross section area (cm2) of tank 1 and tank 2, a2 is the cross section area (cm2) of outlet of tank 2, a12 is the cross section area (cm2) of jointed pipe between tank 1 and tank 2, β2 is the value ratio at the outlet of tank 2, β12 is the value ratio between tank 1 and tank 2, g is the gravity (cm/s2) and k is the gain of pump (cm3/V × s).
According to Equation (1), a linearized model is given by Equation (2).

$$G(s) = \frac{H_2(s)}{U(s)} = \frac{K}{T_{12}T_2s^2+(T_{12}+2T_2)s+1} \qquad (2)$$

where T12 is the time constant between tank 1 and tank 2, and T2 is the time constant of tank 2. T12, T2 and K can be obtained via the Equation (3) as follows.

$$T_{12} = \frac{A_1}{\beta_{12} a_{12}} \sqrt{\frac{2(\overline{h_1} - \overline{h_2})}{g}}, \quad T_2 = \frac{A_2}{\beta_2 a_2} \sqrt{\frac{2\overline{h_2}}{g}}, \quad \mathrm{K} = \frac{kT_2}{A_2} \qquad (3)$$

Where $h_1^{-} and\ h_2^{-}$ is the water level at the operating point of the coupled tank system.
For the coupled tank parameters are obtained by experimental results or data sheet and the parameters including the operating point of the process as shown in Table 1 [1].

**Table 1:** Parameters of coupled tank system

| $A_1, A_2$ | $a_2, a_{12}$ | $\beta_2$ | $\beta_{12}$ | $\overline{h_1}$ | $\overline{h_2}$ | k |
|---|---|---|---|---|---|---|
| 154 | 0.5 | 0.682 | 1.531 | 30 | 25 | 30 |

Using Table 1 and Equation (2), the transfer function in operating points is shown below and this model is simulated in MATLAB.

$$G(s) = \frac{19.8617}{2070.8939\, s^2 + 224.2254\, s + 1}$$

Similarly, hand analysis was performed to validate the form of the final transfer function. The steps were generally the ones taken in the mechanical modeling tutorials. Hand analysis is provided in the following section.

## III. Software components (MATLAB):

In order to simulate the level controller and calculate its parameters and characteristics, the following code was implemented in MATLAB:

```
clear all; clc; close all;
s=tf('s');
t=0:0.01:5;
%define function
%open loop
G=(19.8617)/(2070.8939*s^2+224.2254*s+1)
%closed loop
T=feedback(G,1);
%obtain damping
damp(T)
%obtain overshoot, undershoot, tr, ts, peak, etc
stepinfo(T)
%obtain ess
%e=1/s*s*1/1+openloopgain
%ess=lim(e)as s->zero
%knowing that the open loop gain is G
E=1/(1+G);
ess=dcgain(E);
ess
%plot step response
step(T)

figure %initialize figure
t = 0:0.01:5; %small time for plot
plot(step(T, t)); %step
hold on
title('Step Response base')
legend('Step','Ramp','Parabolic')
xlabel('Time')
ylabel('Amplitude')
```

In this code, we simply initialize the transfer function 'tf' in s domain, followed by identifying the form of this TF as an open loop. Then employed feedback() built-in function to obtain the closed loop. The TF characteristics are extracted in addition to the steady state error with a unit step input. Built-in function damp(T) calculates the poles, damping, frequency, and time constants. Finally, the step response is plotted.

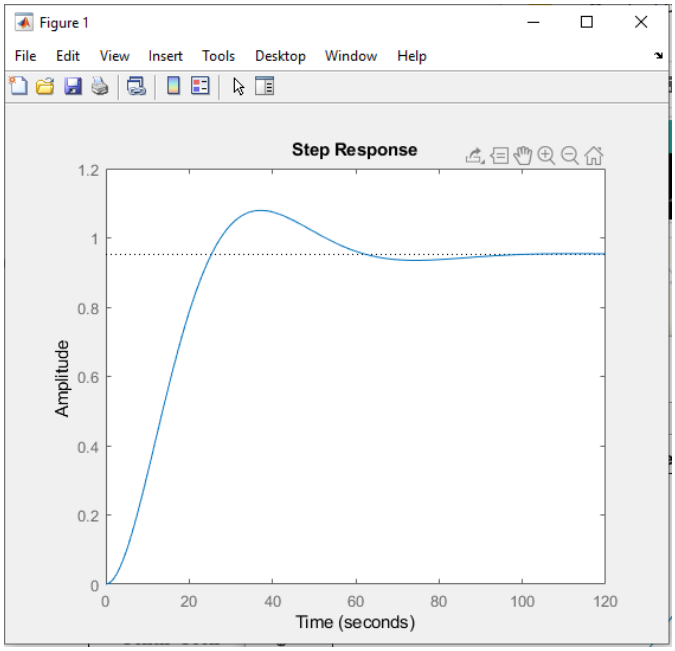
```
G =

          19.86
  ----------------------
  2071 s^2 + 224.2 s + 1

Continuous-time transfer function.
```

```
          Pole              Damping       Frequency      Time Constant
                                        (rad/seconds)      (seconds)

-5.41e-02 + 8.45e-02i      5.39e-01       1.00e-01          1.85e+01
-5.41e-02 - 8.45e-02i      5.39e-01       1.00e-01          1.85e+01
```

```
        RiseTime: 17.1210
    SettlingTime: 57.8193
     SettlingMin: 0.8606
     SettlingMax: 1.0793
       Overshoot: 13.3627
      Undershoot: 0
            Peak: 1.0793
        PeakTime: 37.4284
```

```
ess =

    0.0479
```

## IV. Simulation Results and comparison:



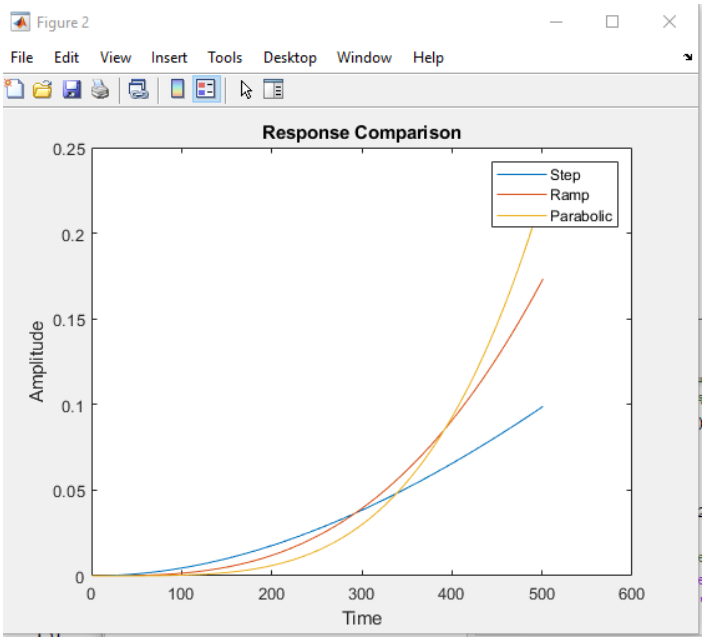



Figure 2(a) figure 2(b)

The simulation results we have obtained are similar to the ones found in [1] where we got settling time of 57.8 while they got 40 using the PSO method and even more (in the range from 75 to 110) using other methods. Rise time on the other hand was found to be 3.8 using PSO while we got 17 similar to the other models in [1]. We may use the PSO technique to gain better results and performance but we know that any way results would face some type of error and lack of performance when done in real time due to the inaccuracy of the equipment we will use.

In the coming phase, we hope to identify the effects of hardware and software (PID tuning in MATLAB) on the performance characteristics of the model. The values and plots obtained from Lee et al [1] are shown below.

**Table 3:** Comparison of time domain specifications for set-point tracking

| Transient Characteristic | Z-N | C-C | IMC | PSO |
|---|---|---|---|---|
| Rise time[s] | 11.07 | 11.33 | 22.22 | 3.80 |
| Peak amplitude | 32.43 | 32.13 | 30.25 | 31.96 |
| Peak time[s] | 29.02 | 29.42 | 46.15 | 9.35 |
| Settling time[s] | 111.19 | 101.58 | 75.58 | 40.01 |

**Table 2:** Comparison of PID controller parameters

| Tuning rules | PID parameters | | |
|---|---|---|---|
| | $K_p$ | $K_i$ | $K_d$ |
| Zeigler-Nichols | 1.4928 | 0.0855 | 6.5159 |
| Cohen–Coon | 1.6712 | 0.0791 | 5.2666 |
| IMC | 1.0153 | 0.0046 | 4.3440 |
| PSO | 12.144 | 0.030 | 9.349 |

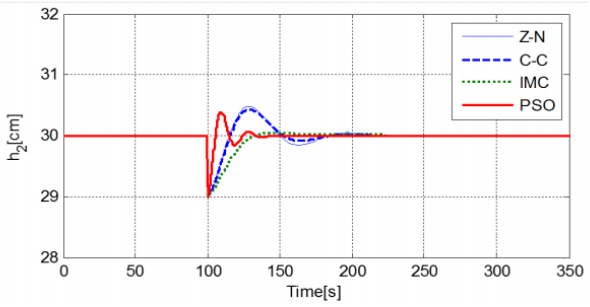



Figure 3:Closed-loop responses for disturbance rejection with PID Controllers

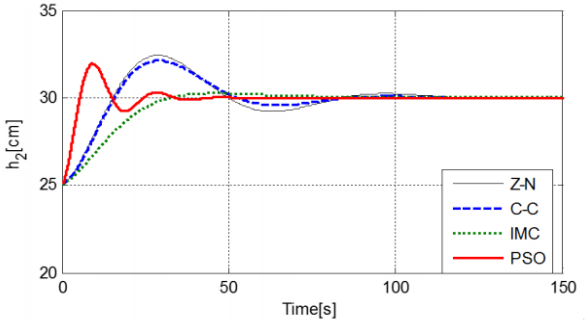



figure 4:Closed-loop responses for set-point tracking PID Controllers

Also the schematic of the PSO based PID control system is shown for convenience from [1].

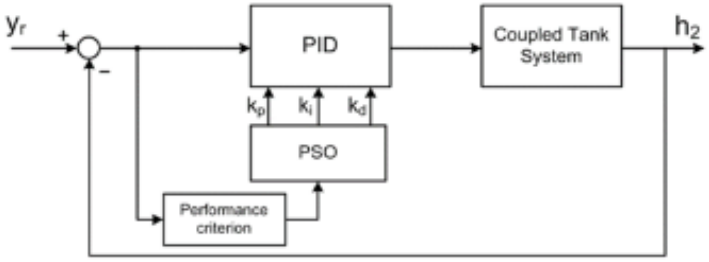



Figure 5: structure of PSO based PID control system

We hope to further improve and compare on these values in the PID phase in MATLAB. Qualitative descriptions of the following parameters are obtained:

## 2nd Order System Time Domain Specs

- The time response characteristics of the system is drawn below.

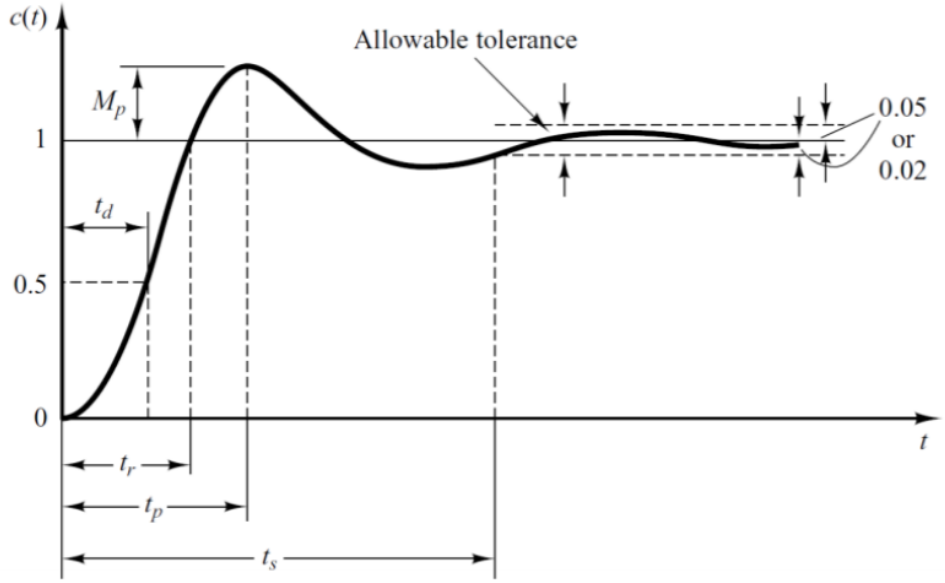



Figure 6: time response

## V. PID controller Matlab quantitative Optimization

In order to simulate the PID controller, first we created the module in Simulink in MATLAB as follows:

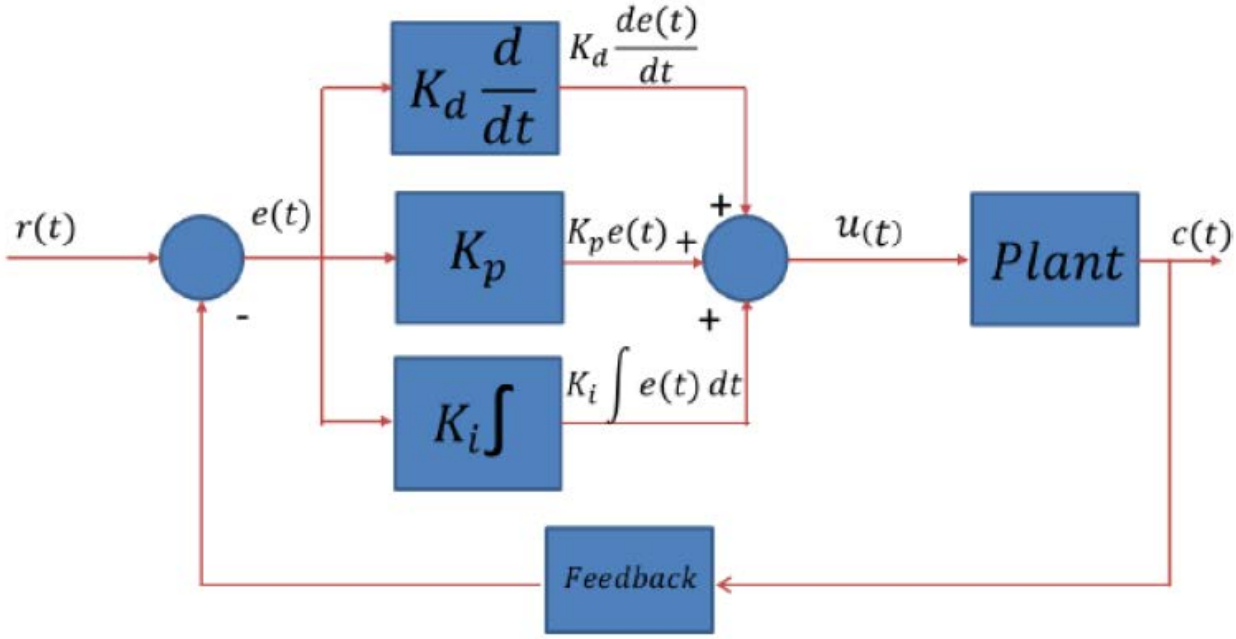



Figure 7

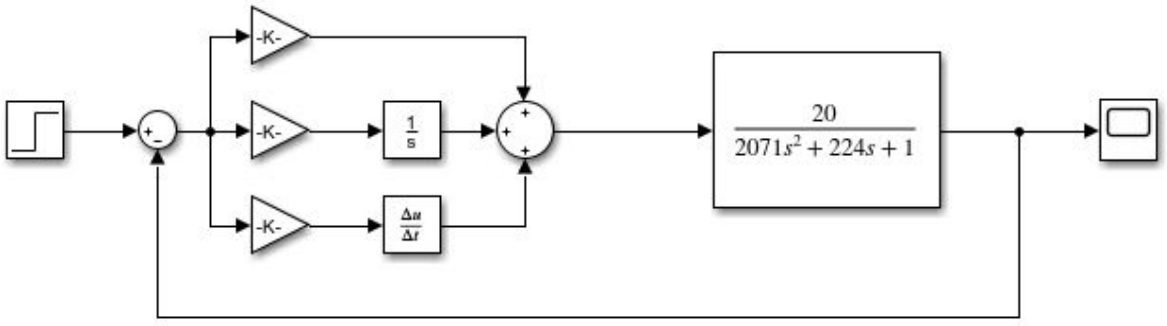



Figure 8

Where K's are the Kp, Ki, Kd values, respectively from top to bottom. We carried out two flows of optimizations, the initial being optimizing from Lee et al. [1] findings where we used the PID values from table two as initial estimates. This was carried out as per tuning rules Zeigler-Nichols (Z-N) &Cohen–Coonn (C-C) with values:

Table 4:

| Tuning rules | PID parameters | | |
|---|---|---|---|
| | $K_p$ | $K_i$ | $K_d$ |
| Zeigler-Nichols | 1.4928 | 0.0855 | 6.5159 |
| Cohen–Coon | 1.6712 | 0.0791 | 5.2666 |

For Zeigler-Nichols and Cohen-Coon based optimizations the initial closed loop TF took the following curve shape, respectively:

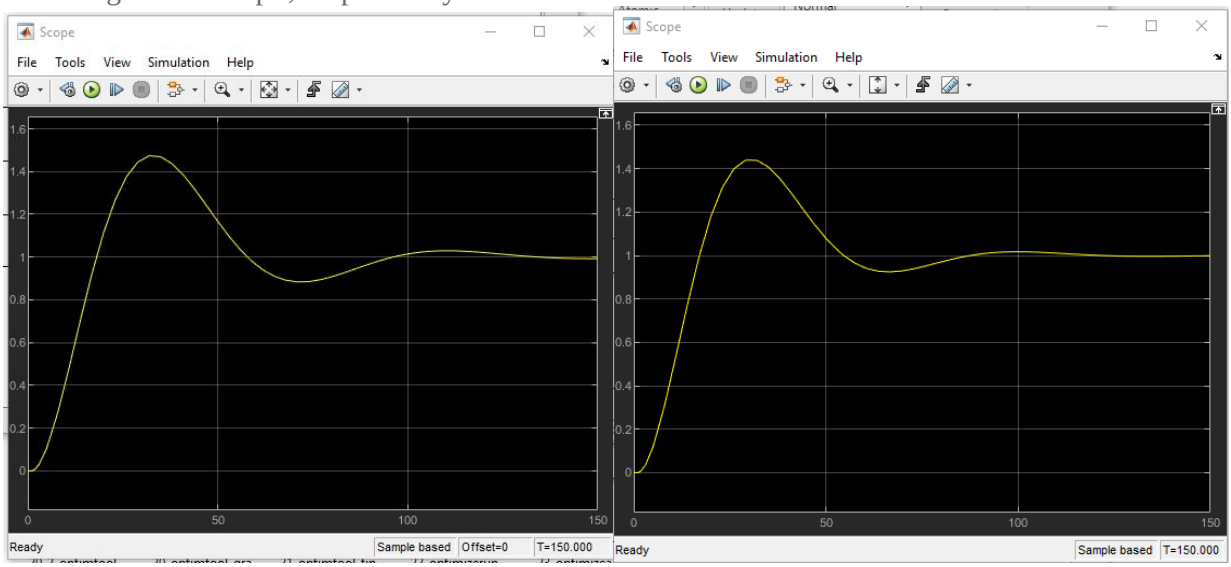



Figure 9: Z-N tuning rules figure10: C-C tuning rules

The following simulink structure was simulated to calculate their time domain responses but also with their performance index from the two of the three major error criterion techniques of Integral of Absolute Error (IAE), Integral Time of Absolute Error (ITAE), while Integral Square of Error (ISE) was not calculated in simulink.

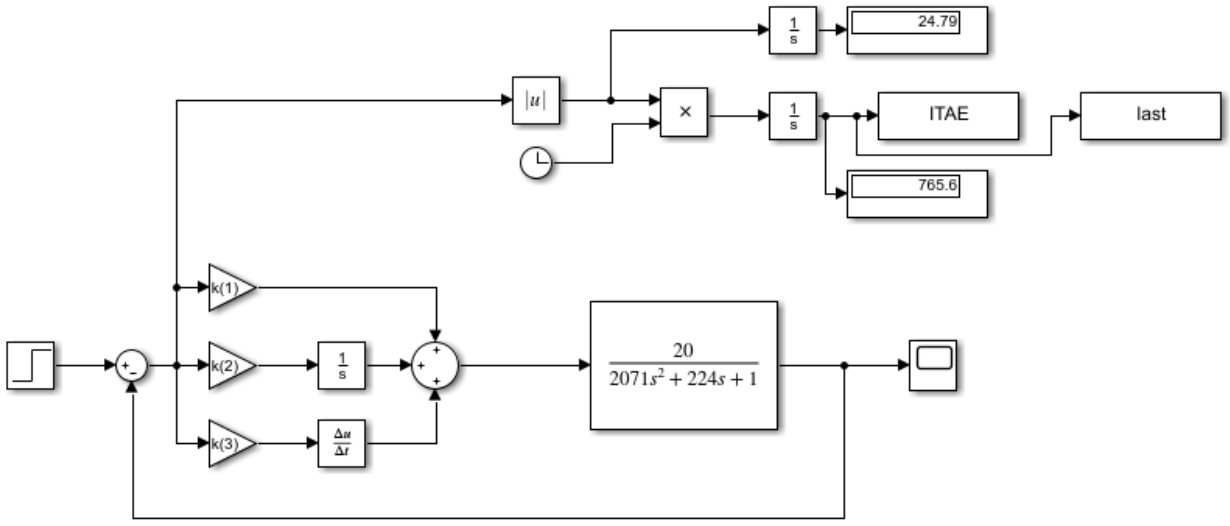



Figure 11

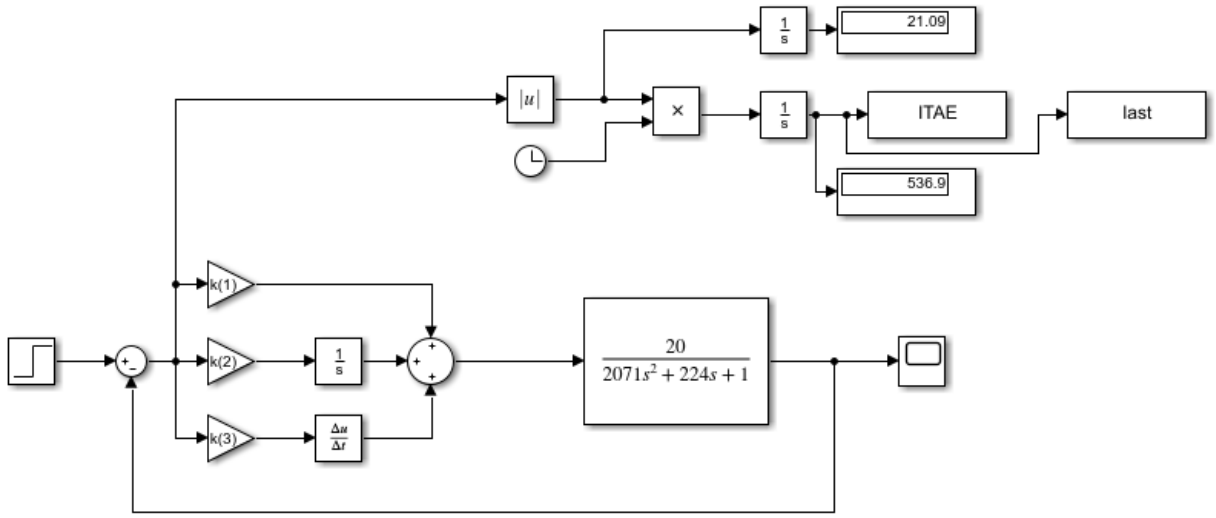



Figure 12

IAE values are shown in top display while ITAE values are shown in bottom display for both Z-N and C-C. ITAE values are sent to the workspace along with the last (instantaneous) value of these values. These values are summarized in the following table:

Table 5

| Performance index | Z-N | C-C |
|---|---|---|
| IAE | 24.79 | 21.09 |
| ITAE | 765.6 | 536.9 |

In order to optimize these values, a function otptimize_PID(k) was implemented that takes the array of k values as an input, these values are manually inserted into the command window.

optimize_PID.m

```
function cost = optimize_PID(k)
    %optimize code that simulates simulink block and retreives last ITAE value
    assignin('base','k',k); %takes k parameters from command window
    sim('Tuning_PID.slx');  %simulates simulink block
    cost=last;              %last element retrieved from simulnk of ITAE

end
```

```
>> k=[1.4928 0.0855 6.5159]

k =

    1.4928    0.0855    6.5159
```

```
>> k=[1.6712 0.0791 5.2666]

k =

    1.6712    0.0791    5.2666
```

Also, the matab script extracts the ITAE matrix which is transposed and saved in array form 'ITAE.mat'. Then calling optimize_PID(k) in the command window, the result of optimization is calculated. Therefore, a quantitative sweep is carried out via the optimtool in Matlab:

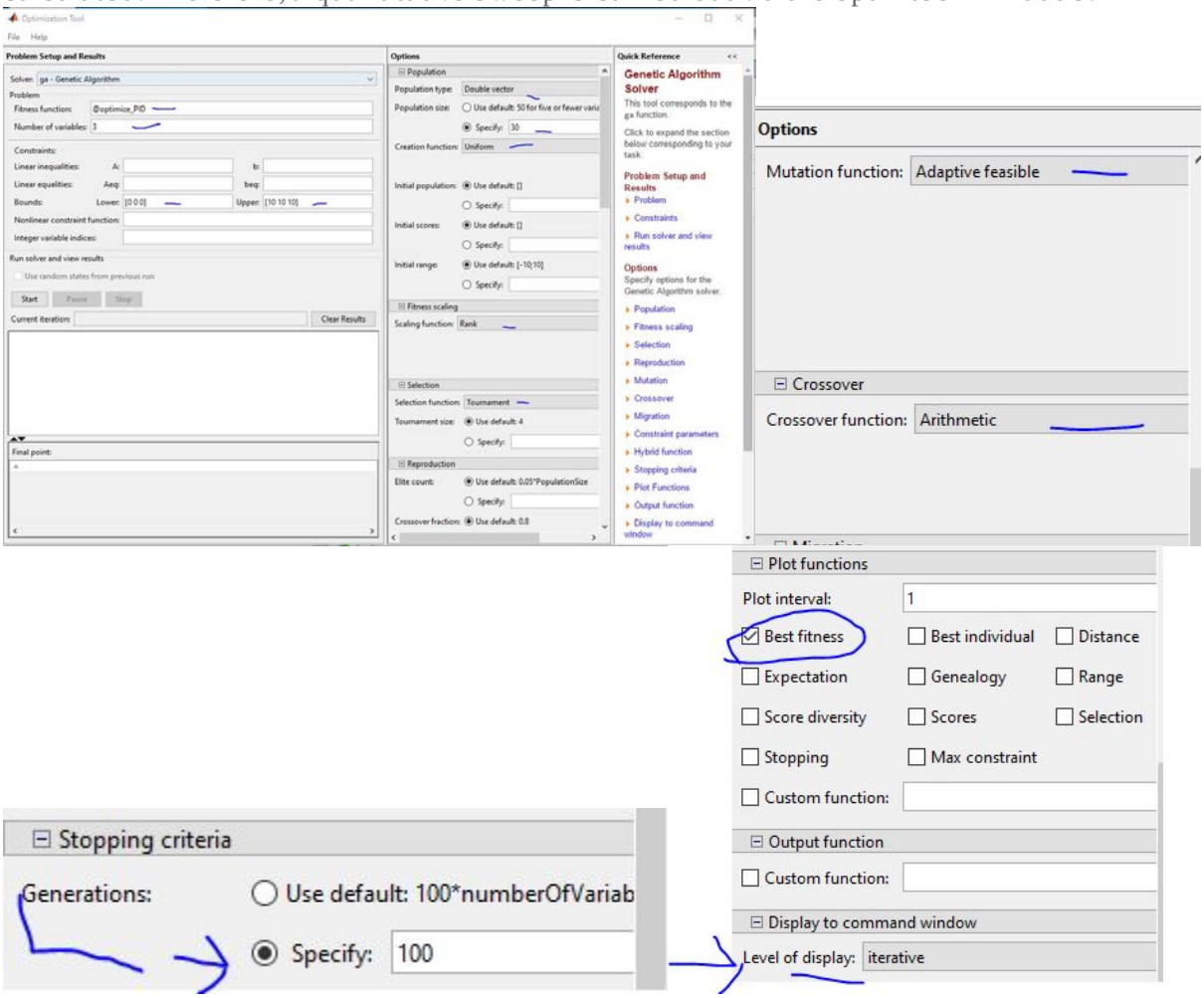



Figure 13

As shown in the optimization tool above, we applied the optimize_PID function as the fitness function, and used the Genetic Algorithm solver with population size 30 and stopping criteria after 100 generations for best values.

Output of genetic algorithm runs for Z-N and C-C runs are shown below, with final points:

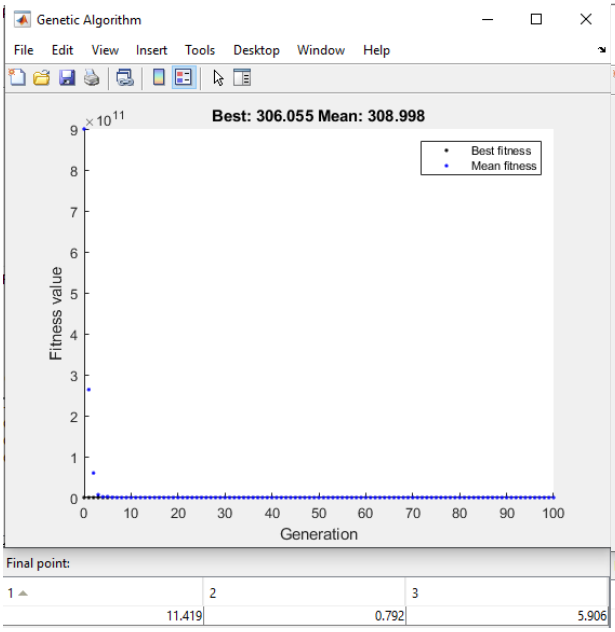



Figure 14

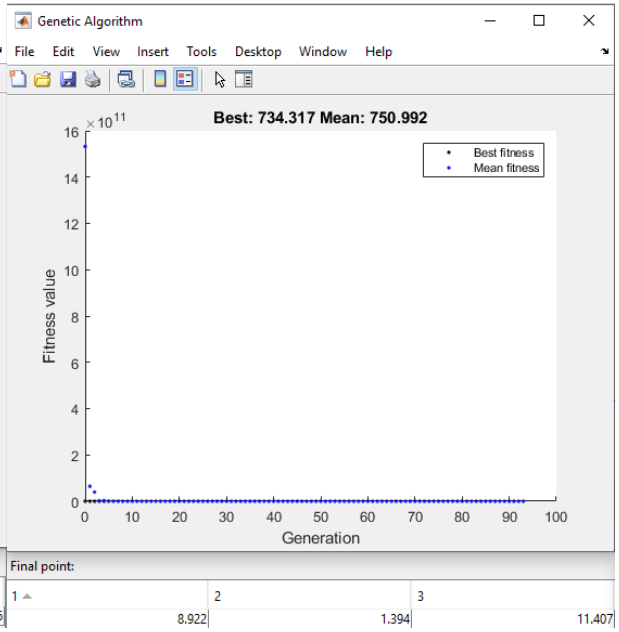



figure 15

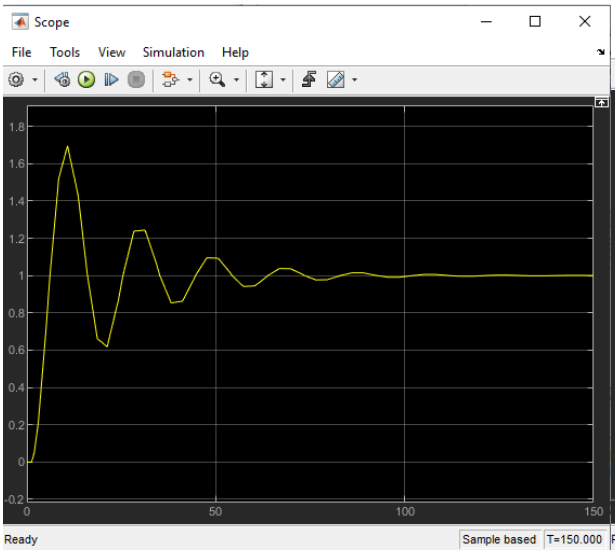



Figure 16

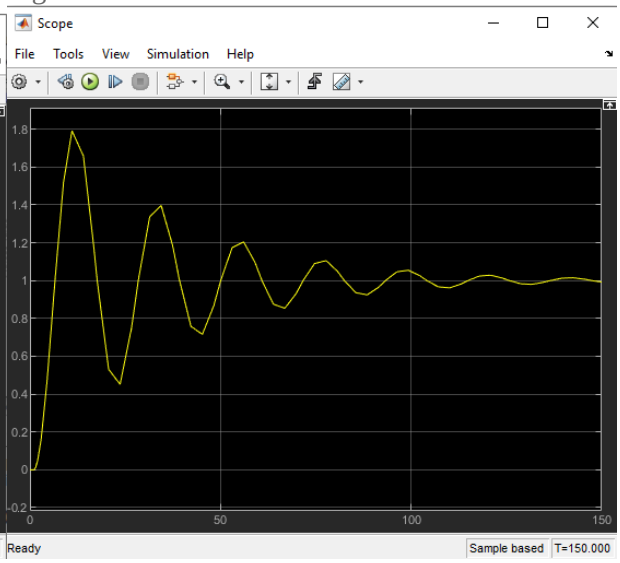



figure 17

From the curve structures, when compared to the initial curves prior to optimization, it can be concluded that the initial values for Z-N and C-C are preferred over these final values.

After plugging in these final points as k values in workspace and running optimization_PID(k) function, IAE and ITAE values along with TF structure were extracted as follows:

Table 6

| Performance index | Z-N | C-C |
|---|---|---|

| | | |
|---|---|---|
| IAE | 14.29 | 22.84 |
| ITAE | 306 | 735.6 |
| PID parameters k=[kp ki kd] | k=[11.419 0.792 5.906] | k=[8.922 1.394 11.407] |

## VI. PID controller Matlab qualitative Optimization

For a qualitative analysis, the TF from phase 1 was edited via the PID tuner in matlab, resulting in the two forms shown below, with their characteristics:

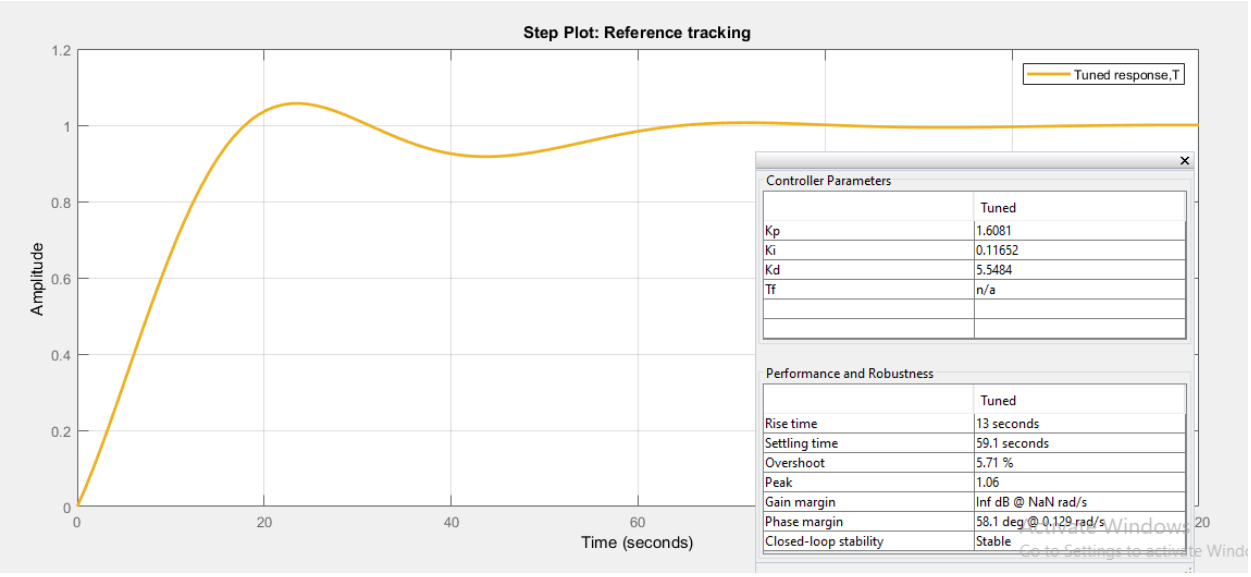



Figure 18 plot 1

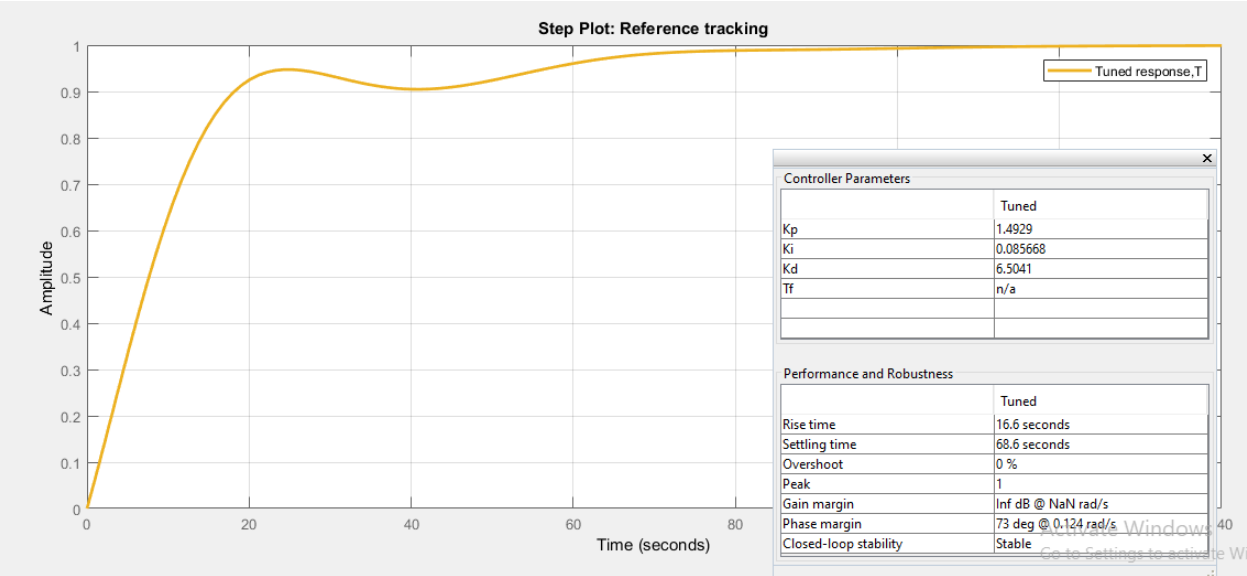



Figure 19 plot 2

After plugging in these k values into the simulink module, the following curves and timing performance values were achieved:

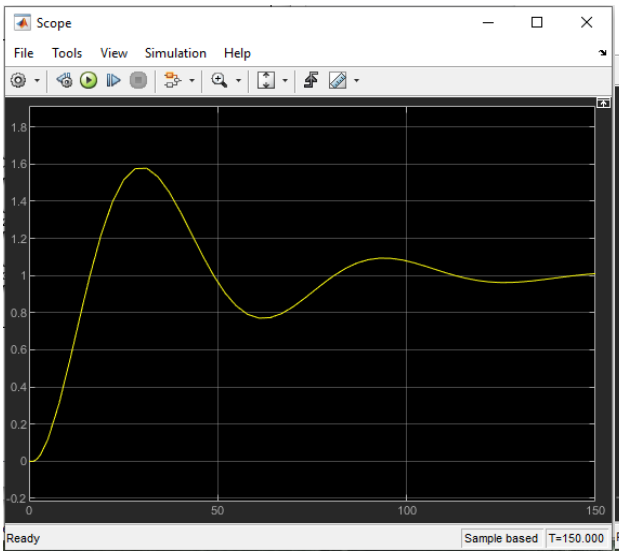



Figure 20 : plot 1

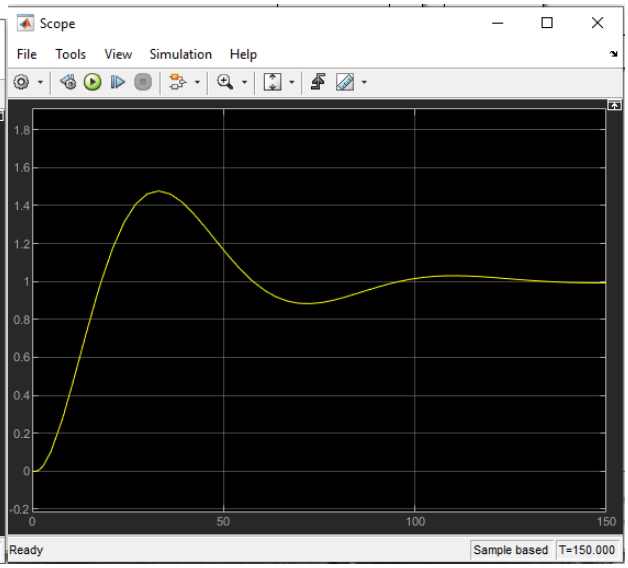



figure 21: plot 2

Table 7

| Performance index | Plot 1 | Plot 2 |
|---|---|---|
| IAE | 28.15 | 24.82 |
| ITAE | 998.7 | 767.3 |
| PID parameters k=[kp ki kd] | k=[1.6081 0.11652 5.5484] | k=[1.4929 0.085668 6.5041] |

After running optimization on these values, the following results were obtained:

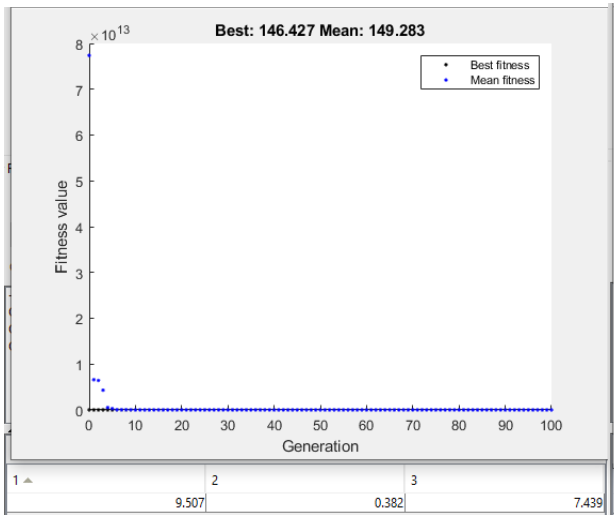



Figure 22

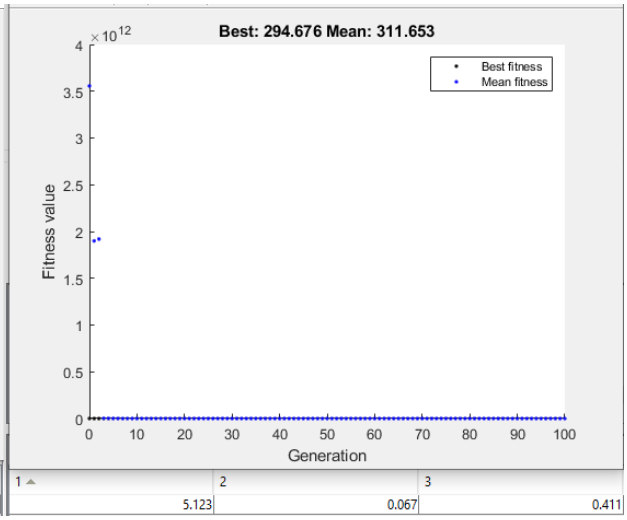



figure 23

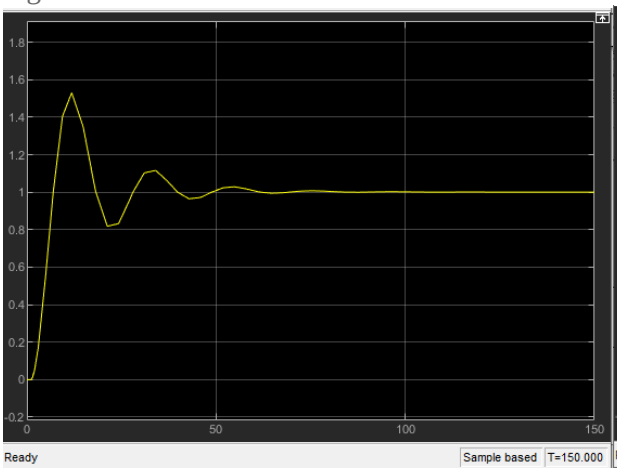



Figure 24 plot 1

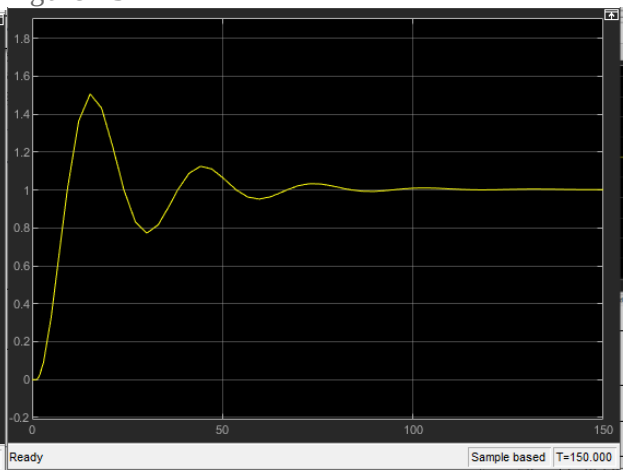



figure 25 plot 2

Table 8

| Performance index | Plot 1 | Plot 2 |
|---|---|---|
| IAE | 9.996 | 13.98 |
| ITAE | 146.6 | 295.1 |
| PID parameters k=[kp ki kd] | k=[9.507 0.382 7.439] | k=[5.123 0.067 0.411] |

## VII. Discussion:

Figures 9 and 10 before the PID tuning after the tuning in figures 16 and 17 oscillation increase,

The rising time decreased while the settling time slightly changed and overshoot increased. Figures 20 and 21 before the PID tuning after the tuning we can see in figures 24 and 25 oscillation increase , maximum overshoot decreased , settling time decreased while rising time decreased, Which indicates that PID tuning has increased the optimization of the system.

## VIII. References:


1. Lee, Yun-Hyung & Ryu, Ki-Tak & Hur, Jae-Jung & So, Myung-Ok. (2014). PSO based tuning of PID controller for coupled tank system. Journal of the Korean Society of Marine Engineering. 38. 1297-1302. 10.5916/jkosme.2014.38.10.1297.
2. Lee, Yun-Hyung & Jin, Gang-Gyoo & So, Myung-Ok. (2014). Level control of single water tank systems using Fuzzy-PID technique. Journal of the Korean Society of Marine Engineering. 38. 550-556. 10.5916/jkosme.2014.38.5.550.
3. Water level controller using arduino. Water level indicator using arduino
4. Jiffy Anna John, Dr. N. E. Jaffar and Prof. Riya Mary Francis, 2015. Modelling and Control of Coupled Tank Liquid Level System using Backstepping Method. International Journal of Engineering Research and, V4(06).