

**In the Name of God**



**Faculty of Electrical Engineering**

## **Design of PID Controller for Inverted Pendulum**

**Student Name:**

**Amirreza Zarrabinia**

**Instructor Name:**

**Dr. Soheil Ganjefar**

**January 2025**

## Table of Contents

<b>Introduction .....</b>	<b>3</b>
<b>1. Nonlinear Simulation of the Inverted Pendulum .....</b>	<b>4</b>
1.1. Method1 (Not suggested) .....	4
1.1.1. System Identification .....	9
1.1.1.1. Three-Parameter Model .....	9
1.1.1.2. Four-Parameter Model .....	14
1.2. Method2 .....	17
1.2.1. System Identification.....	18
1.2.1.1. Integral Identification .....	18
1.2.2. Designing PID Controller for this Method .....	22
1.2.2.1. Ziegler-Nichols PID Tuning in the Time Domain .....	23
1.2.2.2. Ziegler-Nichols PID Tuning in Frequency Domain .....	27
1.2.2.3. Generalized Ziegler-Nichols PID Tuning .....	31
1.2.2.4. PID Design Using the Lambda Tuning Method .....	36
1.2.2.5. PID Design Using the CHR Method .....	36
1.3. Method3.....	39
<b>2. Linear Simulation of the Inverted Pendulum .....</b>	<b>40</b>
2.1. System Identification .....	44
2.2. Designing PID Controller for Linear System .....	48
2.2.1. Ziegler-Nichols PID Tuning in the Time Domain.....	49
2.2.2. Ziegler-Nichols PID Tuning in the Frequency Domain .....	53
2.2.3. Generalized Ziegler-Nichols PID Tuning .....	57
2.2.4. PID Design Using the Lambda Tuning Method.....	61
2.2.5. PID Design Using the CHR Method.....	61
<b>3. Conclusion.....</b>	<b>65</b>

**Introduction:**

In this project, the objective is to control an inverted pendulum system using various PID controllers. The goal is to apply a force, which serves as the system's input, to stabilize the pendulum at its upper equilibrium point (0 degrees) given an initial angle displacement.

The project explores three different methods for system identification. The first method is deemed unsuitable due to the nonlinear behavior of the system during identification. However, for this method, where the segment identified had the shape of S, identification was performed using both the three-component and four-component approaches.

However, the second method attempts to extract a portion of the linear behavior from the nonlinear system, and controllers are designed based on this identified model. Four types of controllers are developed for the system identified using this approach, and their results are discussed in the conclusion.

The third method is theoretically the best for a nonlinear system but could not be implemented successfully due to the unstable and nonlinear behavior of the system.

Subsequently, the nonlinear system is linearized, and four types of controllers are designed for the linearized system as well. The designed controllers from all methods are then applied to the original nonlinear system, and the results are thoroughly analyzed.

## 1. Nonlinear Simulation of the Inverted Pendulum

### 1-1. Method1 (Not suggested)

Note that:

$$x(0) = v(0) = \omega(0) = 0,$$

Figure 1- Initial conditions

Note that the initial angle applied to the system is zero degrees.

To simulate the system, four integrator blocks are required to compute the state variables, along with four function blocks to implement the governing equations of the inverted pendulum.

The figure below illustrates the overall structure of the inverted pendulum simulation in the MATLAB environment:

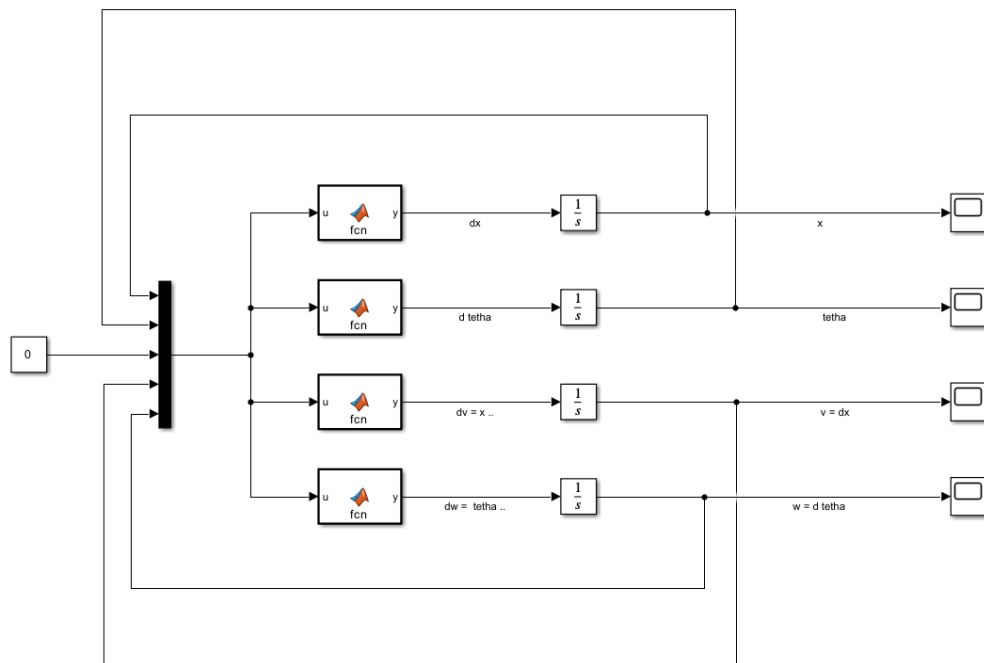


Figure 2- Overall structure

Additionally, the inputs and outputs of each integrator block are clearly indicated in the diagram above.

The governing equations of the inverted pendulum in state-space form are as follows:

$$\begin{cases} \dot{v} = \frac{F + m \ell \omega^2 \sin \theta - m g \sin \theta \cos \theta}{M + m (1 - \cos^2 \theta)} \\ \dot{\omega} = \frac{-F \cos \theta - m \ell \omega^2 \sin \theta \cos \theta + (M + m) g \sin \theta}{\ell [M + m (1 - \cos^2 \theta)]} \\ \dot{x} = v \\ \dot{\theta} = \omega \end{cases}$$

Figure 3- State space

Therefore, the functions are defined based on the equations above.

Note that, considering the inputs of the multiplexer, the following relations are taken into account:

$$\begin{aligned} u(1) &= x \\ u(2) &= \theta \\ u(3) &= F \\ u(4) &= \frac{dx}{dt} = v \\ u(5) &= \frac{d\theta}{dt} = \omega \end{aligned}$$

Equation 1- Defining inputs

Based on these relations, understanding the equations implemented in the function blocks becomes easier.

First Block:

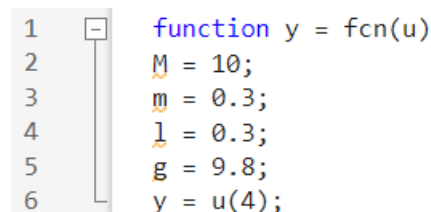


Figure 4- First block's function

Second Block:

```
1 function y = fcn(u)
2 M = 10;
3 m = 0.3;
4 l = 0.3;
5 g = 9.8;
6 y = u(5);
```

Figure 5- Second block's function

Third Block:

```
1 function y = fcn(u)
2 M = 10;
3 m = 0.3;
4 l = 0.3;
5 g = 9.8;
6 y = (u(3) + m*l*u(5)*u(5)*sin(u(2)) - m*g*sin(u(2))*cos(u(2)))/(M + m - m*cos(u(2))*cos(u(2)));
```

Figure 6- Third block's function

Fourth Block:

```
1 function y = fcn(u)
2 M = 10;
3 m = 0.3;
4 l = 0.3;
5 g = 9.8;
6 y = (-u(3)*cos(u(2)) - m*l*u(5)*u(5)*sin(u(2))*cos(u(2)) + M*g*sin(u(2)) + m*g*sin(u(2)))/(l*M + l*m - l*m*cos(u(2))*cos(u(2)));
```

Figure 7- Fourth block's function

The output angle of the pendulum, when a step input is applied, with an initial angle of zero, will be as follows.

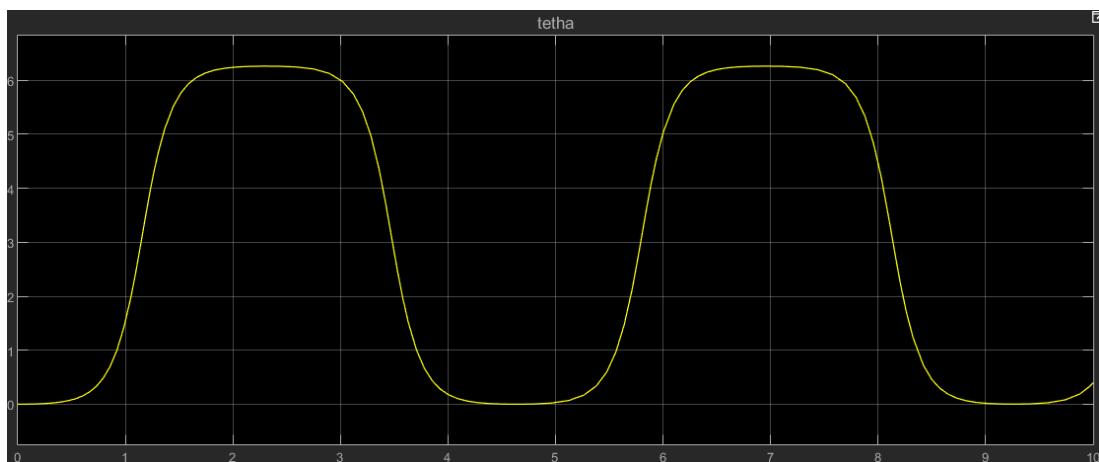


Figure 8- The angle output of nonlinear system with a step input.

The simulated system is as follows:

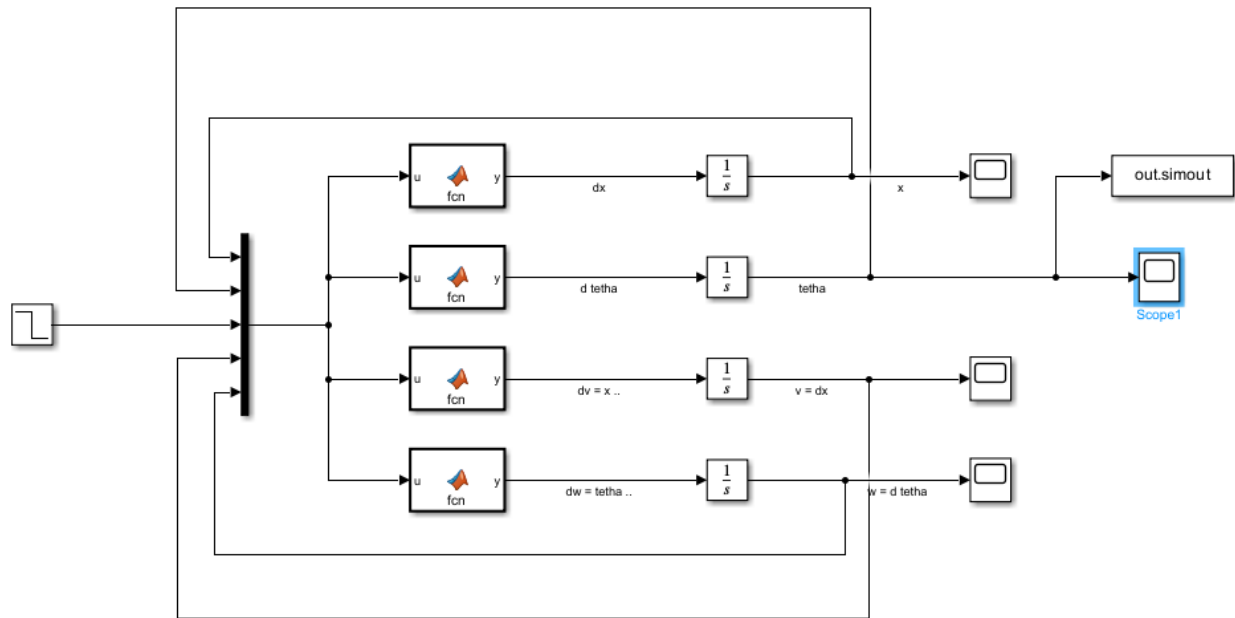


Figure 9- Simulation of the nonlinear pendulum with step input

Therefore, to achieve better control and to ensure that the system's response exhibits linear behavior over a longer period and that the output remains in the s-shape for a longer time, a periodic input with a low amplitude is applied to the system.

A Pulse Generator block is used to generate the pulse, as shown in the figure below, which provides its details.

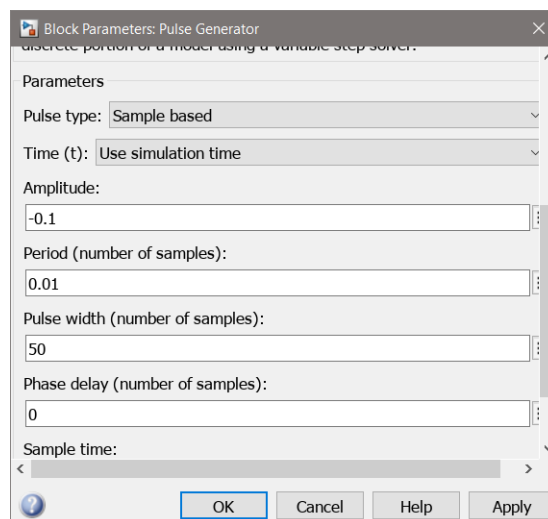


Figure 10- Pulse Generator settings

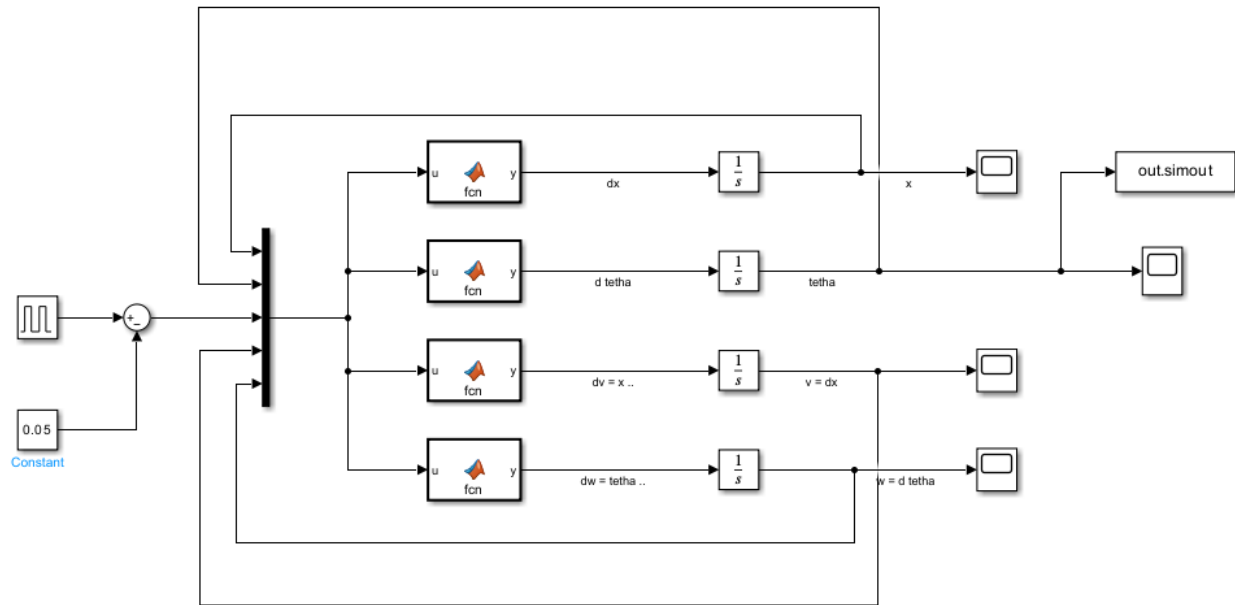


Figure 11- Simulation of the nonlinear pendulum with pulse input

The output is as follows:

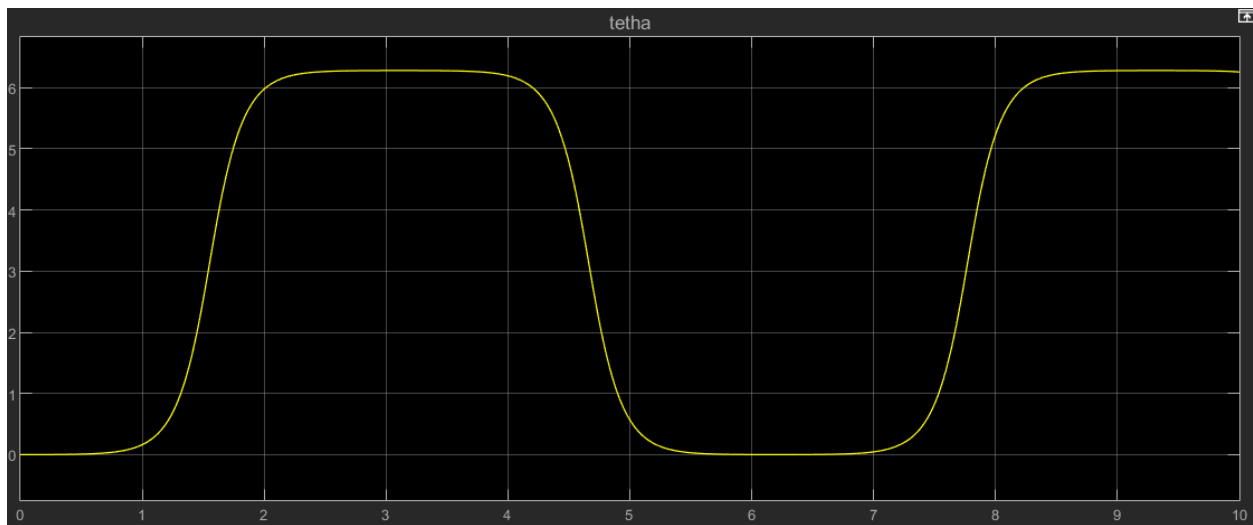


Figure 12- The angle output of nonlinear system with a pulse input

It can be observed that with a pulse input, the system exhibits linear behavior for a longer duration and produces an S-shaped response.

We will use this S-shaped response for identifying the nonlinear system.



## 1-1-1. System Identification

### 1-1-1-1. Three-Parameter Model

We had:

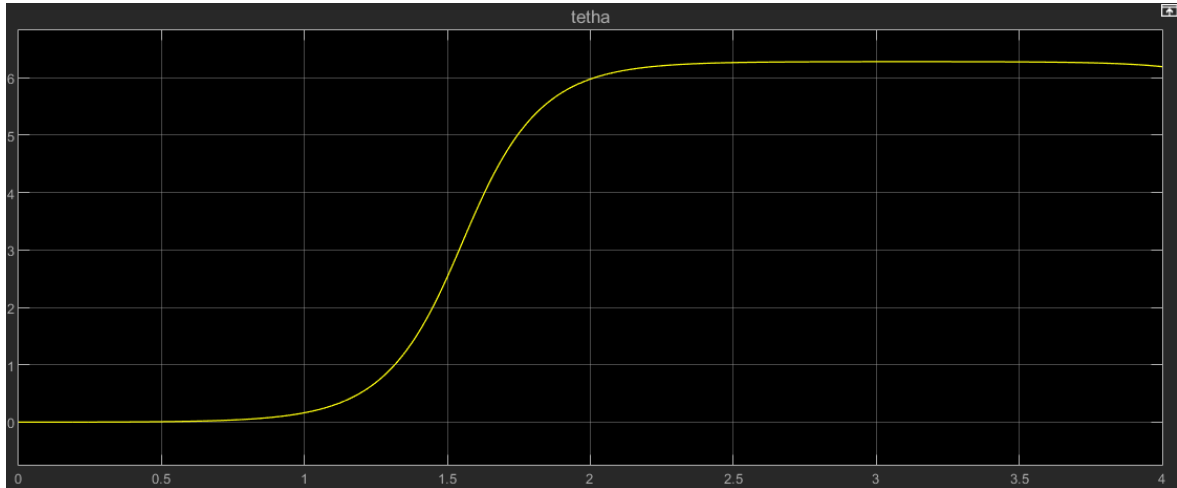


Figure 13- The normalized angle output of nonlinear system with a pulse input

Then the "To Workspace" block is used for analyzing the output data of this system.

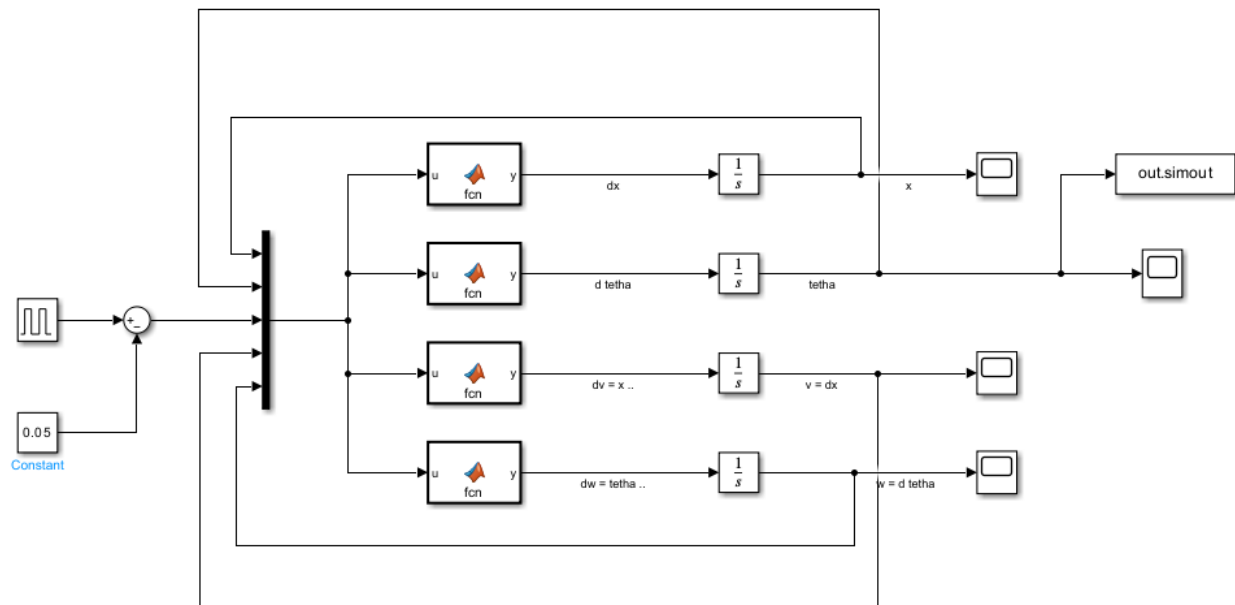


Figure 14- Applying "To Workspace" to the nonlinear model

First, we obtain the output in the MATLAB environment:

```
1  
2
```

```
yout = out.simout;  
plot (yout);
```

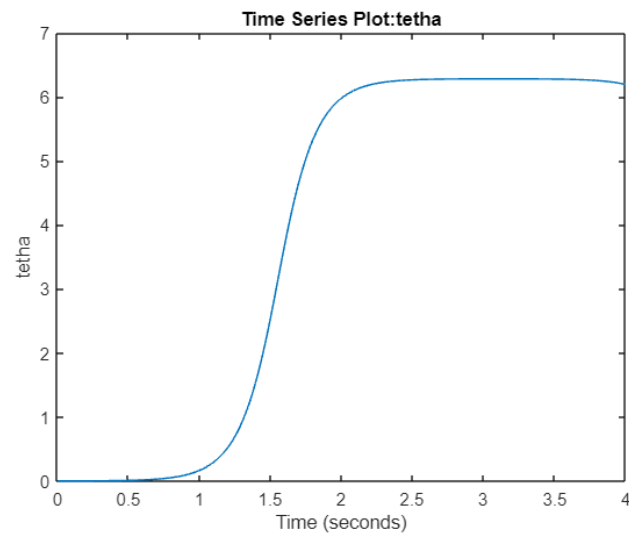


Figure 15- Obtaining the output in the MATLAB

To find the parameters, it is necessary to find the linear equation that is tangent to the system's response at the steepest slope.

For this, we need to take the derivative of the system's response and consider the maximum value of the derivative as the slope, then find the corresponding linear equation.

We proceed as follows:

```
3 signal = yout.Data(:, 1);
4 time = yout.Time;
5 dt = diff(time);
6 dsignal = diff(signal) ./ dt;
7
8 [max_slope, idx] = max(dsignal);
9
10 x_tangent = time(idx);
11 y_tangent = signal(idx);
12
13
14 y = max_slope * (time - x_tangent) + y_tangent;
15
16 plot(time, signal, 'b');
17 hold on;
18 plot(time, y, 'r--');
19 ylim([-1 7]);
```

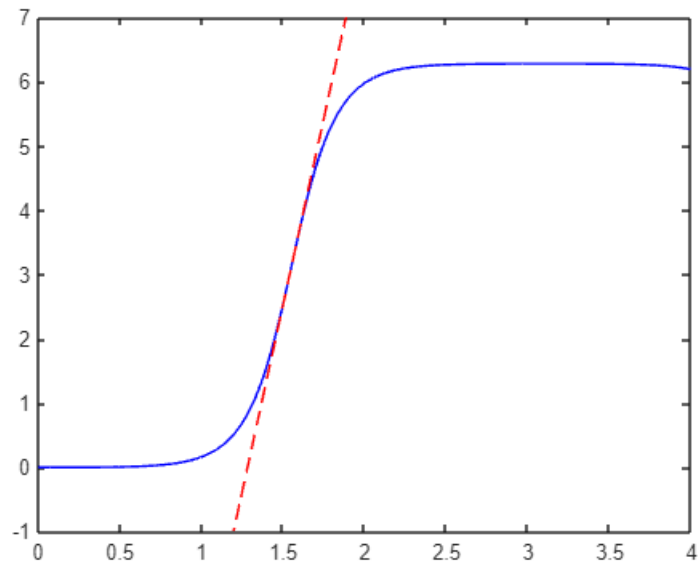


Figure 16- The tangent line to the nonlinear output

The parameter  $L$  can be easily calculated from the equation of this line.

```
20 L = (-y_tangent / max_slope) + x_tangent
L = 1.2824
```

Figure 17- Calculating L for method1

If we want to analyze the response, we can say that the initial angle is 0 degrees, and by applying force, the pendulum moves away from the stable position and then, over time, reaches an angle of 6.28 radian, which corresponds to the initial reference point. It stays in this state for a few seconds.

Now, the response tends toward 6.28. As a result:

$$K = 6.28$$

Now, it is necessary to find  $T$ . In the exercises, we observed that the approximate value of  $T$ , when considering 0.63 of the final value, is closer to the actual response.

```
21 K = 6.28;
22 B = (0.63*K - y_tangent)/max_slope + x_tangent;
23 T = B - L
T = 0.3411
```

Figure 18- Calculating T and defining K for method1

We have:

$$G_3(s) = \frac{6.28}{0.3411s + 1} e^{-1.2824s}$$

Equation 2- The transfer function of the three-parameter system

Therefore, we enter the identified transfer function in MATLAB as follows, and then we observe the output of the main system and the three-parameter identified system together in the scope.

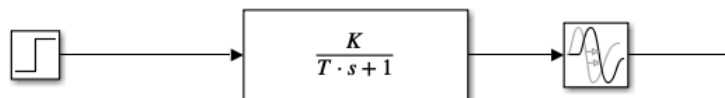


Figure 19- Three-parameter Identified system in the MATLAB

The output is as follows:

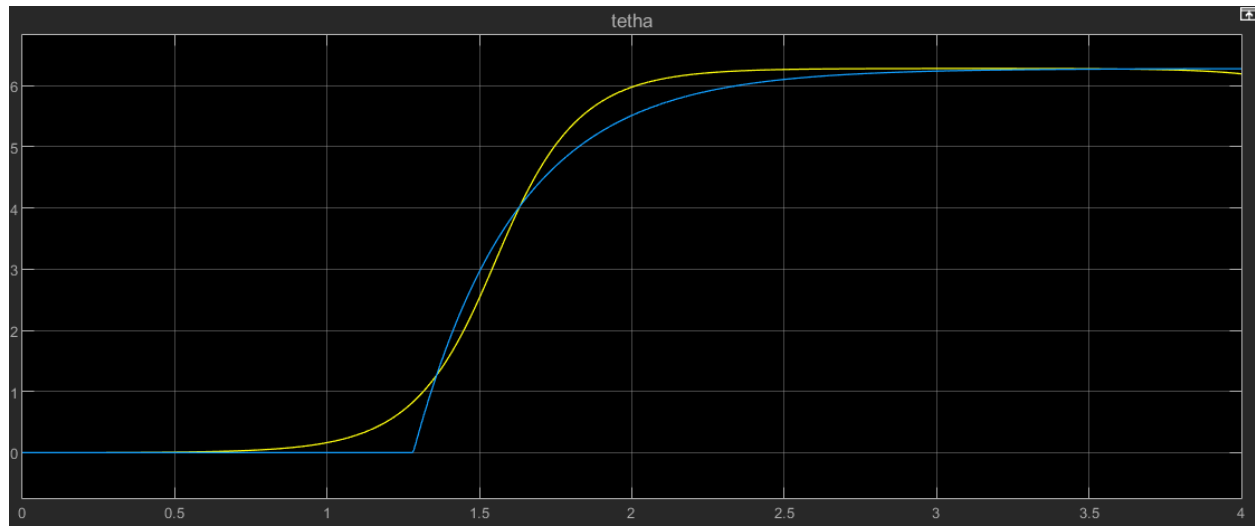


Figure 20- The output of the nonlinear system and the three-parameter system

It can be said that by executing this model, we were able to identify the system to a good extent.

### 1-1-1-2. Four-Parameter Model

To solve nonlinear equations in MATLAB, a command called "fsolve" can be used.

We know that the system response in the four-parameter model is as follows:

$$S(t) = K \left( 1 + \frac{\left( T_2 e^{-(t-L)/T_2} - T_1 e^{-(t-L)/T_1} \right)}{T_1 - T_2} \right)$$

Figure 21- The system response in the four-parameter model

First, it is necessary to calculate the time when the system reaches 50% and 67% of the final value in MATLAB.

```
24 S1 = (0.5*K - y_tangent)/max_slope + x_tangent
    S1 = 1.5531
25 S2 = (0.67*K - y_tangent)/max_slope + x_tangent
    S2 = 1.6451
```

Figure 22- Calculating S1 and S2

We have:

$$S(1.5531) = 0.5K$$

$$S(1.6451) = 0.67K$$

Equation 3- Defining S1 and S2

These parameters are defined in MATLAB.

```
27 S_data = [0.5*K 0.67*K];
28 t_data = [S1 S2];
```

Figure 23- Defining S1 and S2 in MATLAB

Then, the system response, which is  $S$ , is defined as follows:

```
29 fun = @(T) [
30     K * (1 + ((T(2) * exp(-(t_data(1) - L) / T(2)) - T(1) * exp(-(t_data(1) - L) / T(1))) / (T(1) - T(2)))) - S_data(1);
31     K * (1 + ((T(2) * exp(-(t_data(2) - L) / T(2)) - T(1) * exp(-(t_data(2) - L) / T(1))) / (T(1) - T(2)))) - S_data(2)
32 ];
```

Figure 24- Definition of the system response

In MATLAB, the “fsolve” function typically returns only one solution to an equation or system of equations, which depends on the initial guess provided. This is because “fsolve” works based on iterative methods to find the roots and converges to the closest solution to the initial guess.

Therefore, an initial guess must be provided to this command. Note that if this guess is too far off, it may not lead to a solution.

Now, the initial guesses are provided:

```
35 T0 = [0.01, 0.015];
36
37 options = optimoptions('fsolve', 'Display', 'iter', 'TolFun', 1e-10, 'TolX', 1e-10, 'MaxIterations', 1000);
38 solution = fsolve(fun, T0, options);
```

Figure 25- Initial guesses and calling fsolve function

Therefore,  $T1$  and  $T2$  are calculated as follows:

```
39 disp(['T1 = ', num2str(solution(1))]);
    T1 = 0.15933
40 disp(['T2 = ', num2str(solution(2))]);
    T2 = 0.15933
41 T1=0.15933;
42 T2=0.15933;
```

Figure 26- Calculating T1 and T2

Therefore, the transfer function of the four-parameter model will be as follows:

$$G_4(s) = \frac{6.28}{(1 + 0.15933s)^2} e^{-1.2824s}$$

Equation 4- The transfer function of the four-parameter system

Therefore, we enter the identified transfer function in MATLAB as follows, and then we observe the output of the main system and the four-parameter identified system together in the scope.

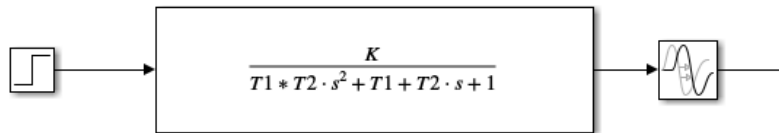


Figure 27- Four-parameter Identified system in the MATLAB

The output is as follows:

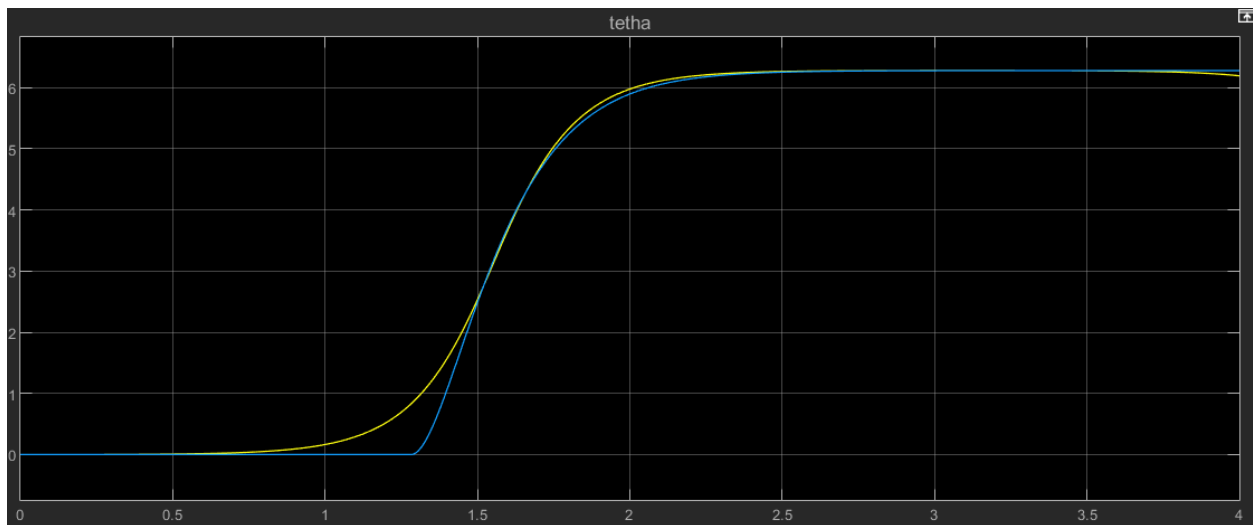


Figure 28- The output of the nonlinear system and the four-parameter system

It is clear that this model performs much better than the three-parameter model and has identified the nonlinear system with very high accuracy.



### The problem with this method:

Using this approach, we are identifying the nonlinear behavior of the system, which is incorrect. In fact, the identification we perform occurs when the pendulum moves away from its stable position and returns to it, during which the system exhibits nonlinear behavior.

### 1-2. Method2

Therefore, it is necessary to identify the system around its equilibrium point. To do this, we need to apply an input to the system so that it oscillates around the equilibrium point. However, the system becomes unstable with the slightest force applied, moving away from equilibrium. Hence, we attempt to analyze the system during the initial moment, which is very short, to remain close to the equilibrium point and assess the linear behavior of the system.

If a pulse with an amplitude of 0.1 and a frequency of 100 is applied to the system, it is possible to identify and analyze the system within the first 2 seconds, during which the pendulum's deviation from the equilibrium point is not significant, and the system exhibits relatively linear behavior.

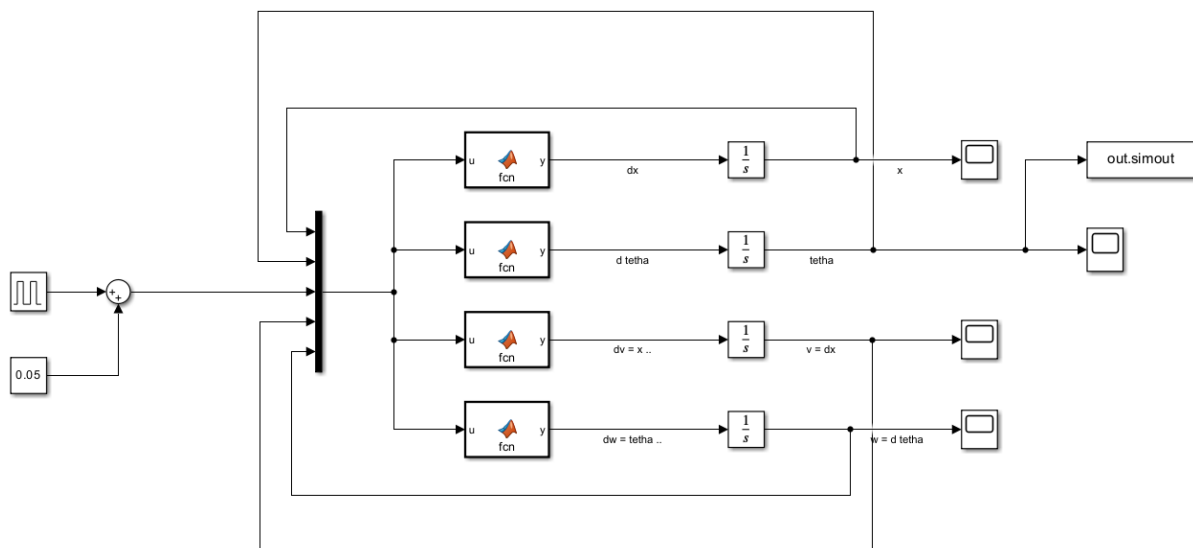


Figure 29- The nonlinear system with a specified pulse input

The output during the first 2.155 seconds is as follows. During this time, the system reaches approximately 6 degrees of deviation, and the pendulum's motion can be considered linear within this range with good approximation. It must be noted that the system should be normalized. Therefore, the output needs to be multiplied by 10.

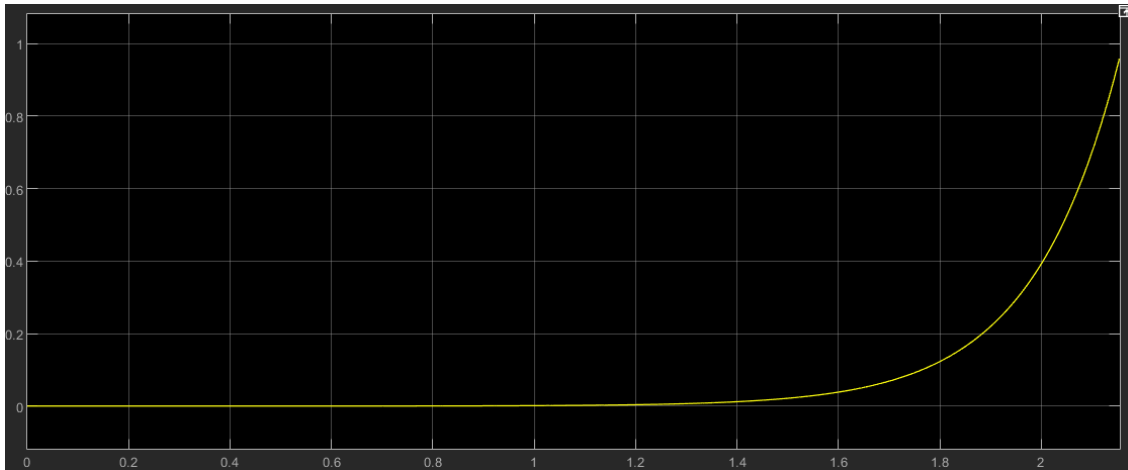
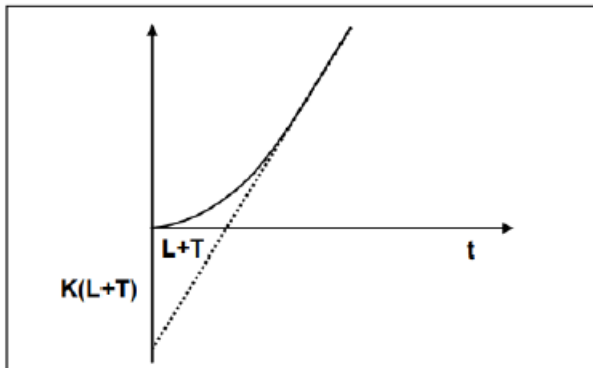


Figure 30- The nonlinear output with pulse input in method2

## 1-2-1. System Identification

### 1-2-1-1. Integral Identification

Since the system's behavior is integral in nature, integral identification is used for this system.



$$G(s) = \frac{Ke^{-sL}}{s(1+sT)}$$

Figure 31- Integral Identification

First, the system must be exported to the MATLAB environment using the "To Workspace" block, and then the necessary analysis is performed to determine the parameters.

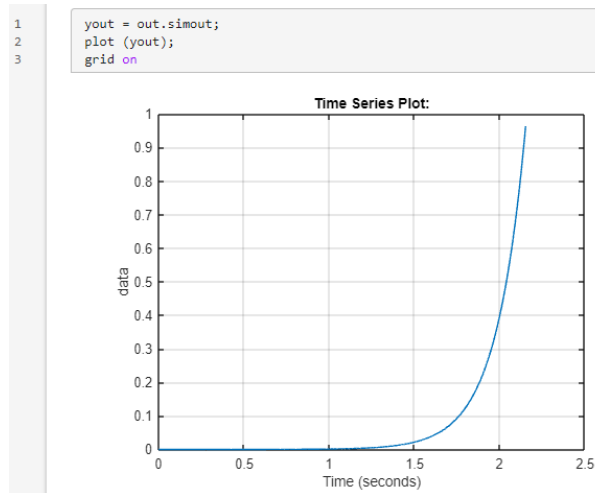


Figure 32- Plotting the output in MATLAB

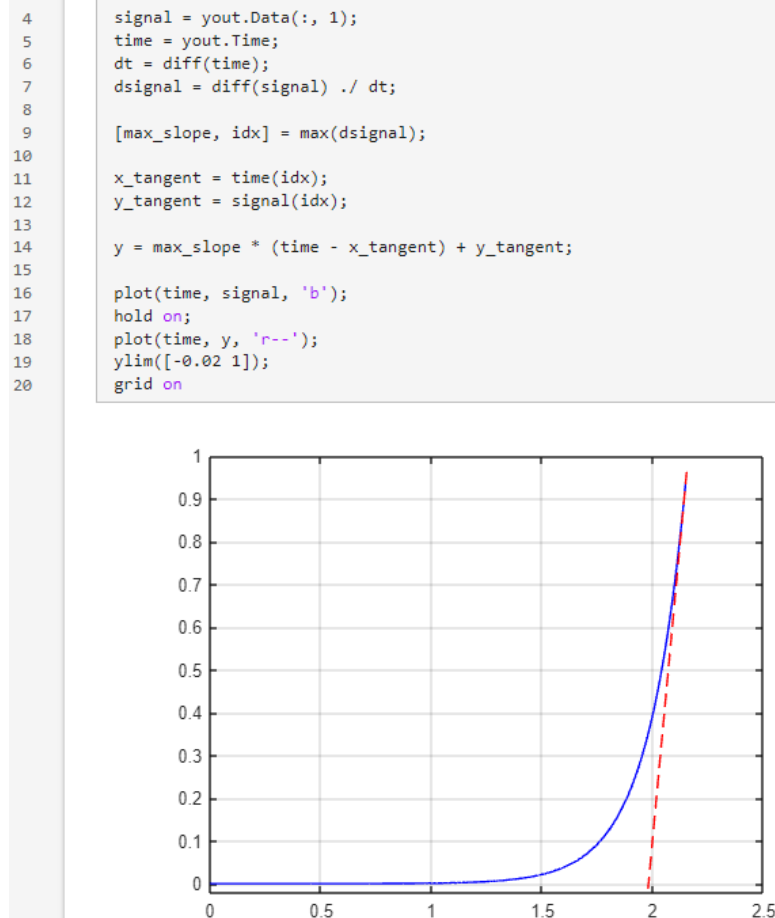


Figure 33- The tangent line to the nonlinear output

Now, according to the calculations we know for integral identification, the necessary parameters for identification are calculated in MATLAB as follows.

```

21 L_Plus_T = (-y_tangent / max_slope) + x_tangent
    L_Plus_T = 1.9823

22 K = (max_slope * (- x_tangent) + y_tangent) / L_Plus_T*(-1)
    K = 5.5792

23 signal_at_L_Plus_T = interp1(time, signal, L_Plus_T)
    signal_at_L_Plus_T = 0.3539

24 T = signal_at_L_Plus_T*exp(1)/K
    T = 0.1724

25 L = L_Plus_T - T
    L = 1.8098

```

Figure 34- Calculating integral identification parameters

Therefore, we have:

$$L + T = 1.9823$$

$$K = 5.5792$$

$$S(L + T) = 0.3539$$

$$T = \frac{S(L + T)e^1}{K} = 0.1724$$

$$L = 1.8098$$

$$G(s) = \frac{5.5792}{s(1 + 0.1724s)} e^{-1.8098s}$$

Equation 5- The transfer function and parameters of integral identification

The transfer function of this identification is entered into MATLAB.

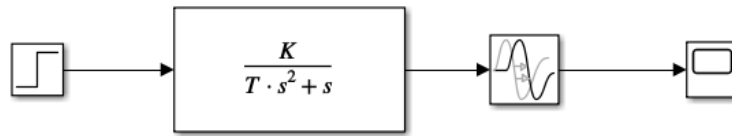


Figure 35- Applying TF in MATLAB

The identified output using the integral method and the main system output are as follows.

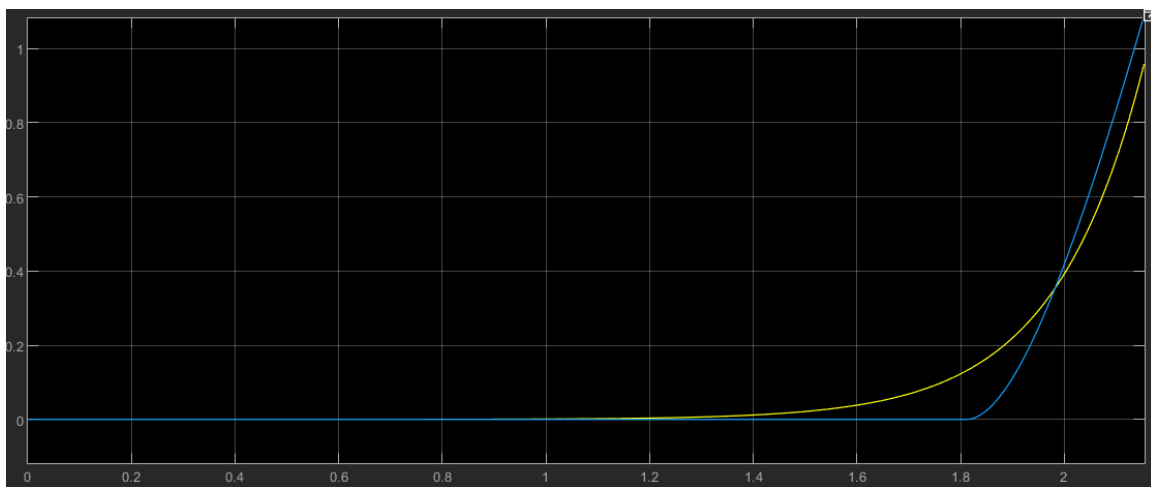


Figure 36- The output of the nonlinear system and the identified system

## 1-2-2. Designing PID Controller for this Method

Now, a PID controller must be designed for this system. First, the final point characteristics are extracted using relay feedback as follows.

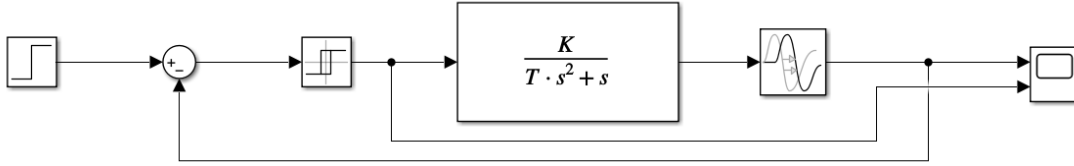


Figure 37- Applying relay feedback on transfer function in MATLAB

The output is as follows:

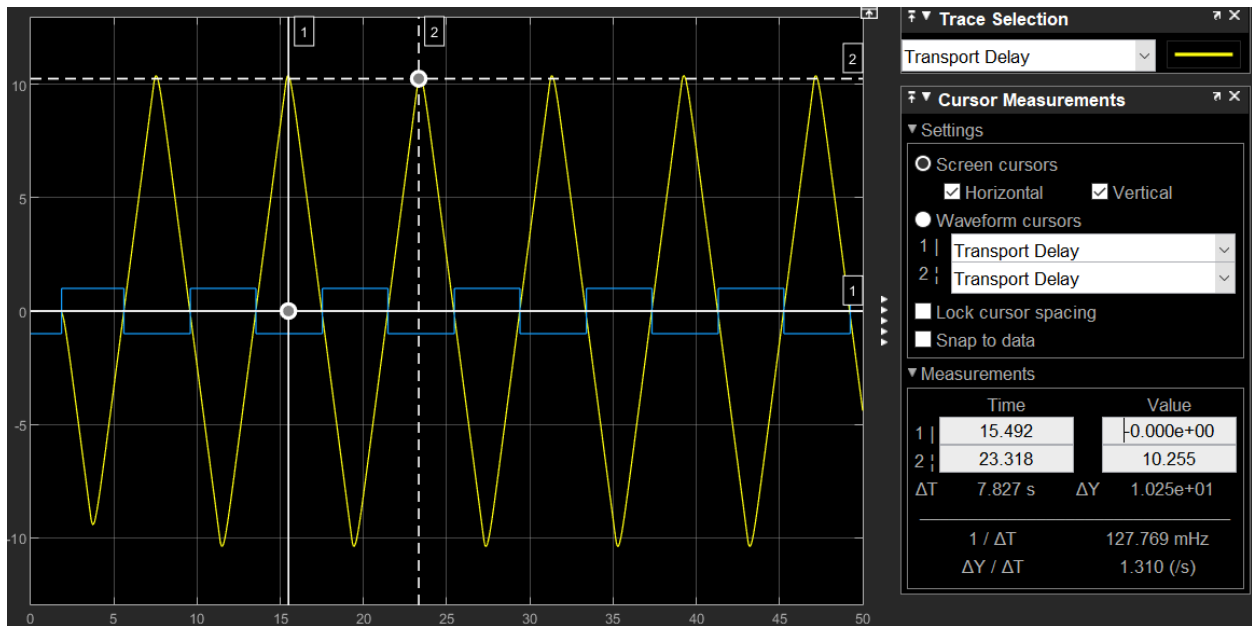


Figure 38- The relay feedback output

From the diagram, the following information is extracted:

$$a = 10.25$$

$$d = 1$$

$$G(jw) = -\frac{\pi a}{4 d} = -8.05$$

$$K_u = 0.124, T_u = 7.827$$

Equation 6- Ultimate point parameters

### 1-2-2-1. Ziegler-Nichols PID Tuning in the Time Domain

Previously we had:

$$L = 1.9823, a = 11.0594$$

Equation 7- Z.N parameters

We know that the Ziegler-Nichols table is as follows:

Tp	Td	Ti	K	کنترل کننده
4L	0	0	1/a	<b>P</b>
5.7L	0	3L	0.9/a	<b>PI</b>
3.4L	L/2	2L	1.2/a	<b>PID</b>

Figure 39- Ziegler-Nichols table in the time domain

Therefore:

$$K = 0.1085, T_i = 3.9646, T_d = 0.99115$$

Equation 8- Z.N PID coefficients

By applying the PID coefficients as follows and forming a closed loop for the identified system, the following results will be obtained:

▼ Compensator formula

$$P \left( 1 + I \frac{1}{s} + D \frac{N}{1 + N \frac{1}{s}} \right)$$

Main Initialization Saturation Data Types State Attributes

Controller parameters

Source: internal

Proportional (P): 0.1085

Integral (I): 1/3.9646 0.25223 ☐ Use I\*Ts (optimal for codegen)

Derivative (D): 0.99115

Filter coefficient (N): 80 ☒ Use filtered derivative

Figure 40- Z.N PID coefficients in MATLAB

Note that the initial angle of the system is applied as follows in the Transport Delay block.

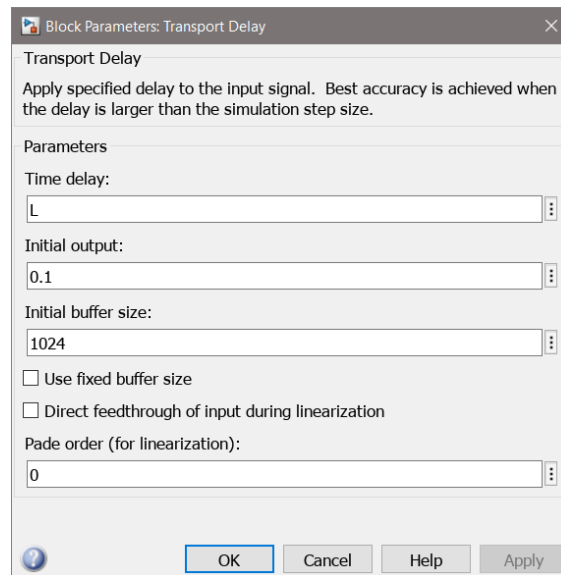


Figure 41- Applying initial conditions in the Transport Delay block.

The block diagram and the output are as follows:

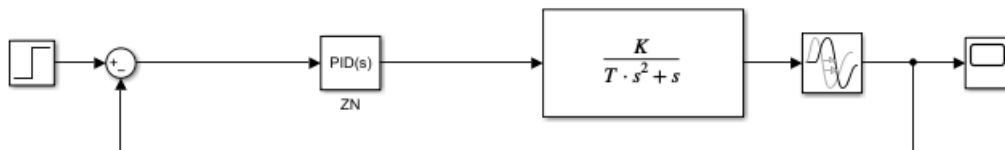


Figure 42- Closed loop system with Z.N PID in MATLAB for identified system

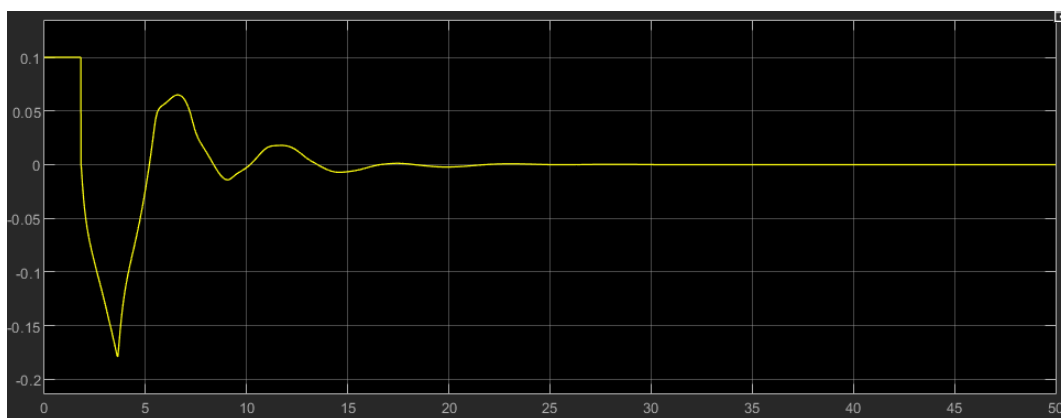


Figure 43- The output of closed loop system with Z.N PID in MATLAB for identified system



It can be observed that the designed PID successfully controls the identified system. The angle it controls is approximately 6 degrees, which corresponds to the range of the system's linear behavior.

Now, the designed PID controller must be applied to the original nonlinear system to determine whether an appropriate response is achieved.

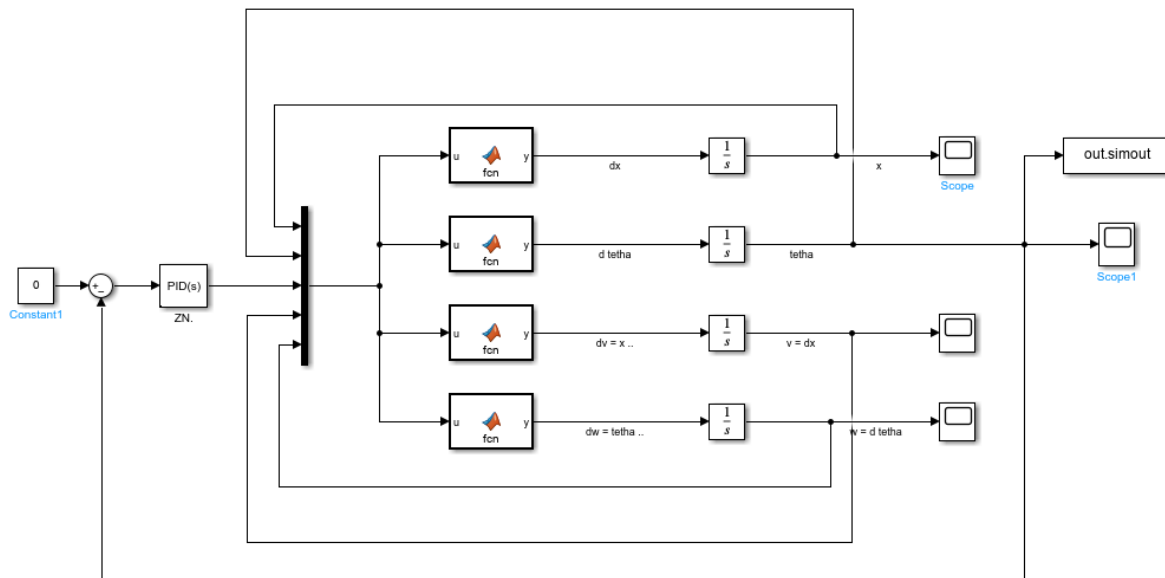


Figure 44- Applying the Z.N PID in MATLAB for the nonlinear system

The output is as follows:

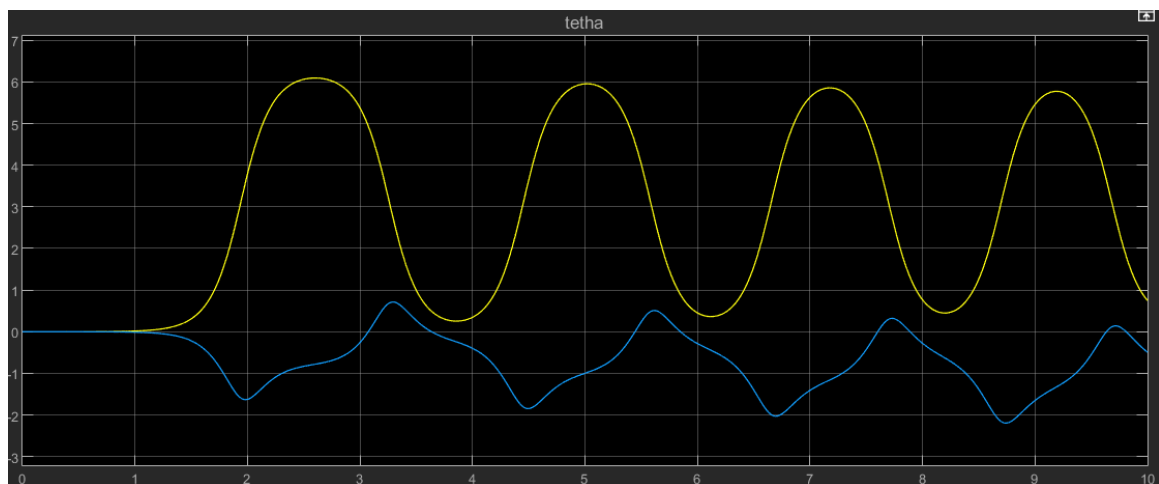


Figure 45- The output of the Z.N PID in MATLAB for the nonlinear system

It is clear that the control signal is weak, and proper control is not achieved. By applying a gain after the PID controller, we can achieve better control of the pendulum.

The output after applying a gain is as follows:

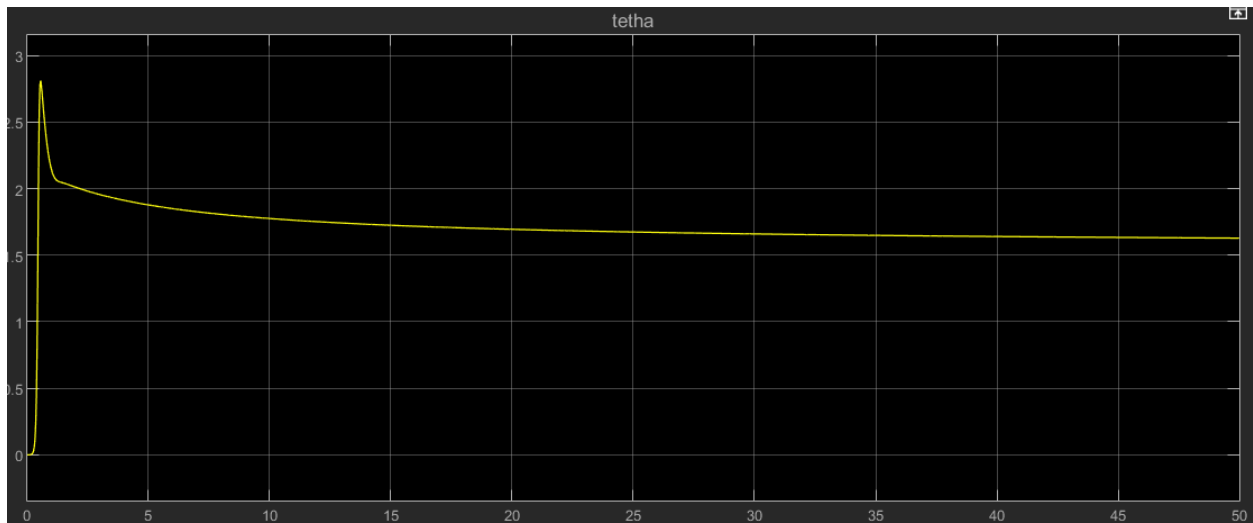


Figure 46- The output of the Z.N PID in MATLAB for the nonlinear system after applying a gain

It can be observed that the system performs better and does not become unstable. However, the angle still cannot be brought to zero. Note that the initial angle is 6 degrees.

### 1-2-2-2. Ziegler-Nichols PID Tuning in Frequency Domain

Based on Equation 6, which defines the ultimate point characteristics, and the Ziegler-Nichols frequency domain table shown below, the PID coefficients are designed.

$T_p$	$T_d$	$T_i$	K	کنترل کننده
$T_u$			$0.5K_u$	<b>P</b>
$1.4T_u$		$0.8T_u$	$0.4K_u$	<b>PI</b>
$0.85T_u$	$0.125T_u$	$0.5T_u$	$0.6K_u$	<b>PID</b>

Figure 47- Ziegler-Nichols table in frequency domain

Therefore:

$$K = 0.0744, T_i = 3.9135, T_d = 0.9784$$

Equation 9- Z.N PID coefficients in frequency domain

By applying the PID coefficients as follows and forming a closed loop for the identified system, the following results will be obtained:

The screenshot shows the MATLAB PID Tuner interface with the following settings:

- Controller parameters:**
  - Source: internal
  - Proportional (P): 0.0744
  - Integral (I): 1/3.9135 (with a small value of 0.25553 displayed next to it)
  - Derivative (D): 0.9784
  - Filter coefficient (N): 100
- Options:**
  - ☐ Use I\*Ts (optimal for codegen)
  - ☒ Use filtered derivative

Figure 48- Z.N PID coefficients in frequency domain in MATLAB

The block diagram and the output are as follows:

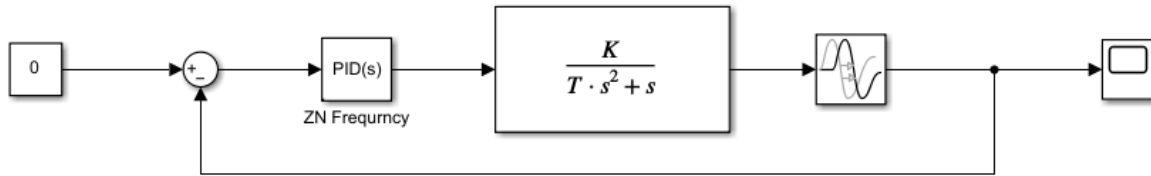


Figure 49- Closed loop system with Z.N PID in frequency domain in MATLAB for identified system

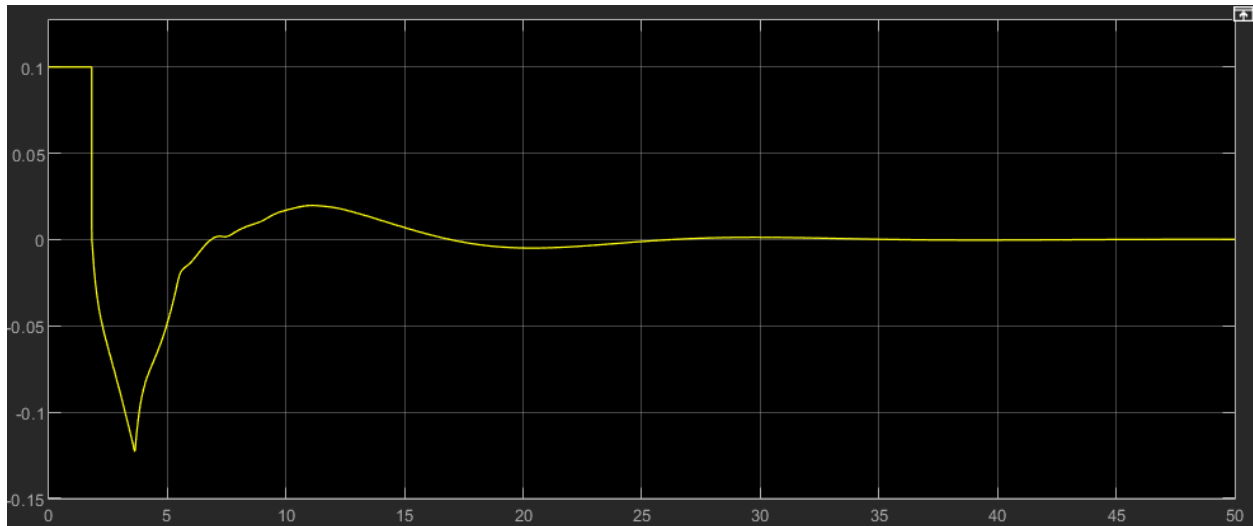


Figure 50- The output of closed loop system with Z.N PID in frequency domain in MATLAB for identified system

It can be observed that the designed PID successfully controls the identified system. The angle it controls is approximately 6 degrees, which corresponds to the range of the system's linear behavior.

Now, the designed PID controller must be applied to the original nonlinear system to determine whether an appropriate response is achieved.

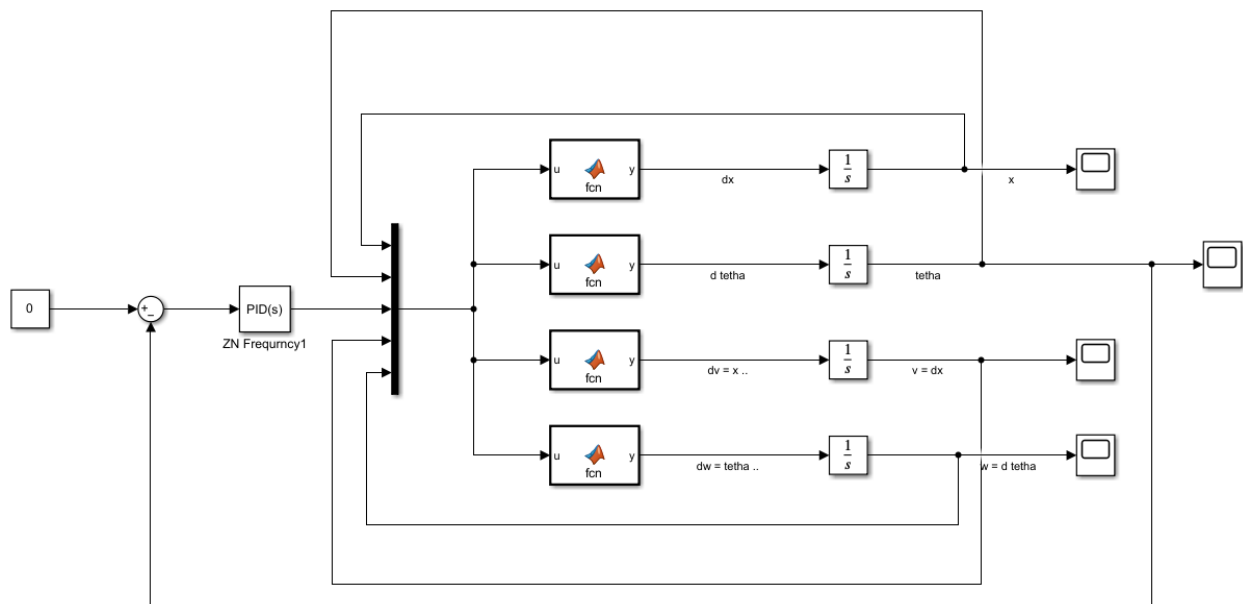


Figure 51- Applying the Z.N PID in frequency domain in MATLAB for the nonlinear system

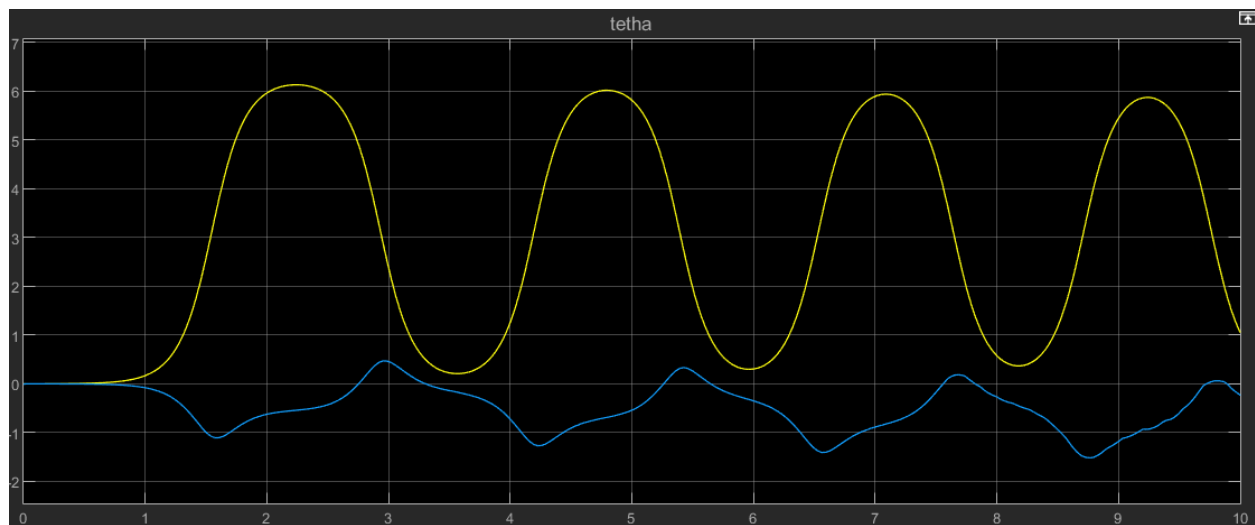


Figure 52- The output of the Z.N PID in frequency domain in MATLAB for the nonlinear system

It is clear that the control signal is weak, and proper control is not achieved. By applying a gain after the PID controller, we can achieve better control of the pendulum.

The output after applying a gain is as follows:

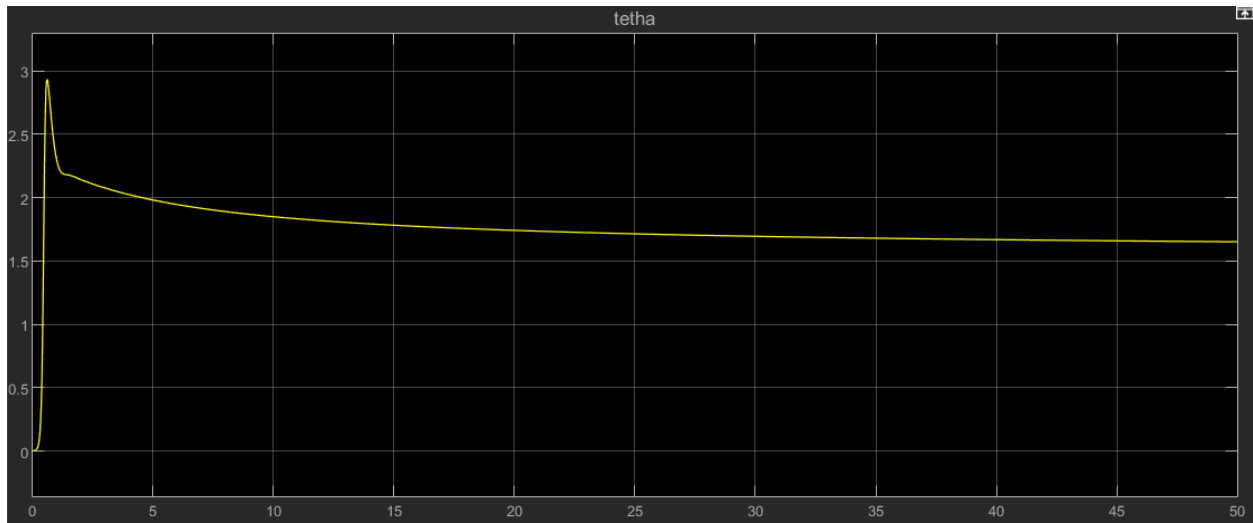


Figure 53- The output of the Z.N PID in frequency domain in MATLAB for the nonlinear system after applying a gain

It can be observed that the system performs better and does not become unstable. However, the angle still cannot be brought to zero. Note that the initial angle is 6 degrees.

### 1-2-2-3. Generalized Ziegler-Nichols PID Tuning

If in the generalized Ziegler-Nichols method, the initial point and final point are considered, the equations to reach point  $B$  will be as follows:

$$\begin{aligned} K &= K_u r_b \cos \varphi_b \\ T_i &= \frac{T_u}{\pi} \left( \frac{1 + \sin \varphi_b}{\cos \varphi_b} \right) \\ T_d &= \frac{\alpha T_u}{\pi} \left( \frac{1 + \sin \varphi_b}{\cos \varphi_b} \right) \end{aligned}$$

Equation 10- Tuning the coefficients using the generalized Ziegler-Nichols method

It should be noted that  $\alpha = 0.25$  is considered.

We want the system to be lead-compensated and improve the controllability of the system. In fact, we aim to eliminate the error from the previous methods and try to reduce the error to 0.

To achieve this,  $\xi$  must be increased, and the angle at point  $B$  must be made larger.

Therefore, the specifications of point  $B$  are considered as follows:

$$r_b = 0.41, \varphi_b = 61^\circ$$

Figure 54- Specifications of point B

By substituting Figure 54 into Equation 10, we will have:

$$\begin{aligned} K &= 0.198K_u \\ T_i &= 1.23T_u \\ T_d &= 0.307T_u \end{aligned}$$

Equation 11- The PID coefficients at point B

In Equation 6, the ultimate point values are calculated. By substituting these values into Equation 11, we will have:

$$K = 0.025, T_i = 9.627, T_d = 2.403$$

Equation 12- Generalized Z.N PID coefficients

By applying the PID coefficients as follows and forming a closed loop for the identified system, the following results will be obtained:

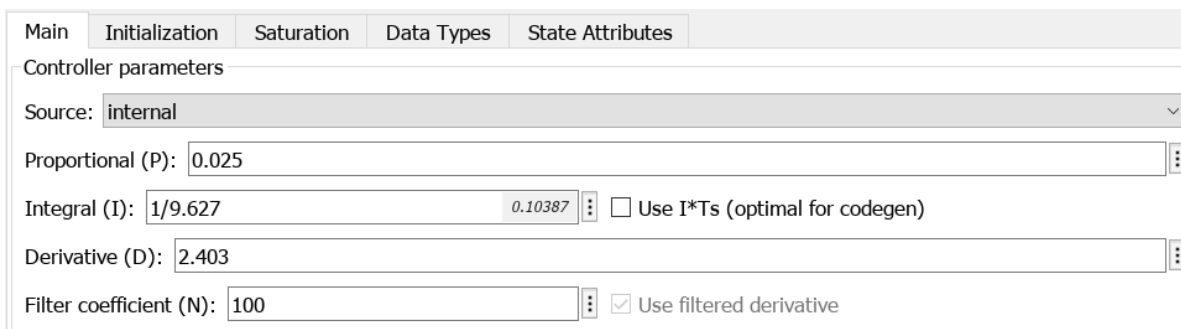


Figure 55- Generalized Z.N PID coefficients in MATLAB

The block diagram and the output are as follows:

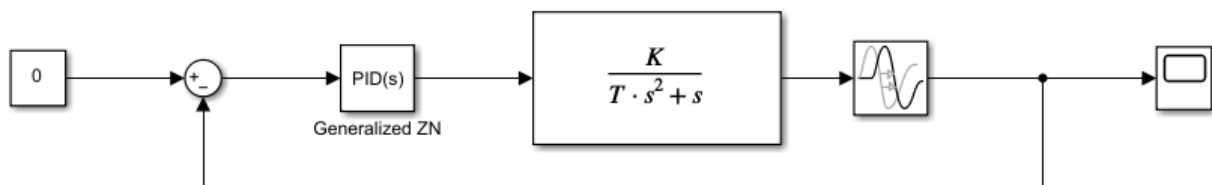


Figure 56- Closed loop system with Generalized Z.N PID in MATLAB for identified system



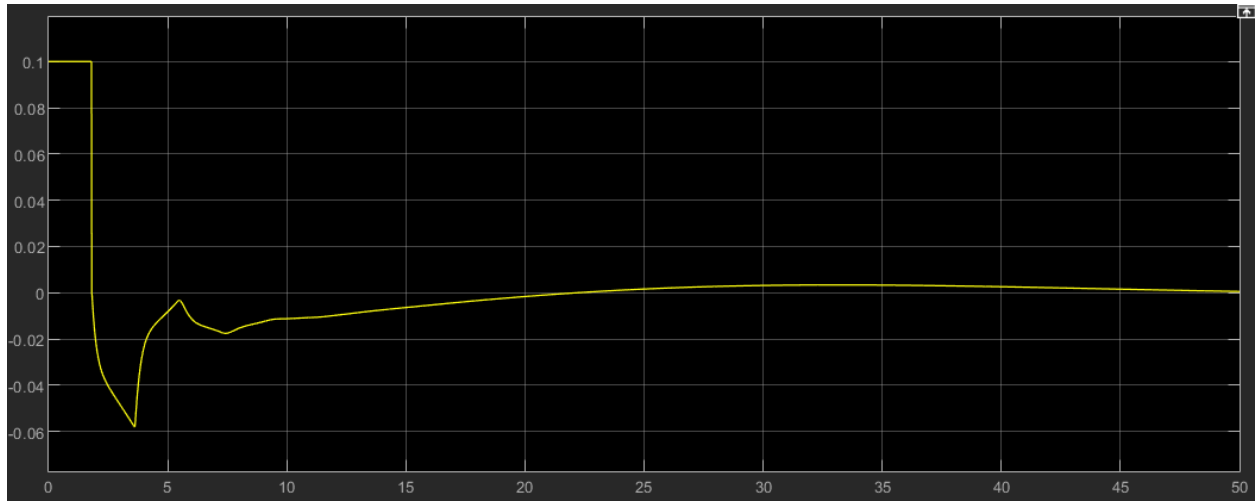


Figure 57- The output of closed loop system with Generalized Z.N PID in MATLAB for identified system

It can be observed that compared to the previous two methods, the overshoot has significantly decreased, but the system takes longer to reach 0.

As a result, it can be said that this point does not provide suitable coefficients for us. Therefore, we repeat the steps above for the following point and consider the angle at point *B* to be less than 25 degrees, which is the angle corresponding to the Ziegler-Nichols method.

$$r_b = 0.5, \varphi_b = 20^\circ$$

Figure 58- New specifications of point B

By following the previous steps, the following PID coefficients can be obtained for the new point *B*:

$$K = 0.058, T_i = 3.558, T_d = 0.889$$

Equation 13- New generalized Z.N PID coefficients

By applying the above PID coefficients, the output will be as follows:

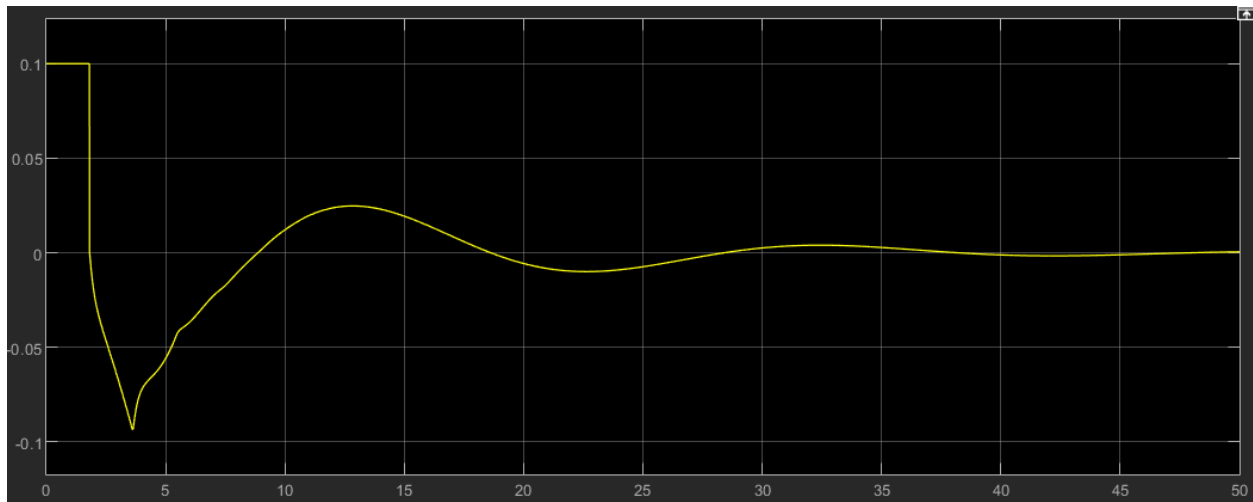


Figure 59- The output of closed loop system with new Generalized Z.N PID in MATLAB for identified system

Now, the designed PID controller must be applied to the original nonlinear system to determine whether an appropriate response is achieved.

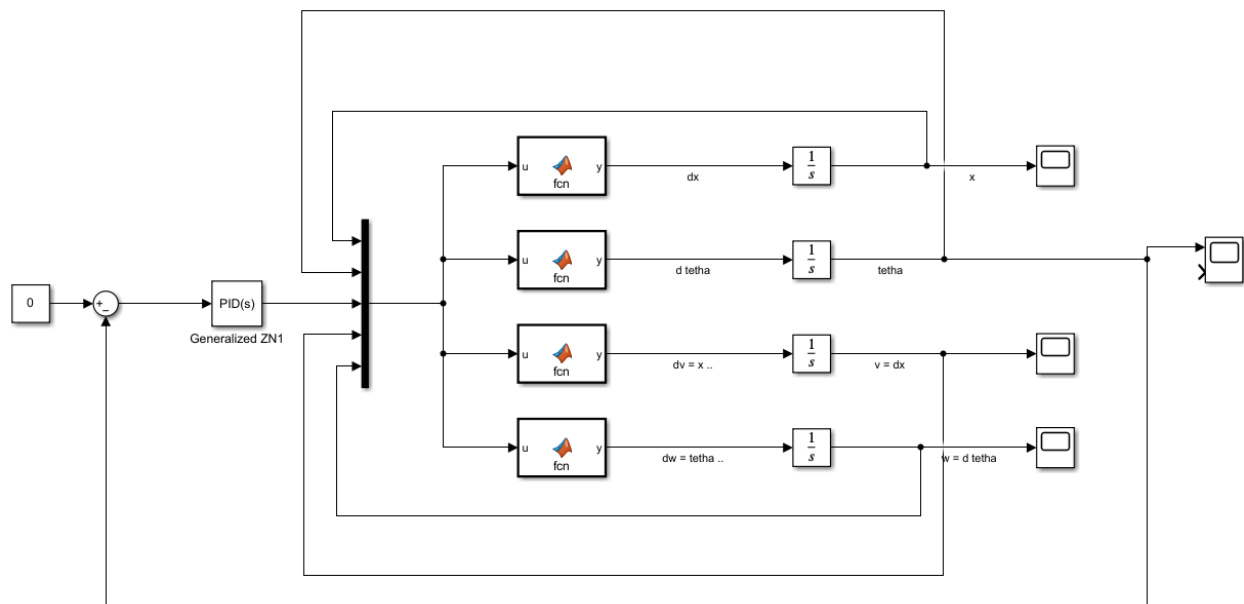


Figure 60- Applying the generalized Z.N PID in MATLAB for the nonlinear system

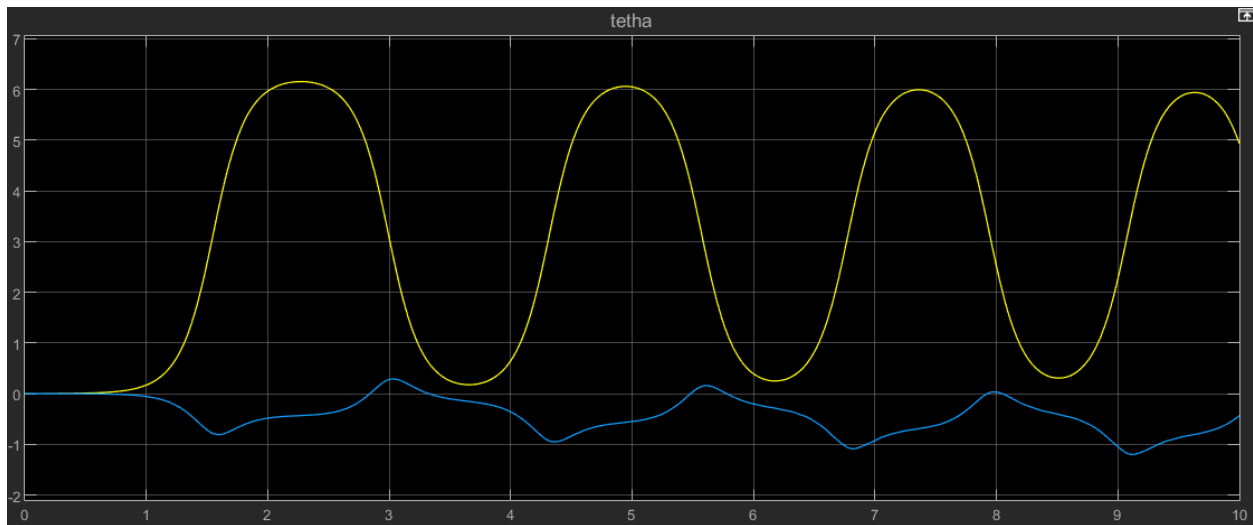


Figure 61- The output of the generalized Z.N PID in MATLAB for the nonlinear system

Once again it is clear that the control signal is weak, and proper control is not achieved. By applying a gain after the PID controller, we can achieve better control of the pendulum.

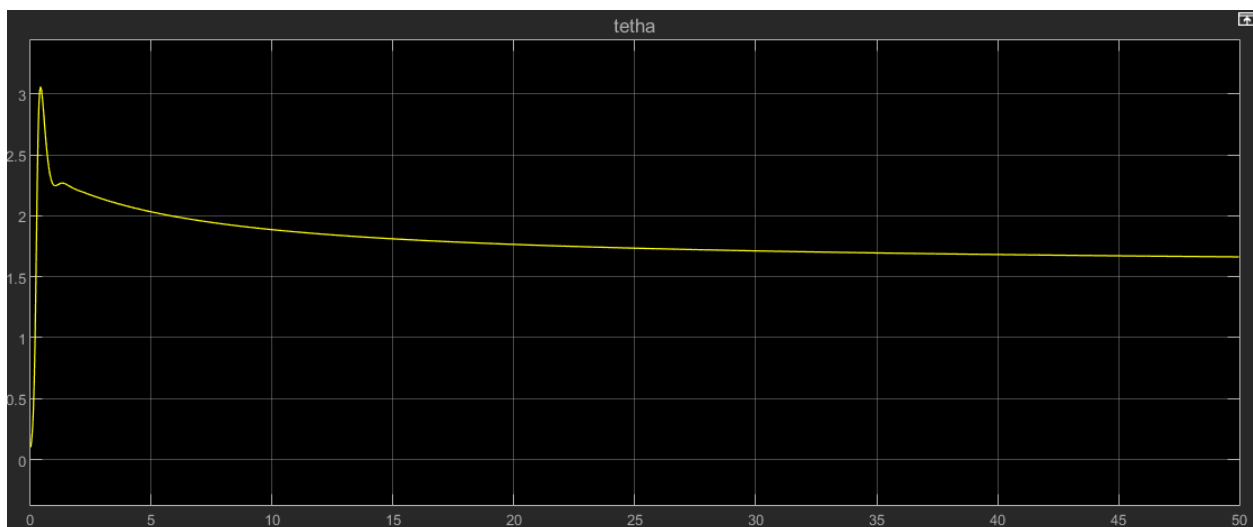


Figure 62- The output of the generalized Z.N PID in MATLAB for the nonlinear system after applying a gain

It can be observed that the system performs better and does not become unstable. However, the angle still cannot be brought to zero. Note that the initial angle is 6 degrees.

#### 1-2-2-4. PID Design Using the Lambda Tuning Method

Since this method is based on a three-parameter model and our identified system is an integral system, it is not possible to apply this PID design method.

#### 1-2-2-5. PID Design Using the CHR Method

Previously, the parameters  $a$  and  $L$  for this system were determined. Now, using these parameters and the table below, which pertains to calculating PID coefficients via the CHR method, the PID coefficients are computed.

K	$T_i$	$T_d$
$a/0.7$	-	-
$a/0.7$	$2.3L$	-
$a/1.2$	$2.0L$	$0.42L$

Figure 63- CHR table

Note that the term  $K$  should be  $1.2/a$ .

The PID coefficients are as follows:

$$K = 0.1085, T_i = 3.9646, T_d = 0.8326$$

Equation 14- CHR PID coefficients

By applying the PID coefficients as follows and forming a closed loop for the identified system, the following results will be obtained:

Main	Initialization	Saturation	Data Types	State Attributes
Controller parameters				
Source: internal				
Proportional (P): 0.1085				
Integral (I): 1/3.9646 0.25223 <input type="checkbox"/> Use I*Ts (optimal for codegen)				
Derivative (D): 0.8326				
Filter coefficient (N): 100 <input checked="" type="checkbox"/> Use filtered derivative				

Figure 64- CHR PID coefficients in MATLAB

The block diagram and the output are as follows:

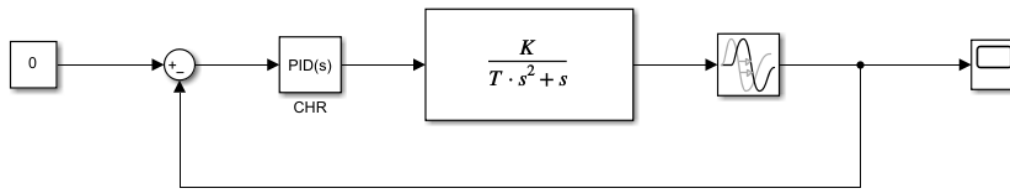


Figure 65- Closed loop system with CHR PID in MATLAB for identified system

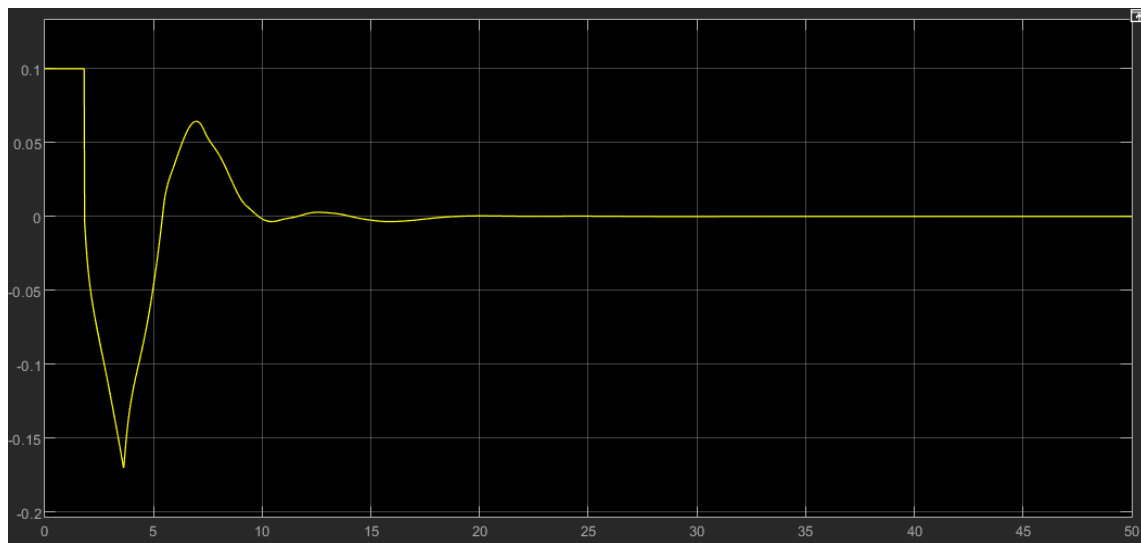


Figure 66- The output of closed loop system with CHR PID in MATLAB for identified system

Now, the designed PID controller must be applied to the original nonlinear system to determine whether an appropriate response is achieved.

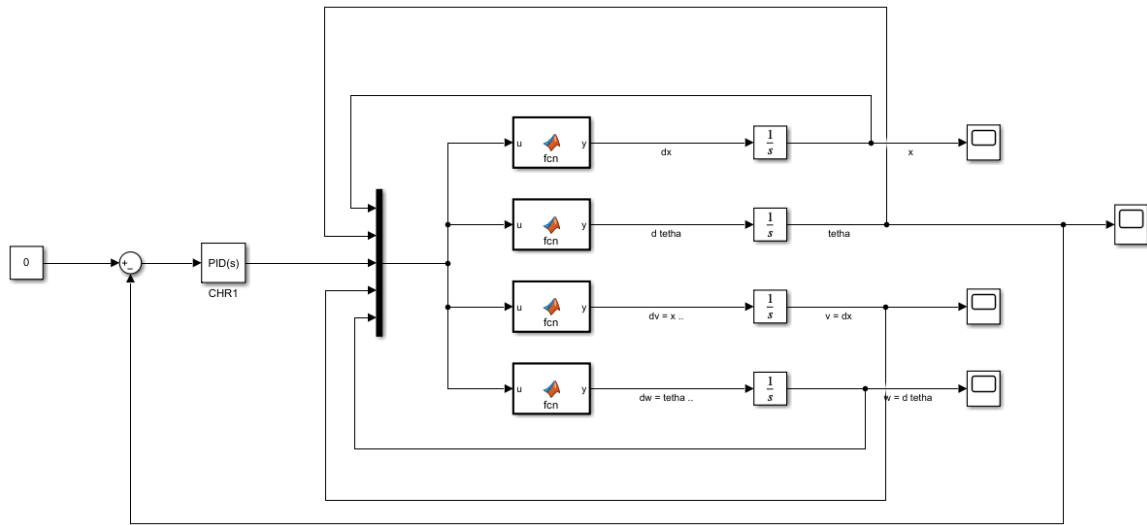


Figure 67- Applying the CHR PID in MATLAB for the nonlinear system

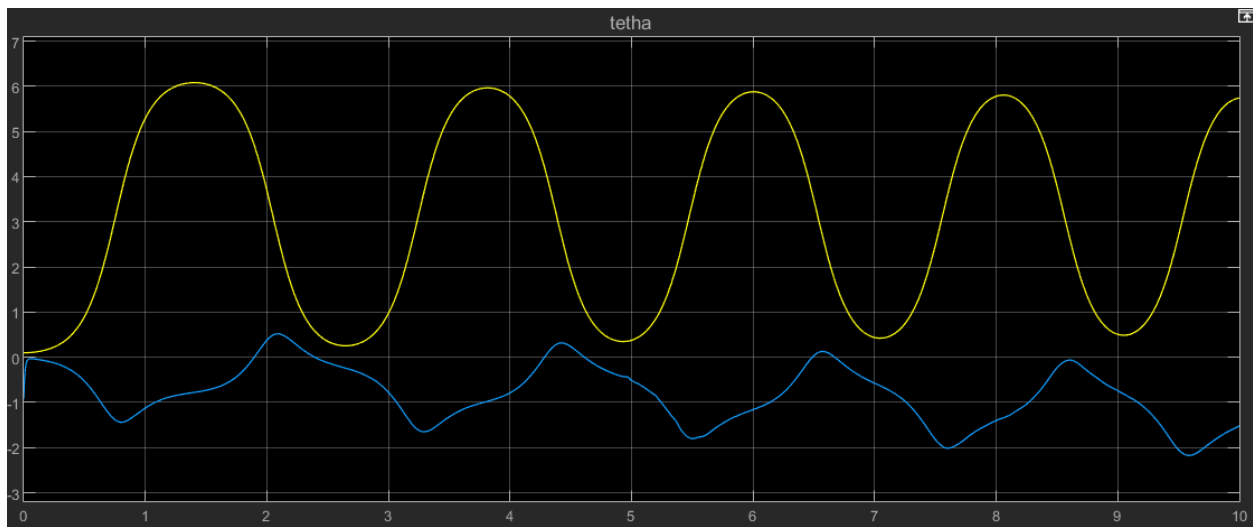


Figure 68- The output of the CHR PID in MATLAB for the nonlinear system

Once again it is clear that the control signal is weak, and proper control is not achieved. By applying a gain after the PID controller, we can achieve better control of the pendulum.

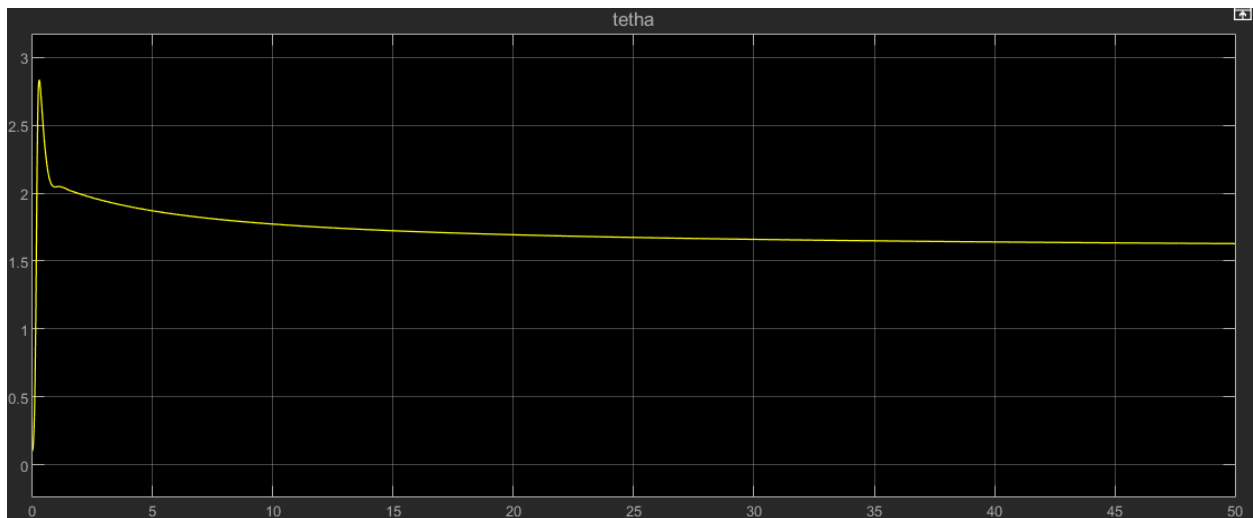


Figure 69- The output of the CHR PID in MATLAB for the nonlinear system after applying a gain

### 1-3. Method3

The best method for identifying a nonlinear system is to apply a pulse input to the system in such a way that the pendulum oscillates around its upper equilibrium point without falling. This approach ensures that the system exhibits the most linear behavior, resulting in the most reliable identification.

In this project, I spent hours calculating and searching for an input that could satisfy these conditions. However, due to the system becoming unstable with even the slightest applied force, I was unable to achieve the desired outcome. Nevertheless, I felt it important to highlight what the optimal approach for obtaining the response would be.

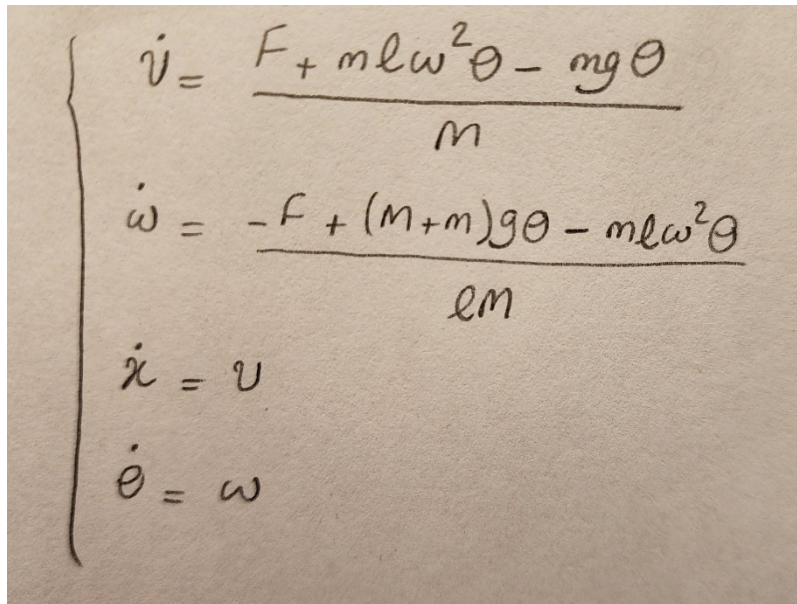
## 2. Linear Simulation of the Inverted Pendulum

In Figure 3, the state-space equations of the inverted pendulum can be observed. To linearize these equations around the equilibrium point, where the angle is zero degrees, the following approximations can be used:

$$\sin \theta = \theta, \cos \theta = 1$$

Equation 15- Approximations for linear system

Thus, the state-space matrices will be as follows:



The image shows a handwritten set of state-space equations for an inverted pendulum system, enclosed in a large left-facing curly bracket. The equations are:

$$\begin{cases} \dot{v} = \frac{F + ml\omega^2\theta - mg\theta}{m} \\ \dot{\omega} = \frac{-F + (M+m)g\theta - ml\omega^2\theta}{lM} \\ \dot{x} = v \\ \dot{\theta} = \omega \end{cases}$$

Figure 70- State space form of linear system



Therefore, the functions are defined based on the equations above.

Note that, considering the inputs of the multiplexer, the following relations are taken into account:

$$\begin{aligned} u(1) &= x \\ u(2) &= \theta \\ u(3) &= F \\ u(4) &= \frac{dx}{dt} = v \\ u(5) &= \frac{d\theta}{dt} = \omega \end{aligned}$$

Equation 16- Defining inputs

The figure below illustrates the overall structure of the inverted pendulum simulation in the MATLAB environment:

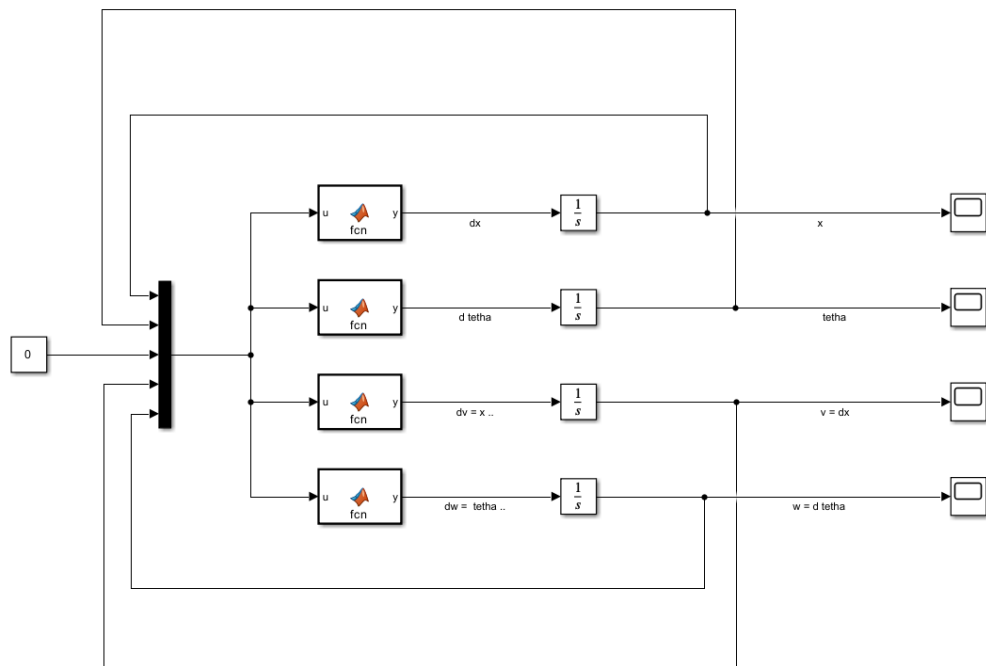


Figure 71- Overall structure

Based on these relations, understanding the equations implemented in the function blocks becomes easier.

First Block:

```

1 function y = fcn(u)
2     M = 10;
3     m = 0.3;
4     l = 0.3;
5     g = 9.8;
6     y = u(4);

```

Figure 72- First block's function

Second Block:

```

1 function y = fcn(u)
2     M = 10;
3     m = 0.3;
4     l = 0.3;
5     g = 9.8;
6     y = u(5);

```

Figure 73- Second block's function

Third Block:

```

1 function y = fcn(u)
2     M = 10;
3     m = 0.3;
4     l = 0.3;
5     g = 9.8;
6     y = ( ((-m*g)/M)*u(2) + (1/M)*u(3) );

```

Figure 74- Third block's function

Fourth Block:

```

1 function y = fcn(u)
2     M = 10;
3     m = 0.3;
4     l = 0.3;
5     g = 9.8;
6     y = ( (((M+m)*g)/(l*M))*u(2) + (-1/(l*M))*u(3) );

```

Figure 75- Fourth block's function

Thus, the output is as follows:

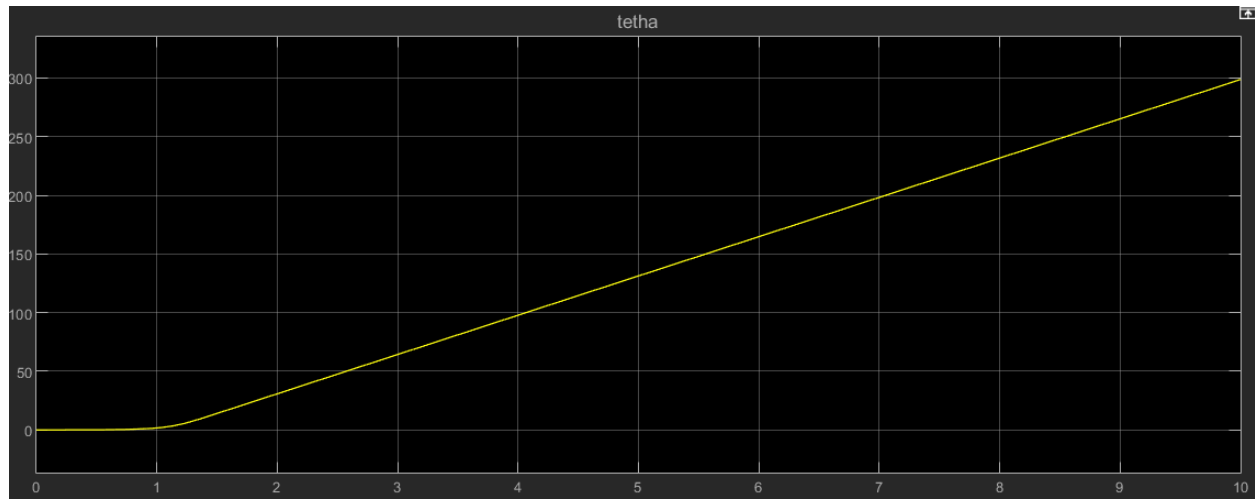


Figure 76- The angle output of linear system with a step input

## 2-1. System Identification

It is evident that the system exhibits integral behavior, and for identification, an integral model can be used, the details of which are shown in Figure 31.

The steps for identification using the integral method have been detailed in Method 2. Here as well, we first transfer the graph to MATLAB using the To Workspace block and then draw the tangent line that touches the graph at its maximum slope. This allows us to determine the parameters required for integral identification.

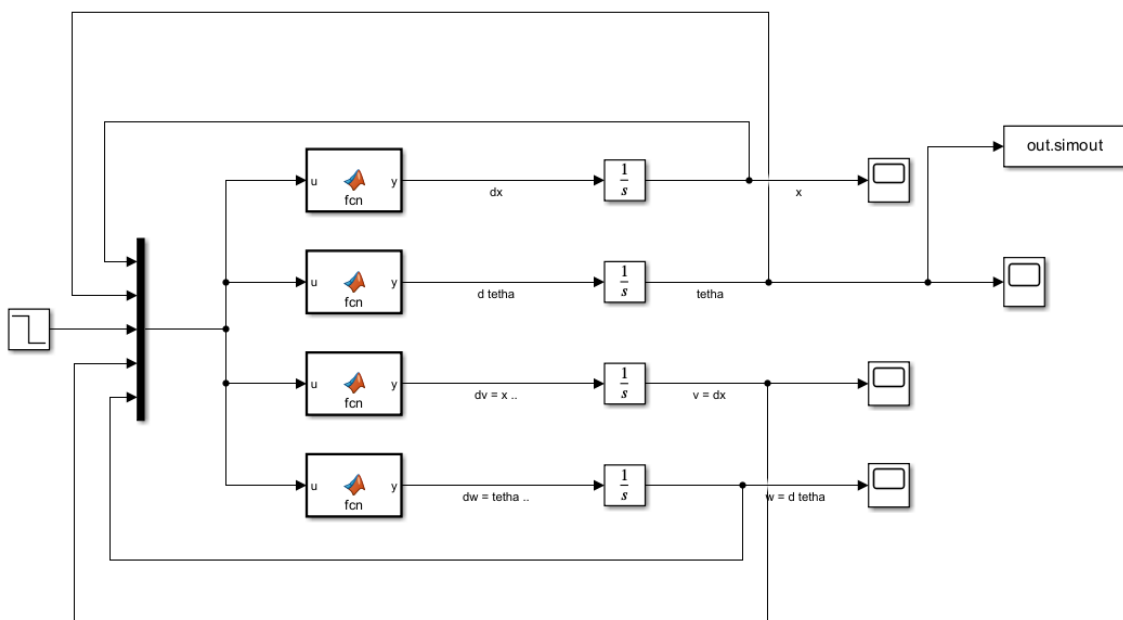


Figure 77- Applying To Workspace block to the linear system

The output is obtained in MATLAB and the tangent line is drawn:

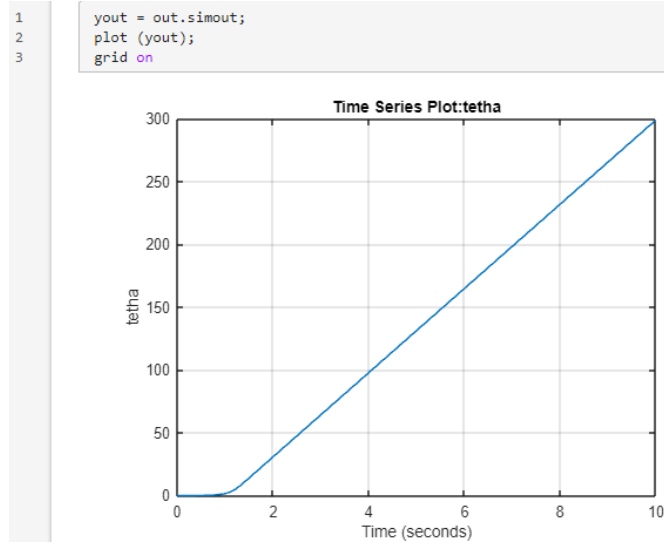


Figure 78- Obtained output in MATLAB

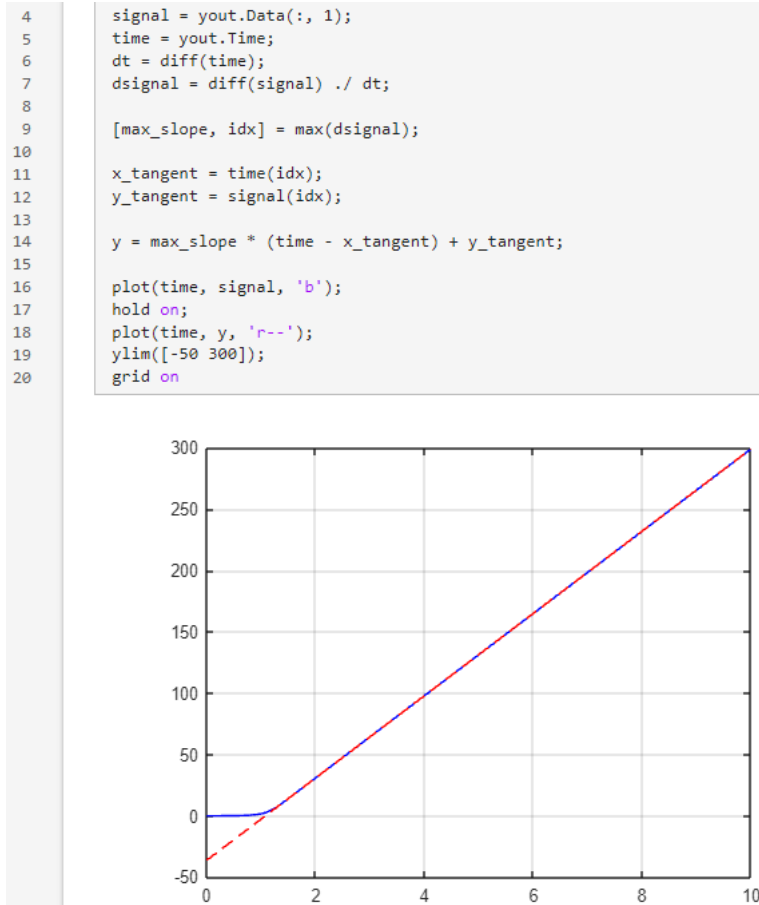


Figure 79- The tangent line

Now, according to the calculations we know for integral identification, the necessary parameters for identification are calculated in MATLAB as follows.

```

21      L_Plus_T = (-y_tangent / max_slope) + x_tangent
      L_Plus_T = 1.0813

22      K = (max_slope * (- x_tangent) + y_tangent) / L_Plus_T*(-1)
      K = 33.4990

23      signal_at_L_Plus_T = interp1(time, signal, L_Plus_T)
      signal_at_L_Plus_T = 2.6216

24      T = signal_at_L_Plus_T*exp(1)/K
      T = 0.2127

25      L = L_Plus_T - T
      L = 0.8686

```

Figure 80- Calculating integral identification parameters for linear system

Therefore, we have:

$$L + T = 1.0813$$

$$K = 33.499$$

$$S(L + T) = 2.6216$$

$$T = \frac{S(L + T)e^1}{K} = 0.2127$$

$$L = 0.8686$$

$$G(s) = \frac{33.499}{s(1 + 0.2127s)} e^{-0.8686s}$$

Equation 17- The transfer function and parameters of integral identification for linear system

The transfer function of this identification is entered into MATLAB.

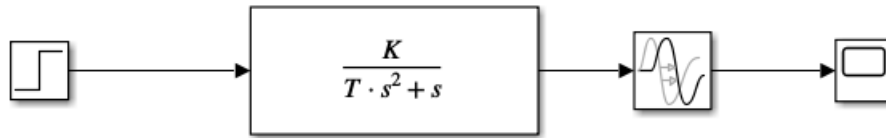


Figure 81- Applying TF in MATLAB

The identified output using the integral method and the main system output are as follows.

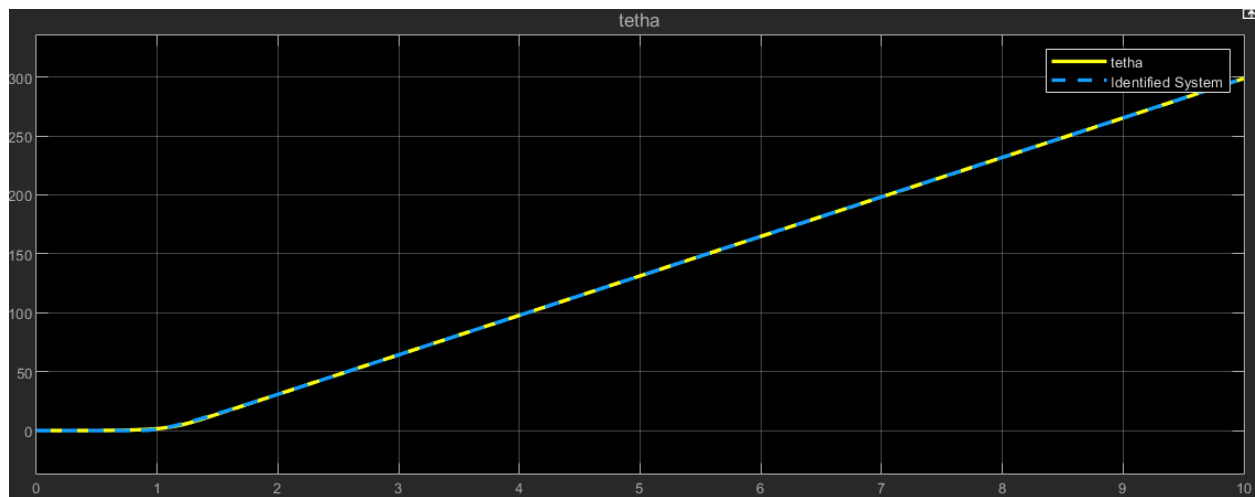


Figure 82- The output of the linear system and the identified system

The identified system has successfully modeled the original linear system.

## 2-2. Designing PID Controller for Linear System

Now, a PID controller must be designed for this system. First, the final point characteristics are extracted using relay feedback as follows.

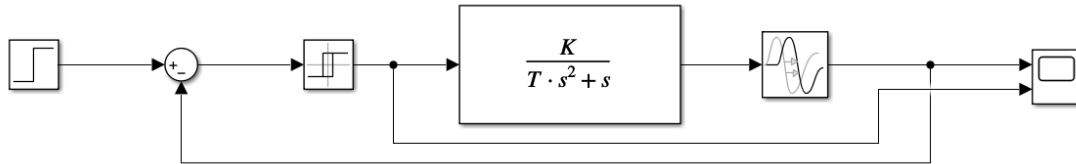


Figure 83- Applying relay feedback on transfer function in MATLAB for the linear system

The output is as follows:

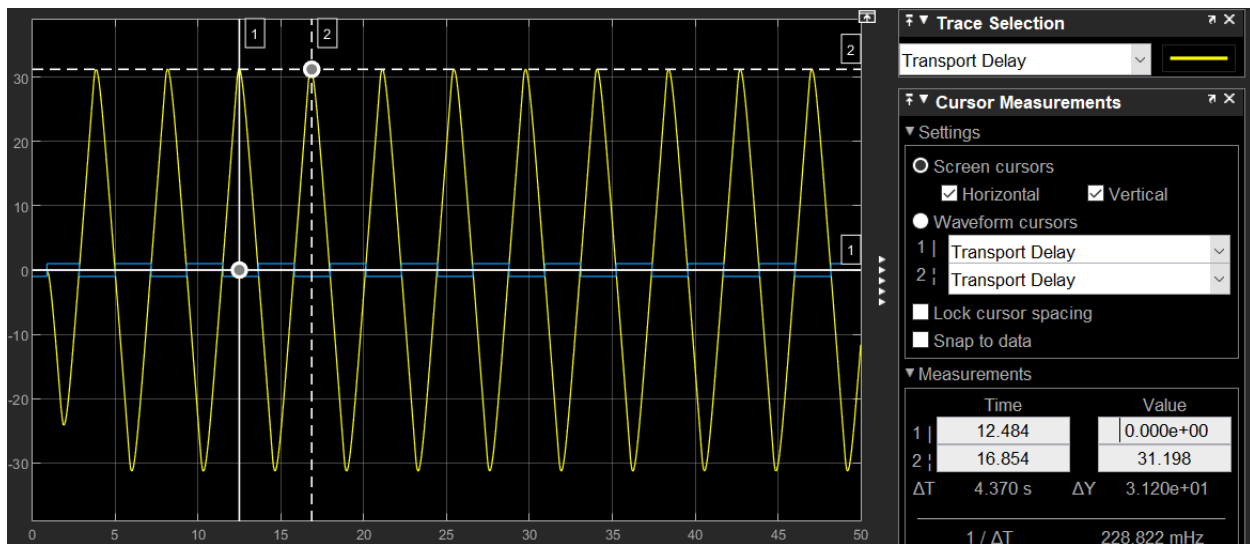


Figure 84- The relay feedback output for the linear system



From the diagram, the following information is extracted:

$$a = 31.2$$

$$d = 1$$

$$G(jw) = -\frac{\pi a}{4d} = -24.5$$

$$K_u = 0.041, Tu = 4.37$$

Equation 18- Ultimate point parameters for identified linear system

### 2-2-1. Ziegler-Nichols PID Tuning in the Time Domain

Previously we had:

$$L = 0.8686, a = 36.2594$$

Equation 19- Z.N parameters for the linear system

We know that the Ziegler-Nichols table is as follows:

Tp	Td	Ti	K	کنترل کننده
4L	0	0	1/a	<b>P</b>
5.7L	0	3L	0.9/a	<b>PI</b>
3.4L	L/2	2L	1.2/a	<b>PID</b>

Figure 85- Ziegler-Nichols table in the time domain for the linear system

Therefore:

$$K = 0.033, T_i = 1.7372, T_d = 0.4343$$

Equation 20- Z.N PID coefficients for the linear system

By applying the PID coefficients as follows and forming a closed loop for the identified system, the following results will be obtained:

Main	Initialization	Saturation	Data Types	State Attributes
Controller parameters				
Source: internal				
Proportional (P): 0.033				
Integral (I): 1/1.7372 0.57564 <input type="checkbox"/> Use I*Ts (optimal for codegen)				
Derivative (D): 0.4343				
Filter coefficient (N): 100 <input checked="" type="checkbox"/> Use filtered derivative				

Figure 86- Z.N PID coefficients in MATLAB for the identified linear system

The block diagram and the output are as follows:

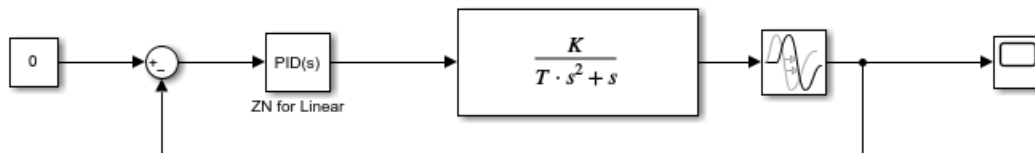


Figure 87- Closed loop system with Z.N PID in MATLAB for identified system

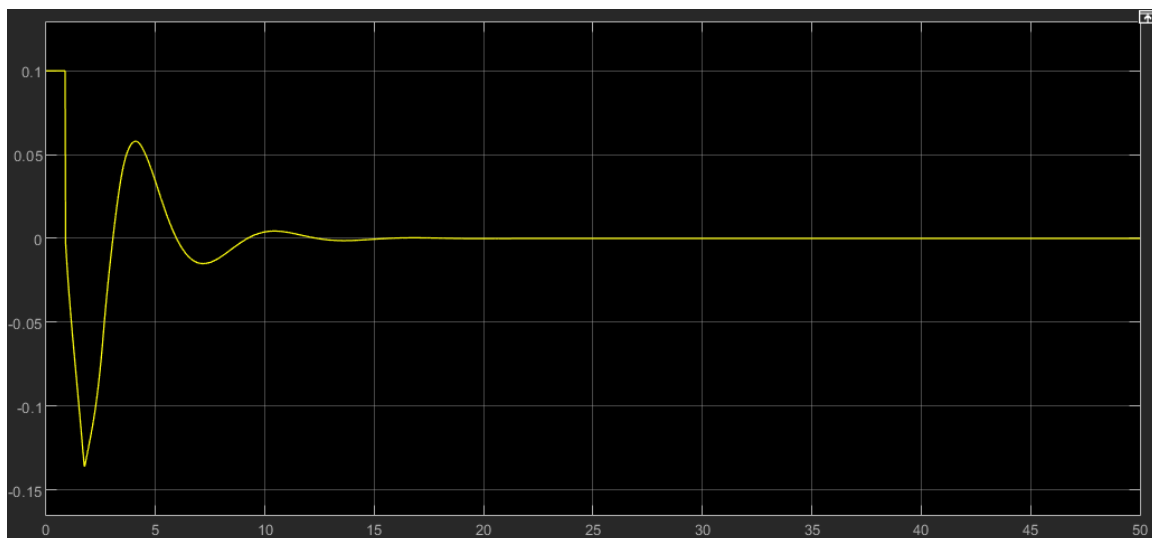


Figure 88- The output of closed loop system with Z.N PID in MATLAB for identified system

It can be observed that the designed PID successfully controls the identified system. The angle it controls is approximately 6 degrees, which corresponds to the range of the system's linear behavior.

Now, the designed PID controller must be applied to the linear system to determine whether an appropriate response is achieved.

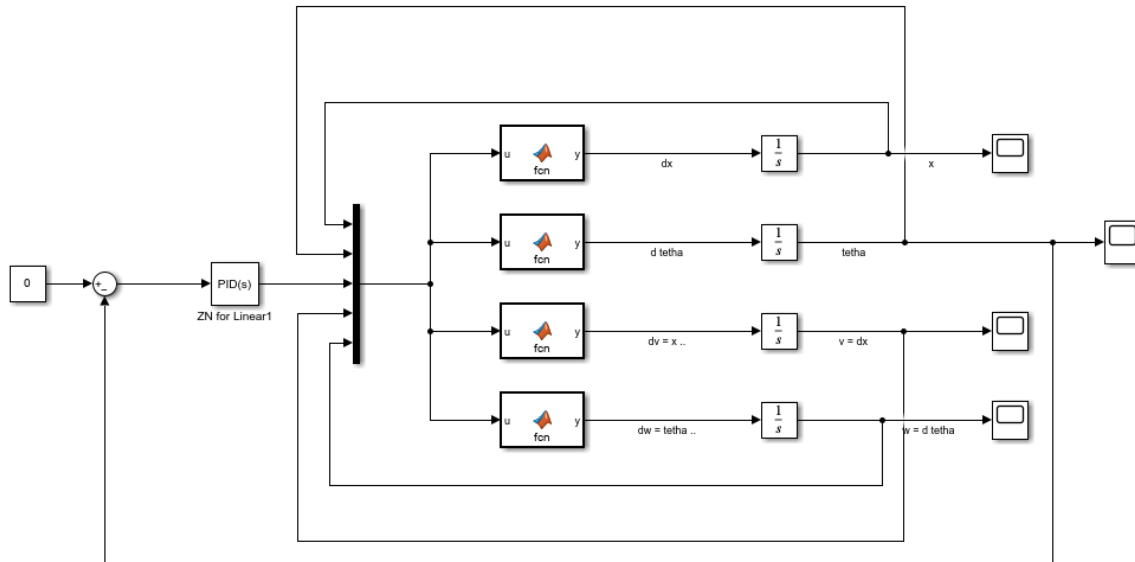


Figure 89- Applying the Z.N PID in MATLAB for the linear system

The output is as follows:

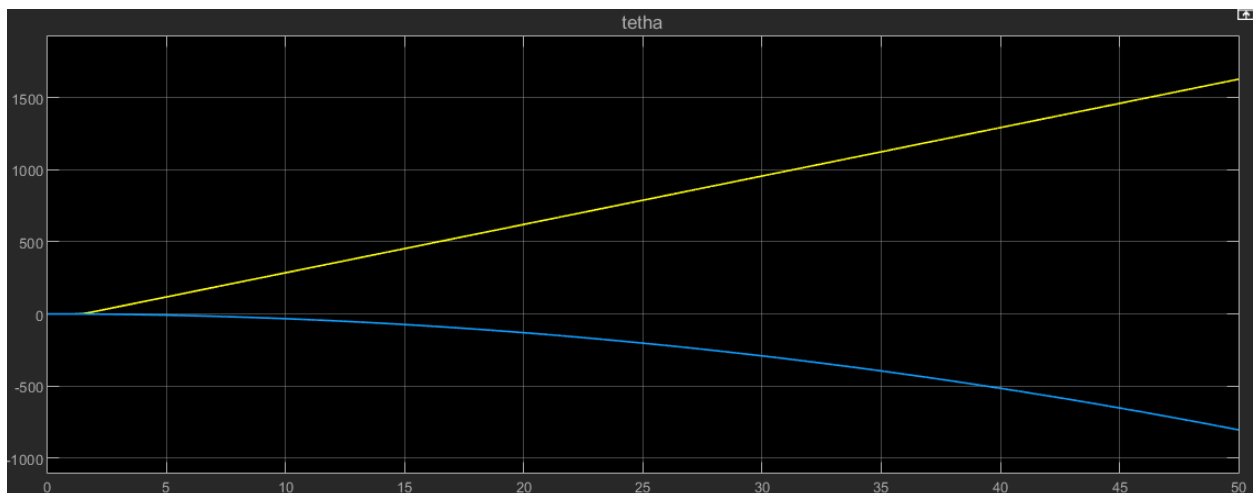


Figure 90- The output of the Z.N PID in MATLAB for the linear system

It is evident that the designed PID controller is not capable of stabilizing the pendulum in the linear model. Next, this PID controller will be applied to the original nonlinear system to evaluate its performance.

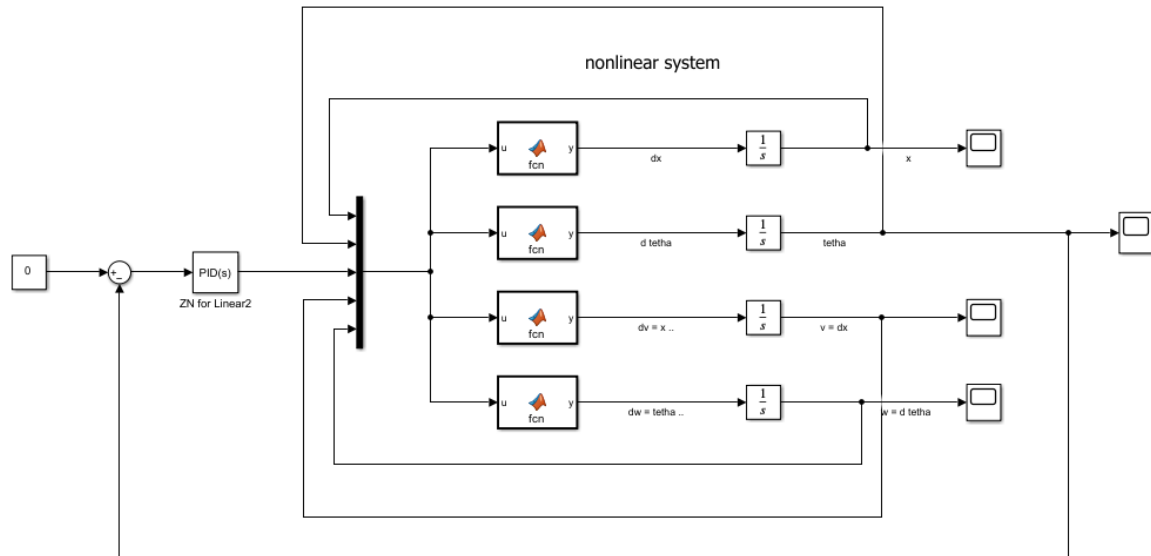


Figure 91- Applying the ZN PID (from identified linear system) on nonlinear system

The output with applying a gain is as follows:

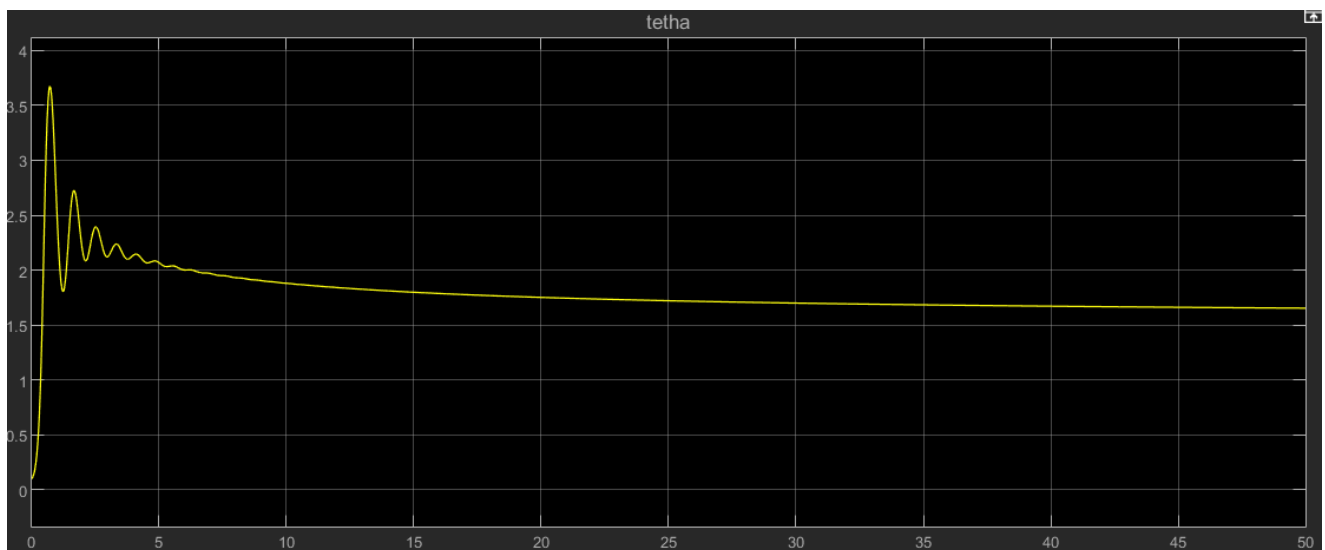


Figure 92- The output of the ZN PID in MATLAB for the nonlinear system after applying a gain

## 2-2-2. Ziegler-Nichols PID Tuning in the Frequency Domain

Based on Equation 18, which defines the ultimate point characteristics, and the Ziegler-Nichols frequency domain table shown below, the PID coefficients are designed.

$T_p$	$T_d$	$T_i$	K	کنترل کننده
$T_u$			$0.5K_u$	<b>P</b>
$1.4T_u$		$0.8T_u$	$0.4K_u$	<b>PI</b>
$0.85T_u$	$0.125T_u$	$0.5T_u$	$0.6K_u$	<b>PID</b>

Figure 93- Ziegler-Nichols table in frequency domain

Therefore:

$$K = 0.025, T_i = 2.185, T_d = 0.5462$$

Equation 21- Z.N PID coefficients in frequency domain for the linear system

By applying the PID coefficients as follows and forming a closed loop for the identified system, the following results will be obtained:

Main	Initialization	Saturation	Data Types	State Attributes
Controller parameters				
Source: internal				
Proportional (P): 0.025				
Integral (I): 1/2.185 0.45767 <input type="checkbox"/> Use I*Ts (optimal for codegen)				
Derivative (D): 0.5462				
Filter coefficient (N): 100 <input checked="" type="checkbox"/> Use filtered derivative				

Figure 94- Z.N PID coefficients in frequency domain in MATLAB for the linear system

The block diagram and the output are as follows:

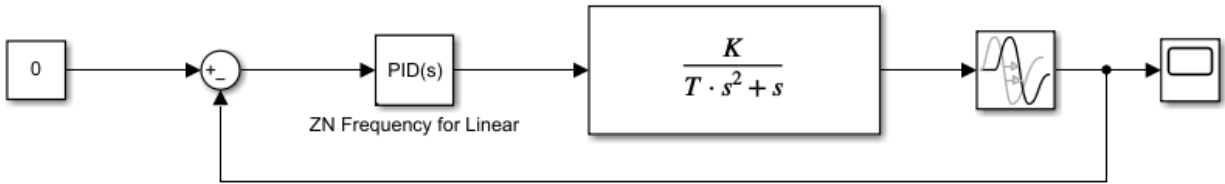


Figure 95- Closed loop system with Z.N PID in frequency domain in MATLAB for identified linear system

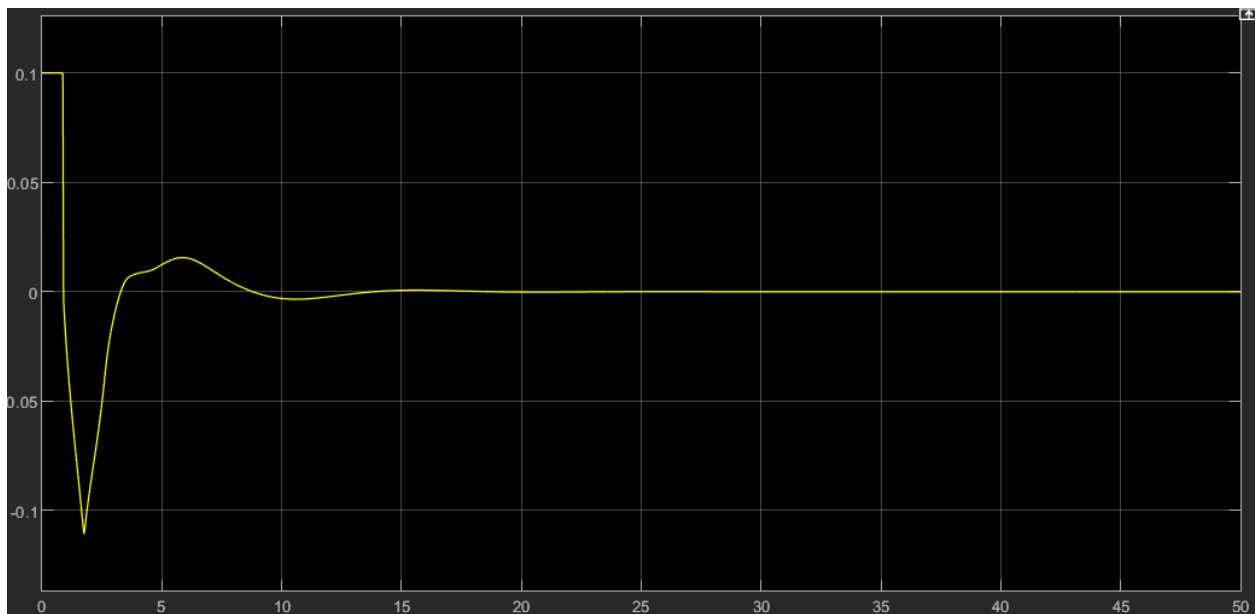


Figure 96- The output of closed loop system with Z.N PID in frequency domain in MATLAB for identified linear system

It can be observed that the designed PID controller is capable of controlling the identified system. However, this is not sufficient; the controller must also be tested on both linear and nonlinear models.

Now, the designed PID controller must be applied to the linear system to determine whether an appropriate response is achieved.

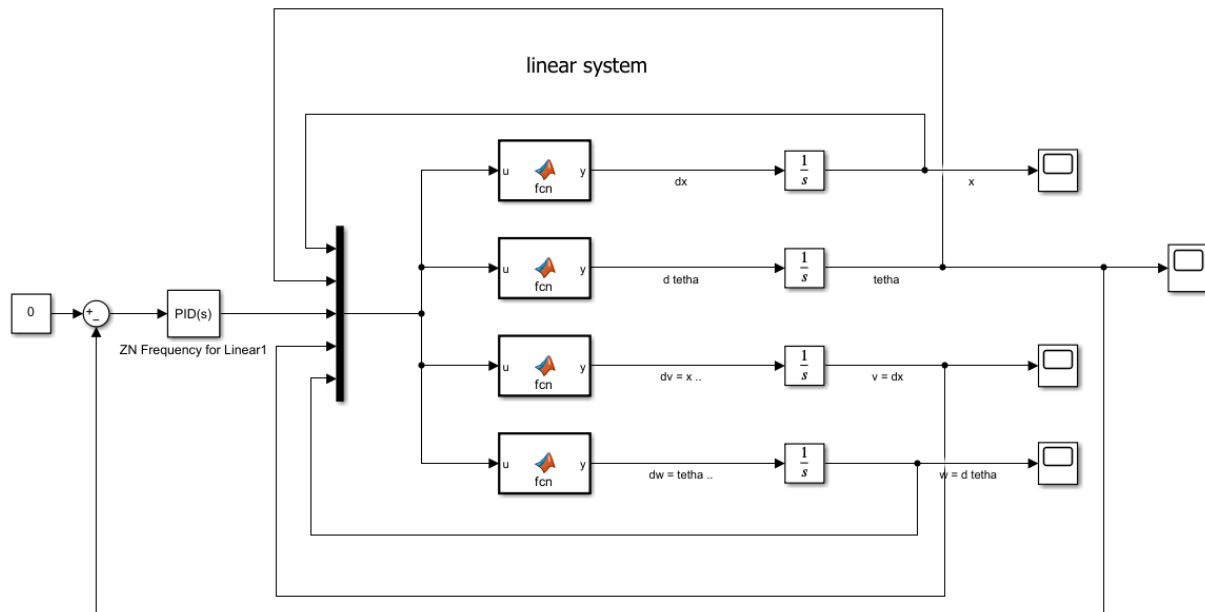


Figure 97- Applying the Z.N PID in frequency domain in MATLAB for the linear system

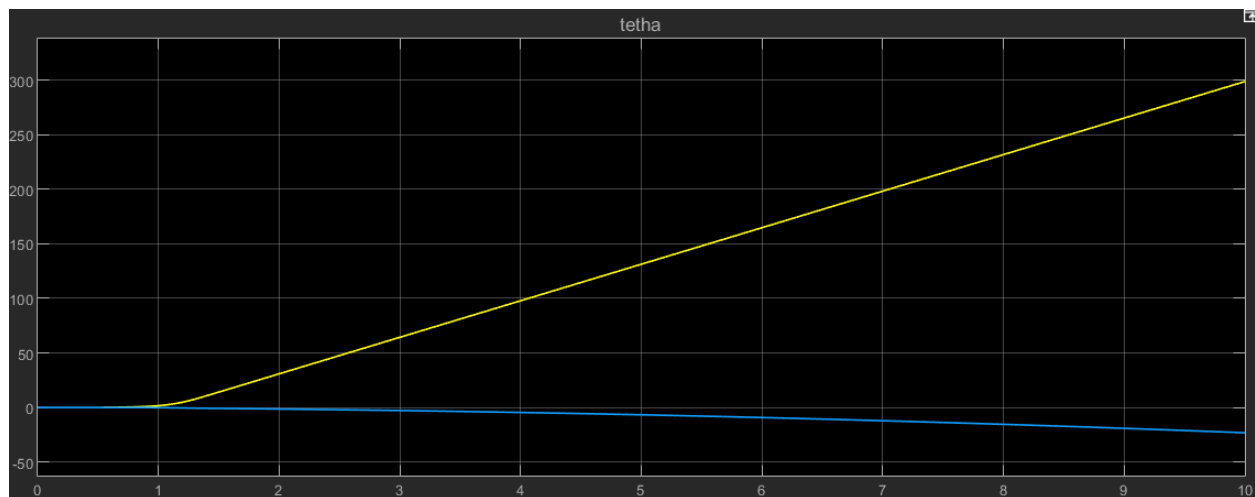


Figure 98- The output of the Z.N PID in frequency domain in MATLAB for the linear system

It is evident that this system cannot be controlled using the PID controller. Nevertheless, the designed PID controller will be applied to the main system, which is nonlinear, to examine the response.

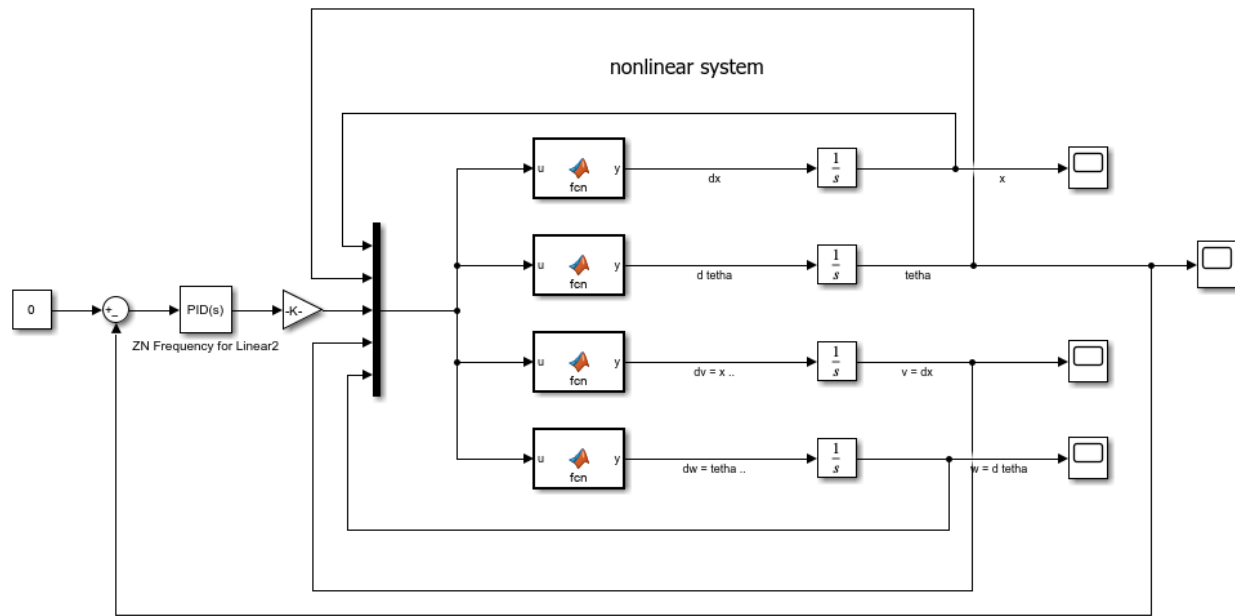


Figure 99- Applying the ZN PID in frequency domain (from identified linear system) on nonlinear system

The output with applying a gain is as follows:

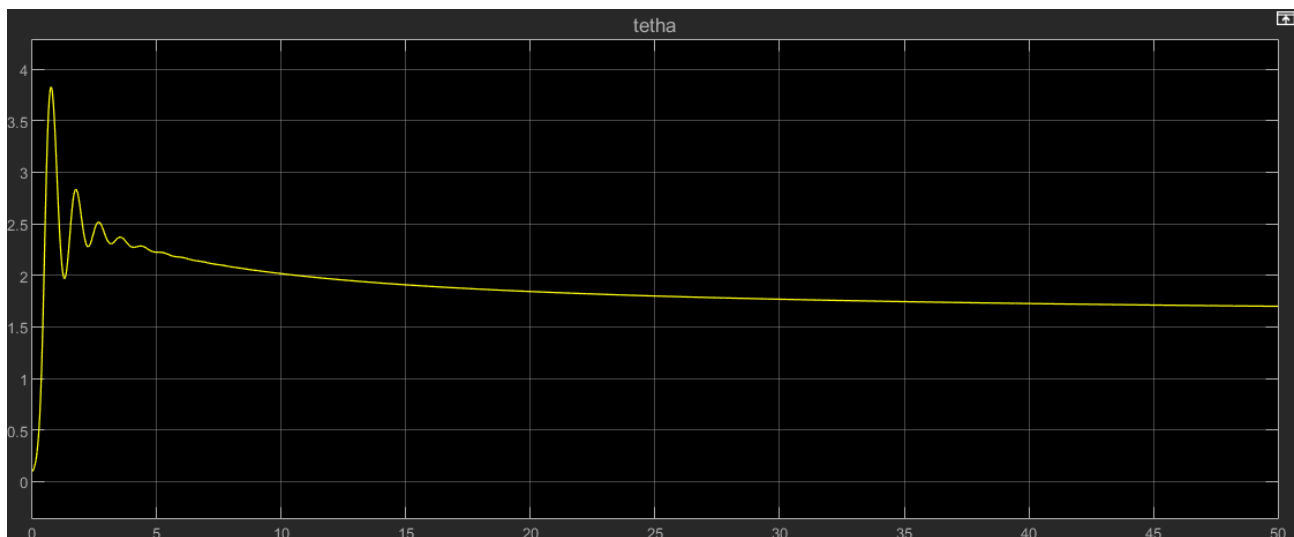


Figure 100- The output of the ZN PID in frequency domain in MATLAB for the nonlinear system after applying a gain



### 2-2-3. Generalized Ziegler-Nichols PID Tuning

If in the generalized Ziegler-Nichols method, the initial point and final point are considered, the equations to reach point  $B$  will be as follows:

$$\begin{aligned} K &= K_u r_b \cos \varphi_b \\ T_i &= \frac{T_u}{\pi} \left( \frac{1 + \sin \varphi_b}{\cos \varphi_b} \right) \\ T_d &= \frac{\alpha T_u}{\pi} \left( \frac{1 + \sin \varphi_b}{\cos \varphi_b} \right) \end{aligned}$$

Equation 22- Tuning the coefficients using the generalized Ziegler-Nichols method

It should be noted that  $\alpha = 0.25$  is considered.

In the PID controller design section using the generalized Ziegler-Nichols method for the nonlinear system, we observed that reducing the angle of point  $B$  resulted in a better response. Therefore, point  $B$  is considered with the following specifications:

$$r_b = 0.5, \varphi_b = 20^\circ$$

Figure 101- Specifications of point B

By substituting Figure 101 into Equation 22, we will have:

$$\begin{aligned} K &= 0.47 K_u \\ T_i &= 0.45 T_u \\ T_d &= 0.11 T_u \end{aligned}$$

Equation 23- The PID coefficients at point B

In Equation 18, the ultimate point values are calculated. By substituting these values into Equation 23, we will have:

$$K = 0.02, T_i = 1.96, T_d = 0.48$$

Equation 24- Generalized Z.N PID coefficients for the linear system

By applying the PID coefficients as follows and forming a closed loop for the identified system, the following results will be obtained:

Main	Initialization	Saturation	Data Types	State Attributes
Controller parameters				
Source: internal				
Proportional (P): 0.02				
Integral (I): 1/1.96 0.5102 <input type="checkbox"/> Use I*Ts (optimal for codegen)				
Derivative (D): 0.48				
Filter coefficient (N): 100 <input checked="" type="checkbox"/> Use filtered derivative				

Figure 102- Generalized Z.N PID coefficients in MATLAB for the linear system

The block diagram and the output are as follows:

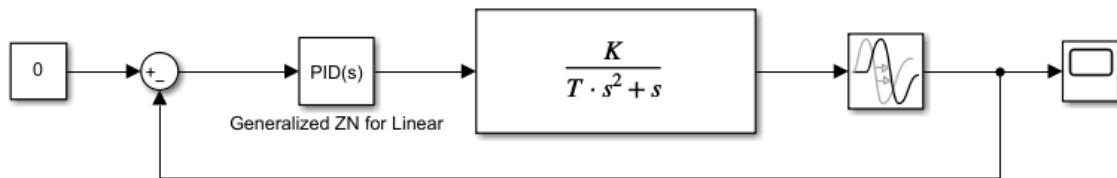


Figure 103- Closed loop system with generalized Z.N PID in MATLAB for identified linear system

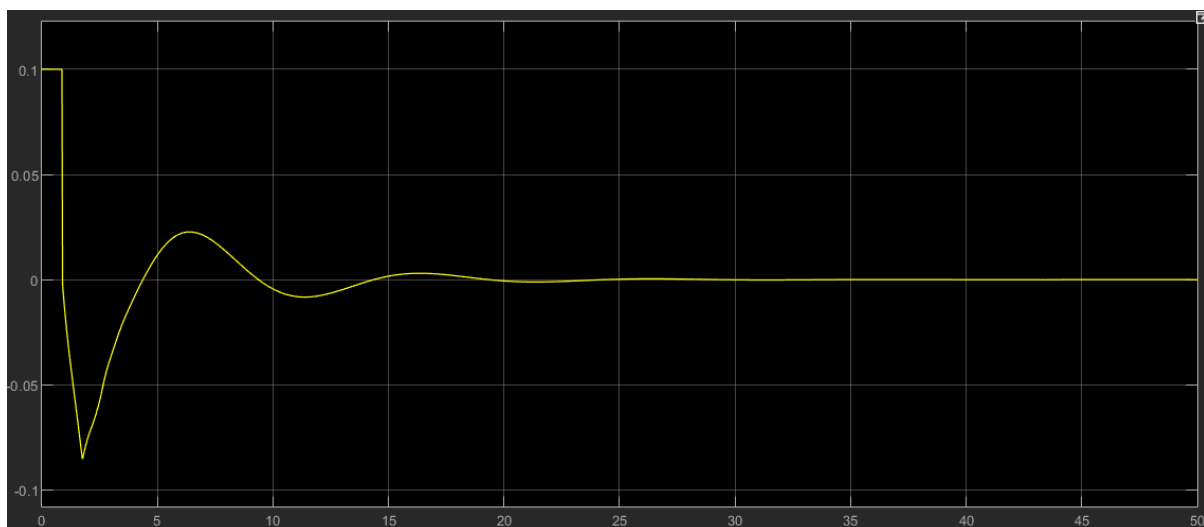


Figure 104- The output of closed loop system with generalized Z.N PID in MATLAB for identified linear system

Now, the designed PID controller must be applied to the linear system to determine whether an appropriate response is achieved.

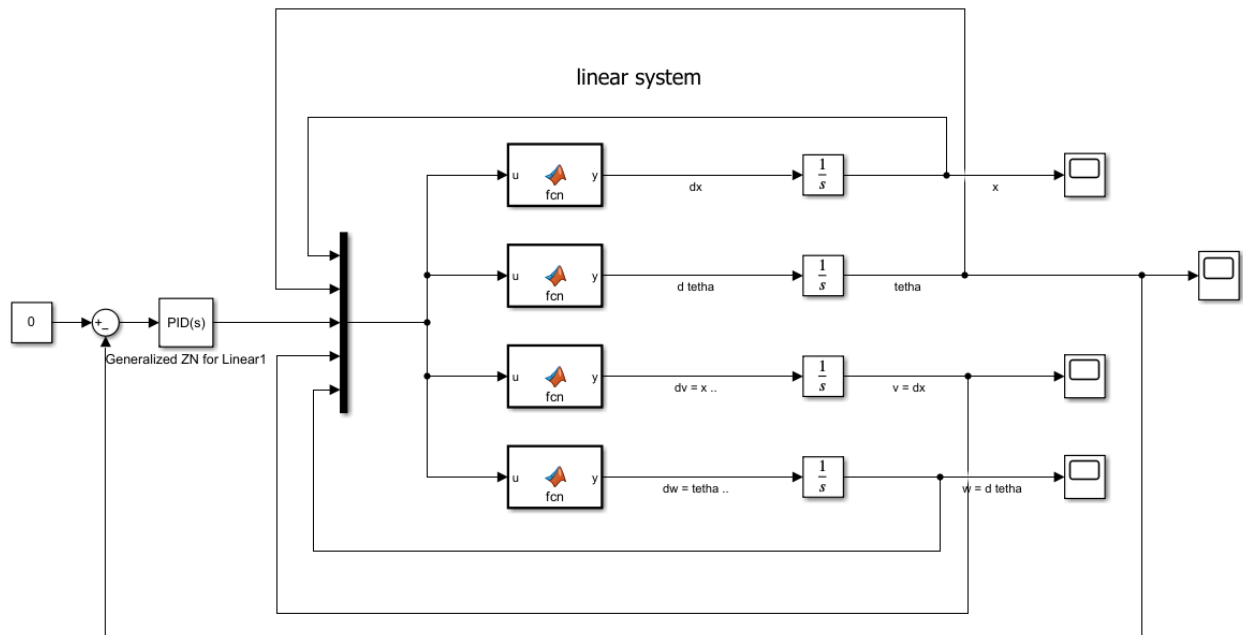


Figure 105- Applying the generalized Z.N PID in MATLAB for the linear system

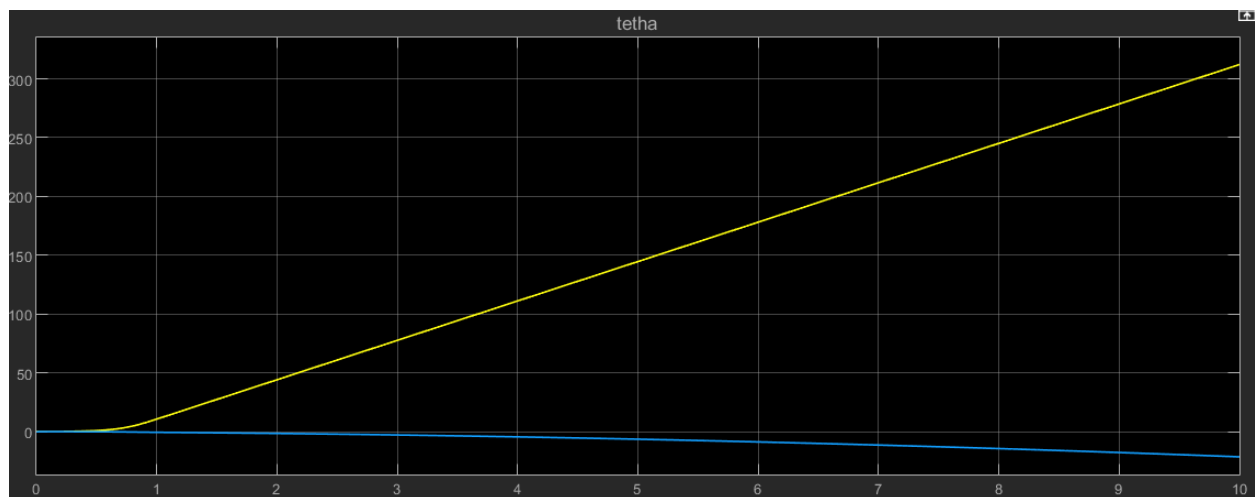


Figure 106- The output of the generalized Z.N PID in MATLAB for the linear system

It is evident that this system cannot be controlled using the PID controller. Nevertheless, the designed PID controller will be applied to the main system, which is nonlinear, to examine the response.

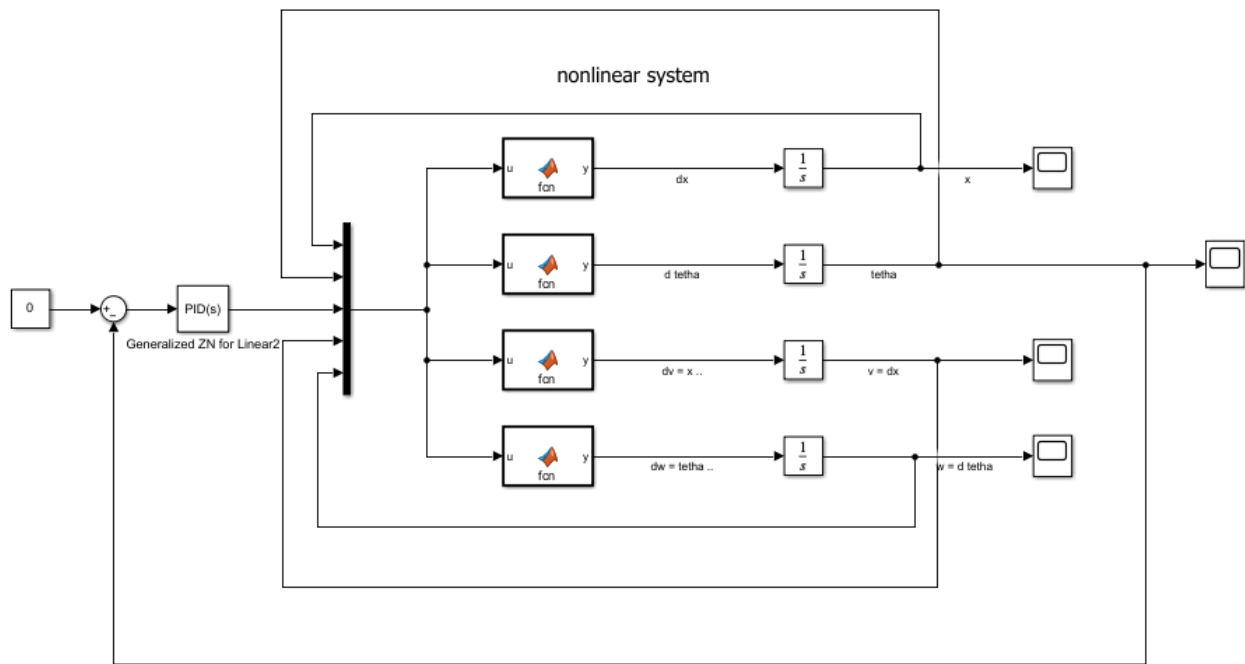


Figure 107- Applying the generalized ZN PID (from identified linear system) on nonlinear system

The output with applying a gain is as follows:

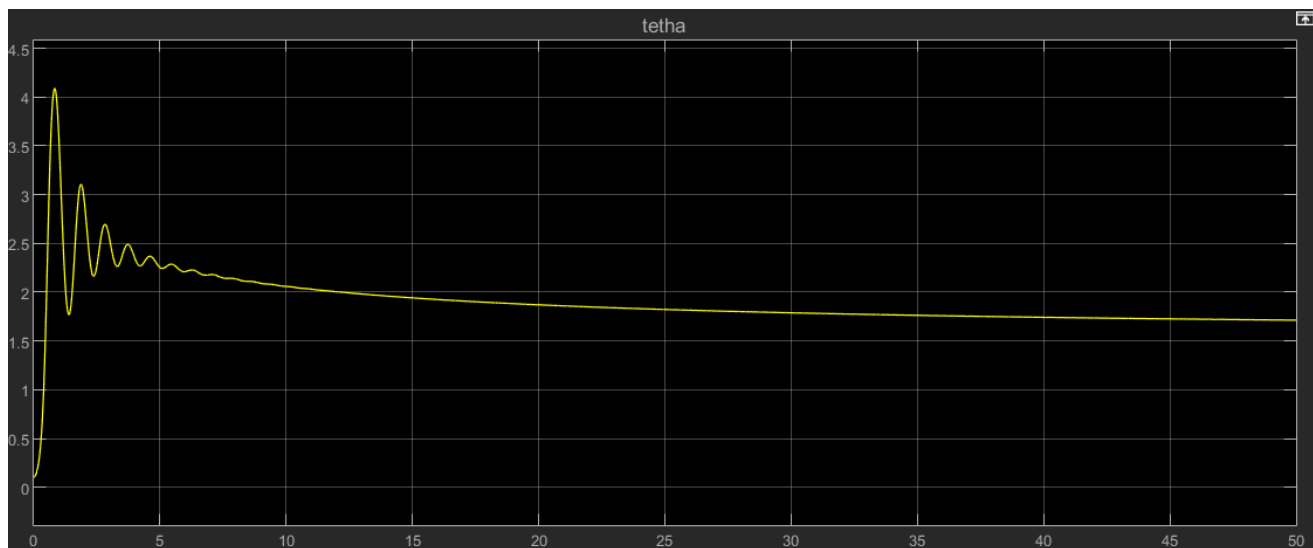


Figure 108- The output of the generalized ZN PID in MATLAB for the nonlinear system after applying a gain

## 2-2-4. PID Design Using the Lambda Tuning Method

Since this method is based on a three-parameter model and our identified system is an integral system, it is not possible to apply this PID design method.

## 2-2-5. PID Design Using the CHR Method

Previously, the parameters  $a$  and  $L$  for this system were determined. Now, using these parameters and the table below, which pertains to calculating PID coefficients via the CHR method, the PID coefficients are computed.

K	$T_i$	$T_d$
$a/0.7$	-	-
$a/0.7$	$2.3L$	-
$a/1.2$	$2.0L$	$0.42L$

Figure 109- CHR table

Note that the term  $K$  should be  $1.2/a$ .

The PID coefficients are as follows:

$$K = 0.033, T_i = 1.7372, T_d = 0.3648$$

Equation 25- CHR PID coefficients

By applying the PID coefficients as follows and forming a closed loop for the identified system, the following results will be obtained:

The screenshot shows the 'Controller Parameters' block in MATLAB Simulink. The 'Source' is set to 'internal'. The 'Proportional (P)' gain is 0.033. The 'Integral (I)' gain is 1/1.7372, with a preview of 0.57564. The 'Derivative (D)' gain is 0.3648. The 'Filter coefficient (N)' is 100. There are checkboxes for 'Use I\*T\_s (optimal for codegen)' (unchecked) and 'Use filtered derivative' (checked).

Parameter	Value
Source	internal
Proportional (P)	0.033
Integral (I)	1/1.7372 (0.57564)
Derivative (D)	0.3648
Filter coefficient (N)	100

Figure 110- CHR PID coefficients in MATLAB for the linear system

The block diagram and the output are as follows:

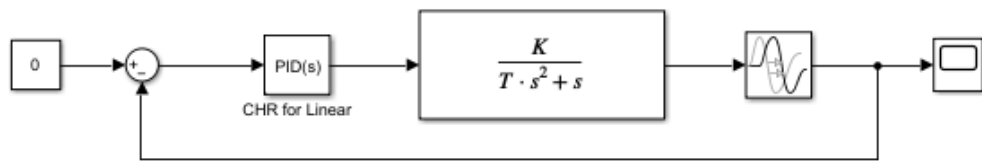


Figure 111- Closed loop system with CHR PID in MATLAB for identified linear system

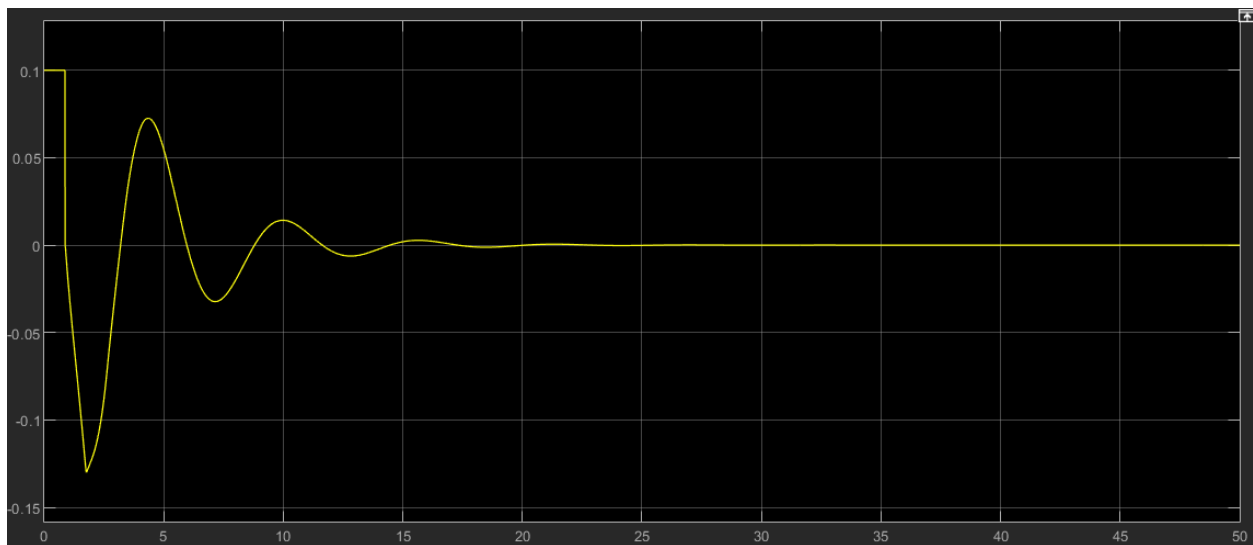


Figure 112- The output of closed loop system with CHR PID in MATLAB for identified linear system

Now, the designed PID controller must be applied to the linear system to determine whether an appropriate response is achieved.

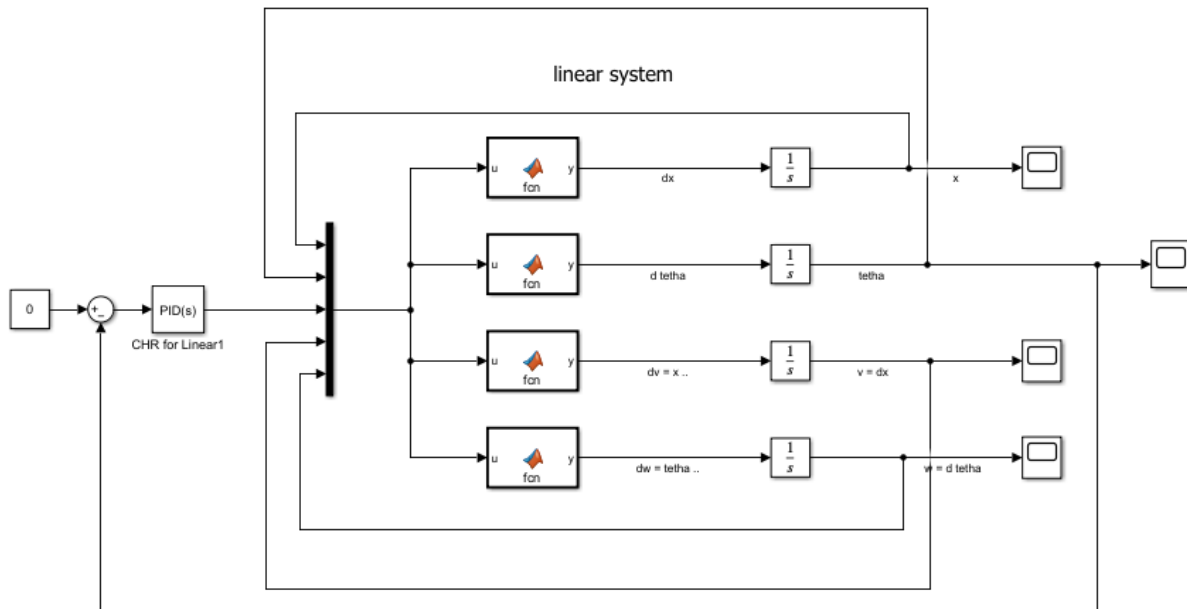


Figure 113- Applying the CHR PID in MATLAB for the linear system

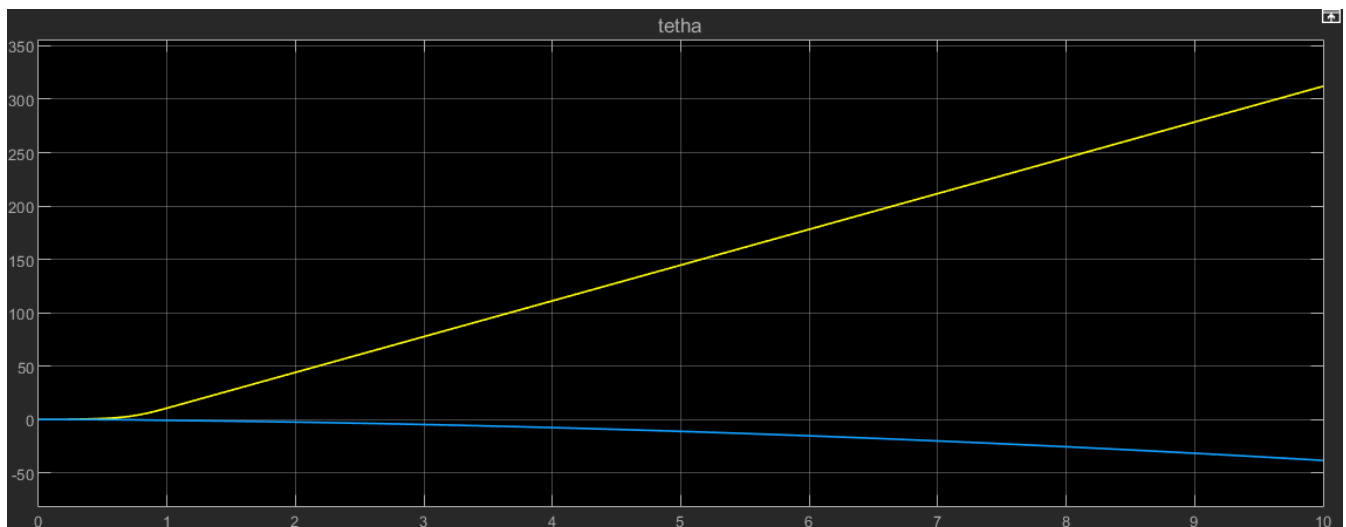


Figure 114- The output of the CHR PID in MATLAB for the linear system

Once again it is evident that this system cannot be controlled using the PID controller. Nevertheless, the designed PID controller will be applied to the main system, which is nonlinear, to examine the response.

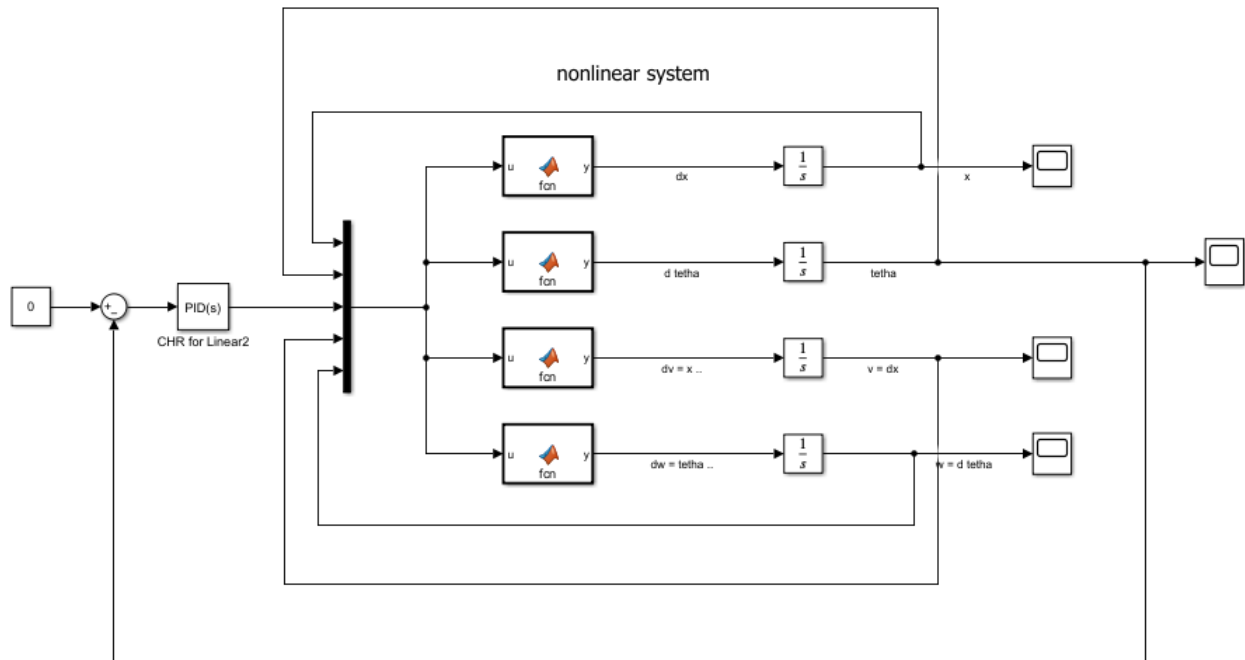


Figure 115- Applying the CHR PID (from identified linear system) on nonlinear system

The output with applying a gain is as follows:

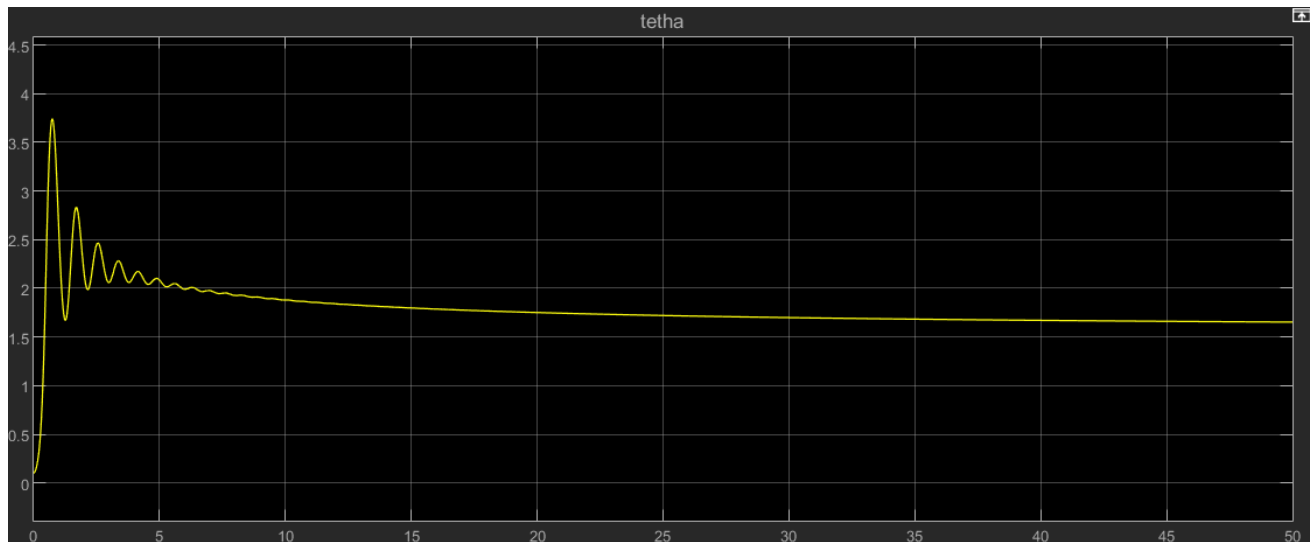


Figure 116- The output of the CHR PID in MATLAB for the nonlinear system after applying a gain



### 3.Conclusion

In this project, eight PID controllers were designed. Four of them were based on the second identification method and were applied to the nonlinear system.

Initially, a PID controller was designed using the Ziegler-Nichols time-domain method and applied to the identified system. The result, shown in Figure 43, indicates that the controller was properly designed. The identified system, which is the same system used to derive the controller parameters, exhibited appropriate behavior, successfully stabilizing the system for an initial angular displacement of 6 degrees.

Subsequently, this controller was applied to the original nonlinear system. At first, the control signal was too weak to stabilize the pendulum effectively. However, after amplifying the control signal by adding a gain, the system's performance improved. Despite this improvement, the performance was still not ideal. Although the system was stabilized, it exhibited steady-state error. This behavior is illustrated in Figure 45.

Due to the nonlinear nature of the original system and the lack of specific training in nonlinear control within the scope of this course, I was unable to achieve perfect control over the original system.

Additionally, PID controllers were designed using the Ziegler-Nichols frequency-domain, generalized Ziegler-Nichols, and CHR methods. Their performance, both on the identified system and on the original nonlinear system, was similar to that of the controller designed using the Ziegler-Nichols time-domain method.

Type of PID	Performance	
	Identified	Nonlinear
<b>Ziegler-Nichols Time Domain</b>	Proper control with an initial deviation of 6 degrees (Figure43)	Stabilizing the system but with steady-state error (Figure45)
<b>Ziegler-Nichols Frequency Domain</b>	Proper control with an initial deviation of 6 degrees (Figure50)	Stabilizing the system but with steady-state error (Figure53)
<b>Generalized Ziegler-Nichols</b>	Proper control with an initial deviation of 6 degrees (Figure59)	Stabilizing the system but with steady-state error (Figure62)
<b>CHR</b>	Proper control with an initial deviation of 6 degrees (Figure66)	Stabilizing the system but with steady-state error (Figure69)

Table 1- Total performance of method2

Four other controllers were designed using the same methods described above. However, this time, the nonlinear system was linearized first, and then identification was performed on the linearized system. Using the identified parameters, such as the ultimate point, system delay, and other relevant characteristics, the required parameters for designing the controllers were extracted.

These controllers, like those designed for systems identified from the nonlinear system, performed well on the systems they were initially designed for, namely, the identified systems.

After ensuring the effectiveness of these controllers, they were first applied to the linearized system and then to the original nonlinear system. When applied to the linearized system, the response was not satisfactory. Thus, they were applied to the original system since the primary goal of this project was to control the original system.

Upon applying these controllers to the nonlinear system, it was observed that the control signal was still too weak to adequately control the system. By introducing a gain and amplifying the control signal, an attempt was made to improve the control. Although the system was stabilized, it still fell short of an ideal level of control, retaining a steady-state error.

Type of PID	Performance		
	Identified	Linear	Nonlinear
<b>Ziegler-Nichols Time Domain</b>	Proper control with an initial deviation of 6 degrees (Figure88)	The PID controller was unable to perform the control (Figure90)	Stabilizing the system but with steady-state error (Figure92)
<b>Ziegler-Nichols Frequency Domain</b>	Proper control with an initial deviation of 6 degrees (Figure96)	The PID controller was unable to perform the control (Figure98)	Stabilizing the system but with steady-state error (Figure100)
<b>Generalized Ziegler-Nichols</b>	Proper control with an initial deviation of 6 degrees (Figure104)	The PID controller was unable to perform the control (Figure106)	Stabilizing the system but with steady-state error (Figure108)
<b>CHR</b>	Proper control with an initial deviation of 6 degrees (Figure112)	The PID controller was unable to perform the control (Figure114)	Stabilizing the system but with steady-state error (Figure116)

Table 2- Total performance of linear system

It can be said that the controllers designed in the linear section had more overshoot compared to method 2.

In conclusion, the PID controller did not perform adequately for controlling a nonlinear system like the inverted pendulum. Significant efforts were made in this project to control the system by any means, but the results were limited to stabilizing the system without achieving the desired level of control.