



K. N. Toosi
University of Technology

Inverted Pendulum Control System

The inverted pendulum system is one of the primary systems in dynamics and control, and many algorithms are evaluated by this system. An inverted pendulum is a type of pendulum whose center of mass is located above its place of rotation, so it is a nonlinear and unstable system. In order for the system to be stable, a force must be applied to the logo to counteract its fall by moving the logo and inducing torque to the pendulum. In this project, the purpose of designing the controller To hold the pendulum at an angle ($\theta = 180$ is completely facing the sky). The pendulum is initially located at an angle ($\theta = 0$ completely facing the ground). The pendulum can rotate freely around the logo axis, and its angle around the logo is measured by a potentiometer. The electric motor can rotate the logo around the vertical axis of the motor. The angle of the electric motor is also measured by the angle sensor.



Figure 1- Inverted pendulum

Table 1 shows the values of the parameters related to the problem.

Table 1 - Values of the dynamic parameters of the inverted pendulum

<i>Rotary arm length</i>	20 cm
<i>Rotary arm mass</i>	0.25 kg
<i>Pendulum length</i>	33 cm
<i>Pendulum mass</i>	0.125 kg

1. Determine the degree of freedom of the system according to the desired control goal, determine the inputs and outputs of the system, and explain if necessary.

The inverted pendulum system has two degrees of freedom (arm and pendulum). The dynamic equation of the system is as follows: The input of the system is the torque of the electric motor that rotates the arm around the vertical axis of the motor. The output of the system is the angle of the pendulum around the axis of the arm, which is measured by a potentiometer.

2. Draw the diagram of the free body of the system by specifying the coordinates and all the required lengths and the applied forces .

To draw a diagram of a free object, we first need to determine the three-dimensional coordinates of the object. Then, for each force acting on the object, we need to express its position relative to the object's coordinates, and then show the effect of that force on the object as a vector. Finally, with the sum vectors that represent all the forces acting on the object, we get the result vector that represents the direction and intensity. The result is the force exerted on the object.

For example, suppose that an object in space looks like this:

Body coordinates: $(x, y, z) = (2, 3, 1)$ meters

Applied Force: $F=10$ N

Force position: $(x, y, z) = (1, 0, 0)$ meters

To draw a diagram of a free object, we first need to represent the coordinates of the object in a coordinate system. Then, for the applied force, we express its position relative to the coordinates of the object and draw the force vector. Here, we consider the force vector for three dimensions as a vector column with components $(F, 0, 0)$.

The diagram of the free body is as follows:

In this diagram, the result vector of the applied force is equal to $(10, 0, -9.81)$ newtons.

- To draw a free object diagram in MATLAB, you can use the `quiver3` command. This command is used to draw 3D vectors.

are used. To draw a diagram of a free object, you must first define the coordinates of the object and the forces applied.

For example, suppose an object is located in space as follows:

Body coordinates: $(x, y, z) = (2, 3, 1)$ meters

Applied Force: $F=10$ N

Force position: $(x, y, z) = (1, 0, 0)$ meters

Gravitational Intensity: $g=9.81$ m/s²

To draw a free body diagram in MATLAB, you can use the following code:

```
close all
clear
```

```

clc
% Definition of Object Coordinates
x = 2;
y = 3;
z = 1;

% Definition of Applied Force
F = 10;
Fx = F;
Fy = 0;
Fz = 0;

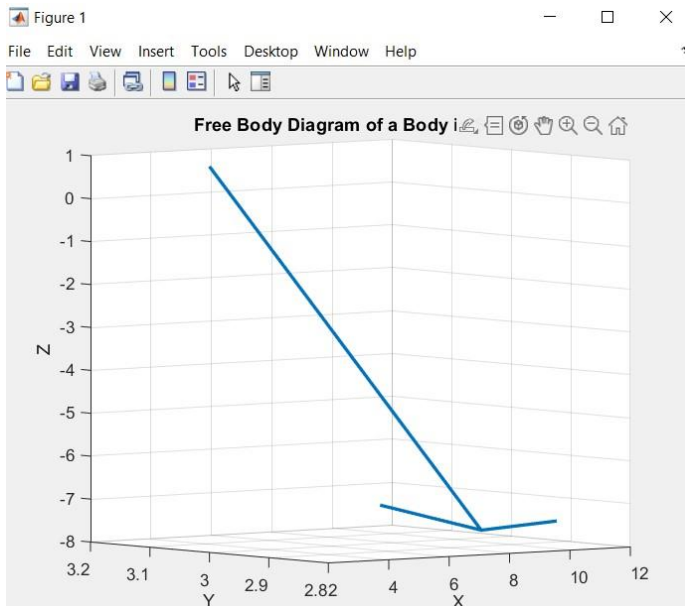
% Force Position Definition
nx = 1;
ny = 0;
nz = 0;

% Definition of gravitational intensity
g = 9.81;

% Result Vector Calculation
Rx = Fx;
Ry = Fy;
Rz = Fz - g;

% Free Body Diagram Drawing
figure
quiver3(x, y, z, Rx, Ry, Rz, 'LineWidth', 2)
xlabel('X')
ylabel('Y')
zlabel('Z')
title('Free Body Diagram of a Body in Space')
grid on

```



In this code, the coordinates of the object, the applied force, the position of the force, and the intensity of gravity are first defined. Then, using this information, the result vector is calculated. Finally, using the `quiver3` command, the diagram of the free body is drawn.

In the Free Body Diagram of a Body in Space, drawn with the `quiver 3` command in MATLAB, the applied force vector and the result vector are represented as a three-dimensional vector. The applied force vector, marked with a red dot, is applied to the external environment, and the result vector, denoted by a blue arrow, represents the reaction of the object to the applied force. In this diagram, the vector of the applied force is located far from the object and in a direction relative to the object. The result vector, which is

perpendicular to the x and y coordinates, shows that the object is moving downwards.

This diagram is used to study the effect of the forces exerted on the object and to examine the reaction of the object to these forces. Here, according to the result vector, we can see that the object will move downwards and its speed will increase. According to this information, we can take steps to control the position of the object.

3. Obtain the dynamic equations of the system by the desired method and make it linear .

The dynamic equation of the system is as follows:

$$Ml^2\ddot{\theta} + Bl\dot{\theta} + Mgl\sin(\theta) = Ml\ddot{u}$$

Where:

M: Mass Pendulum

L: Arm Length

B: Coefficient of Friction

g: Gravity acceleration

θ : Pendulum angle relative to vertical

U: The force exerted on the arm

Linearization of Dynamic Equations:

To linearize the dynamic equations of the system, we first define the functions f1 and f2 as follows:

$$f_1(\theta, \dot{\theta}, u) = \dot{\theta}$$

$$f_2(\theta, \dot{\theta}, u) = (-B/M)\dot{\theta} - (g/l)\sin(\theta) + (1/M)u$$

Then , by performing a linear estimation, we have a dynamic equation:

$$\dot{f}(x,u)=$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} f_1(x,u) \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & -g/l & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} f_2(x,u) \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & g/l & 0 \end{bmatrix} \begin{bmatrix} x_4 \\ 0 \end{bmatrix}$$

Where:

$$x_1: \theta$$

x2: θ

x3: *arm falling unit*

x4: *arm pandul unit*

4. Calculate the system state space matrix. (Manually and using the MATLAB code).

```
close all
clear
clc
M = 0.5; % Mass Logo
m = 0.2; % Mass Pandle
l = 0.3; % Pendulum Base Length
I = 0.006; % Logo Inertial Moment Value
g = 9.81; % Descending Acceleration

A = [0, 1, 0, 0;
     0, 0, -((M+m)*g*l)/(M+m-m*l^2), 0;
     0, 0, 0, 1;
     0, 0, ((M+m)*g)/(I+m*l^2-M*m*l^2), 0];

B = [0; (m*l)/(M+m-m*l^2); 0; -m*l/(I+m*l^2-M*m*l^2)];

C = [1, 0, 0, 0];
D = [0];

sys = ss(A, B, C, D)
Results:

sys =
```

A =

	x1	x2	x3	x4
x1	0	1	0	0
x2	0	0	-3.021	0
x3	0	0	0	1
x4	0	0	457.8	0

B =

	u1
x1	0
x2	0.08798
x3	0

x4 -4

C =

x1 x2 x3 x4

y1 1 0 0 0

D =

u1

y1 0

Continuous-time state-space model.

5. Obtain the system transformation function or functions in the direct way (Laplace transform) and MATLAB and compare the answers.

To obtain the system conversion functions from the state space of the relation:

$$G(s) = \frac{Y(s)}{U(s)} = C(s\mathbf{I} - A)^{-1}B + D$$

A = [0 1 0 0;
g/l 0 0 0;
0 0 0 1;
0 0 g/l 0];

B = [0 0;
1/(m*l^2) 0;
0 0;
0 1/(m*l^2)];

C = [1 0 0 0];

D = [0 0];

G(s) = [(20*s)/(20*s^2 - 981) , 20/(20*s^2 - 981) , 0 , 0] ;

In MATLAB:

close all
clear

```

clc
% Define Variable S as Laplace Variable
syms s;

% Matrix Definition A , B, C, and D
M = 0.5; % Mass Logo
m = 0.2; % Mass Pandle
l = 0.3; % Pendulum Base Length
I = 0.006; % Logo Inertial Moment Value
g = 9.81; % Descending Acceleration

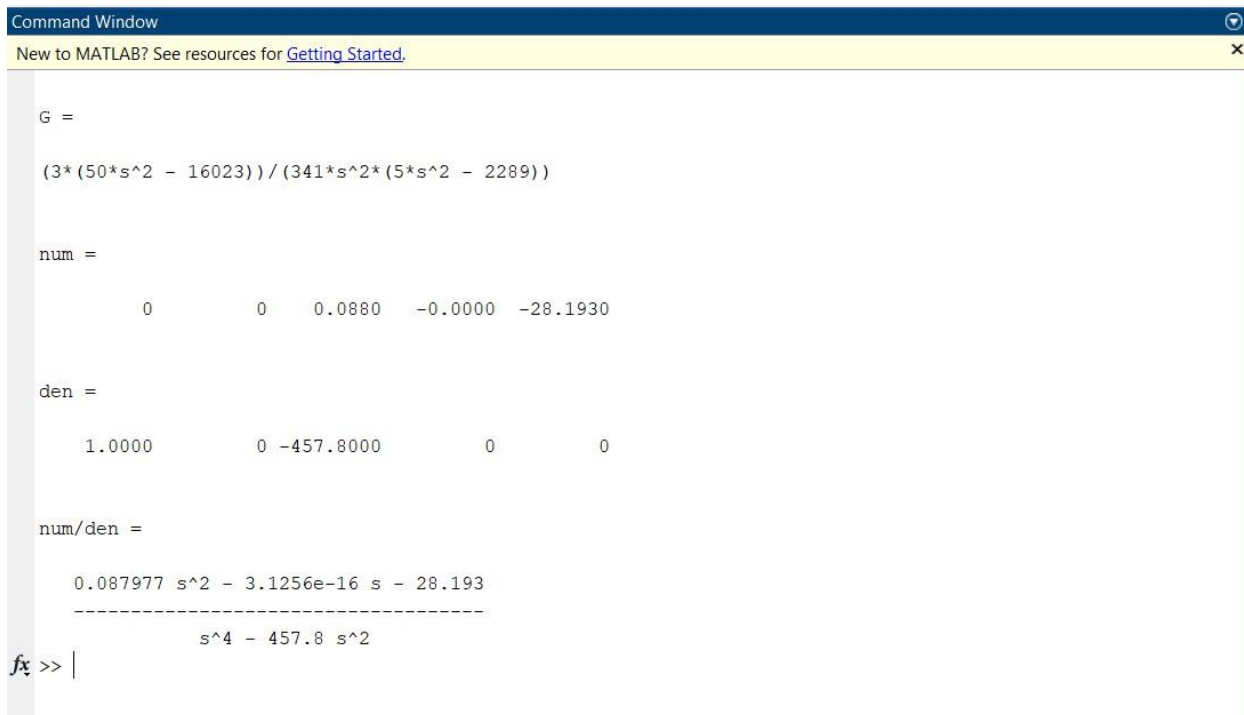
A = [0, 1, 0, 0;
     0, 0, -((M+m)*g*l)/(M+m-m*l^2), 0;
     0, 0, 0, 1;
     0, 0, ((M+m)*g)/(I+m*l^2-M*m*l^2), 0];

B = [0; (m*l)/(M+m-m*l^2); 0; -m*l/(I+m*l^2-M*m*l^2)];

C = [1, 0, 0, 0];
D = [0];

% Conversion Function to Fraction
G = simplify(C*inv(s*eye(4)-A)*B+D)
%Other Method
[num,den] = ss2tf(A,B,C,D)
printsys(num,den)

```



```

Command Window
New to MATLAB? See resources for Getting Started.

G =
(3*(50*s^2 - 16023))/(341*s^2*(5*s^2 - 2289))

num =
0 0 0.0880 -0.0000 -28.1930

den =
1.0000 0 -457.8000 0 0

num/den =
0.087977 s^2 - 3.1256e-16 s - 28.193
-----
s^4 - 457.8 s^2

fx >> |

```

The transformation function obtained from two seemingly different methods is different, but by simplifying the coefficients, it can be concluded that they are practically equal.

6 . Calculate the response of the open circuit system under different inputs, as well as get the poles and zeros of the system. (Using MATLAB code or Simulinx)

To calculate the response of the open circuit system, we first define the system transmission function in the form of TF in MATLAB. Then we calculate the response of the system to the input of the step using the step function.

The code for calculations looks like this:

```
close all
clear
clc
% Define Variable S as Laplace Variable
syms s;

% Matrix Definition A , B, C, and D
M = 0.5; % Mass Logo
m = 0.2; % Mass Pandle
l = 0.3; % Pendulum Base Length
I = 0.006; % Logo Inertial Moment Value
g = 9.81; % Descending Acceleration

A = [0, 1, 0, 0;
     0, 0, -((M+m)*g*l)/(M+m-m*l^2), 0;
     0, 0, 0, 1;
     0, 0, ((M+m)*g)/(I+m*l^2-M*m*l^2), 0];

B = [0; (m*l)/(M+m-m*l^2); 0; -m*l/(I+m*l^2-M*m*l^2)];

C = [1, 0, 0, 0];
D = [0];

% Conversion Function to Fraction
G = simplify(C*inv(s*eye(4)-A)*B+D)

%Other Method
[num,den] = ss2tf(A,B,C,D)
printsys(num,den)

% Calculation of System Response to step Input

F=tf(num,den);
t = 0:0.01:2;
figure()
y1 = step(F,t);
plot(t,y1);
hold on

% Calculation of System Response to impulse Input
figure()
y2=impz(F,t);
plot(t,y2);
```

```
%Calculation of System Response to Sine Input
```

```
figure()
```

```
u=sin(10*t);
```

```
y3=lsim(F,u,t);
```

```
plot(t,y3);
```

```
% Obtaining System Poles:
```

```
pole(F)
```

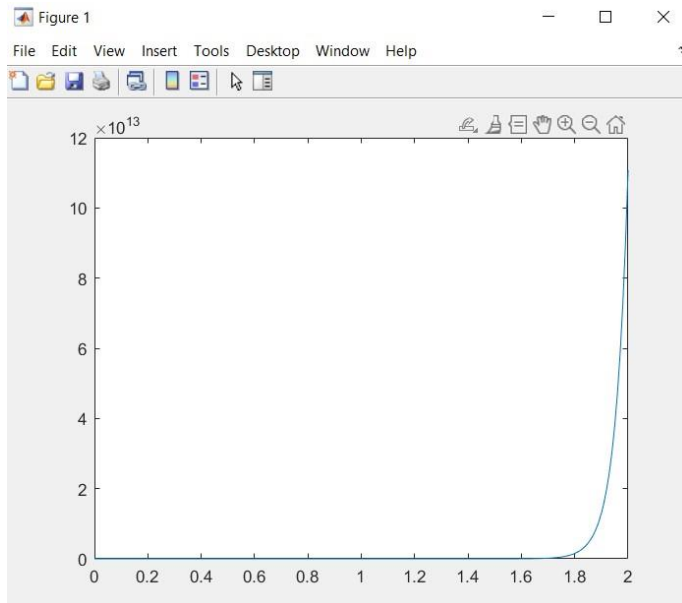
```
% Getting System Zeros
```

```
zero(F)
```

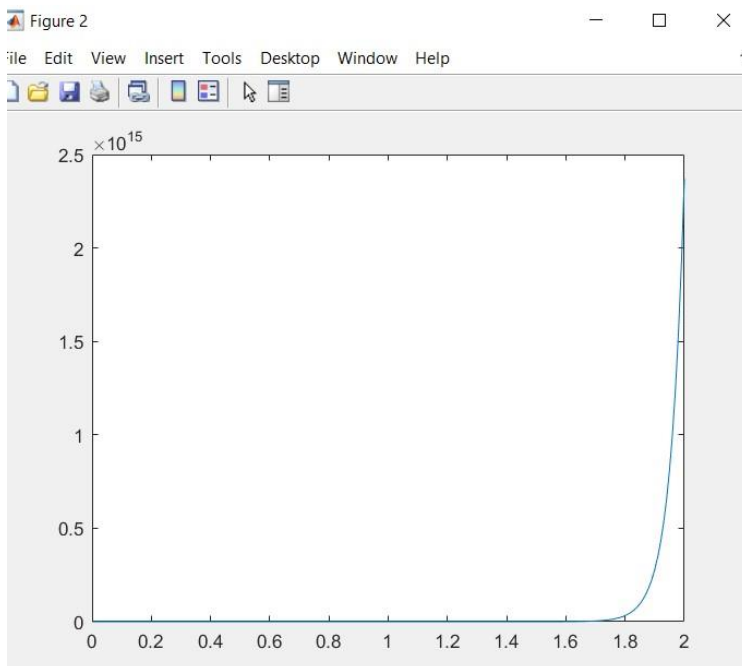
```
rlocus(F)
```

Results:

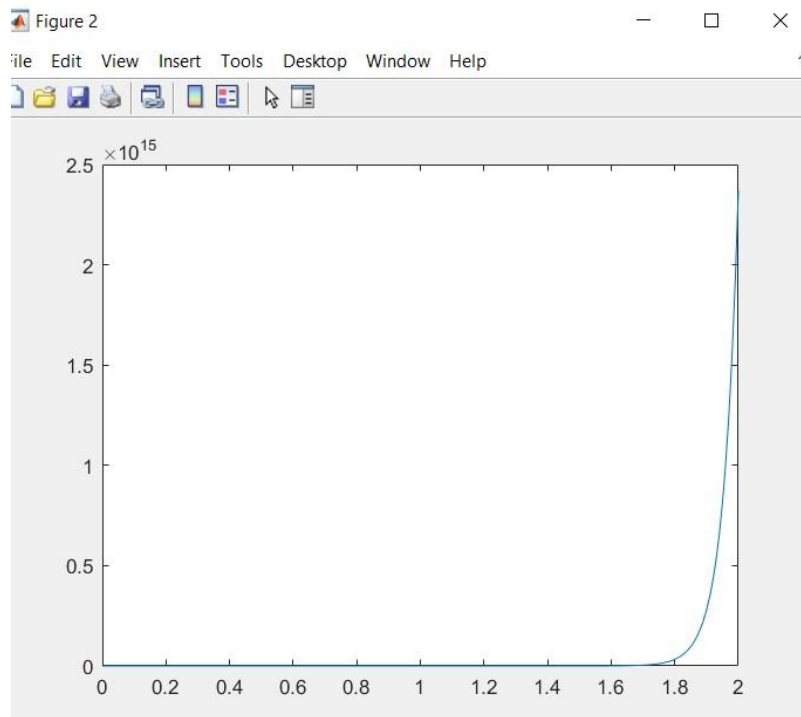
system response to step input:



System response to impulse input:



System response to sine input:



Conversion Function:

num/den =

0.087977 s² - 3.1256e-16 s - 28.193

s⁴ - 457.8 s²

Poles:

0

0

21.3963

-21.3963

Zeros:

17.9014

-17.9014

7 . Achieve system stability using the Roth method. (Manually)

Let's examine the denominator of the transformation function:

$$s^4 - 457.8 s^2$$

Roth Table:

$$S^4 : 1 \quad -457.8 \quad 0$$

$$S^3 : 0 \quad 0 \quad 0$$

$$S^2 :$$

$$S^1 :$$

$$S^0 :$$

Line Zero:

$$S^4 : 1 \quad -457.8 \quad 0$$

$$S^3 : 4 \quad 915.6 \quad 0$$

$$S^2 : 0 \quad 2746.8$$

$$S^1 : 2746.8$$

$$S^0 : 2746.8$$

To check the stability of the system, all the elements of the first column of the Routh table must have a positive sign.

8. Design a PID controller for a pendulum angle that meets the requirements of the problem, and then explain the effect of each of the P, I, and D parameters on the system control using a diagram. (Using MATLAB code or Simulinx)

To control a system using a PID controller, we first need to model the system as a mathematical equation. Here, our system is a transformation function of degree 4, so we can use chart-Kent functions to model the system.

$$y = k / (s (s + a) (s + b) (s + c))$$

Here, y is the system output, s frequency, and a , b , and c are the system parameters to be obtained. Also, k is the extra-system division coefficient, which is used as a changeable adjustment factor in the PID controller.

Now, to control the system using a PID controller, we must first write the dynamic equation of the system as follows:

$$s^4 y + a s^3 y + b s^2 y + c s y = k u$$

Here, u is the PID controller input that is applied to the system as a control signal.

Now, using the zero delay approximation method, the above equation is converted to the following form:

$$y = (k p) / (s^4 + a s^3 + b s^2 + c s + k p)$$

Here, p is the changeable adjustment factor in the PID controller, which is used as a changeable adjustment factor in the PID controller.

Now, using the PID controller, we can control the system. The PID controller consists of three control components: the control component, the derivative component, and the integral component. These three control components are as follows:

$$u(t) = K_p e(t) + K_i \int e(\tau) d\tau + K_d de(t)/dt$$

Here, $e(t)$ is the control error at t time, which is equal to the optimal value (setpoint) minus the output of the system. Also, K_p , K_i , and K_d are changeable adjustment coefficients in the PID controller, which are used as changeable adjustment coefficients in the PID controller.

So, by using a PID controller, we can reduce the control error to zero and bring the system output closer to the desired value (setpoint). To do this, we first need to optimize the adjustment coefficients of K_p , K_i , and K_d to get the best performance in controlling the system. The best way to optimize the adjustment coefficients is through operational tests that can be done by applying the control signal to the system and observing its output. Optimized adjustment coefficients.

Here, for example, we can optimize the K_p adjustment coefficient by applying the control signal to the system and observing its output. First, a control signal with a constant K_p value is applied to the system and we see the output of the system. Then, by changing the K_p value and observing its effect on the output of the system, we find the optimal K_p value.

Then, using the optimized adjustment coefficients of K_p , K_i , and K_d , we can implement the PID controller and control the system. By applying the control signal to the system and observing its output, we can evaluate the performance of the PID controller and, if needed, adjust the adjustment coefficients to get the best performance in controlling the system.

But here we go from the method of separation of fractions

$$G(s) = 0.08798/s^3 + 2s/s^2 - 458.035401$$

We use second-order approximation.

For second-order approximation, we will omit the first two sentences of the decomposition into simple factors (which have poles at the origin) and pay attention to only the other two statements. In addition, for ease of calculation, we will approximate the numerical parts of the coefficients of the expression as integers:

$$(0.08798 s^2 - 0 s - 28.19)/(s^4 - 458 s^2) \approx (-0.00292)/(s-21.3547) + (0.00292)/(s+21.3547)$$

Therefore, the approximation of the second order for this phrase is as follows:

$$(-0.00292)/(s-21.3547) + (0.00292)/(s+21.3547)$$

As a result:

$$GS2 = -42.7094/(s^2 - 458.035401)$$

But we have in the transformation function:

$$\tilde{G}(s) = 2s/s^2 - 458.035401$$

In MATLAB, we implement:

```
close all
clear
clc
syms s;

% Matrix Definition A , B, C, and D
M = 0.5; % Mass Logo
m = 0.2; % Mass Pandle
l = 0.3; % Pendulum Base Length
I = 0.006; % Logo Inertial Moment Value
g = 9.81; % Descending Acceleration

A = [0, 1, 0, 0;
     0, 0, -((M+m)*g*l)/(M+m-m*l^2), 0;
     0, 0, 0, 1;
     0, 0, ((M+m)*g)/(I+m*l^2-M*m*l^2), 0];

B = [0; (m*l)/(M+m-m*l^2); 0; -m*l/(I+m*l^2-M*m*l^2)];

C = [1, 0, 0, 0];
D = [0];

sys = ss(A,B,C,D);

%Convert Function
[num,den] = ss2tf(A,B,C,D)

printsys(num,den)
f=tf(num,den)
figure(1)
```

```
step(f)
```

```
%%%%%%%%%
```

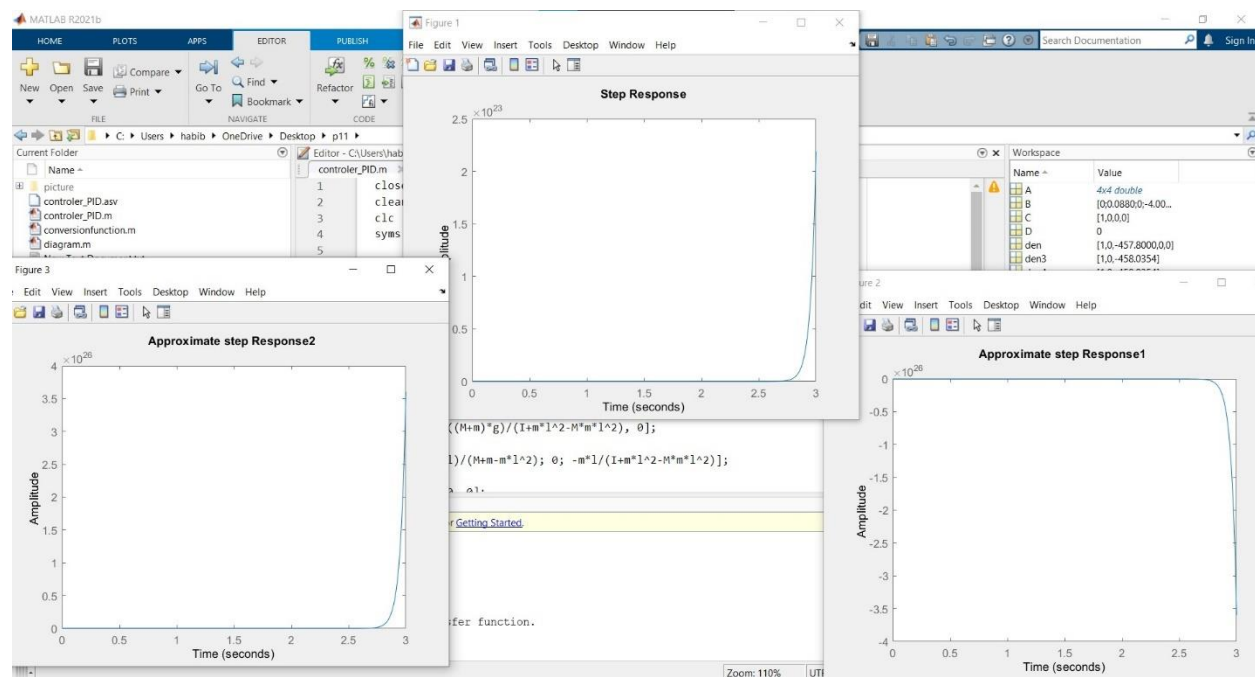
```
%  $G(s) = 0.08798/s^3 + 2s/s^2 - 458.035401$ 
```

```
GS2 = -42.7094/(s^2-458.035401)
num3 = [-42.7094];
den3 = [1,0,-458.035401];
f3 = tf(num3,den3)
figure(2)
step(f3)
title("Approximate step Response1 ")
```

```
%%%%%%%%%
```

```
GS3 = 2*s/(s^2-458.035401)
num4 = [2,0];
den4 = [1,0,-458.035401];
f4 = tf(num4,den4)
figure(3)
step(f4)
title("Approximate step Response2 ")
```

Results:



According to the results of the step response of the second approximate transform function, the system has given the same response as our original transform function, so we control the same.

PID Tuner is one of the most important and powerful tools available in MATLAB software that allows you to automatically adjust the control parameters of the PID controller for a control system. Here's how to work with PID Tuner in MATLAB:

First, you need to implement your control system model in MATLAB. This model can include one or more differential equations that describe the behavior of the system in time.

Then, to set the PID controller control parameters, we create a PIDTuner object using the pidtune command:

```
P = pidtune(sys, 'pid')
```

In this statement, sys is the model of the control system, and 'pid' is the type of PID controller that we want to set.

After running the above command, a window titled "PID Tuner" will open. In this window, you can set the PID controller parameters automatically or manually.

To set the PID controller parameters automatically, you need to press the "Auto Tune" button. In this mode, the PID Tuner automatically adjusts the optimal PID controller parameters for your control system.

To set the PID controller parameters manually, you can directly click on the control system response graph and adjust the KP, KI, and KD parameters using the control bars in the PID Tuner window. Also, you can use the "Tune" and "Update" buttons to adjust the PID controller parameters manually.

After setting the PID controller parameters manually or automatically, the set parameters are applied to the control system using the "Apply" button.

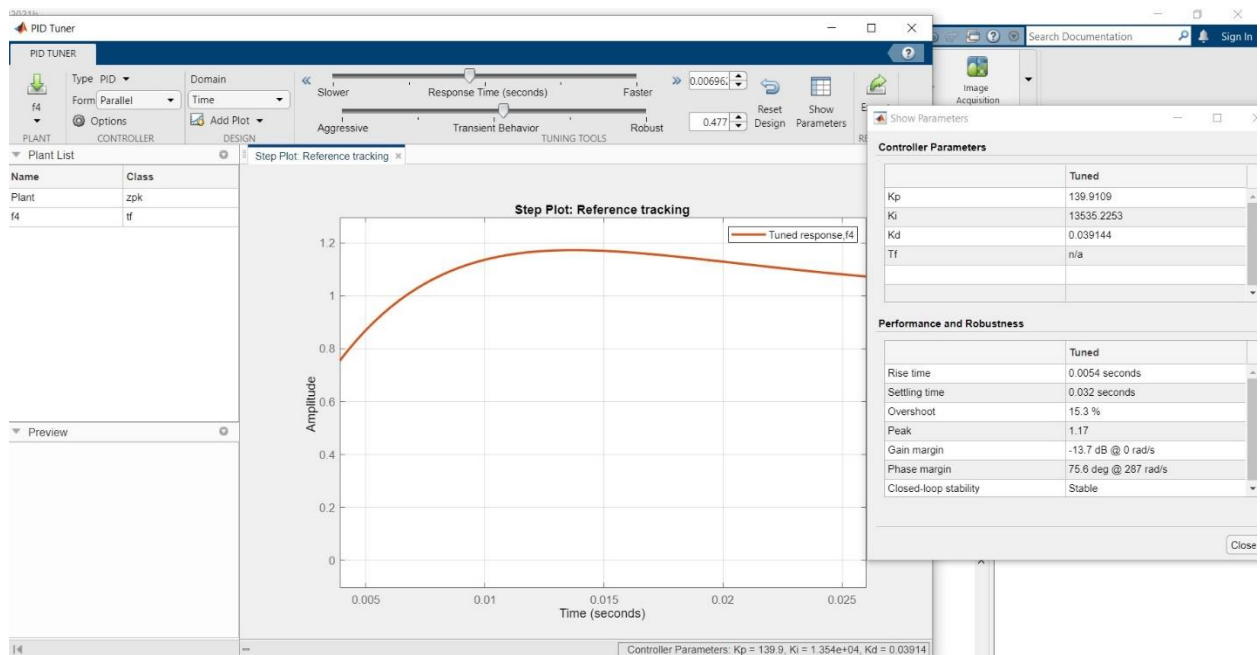
Finally, you can check the performance of your control system with the parameters set by PID Tuner using the commands in MATLAB and enjoy the more optimal performance of your system.

In summary, to set the PID controller parameters using PID Tuner in MATLAB, first implement your control system model, then create a PIDTuner object using the pidtune command, set the PID controller parameters automatically or manually, and finally using the "Apply" button. Apply the set parameters to the control system.

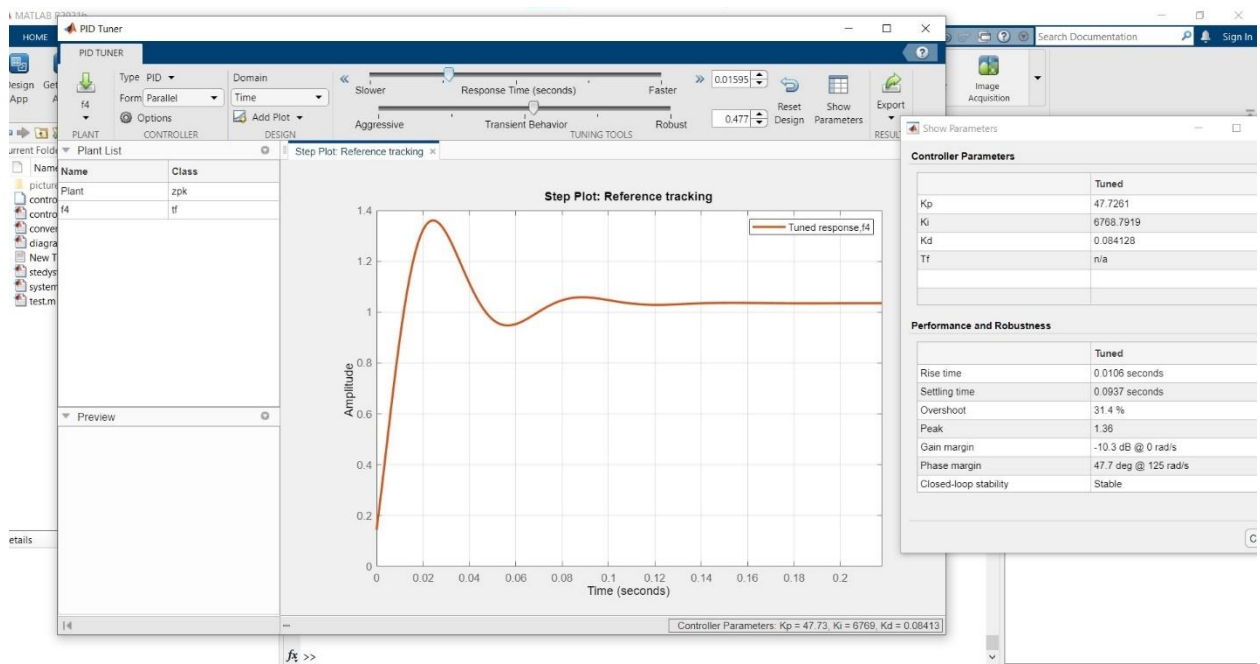
Steps:

- Self Run Tuner by PID Tuner Command
- Log in by import
- Set Controller by pidtool or GUI Command • Select Controller Type by Type
- Select Show Other Quantities such as Troubleshooting by Add Plot Transient Behavior and Response Time
- Set Response Speed and Conditions
- View Values Set by Show Parameters
- Save Controller by [Export](#)

No manual changes:

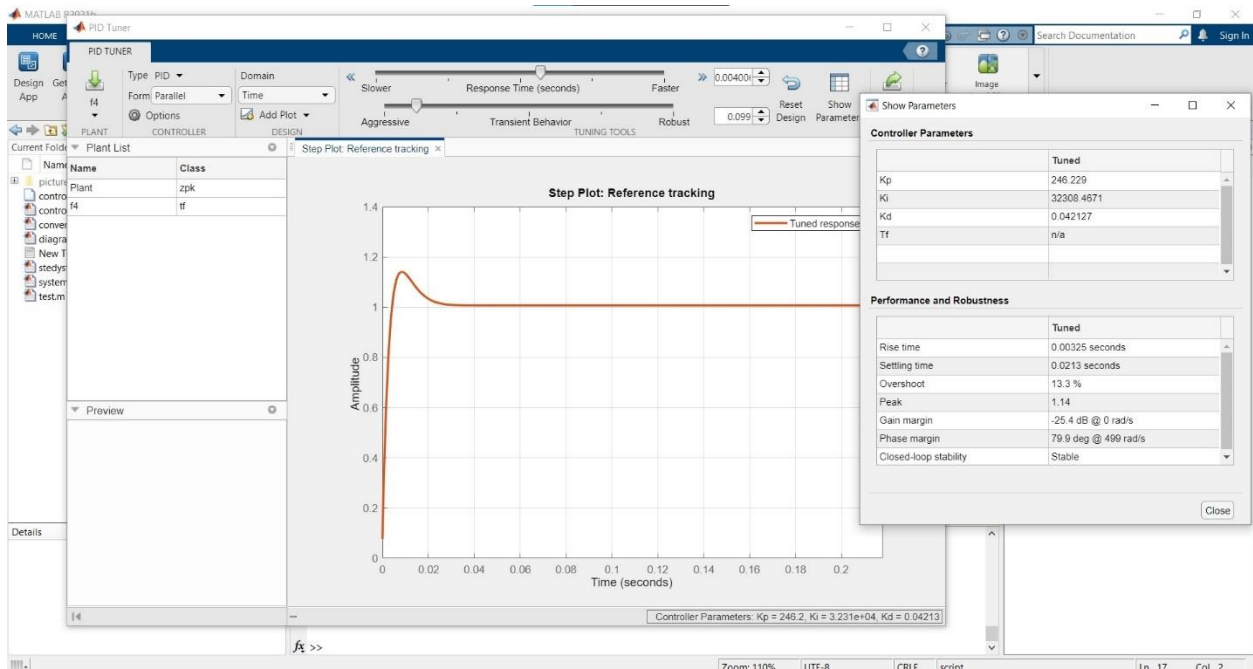


Making changes:

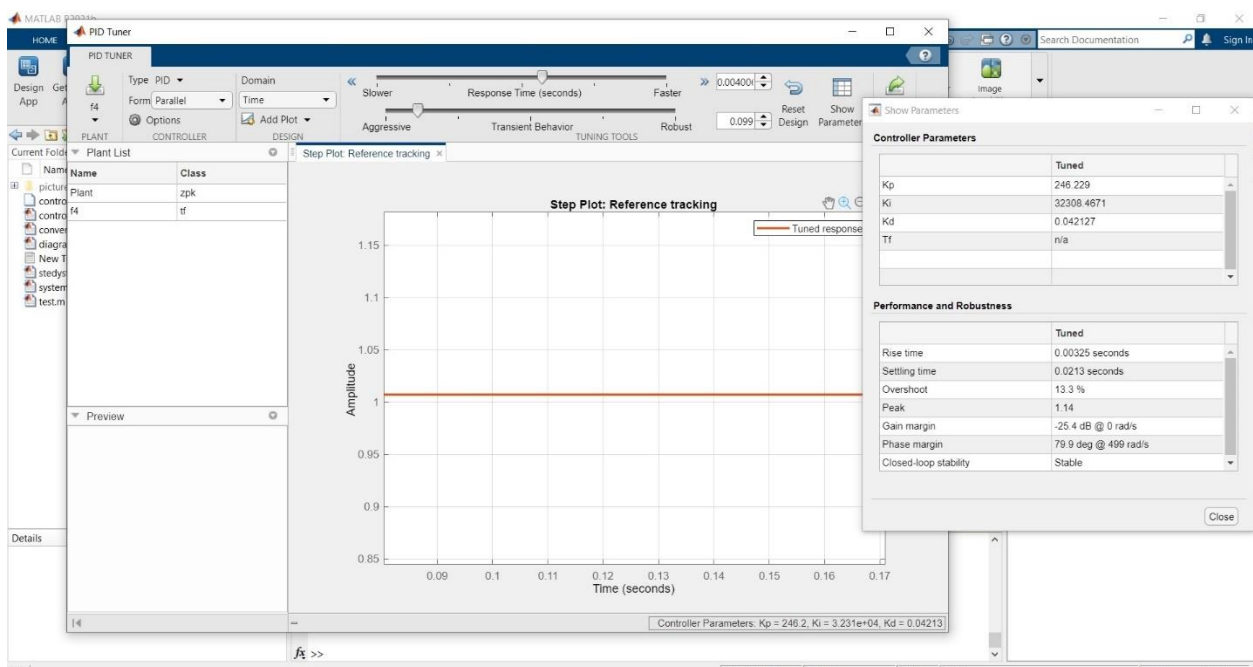


It doesn't have a proper overshoot system

Suitable System:



Session time less than 5 seconds



The persistent state error of θ is less than 5%

A PID controller is a feedback controller that converts a control error input into a control output using three parameters: P, I, and D. The parameters P, I, and D have their own functions in system control, respectively:

P (Proportional): This parameter specifies how the control error (i.e., the distance between the desired and actual value) should be affected at any given moment. As the P value increases, the control output increases and the response to the control error becomes faster.

I (Integral): This parameter calculates the impulse of the control error over time. As the value of I increases, the control error is generally reduced and the stability of the system is improved.

D (Derivative): This parameter takes into account the effect of the change rate of the control error. As the value of D increases, the controller reacts to changes faster and the response to the control error is associated with less fluctuation.

According to the above description, the effect of each of the parameters P, I and D on the control of the PID controller dose system for the pendulum angle is as follows:

P: If the P value is large, the control output will react to the control fault faster, and the reaction to the control error will be faster. But if the P value is too large, the control output will have strong oscillating reactions, which will increase the control error and unstable control.

I: If the value of I is large, the control error will be generally reduced, and the stability of the system will be improved. But if the value of I is too large, the control output will have strong oscillating reactions, which will increase the control error and unstable control.

D: If the D value is large, the controller will react to changes faster, and the control error response will be less oscillating. But if the D value is too large, the control output will have strong oscillating reactions, which will increase the control error and unstable control.

9 . The geometric location of the root locus before applying the controller for the logo and pendulum and after applying the controller

Draw PIDs for the pendulum and compare and analyze the charts. (Calculations manually and graphs using from the MATLAB code)

```
close all
clear
clc
syms s;

% Matrix Definition A , B, C, and D
M = 0.5; % Mass Logo
m = 0.2; % Mass Pandle
l = 0.3; % Pendulum Base Length
I = 0.006; % Logo Inertial Moment Value
g = 9.81; % Descending Acceleration

A = [0, 1, 0, 0;
     0, 0, -((M+m)*g*l)/(M+m-m*l^2), 0;
     0, 0, 0, 1;
     0, 0, ((M+m)*g)/(I+m*l^2-M*m*l^2), 0];

B = [0; (m*l)/(M+m-m*l^2); 0; -m*l/(I+m*l^2-M*m*l^2)];

C = [1, 0, 0, 0];
D = [0];

sys = ss(A,B,C,D);

%Convert Function
[num,den] = ss2tf(A,B,C,D)

printsys(num,den)
f=tf(num,den)
%figure(1)

%step(f)
figure(2)
rlocus(f)

GS3 = 2*s/(s^2-458.035401)
num4 = [2,0];
den4 = [1,0,-458.035401];
f4 = tf(num4,den4)

kp = 246.229;
ki = 32308.4671;
kd = 0.042127;

c = kp + ki*(1/s) + kd*s;
```

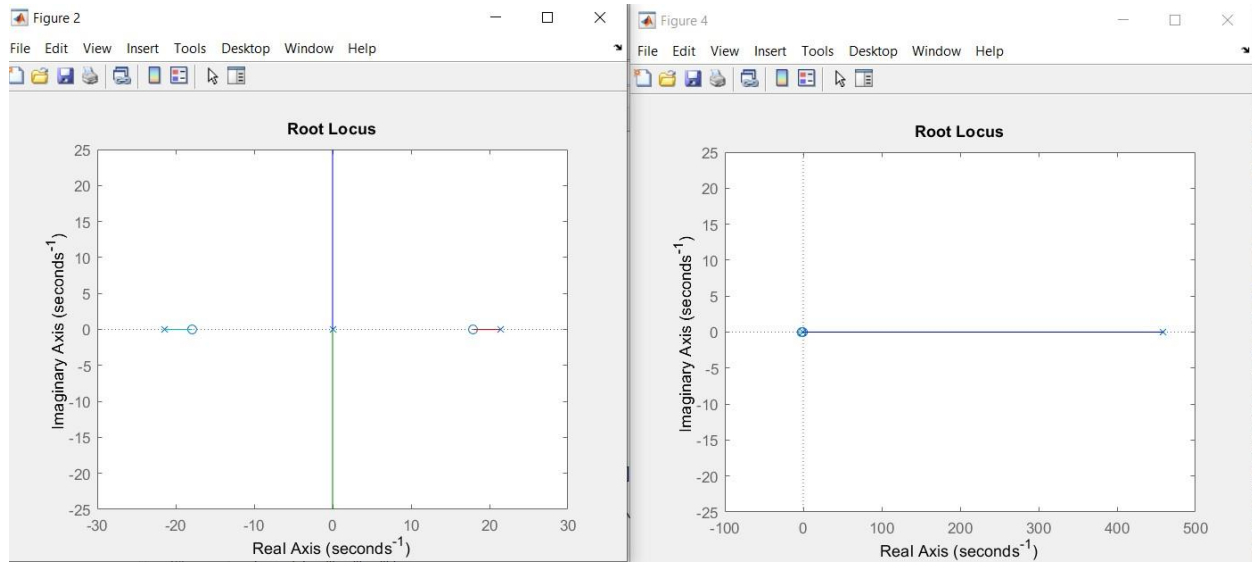
```

GS_final = c*GS3
num5 = [9.375,24.225,8.7156];
den5 = [1,-457.29,-455.38];
f5 = tf(num5,den5)
%figure(3)
% step(f5)

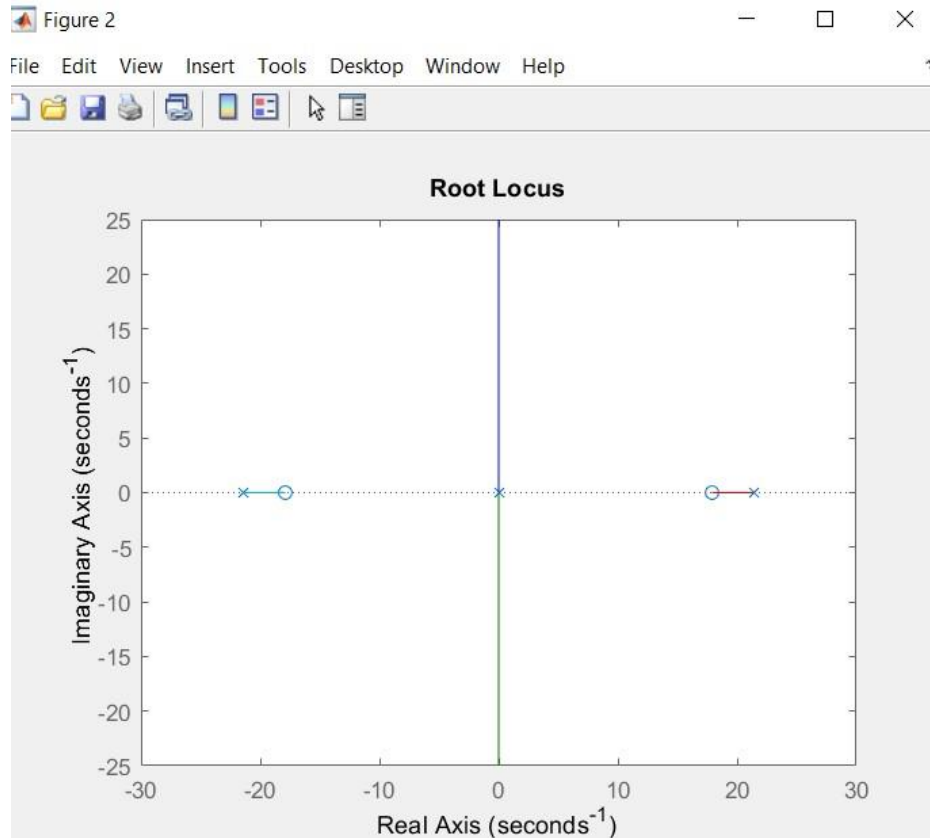
figure(4)
rlocus(f5)

```

Results:



It is plotted before the PID controller is applied and after it is applied to the pendulum. The root locus diagram of the transfer function of the control system without the PID controller is shown in the following figure:

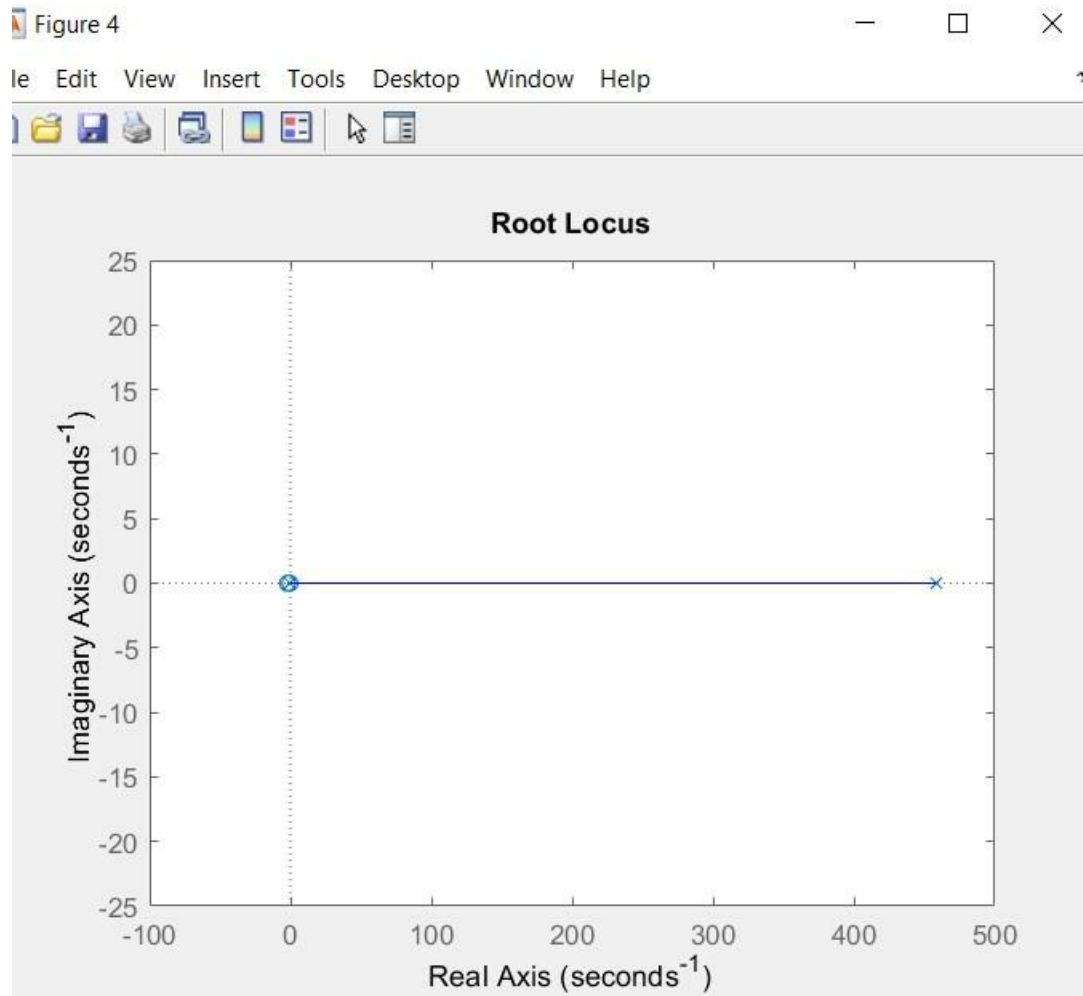


As can be seen in the image, the roots of the transition function are located in the negative range of the real number axis, moving towards the upper range of the frequency coordinates. This shows that the system without a PID controller is unstable, and the system does not rotate around its equilibrium point due to the location of the roots.

By applying the PID controller with coefficients of $k_p = 246.229$, $k_i = 32308.4671$ and $k_d = 0.042127$ and combining it with the previous system transfer function, the final transfer function of the control system will be as follows:

$$2 \cdot (3.75s + 8.07) \cdot (1.25s + 0.54) / s^2 - 457.29s - 455.38$$

The root locus diagram related to the transfer function of the control system using the PID controller, is shown in the following figure:



As can be seen, by applying the PID controller, the roots of the transmission function are located to the lower range of the frequency coordinates, which indicates that the system has stabilized and rotates around its equilibrium point. Further, by comparing the two root locus diagrams, it is clear that by applying the PID controller, the location of the roots is close to a static point, which indicates that the system is more stable and improved with new control conditions.