Magnetic Levitation System

Modern Control Systems

Phase 1

Prof. Kebriaei

December 24, 2022

Shiva Shakeri

Contents

1. Introduction

Magnetic levitation system is known as electro-mechanical systems in which an object is levitate in specific region without using any support. Now a days this technology is spreading over a wide range because of its contact-less and friction-less properties as it removes energy losses which occur due to friction. Since this technology was developed, researchers have analyzed a large number of such systems. In the analysis of such systems most of the work has been done on designing of controller as they require appropriate control action and are more complex, such as State space controller, feedback linearization is used to control the position of ball. However, there are mostly highly non-linear systems and open loop unstable. This nonlinearity and unstability feature of magnetic levitation systems makes their control and modeling very challenging.

The problem of controlling the magnetic field by control method is taken up to levitate a metal hollow sphere, here. The control problem is to supply controlled position of the ball such that the magnetic force on the levitated body and gravitational force acting on it are exactly equal. Thus the magnetic levitation system is inherently unstable without any control action.

The objective of the work can be directed to design controller considering the linear model as well as nonlinear model. The controller designed is validated by simulating and implementing on real time system.

2. Magnetic Levitation System

2.1. Model Diagram

An overview of the general structure of a magnetic levitation system can be found in Figure 1. Basically, an electromagnet and a metal ball make up the core of this system.

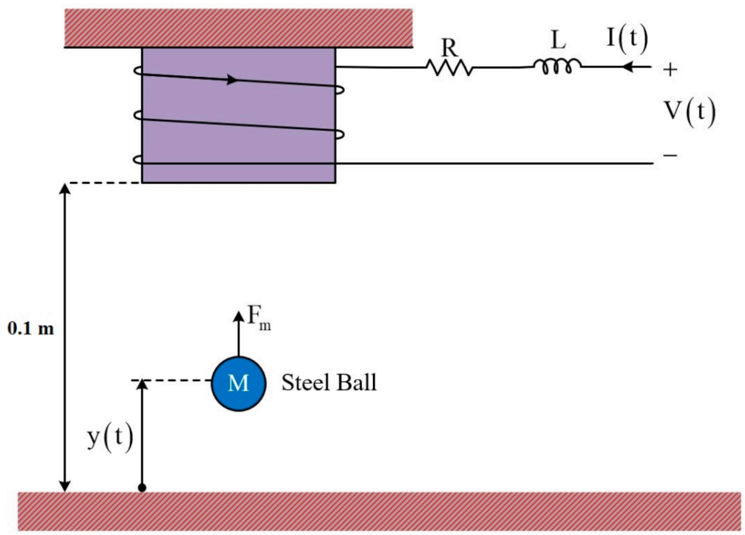

Figure 1. Model of Magnetic Levitation System

2.2. Model Parameters

The circuit's resistance (R) and inductance (L) were measured with ohms meter and inductance meter respectively. In this experiment, the object's mass (M) is measured, placed under the electromagnetic pole on a known distance (y) and current through the electromagnet ($I(t)$) is gradually increased up to the point when it picks up the object. According to table 1, the parameters which were obtained are as follows.

Table 1. MagLev System Model Parameters

R	Circuit Resistance	50Ω
L	Winding Inductance	$0.2H$
g	Gravitational Constant	$9.8\frac{m}{s^2}$
M	Metal Ball Mass	$0.491Kg$
c	Electromagnetic Force Constant	$0.3\frac{N \times m}{A^2}$

f_v	Air Resistance(Friction)	$0.04\frac{N \times s}{m}$
$y^\star$	Output Operating Point	$0.06m$

3. State-Space and Input

In this system, there are three states, one controlling input, and one output. The parameters x_1 as the position of the metal ball, x_2 as the velocity of the metal ball, and x_3 as winding's current are state parameters. The input is shown by parameter u, and presents the input voltage. The input is shown by parameter 'u' which presents the input voltage. The 'y' parameter shows the position of the metal ball as the output of this system.

As a result of writing the dynamic and physical equations for this system, we can find the state equations as follows.

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = -g + \frac{c}{M}\frac{x_3^2}{0.1 - x_1} - \frac{f_v x_2}{M} \\ \dot{x_3} = \frac{1}{L}(-Rx_3 + u) \end{cases} \qquad \{y = [1,0,0]x$$

4. Equilibrium Points

At equilibrium points, the system doesn't change at all, and that means that the parameters of the system should be equal to zero at that point. In order to find the equilibrium points of the system, we set the state equations to zero. Appendix A contains the Matlab code for this part.

$$\begin{cases} x_2 = 0 \\ -g + \frac{c}{M}\frac{x_3^2}{0.1 - x_1} - \frac{f_v x_2}{M} = 0 \\ \frac{1}{L}(-Rx_3 + u) = 0 \end{cases}$$

Then, by rewriting the equations using parameters:

$$\begin{cases} x_2 = 0 \\ -9.8 + \frac{0.3}{0.491}\frac{x_3^2}{0.1-0.06} - \frac{0.04x_2}{0.491} = 0 \\ \frac{1}{0.2}(-50x_3 + u) = 0 \end{cases}$$

By solving the equations, x_1, x_2, and x_3 are:

$$\begin{cases} x_1 = 0.06 \\ x_2 = 0 \\ x_3 = -0.800983 \end{cases} \quad \text{,and} \quad \begin{cases} x_1 = 0.06 \\ x_2 = 0 \\ x_3 = 0.800983 \end{cases}$$

There are two values of x_3, which show the current, meaning that the current can flow clockwise or counterclockwise. The next section of the problem will be solved using the positive value of the current flow.

5. Linearization

In order to determine a state-space equation for the system, the system has to be linearized. Assume that $\dot{x_1} = f(x_1, x_2, x_3)$, $\dot{x_2} = g(x_1, x_2, x_3)$, and $\dot{x_3} = h(x_1, x_2, x_3)$. Appendix A contains the Matlab code for this part. The Jacobian Matrix is:

$$A = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} \\ \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} & \frac{\partial g}{\partial x_3} \\ \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} & \frac{\partial h}{\partial x_3} \end{bmatrix} \rightarrow A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{0.611x_3^2}{(0.1-x_1)^2} & -0.0814 & \frac{1.222x_3}{-0.1-x_1} \\ 0 & 0 & -250 \end{bmatrix}$$

In order to linearize the system, the equilibrium points take place in the Jacobian Matrix.

$$A_1 = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & -24.47 \\ 0 & 0 & -250 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & 24.47 \\ 0 & 0 & -250 \end{bmatrix}$$

Therefore the linearized system is:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & -24.47 \\ 0 & 0 & -250 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} u, \ \dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & 24.47 \\ 0 & 0 & -250 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} u$$

In the next step, the eigenvalues of Jacobin Matrices are calculated.

$$eig(A_1) = \begin{cases} \lambda_1 = -250 \\ \lambda_2 = -15.7, \\ \lambda_3 = 15.61 \end{cases} \qquad eig(A_2) = \begin{cases} \lambda_1 = -250 \\ \lambda_2 = -15.7 \\ \lambda_3 = 15.61 \end{cases}$$

The eigenvalues of the linearized matrix are both real positive and negative values. Therefore, the linearized system is unstable and the equilibrium points for the non-linear system is also unstable.

6. State-Space Analysis

Controllability and observability of control systems are two of the major concepts of modern control system theory. These concepts were introduced by R. Kalman in 1960. The following are some definitions that can be used to describe them. Appendix A contains the Matlab code for this part.

- Controllability: In order to be able to do whatever we want with the given dynamic system under control input, the system must be controllable.
- Observability: In order to see what is going on inside the system under observation, the system must be observable.

6.1. Controllability

As a result of applying the theorem 1, we can determine whether the system is controllable.

Theorem 1. A n-dimensions time-Invariant system,

$$\dot{x}(t) = Ax(t) + Bu(t),$$

is totally controllable, if and only if the column vectors of the controllability matrix,

$$\Phi_c = [B, AB, \dots, A^{n-1}B],$$

span the n-dimensional space.

As a result, the controllability matrix of the system should be constructed as follows: $\Phi_c = [B, AB, A^2B]$ where the matrix $B = [0,0,5]^T$.

$$A_1B = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & -24.47 \\ 0 & 0 & -250 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 \\ -122.35 \\ -1250 \end{bmatrix}$$

$$A_2B = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & 24.47 \\ 0 & 0 & -250 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 \\ 122.35 \\ -1250 \end{bmatrix}$$

$$A_1^2B = \begin{bmatrix} 245 & -0.0814 & -24.47 \\ -19.943 & 245 & 6119.5 \\ 0 & 0 & 62500 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} = \begin{bmatrix} -122.35 \\ 30597.45 \\ 312500 \end{bmatrix}$$

$$A_2^2B = \begin{bmatrix} 245 & -0.0814 & 24.47 \\ -19.943 & 245 & -6119.5 \\ 0 & 0 & 62500 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} = \begin{bmatrix} 122.35 \\ -30597.45 \\ 312500 \end{bmatrix}$$

$$\rightarrow \Phi_{c1} = \begin{bmatrix} 0 & 0 & -122.35 \\ 0 & -122.35 & 30597.45 \\ 5 & -1250 & 312500 \end{bmatrix}, \Phi_{c2} = \begin{bmatrix} 0 & 0 & 122.35 \\ 0 & 122.35 & -30597.45 \\ 5 & -1250 & 312500 \end{bmatrix}$$

By calculating the determinant of the controllability matrices:

$$|\Phi_{c1}| = |\Phi_{c2}| = -74847.61 \neq 0$$

As a result of the *theorem 1*, the system is controllable if and only if $|\Phi_c| \neq 0$. Therefore since the value of $L = 0.2H$, **the system is controllable**.

6.2. Observability

By using the theorem 2, one can determine the observability of a system.

Theorem 2. A n-dimensions time-Invariant system,

$$\dot{x}(t) = Ax(t) + Bu(t),$$

is totally observable if and only if the column vectors of the observability matrix,

$$\Phi_o = [C, CA, \ldots, CA^{n-1}],$$

span the n-dimensional space.

As a result, the controllability matrix of the system should be constructed as follows: $\Phi_o = [C, CA, CA^2]$ where $C = [1 \quad 0 \quad 0]$.

$$CA_1 = [1 \quad 0 \quad 0] \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & -24.47 \\ 0 & 0 & -250 \end{bmatrix} = [0 \quad 1 \quad 0]$$

$$CA_2 = [1 \quad 0 \quad 0] \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & 24.47 \\ 0 & 0 & -250 \end{bmatrix} = [0 \quad 1 \quad 0]$$

$$CA_1^2 = [1 \quad 0 \quad 0] \begin{bmatrix} 245 & -0.0814 & -24.47 \\ -19.943 & 245 & 6119.5 \\ 0 & 0 & 62500 \end{bmatrix} = [245 \quad -0.0814 \quad -24.47]$$

$$CA_2^2 = [1 \quad 0 \quad 0]\begin{bmatrix} 245 & -0.0814 & 24.47 \\ -19.943 & 245 & 6119.5 \\ 0 & 0 & 62500 \end{bmatrix} = [245 \quad -0.0814 \quad 24.47]$$

$$\rightarrow \Phi_{o1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 245 & -0.0814 & -24.47 \end{bmatrix}, \Phi_{o2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 245 & -0.0814 & 24.47 \end{bmatrix}$$

The observability matrix has the rank 3 and its vectors are linearly independent. Therefore the matrix Φ_o spans the n-dimensional space and **the system is always observable**.

6.3. Minimal Realization

The following theorem which is due to Kalman gives a characterization of minimal state space realizations:

Theorem 3. A realization (A,B,C,D) is minimal if and only if it is controllable and observable.

As a result of this theorem, the presented state-space is minimal.

6.4. State Transition Matrix

The state-transition matrix is a matrix whose product with the state vector x at an initial time t_0 gives x at a later time t. The state-transition matrix can be used to obtain the general solution of linear dynamical systems.

First, by solving the homogeneous state equations using $u(t) = 0$: $\dot{x}(t) = Ax(t)$. The final solution of this equation is $x(t) = \exp[A(t - t_0)]x(t_0)$; where t_0 is the initial time and $x(t_0)$ is the initial state vector. The $\exp[A(t - t_0)]$ can be written as follows:

$$e^{A(t-t_0)} = \exp[A(t - t_0)] = I + \frac{A(t - t_0)}{1!} + \frac{A^2(t - t_0)^2}{2!} + \ldots + \frac{A^k(t - t_0)^k}{k!}$$

Therefore, the matrix $\exp[A(t - t_0)]$ is squared and its dimensions equals to the dimensions of matrix A. This matrix is called “state-transition matrix,” and is shown by:

$$\Phi(t) = e^{At} = \exp[At]$$

The state transition matrix can also be calculated by this formula:

$$e^{At} = \mathscr{L}^{-1}[(s\mathbf{I} - A)^{-1}]$$

By applying this formula to the system:

$$e^{A_1 t} = \mathscr{L}^{-1}(\begin{bmatrix} s & -1 & 0 \\ -245 & s+0.0814 & 24.47 \\ 0 & 0 & s+250 \end{bmatrix}^{-1})$$

By using the Sylvester interpolation method to calculate e^{At}:

$$A_1 = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & -24.47 \\ 0 & 0 & -250 \end{bmatrix}, eig(A_1) = \begin{cases} \lambda_1 = -250 \\ \lambda_2 = -15.7 \\ \lambda_3 = 15.61 \end{cases}$$

$$\begin{vmatrix} 1 & \lambda_1 & \lambda_1^2 & e^{\lambda_1 t} \\ 1 & \lambda_2 & \lambda_2^2 & e^{\lambda_2 t} \\ 1 & \lambda_3 & \lambda_3^2 & e^{\lambda_3 t} \\ I & A & A^2 & e^{At} \end{vmatrix} = 0 \rightarrow \begin{vmatrix} 1 & -250 & 62500 & e^{-250t} \\ 1 & -15.7 & 246.49 & e^{-15.7t} \\ 1 & 15.61 & 243.67 & e^{15.61t} \\ I & A & A^2 & e^{At} \end{vmatrix} = 0$$

$1948496.6721e^{(At)} + (265.61A^2e^{-15.7t} - 31.31A^2e^{-250t} - 234.3A^2e^{15.61t} + 62256.33Ae^{-15.7t} - 2.82Ae^{-250t} - (62253.51Ae^{15.61t} - 1036542.5Ie^{-15.7t} + 76733.279Ie^{-250t} - 919627.5Ie^{15.61t} = 0$

This equation can be solved for e^{At}.

7. Transfer Function

The transfer function of an linearized system $\dot{x} = Ax + Bu$, and $y = Cx$ is

$$g(s) = c(sI - A)^{-1}b$$

$$g_1(s) = [1 \quad 0 \quad 0] \begin{bmatrix} s & -1 & 0 \\ -245 & s+0.0814 & 24.47 \\ 0 & 0 & s+250 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} = \frac{-60073.7}{(491s^3 + 122790s^2 - 110295s - 30073750)}$$

$$g_1(s) = [1 \quad 0 \quad 0] \begin{bmatrix} s & -1 & 0 \\ -245 & s+0.0814 & -24.47 \\ 0 & 0 & s+250 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} = \frac{60073.7}{(491s^3 + 122790s^2 - 110295s - 30073750)}$$

The system has no zeros, and the poles of the both transfer functions are -250, -15.69, and 15.69. Appendix A contains the Matlab code for this part.

8. PID Controller

8.1. Introduction

A proportional–integral–derivative controller (PID controller or three-term controller) is a control loop mechanism employing feedback that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an error value as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), hence the name.

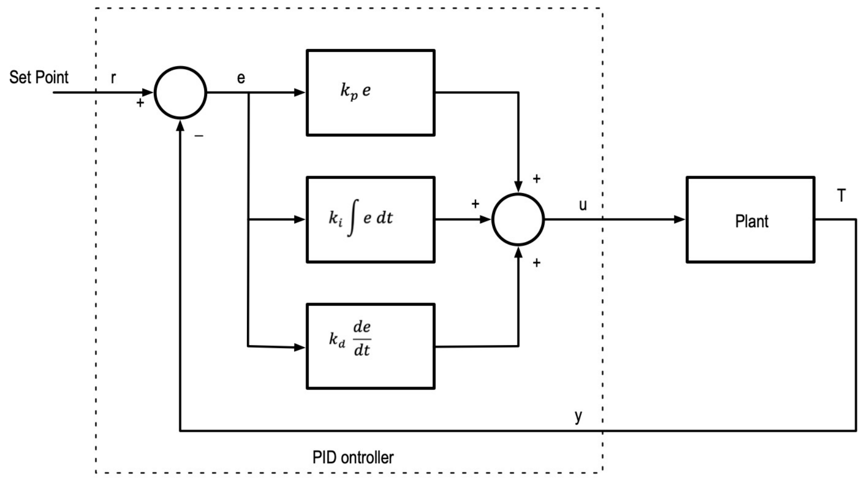

Figure 2. A Block Diagram of a PID Controller

In practical terms, PID automatically applies an accurate and responsive correction to a control function. An everyday example is the cruise control on a car, where ascending a hill would lower speed if constant engine power were applied. The controller's PID algorithm restores the measured speed to the desired speed with minimal delay and overshoot by increasing the power output of the engine in a controlled manner.

The distinguishing feature of the PID controller is the ability to use the three *control terms* of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. The block diagram in Fig. 2 shows the principles of how these terms are generated and applied.

8.2. PID Controller Tuning

First the open-loop(Fig. 3) , using the transfer function $g(s)$, where

$$g_1(s) = \frac{Y(s)}{X(s)} = \frac{-60073.7}{(491s^3 + 122790s^2 - 110295s - 30073750)}$$

$$g_2(s) = \frac{Y(s)}{X(s)} = \frac{60073.7}{(491s^3 + 122790s^2 - 110295s - 30073750)}$$

The step responses of two systems (g1 and g2) is shown in Fig. 4 and Fig. 5 respectively.

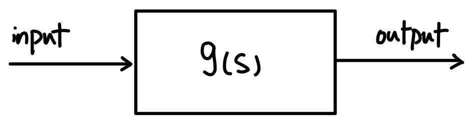

Figure 3. Open-Loop System Diagram

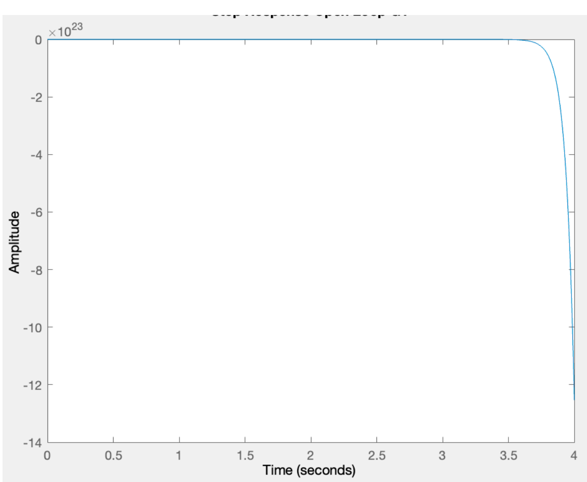

Figure 4. Step Response of G_1

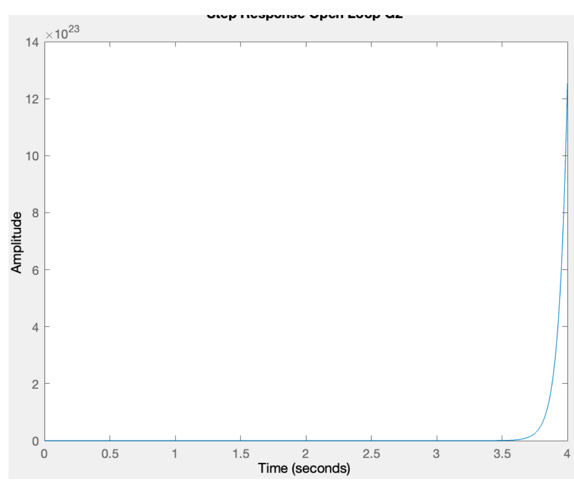

Figure 5. Step Response of G_2

8.2.1. Proportional Control

The proportional controller (K_p) reduces the rise time, increases the overshoot, and reduces the steady-state error. The closed-loop transfer function of our unity-feedback system with a proportional controller is the following:

$$T(s) = \frac{K_pG(s)}{1 + K_pG(s)} = \frac{60037.7 + K_p}{(491s^3 + 122790s^2 - 110295s + (-30073750 + K_p))}$$

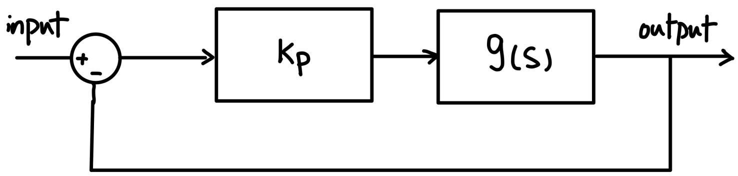

Figure 6. Proportional Controller Diagram

The step response of this system when $K_p = 500$ is shown in Fig 7 and Fig 8.

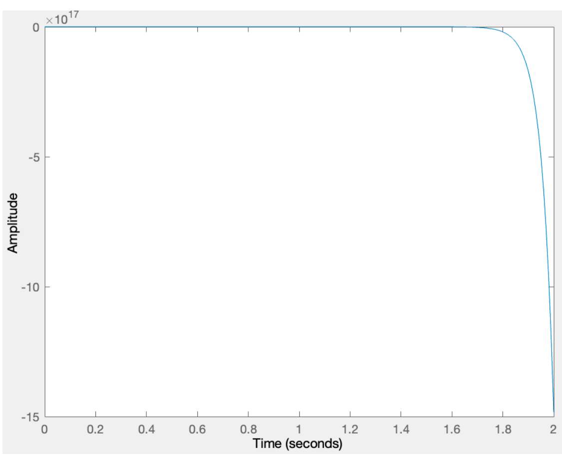

Figure 7. Step Response of K_pG_1

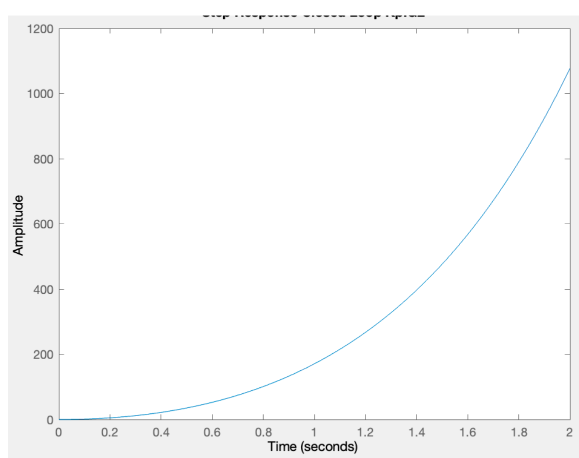

Figure 8. Step Response of K_pG_2

The step response of this system when $K_p = 1600$ for G_2 is shown in Fig 9.

In Fig. 9 it is shown that the range of step response's amplitude is more bounded by choosing the right proportional constant, but as the above figures show, the system is still unstable.

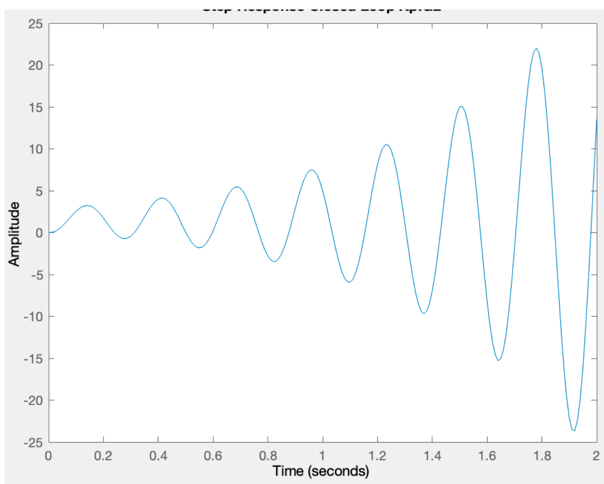

Figure 9. Step Response of $(K_p = 1600)G_2$

8.2.2. Proportional-Derivative Control

Now, let's take a look at PD control. The addition of derivative control (K_d) tends to reduce both the overshoot and the settling time. The closed loop transfer function of the given system with a PD controller is:

$$T_3(s) = \frac{(K_d s + K_p)G(s)}{1 + (K_d s + K_p)G(s)} = \frac{K_d s + (60073.7 + K_p)}{(491s^3 + 122790s^2 + (-11029 + K_d)s + (-30073750 + K_p)}$$

The step response of this system when $K_p = 1600$ and $K_d = 500$ for G_2 is shown in Fig. 10. The stepinfo() in this system will be in table 2.

Table 2. Step Info T_3

RiseTime	0.5010
SettlingTime	1.2261
SettlingMin	1.3100
SettlingMax	1.4549
Overshoot	0
Undershoot	0
Peak	1.4549
PeakTime	3.0464

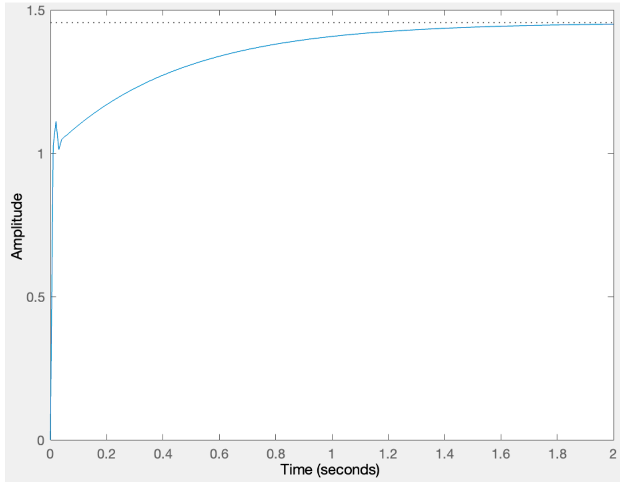

Figure 10. Step Response of T_3

8.2.4. Proportional-Integral-Derivative Control

Now, let's examine PID control. The closed-loop transfer function of the given system with a PID controller is:

$$T_4(s) = \frac{(K_d s + K_p)G(s)}{1 + (K_d s + K_p)G(s)} = \frac{K_d s + (60073.7 + K_p)}{(491s^3 + 122790s^2 + (-11029 + K_d)s + (-30073750 + K_p)}$$

By setting the PID constants $K_p = 1600$, $K_i = 2600$, and $K_d = 245$, the step response of the system is shown in Fig. 11.

The stepinfo() in this system will be in table 3.

Table 3. Step Info T_4

RiseTime	0.0117
SettlingTime	1.1550
SettlingMin	0.9167
SettlingMax	1.3107
Overshoot	31.0718
Undershoot	0
Peak	1.3107
PeakTime	0.3212

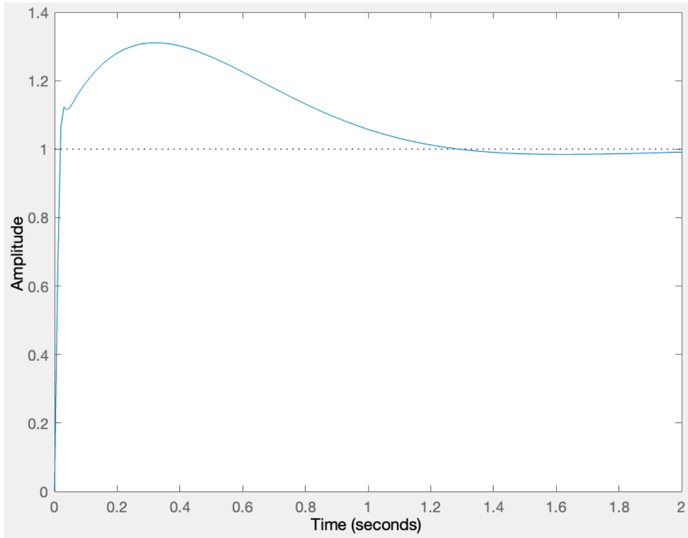

Figure 11. Step Response of T_4

Rising time, settling time, and steady-state error are small, but there exists a large value of overshoot.

9. Non-Linear Model Results in Simulink

The non-linear model of the Magnetic Levitation System is designed in Simulink and it's shown in Fig. 12. The parameters are assigned by the values in Table. 1.

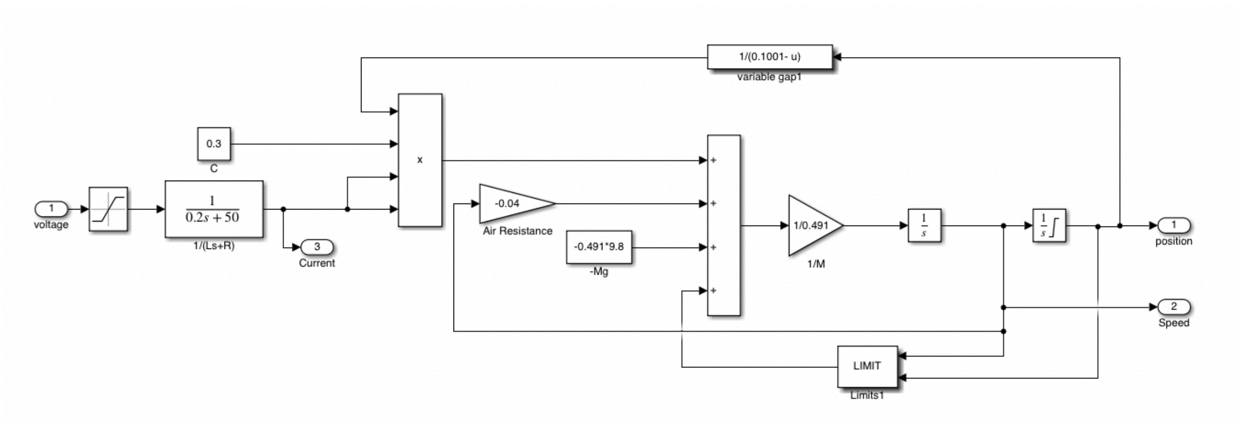

Figure 12. Non-Linear Model of the Magnetic Levitation System Diagram

The completed system is shown in Fig. 13. The input is considered as step and the 3D simulation of ball transition in the system is in the output to show better vision of this system.

First, the ball's position can be seen in Fig. 13, where the ball's position is 0 based on the definition of output in section 2.

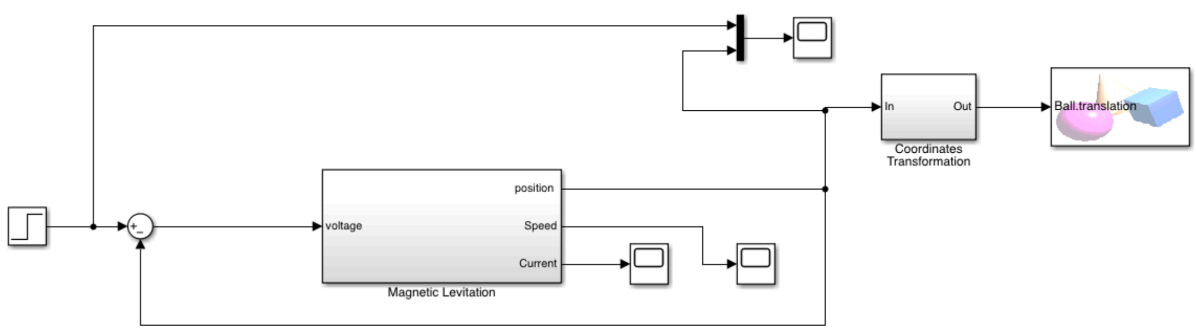

Figure 13. Magnetic Levitation System Diagram

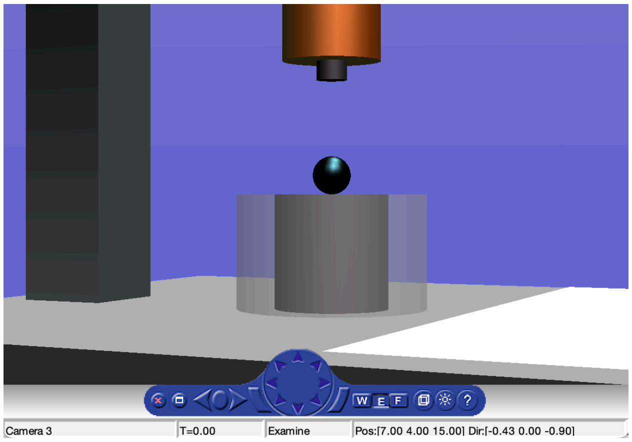

Figure 14. Ball’s Position T=0.00

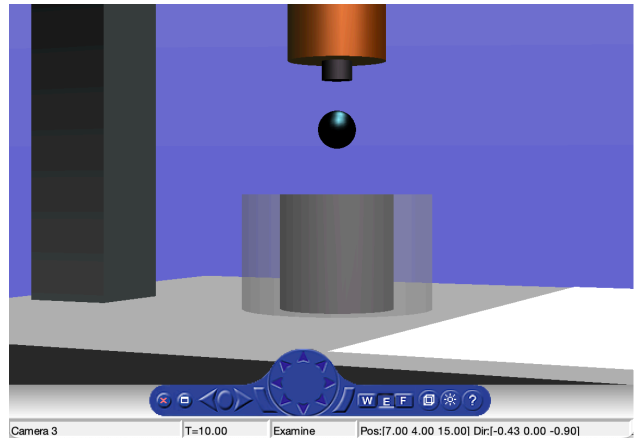

Figure 17. Ball’s Position T=10.00

In the next step the PID controller is added to the system(Fig. 15.) The PID block in Matlab is added and is tuned as Fig. 16. By adding the PID controller to the system, the ball moves upward and stays stable at the position shown in Fig. 17.

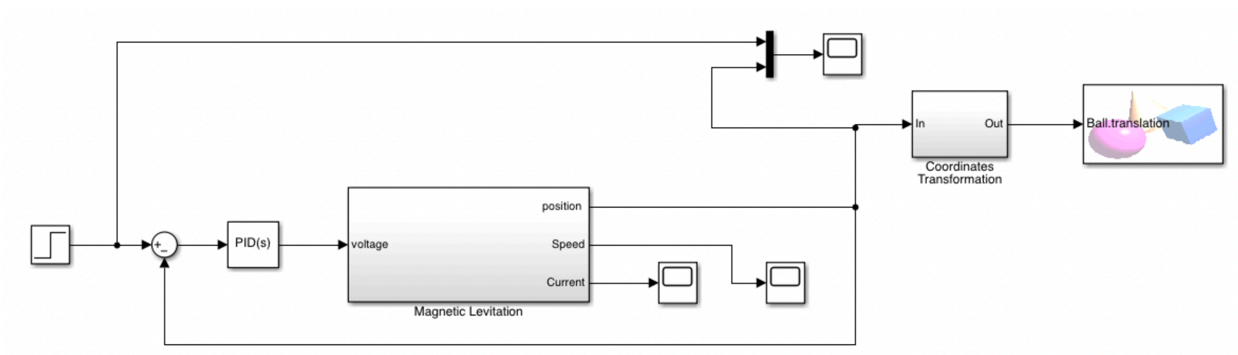

Figure 15. Magnetic Levitation System Diagram and PID Controller

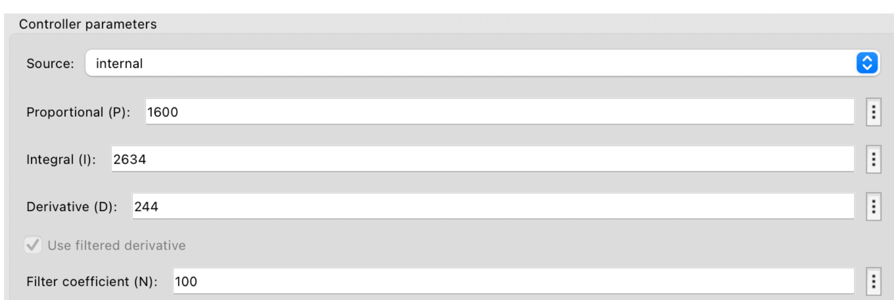

Figure 16. PID Controller Parameters

Next, the step responses of position, velocity, and current(states) are shown in Fig. 18, Fig. 19, and Fig. 20 respectively.

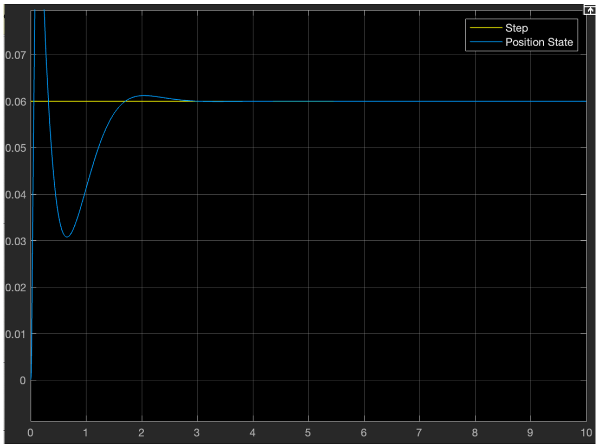

Figure 18. Ball's Position Step Response

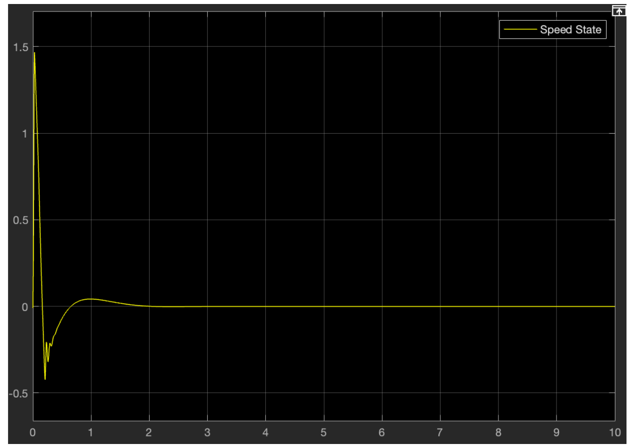

Figure 19. Ball's Speed Step Response

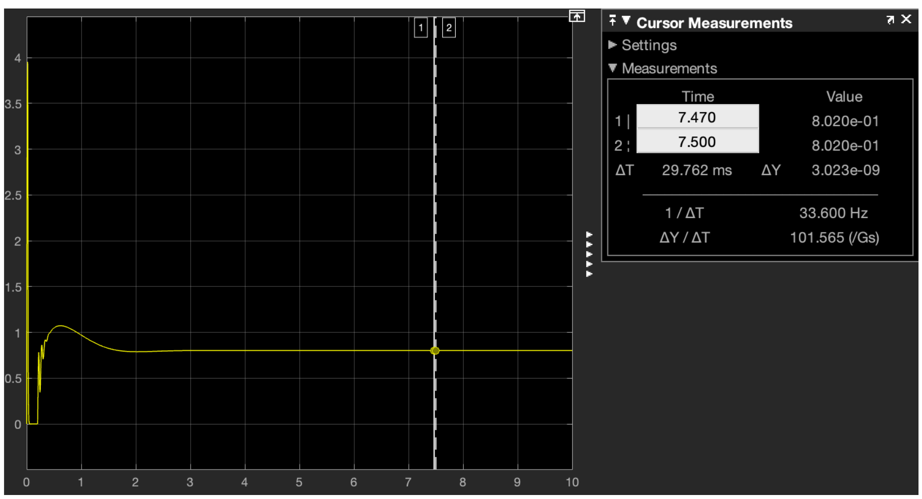

Figure 20. Winding's Current Step Response

In section 4, the equilibrium points were calculated. The stable values of x_1 as the ball's position was 0.06, x_2 was 0, and x_3 as the winding's current was 0.8009. The above figures confirm that the system with the designed PID controller will have those values when it's stable.

Appendix A.

Matlab codes of project sections exist in this appendix.

1. Equilibrium Points (4)

```
syms x1 x2 x3 u

R = 50;
L = 0.2;
g = 9.8;
M = 0.491;
c = 0.3;
fv = 0.04;

eq1 = x2 == 0;
eq2 = x1 == 0.06;
eq3 = -g+(c/M)*(x3^2/(0.1-x1))-(fv*x2/M) == 0;
eq4 = (1/L)*(-R*x3+u) == 0;

stateEqs = [eq1, eq2, eq3, eq4];
sol = solve(stateEqs);

sol.x1
sol.x2
sol.x3
```

```
ans =

3/50
3/50


ans =

0
0


ans =

-(7*7365^(1/2))/750
 (7*7365^(1/2))/750
```

2. Linearization (5)

```
f = x2;
g = -g+(c/M)*(x3^2/(0.1-x1))-(fv*x2/M);
h = (1/L)*(-R*x3+u);

A = jacobian([f, g, h], [x1, x2, x3]);
A

A_1 = subs(A, {x1, x2, x3}, {xe1(1), xe2(1),xe3(1)})
A_2 = subs(A, {x1, x2, x3}, {xe1(2), xe2(2),xe3(2)})

eigA1 = eig(A_1)
eigA2 = eig(A_2)
```

```
A =

[                          0,      1,                          0]
[(300*x3^2)/(491*(x1 - 1/10)^2), -40/491, -(600*x3)/(491*(x1 - 1/10))]
[                          0,      0,                       -250]


A_1 =

[  0,      1,                    0]
[245, -40/491, -(140*7365^(1/2))/491]
[  0,      0,                 -250]


A_2 =

[  0,      1,                   0]
[245, -40/491, (140*7365^(1/2))/491]
[  0,      0,                -250]


eigA1 =

                          -250
- (3*6562805^(1/2))/491 - 20/491
  (3*6562805^(1/2))/491 - 20/491


eigA2 =

                          -250
- (3*6562805^(1/2))/491 - 20/491
  (3*6562805^(1/2))/491 - 20/491
```

3. State-Space Analysis (6)

```
% Controllability
% We can also use the command "ctrb(A, B)"
B = [0; 0; 5];

A_1B = A_1 * B;
A_2B = A_2 * B;

A_12B = A_1^2 * B;
A_22B = A_2^2 * B;

Phi_c1 = [B A_1B A_12B]
Phi_c2 = [B A_2B A_22B]

det_Phi_c1 = det(Phi_c1)
det_Phi_c2 = det(Phi_c2)

% Observability
% We can also use the command "obsv(A, C)"
C = [1 0 0];

A_1C = C * A_1;
A_2C = C * A_2;

A_12C = C * A_1^2;
A_22C = C * A_2^2;

Phi_o1 = [C; A_1C; A_12C]
Phi_o2 = [C; A_2C; A_22C]

rank_Phi_o1 = rank(Phi_o1)
rank_Phi_o2 = rank(Phi_o2)

% State Transition Matrix
syms s

eA1t = ilaplace(inv((s*eye(size(A_1,1))-A_1)))
eA2t = ilaplace(inv((s*eye(size(A_2,1))-A_2)))
```

```
Phi_c1 =

[0,                     0,        -(700*7365^(1/2))/491]
[0, -(700*7365^(1/2))/491, (85953000*7365^(1/2))/241081]
[5,                 -1250,                       312500]


Phi_c2 =

[0,                    0,          (700*7365^(1/2))/491]
[0, (700*7365^(1/2))/491, -(85953000*7365^(1/2))/241081]
[5,                -1250,                        312500]


det_Phi_c1 =

-36750000/491


det_Phi_c2 =

-36750000/491


Phi_o1 =

[  1,       0,                      0]
[  0,       1,                      0]
[245, -40/491, -(140*7365^(1/2))/491]


Phi_o2 =

[  1,       0,                     0]
[  0,       1,                     0]
[245, -40/491, (140*7365^(1/2))/491]


rank_Phi_o1 =

     3


rank_Phi_o2 =

     3
```

```
eA1t =

[exp(-(20*t)/491)*(cosh((3*6562805^(1/2)*t)/491) + (4*6562805^(1/2)*sinh((3*6562805^(1/2)*t)/491))/3937683),                              (491*6562805^(1/2)*exp(-(20*t)/491)*sinh((3*6562805^(1/2)*t)/491))/19688415,        (4*7365^(1/2)*exp(-(20*t)/491)*(cosh((3*6562805^(1/2)*t)/491) - (8182*6562805^(1/2)*sinh((3*6562805^(1/2)*t)/491))/1312561))/873063 - (4*7365^(1/2)*exp(-250*t))/873063]
[                          (24059*6562805^(1/2)*exp(-(20*t)/491)*sinh((3*6562805^(1/2)*t)/491))/3937683, exp(-(20*t)/491)*(cosh((3*6562805^(1/2)*t)/491) - (4*6562805^(1/2)*sinh((3*6562805^(1/2)*t)/491))/3937683), (1000*7365^(1/2)*exp(-250*t))/873063 - (1000*7365^(1/2)*exp(-(20*t)/491)*(cosh((3*6562805^(1/2)*t)/491) - (8353*6562805^(1/2)*sinh((3*6562805^(1/2)*t)/491))/328140250))/873063]
[                                                                                      0,
0,
exp(-250*t)]


eA2t =

[exp(-(20*t)/491)*(cosh((3*6562805^(1/2)*t)/491) + (4*6562805^(1/2)*sinh((3*6562805^(1/2)*t)/491))/3937683),                              (491*6562805^(1/2)*exp(-(20*t)/491)*sinh((3*6562805^(1/2)*t)/491))/19688415,        (4*7365^(1/2)*exp(-250*t))/873063 - (4*7365^(1/2)*exp(-(20*t)/491)*(cosh((3*6562805^(1/2)*t)/491) - (8182*6562805^(1/2)*sinh((3*6562805^(1/2)*t)/491))/1312561))/873063]
[                          (24059*6562805^(1/2)*exp(-(20*t)/491)*sinh((3*6562805^(1/2)*t)/491))/3937683, exp(-(20*t)/491)*(cosh((3*6562805^(1/2)*t)/491) - (4*6562805^(1/2)*sinh((3*6562805^(1/2)*t)/491))/3937683), (1000*7365^(1/2)*exp(-(20*t)/491)*(cosh((3*6562805^(1/2)*t)/491) - (8353*6562805^(1/2)*sinh((3*6562805^(1/2)*t)/491))/328140250))/873063 - (1000*7365^(1/2)*exp(-250*t))/873063]
[                                                                                      0,
0,
exp(-250*t)]
```

4. Transfer Function (7)

```
g1 = C * (inv((s*eye(size(A_1,1))-A_1))) * B
g2 = C * (inv((s*eye(size(A_2,1))-A_2))) * B

poleq = 491*s^3 + 122790*s^2 - 110295*s - 30073750;
solve(poleq, s)
```

```
g1 =

(700*7365^(1/2))/(- 491*s^3 - 122790*s^2 + 110295*s + 30073750)


g2 =

-(700*7365^(1/2))/(- 491*s^3 - 122790*s^2 + 110295*s + 30073750)


ans =

                          -250
- (3*6562805^(1/2))/491 - 20/491
  (3*6562805^(1/2))/491 - 20/491
```

5. PID Controller

```
G1 = -60073.7 / (491*s^3 + 122790*s^2 - 110295*s - 30073750);
G2 = 60073.7 / (491*s^3 + 122790*s^2 - 110295*s - 30073750);

t = 0:0.01:2;
figure;
step(G1, t)
title('Step Response Open Loop G1')

figure;
step(G2, t)
title('Step Response Open Loop G2')
% P
Kp = 1600;
C = pid(Kp);

T1 = feedback(C*G1,1);
T2 = feedback(C*G2,1);

figure;
step(T1,t)
title('Step Response Closed Loop Kp.G1')
figure;
step(T2,t)
title('Step Response Closed Loop Kp.G2')
%PD
Kp = 1600;
Kd = 500;

C = pid(Kp,0,Kd);
T3 = feedback(C*G2,1);
figure;
stepinfo(T3)
step(T3,t)
title('Step Response Closed Loop Kd.Kp.G2')
%PID
Kp = 1600;
Ki = 2600;
Kd = 245;

C = pid(Kp,Ki,Kd);
T4 = feedback(C*G2,1);
figure;
stepinfo(T4)
step(T4,t)
title('Step Response Closed Loop Kd.Ki.Kp.G2')
```