Control Systems - II

Dynamic Modelling & Control of a Magnetic Levitating System

Submitted To:

Prof. Dr. Abdullah Sheeraz

Submitted By:

Muhammad Haider Ali

Department of Mechatronics & Control Engineering, University of Engineering & Technology, Lahore

January 09, 2025

I. Introduction:

Magnetic levitation (maglev) systems are sophisticated electromechanical arrangements that facilitate the suspension of objects on a non-contact basis using the repulsive or attractive forces produced by magnetic fields. These systems find extensive application in uses from high-speed trains to precision manufacturing and experimental physics because they can eliminate mechanical friction and offer high stability and responsiveness.

This project involves modeling and analysis of a vertical magnetic levitation system where a ferromagnetic ball is suspended under an electromagnet. The main goal is to keep the ball's position at a specified reference point by regulating the voltage input to the electromagnet. The system's inherent nonlinearity and open-loop instability make it an interesting topic for dynamic analysis and control system design.

To study the behavior of the maglev system, a nonlinear state-space model is obtained from first principles based on Newton's laws of motion and Kirchhoff's laws of electrical circuits. There are three state variables in the system: the vertical position of the ball, the velocity of the ball, and the coil current. The input to the system is the voltage applied to the coil, and the output is the ball's position. These equations provide the basis for additional control design, simulation, and experimental verification.

The report includes derivation of the state-space equations, the underlying physical principles, and a basis for future implementation of control measures to stabilize the levitating object.

II. Magnetic Levitation System:

Block Diagram:

The simplest overview of a magnetic levitation system can be illustrated in Fig:1.1. An electromagnet and a steel ball are the primary two components of the system. The electromagnet is usually made of a ferromagnetic core.

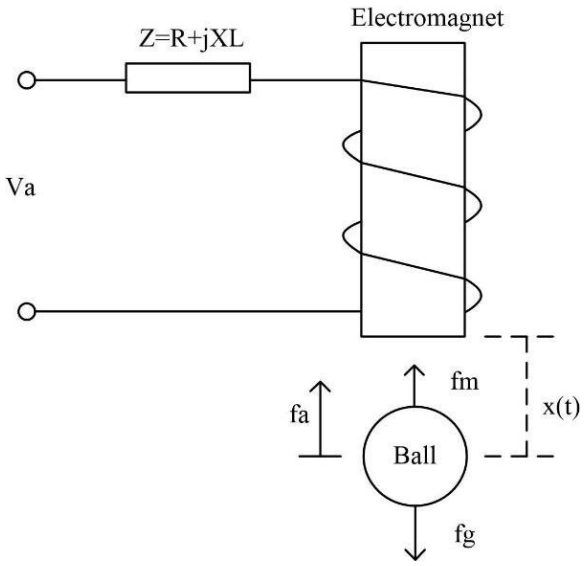

Fig1.1: Maglev System Block Diagram

Model Parameters:

Following are the sample parameters which are used for the analysis of the model. The resistance (R) and inductance (L) are measured. The object's mass (M) is taken which is placed under the electromagnetic pole on a known distance (y) and current through the electromagnet is gradually increased up to the point where it picks up the object. According to Table1.1, the parameters are given as follows:

R	Electrical Resistance	50 Ω
L	Winding Inductance	0.2H
g	Gravitational Const.	$9.8\, m/s^2$
M	Steel Ball Mass	0.491kg
c	Electromagnetic Force Const.	$\frac{0.3\, N \cdot m}{A^2}$
f_v	Air Resistance (Friction)	$\frac{0.04\, N \cdot s}{m}$
y*	Output Operating Point	0.06m

Table1.1: Maglev Model Parameters

State Space Equations:

The dynamic system equations are derived using Newton's second law for translational motion and Kirchhoff's voltage law for the electrical subsystem.

In this system, there are three states, one controlling input and one output. Let the state variables be defined: x1 as position of the ball (m), x2 as velocity of the ball (m/s), x3 as current in the

electromagnetic coil (A). The input of the system is voltage to the coil (V). The output of the system is y = x1 (The ball's position).

Kinematic Equation:

The rate of change of position is the velocity:

$$\dot{x_1} = x_2$$

Mechanical Dynamics:

Using Newton's second law:

$$Mx_1'' = F_{mag} - Mg - f_v x_1'$$

Where, F_{mag} is the upward force exerted by the coil.

The magnetic force is modeled as:

$$F_{\text{mag}} = \frac{cx_3^2}{h - x_1}$$

Substitute into Newton's law and divide by M:

$$\dot{x_2} = -g + \frac{c}{M} \cdot \frac{x_3^2}{h - x_1} - \frac{f_v}{M} x_2$$

Electrical Dynamics:

From Kirchhoff's voltage law to the coil circuit and as the electromagnet coil behaves like an **RL circuit**:

$$u = L\dot{x_3} + Rx_3$$

Solving for $\dot{x_3}$,

$$\dot{x_3} = \frac{1}{L}(u - Rx_3)$$

As a result of writing the dynamic and physical equations for this system, we found the state equations.

Equilibrium Points:

At the **equilibrium points** of the system, the state variables (position, velocity, and current) do not change over time. This means that the **derivatives** of these state variables with respect to time ($\dot{x_1},\ \dot{x_2},\ \dot{x_3}$) must all be **zero**. These conditions can be used to solve the equilibrium values of the system parameters. Appendix A contains the MATLAB code for this section.

$$\dot{x_1} = x_2 \quad \Rightarrow \quad x_2 = 0$$

$$0 = -g + \frac{c}{M} \cdot \frac{x_3^2}{h - x_1} \quad \Rightarrow \quad g = \frac{c}{M} \cdot \frac{x_3^2}{h - x_1}$$

$$0 = \frac{1}{L}(u - Rx_3) \quad \Rightarrow \quad u = Rx_3$$

By putting the values from Table1.1. And solving for x1, x2 and x3. We get:

$$x_1 = 0.06;\ x_2 = 0;\ x_3 = \pm 0.800983$$

There are two values of x3, 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.

There are two values of x3, which means that the current can flow in either clockwise or anticlockwise direction. In the next section, we will solve using the positive value of the current.

III. Linearization:

To analyze the stability of the system, we first need to linearize the nonlinear state-space model near the equilibrium points. This entails finding the Jacobian matrix at the equilibrium points, which represents the system dynamics around these points. The behavior of the system at equilibrium can be explored by computing the eigenvalues of the Jacobian matrix, and it tells us whether the equilibrium is stable or not. Appendix A contains the MATLAB code for this part.

Jacobian Matrix:

The Jacobian matrix is defined as the matrix of partial derivatives of the system's state equations with respect to the state variables. Given the nonlinear state-space equations:

$$\dot{x_1} = x_2$$

$$\dot{x_2} = -g + \frac{c}{M} \cdot \frac{x_3^2}{h - x_1} - \frac{f_v}{M} x_2$$

$$\dot{x_3} = \frac{1}{L}(u - Rx_3)$$

Let the state vector be x = $\begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix}$, and the input be u.

The Jacobian matrix A is a 3×3 matrix that consists of the partial derivatives of the right-hand sides of the equations with respect to x1, x2 and x3. The Jacobian matrix A is defined as:

$$A = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} \\ \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} & \frac{\partial g}{\partial x_3} \\ \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} & \frac{\partial h}{\partial x_3} \end{bmatrix}$$

Where,

$$f(x_1, x_2, \dot{x_3}) = x_1 = x_2$$

$$g(x_1, x_2, \dot{x_3}) = x_2 = -g + \frac{c}{M} \cdot \frac{x_3^2}{h - x_1} - \frac{f_v}{M} x_2$$

$$h(x_1, x_2, x_3) = \dot{x_3} = \frac{1}{L}(u - Rx_3)$$

Calculating the Jacobian Matrix at Equilibrium Points:

At the equilibrium points, we set the derivatives to zero. This gives us the equilibrium values of the state variables:

$$x_1 = 0.06;\ x_2 = 0;\ x_3 = \pm 0.800983$$

The Jacobian matrix evaluated at these equilibrium points is:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{c}{M} \cdot \frac{x_3^2}{(h - x_1)^2} & -\frac{f_v}{M} & \frac{2cx_3}{M(-h - x_1)} \\ 0 & 0 & -R/L \end{bmatrix}$$

Evaluating the Jacobian matrix at the equilibrium points $x_3 = \pm 0.800983$

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

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

Therefore, the linearized system is:

$$\dot{x_1} = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & -24.47 \\ 0 & 0 & -250 \end{bmatrix} \text{x} + \begin{bmatrix} 0 \\ 0 \\ 1/L \end{bmatrix} \text{u}$$

$$\dot{x_1} = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & 24.47 \\ 0 & 0 & -250 \end{bmatrix} \text{x} + \begin{bmatrix} 0 \\ 0 \\ 1/L \end{bmatrix} \text{u}$$

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

Eigenvalue Calculation:

Once the Jacobian matrices are computed, the eigenvalues of the matrix are found to determine the stability of the system. The eigenvalues for the two cases are both real and mixed (positive and negative), indicating that the linearized system is unstable.

The eigenvalues λ are calculated by solving the characteristic equation:

$$\det(\text{A} - \lambda\text{I}) = 0$$

where, I is the identity matrix.

$$\text{eig}(\text{A}_1) = \begin{matrix} \lambda_1 = -250 \\ \lambda_2 = -15.7 \\ \lambda_3 = 15.61 \end{matrix} \qquad eig(A_2) = \begin{matrix} \lambda_1 = -250 \\ \lambda_2 = -15.7 \\ \lambda_3 = 15.61 \end{matrix}$$

The **eigenvalues** of the linearized system are real, with both positive and negative values, indicating that the equilibrium points for the nonlinear system are **unstable**.

IV. State-Space Analysis:

Controllability and Observability:

Controllability and observability are basic notions in contemporary control theory, which were introduced by R. Kalman in 1960. These notions tell us how well we can control or observe the internal states of a system. MATLAB code for this section is given in Appendix A.

Controllability:

Definition: A system is *controllable* if it is possible to move the system from any initial state to any desired final state within finite time using an appropriate input.

We use **Theorem 1** to determine controllability:

Theorem 1: A time-invariant system

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

is controllable if the controllability matrix

$$\Phi_c = [B, AB, A^2B, \dots, A^{n-1}B]$$

has full rank (i.e., rank n for an n-dimensional system).

For our linearized system, with input matrix

$$\mathrm{B} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$$

For Matrix A_1:

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

$$A_1^2B = A_1A_1B = \begin{bmatrix} -122.35 \\ 30597.45 \\ 312500 \end{bmatrix}$$

$$\Phi_{c_1} = \begin{bmatrix} 0 & 0 & -122.35 \\ 0 & -122.35 & 30597.45 \\ 5 & -1250 & 312500 \end{bmatrix}$$

$$\det\left(\Phi_{c_1}\right) = -74847.61 \neq 0$$

Since the determinant is non-zero, the system is **controllable**.

Observability:

Definition: A system is *observable* if it is possible to determine the internal state of the system using only the output measurements over time.

Theorem 2: A time-invariant system

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

is observable if the observability matrix

$$\Phi_0 = [C, CA, CA^2, \dots, CA^{n-1}] \text{ has full rank n.}$$

Given:

$$C = [1 \quad 0 \quad 0]$$

$$CA_1 = [0 \quad 1 \quad 0]$$

$$CA_1^2 = [245 \quad -0.0814 \quad -24.47]$$

$$\Phi_{0_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 245 & -0.0814 & -24.47 \end{bmatrix}$$

The matrix has full rank (3), so the system is **observable**.

Minimal Realization:

Theorem 3: A state-space realization (A, B, C, D) is minimal if and only if it is **both controllable and observable**.

Since our system satisfies both conditions, the realization is **minimal**.

State Transition Matrix:

The **state transition matrix** is defined as:

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

This matrix relates the state at time t_0 to the state at time t:

$$x(t) = \Phi(t - t_0)\, x(t_0)$$

It can be computed using the **matrix exponential**:

$$e^{At} = I + At + \frac{A^2t^2}{2!} + \frac{A^3t^3}{3!} +$$

Or using the **Laplace transform**:

$$e^{At} = \mathcal{L}^{-1}\{(sI - A)^{-1}\}$$

Example for matrix A_1:

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

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

$$\mathrm{A}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 245 & -0.0814 & -24.47 \\ 0 & 0 & -250 \end{bmatrix}, \quad \mathrm{eig}(\mathrm{A}_1) = \begin{matrix} \lambda_1 = -250 \\ \lambda_2 = -15.7 \\ \lambda_3 = 15.61 \end{matrix}$$

$$\begin{aligned} &1948496.6721e^{At} \\ &\quad + (265.61A^2e^{-15.7t} - 31.31A^2e^{-250t} - 234.3A^2e^{15.61t} + 62256.33Ae^{-1\ .7t} \\ &\quad - 2.82e^{-250t}) \\ &\quad - (62253.51Ae^{15.61t} - 1036542.5Ie^{-15.7t} + 76733.279Ie^{-250t} \\ &\quad - 919627.5Ie^{15.61}\ ) = 0 \end{aligned}$$

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

V. Transfer Function:

The transfer function is derived from the linearized system using the formula:

$$G(s) = C(sI - A)^{-1}b$$

$$G_1(s) = \frac{-60073.7}{491s^3 + 122790s^2 - 110295s - 30073750}$$

Using MATLAB (see Appendix A), the eigenvalues (poles) of both systems are:

λ1=−250, **λ2**=−15.7, **λ3**=15.61

The system has no zeros and since one of the poles is positive, the system is **unstable**.

VI. PID Controller:

Introduction:

A proportional–integral–derivative controller (PID controller or three-term controller) is a feedback-based control loop mechanism commonly used to manage machines and processes that require continuous control and automatic adjustment. It is typically used in industrial control systems and various other applications where constant control through modulation is necessary without human intervention. The PID controller automatically compares the desired target value (setpoint or SP) with the actual value of the system (process variable or PV). The difference between these two values is called the error value, denoted as e(t).

It then applies corrective actions automatically to bring the PV to the same value as the SP using three methods: The proportional (P) component responds to the current error value by producing an output that is directly proportional to the magnitude of the error. This provides immediate correction based on how far the system is from the desired setpoint. The integral (I) component, in turn, considers the cumulative sum of past errors to address any residual steady-state errors that persist over time, eliminating lingering discrepancies. Lastly, the derivative (D) component predicts future error by assessing the rate of change of the error, which helps to mitigate overshoot and enhance system stability, particularly when the system undergoes rapid changes. The PID controller reduces the likelihood of human error and improves automation.

A common example is a vehicle's cruise control system. When a vehicle encounters a hill, its speed may decrease due to constant engine power. The PID controller adjusts the engine's power output to restore the vehicle to its desired speed, doing so efficiently with minimal delay and overshoot. **[1]**

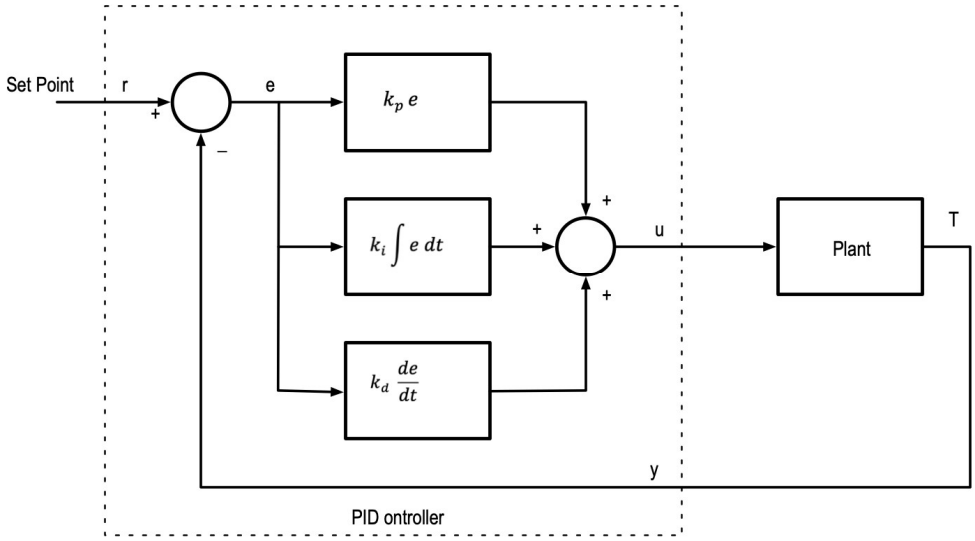

Fig1.2: PID Controller Block Diagram **[2]**

PID Controller Tuning:

First, for the open-loop (Fig. 1.3), using the transfer function G(s), where

$$G_1(s) = \frac{-6007\ .7}{491s^3+122790s^2-110295s-30073750}$$

$$G_2(s) = \frac{60073.7}{491s^3+122790s^2-110295s-30073750}$$

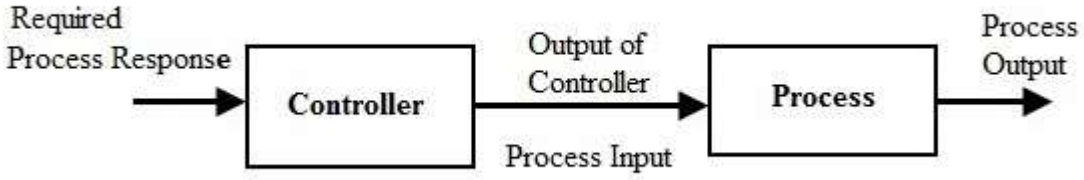

Fig1.3: Open Loop System Block Diagram **[3]**

The step responses of two systems (G1 and G2) are shown in Fig. 1.4 and Fig. 1.5 respectively. See MATLAB code in Appendix for this part.

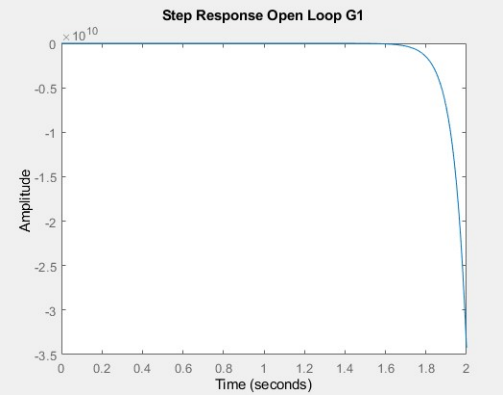

Fig1.4: Step Response Open Loop G1(s)

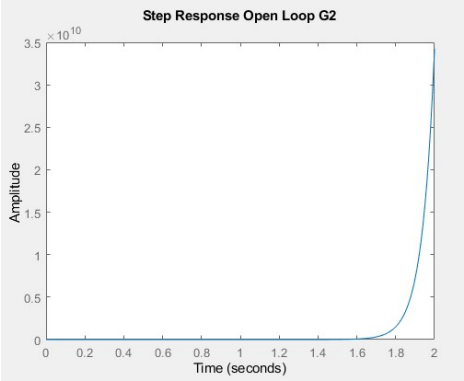

Fig1.5: Step Response Open Loop G2(s)

Proportional Control:

A proportional controller (K_p) applies a control signal proportional to the error between the reference input and the system output. It reduces the rise time, increases the overshoot, and decreases the steady-state error. The closed-loop transfer function of a unity-feedback system with a proportional controller is given by:

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

The response of the system when K_p = 500 is shown in Fig1.6 and Fig1.7.

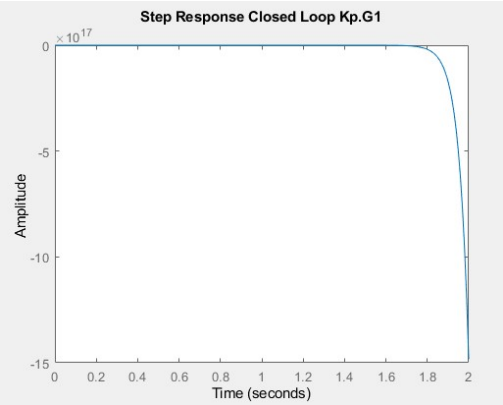

Fig1.6: Step Response of K_pG1(s)

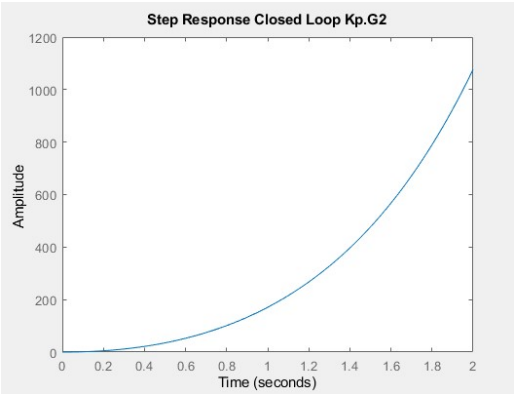

Fig1.7: Step Response K_pG2(s)

The step response of this system when K_p = 1600 for G2 is shown in Fig1.8.

As shown in Fig1.8, increasing the proportional gain helps in bounding the amplitude of the step response, resulting in a more limited output range. However, despite this improvement in response behavior, the system remains unstable, as evident from the continued divergence in the time-domain response.

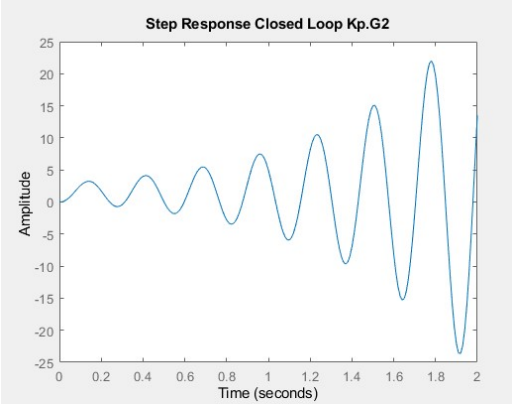

Fig1.8: Step Response K_p = 1600 for G2(s)

Proportional-Derivative Control:

The inclusion of derivative control in the system helps in reducing both the overshoot and the settling time, leading to a more stable response. The closed-loop transfer function of the system with a PD controller is given by:

$$T_3(s) = \frac{(K_d s + K_p)G(s)}{1 + (K_d s + K_p)G(s)}$$

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

The step response of this system when K_p = 1600 and K_d = 500 for G2 is shown in Fig1.9.

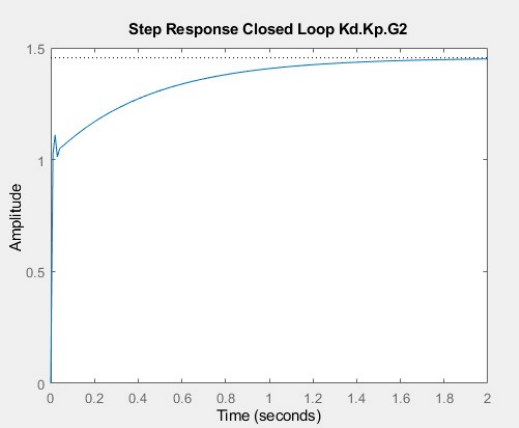

Fig1.8: Step Response K_p = 1600, K_d = 500 for G2(s)

The stepinfo() for this system will be in Table1.2.

Rise Time	0.5010
Settling Time	1.2261
Settling Min	1.3100
Settling Max	1.4549
Overshoot	0
Undershoot	0
Peak	1.4549
Peak Time	3.0464

Table1.2: Step Info T_3

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 parameters, K_p = 1600, K_d = 245 and K_i = 2600 the step response of the system is shown in Fig1.9.

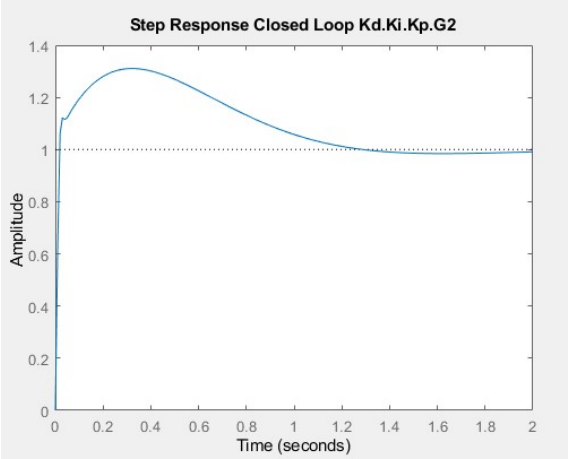

Fig1.8: Step Response K_p = 1600, K_d = 245, K_i = 2600 for G2(s)

The stepinfo() for this system will be in Table1.3.

Rise Time	0.0117
Settling Time	1.1550
Settling Min	0.9167
Settling Max	1.3107
Overshoot	31.0718
Undershoot	0
Peak	1.3107
Peak Time	0.3212

Table1.3: Step Info T_4

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

VII. Non-Linear Model Results in Simulink:

The non-linear model of the Magnetic Levitation System is designed in Simulink and it's shown in Fig2. The parameters are assigned by the values in Table1.1. The completed system is shown in Fig2. The input is considered as step and the 3D simulation of ball translation in the system is shown in the output to show better vision of this system. **[4]** 1 DOF PID Controller is used from MATLAB in-built blocks and the constant values are set which are shown in Fig2.1.

First, the ball's position can be seen in Fig2.2, where the ball's position is 0 based on the definition of output in section 2, when T = 0. And the ball's position is shown in Fig2.3, when T=10. The PID values are tuned and then decided which will enable the ball to levitate with stability. The ball moves upward and stays stable at the position as seen in Fig2.3.

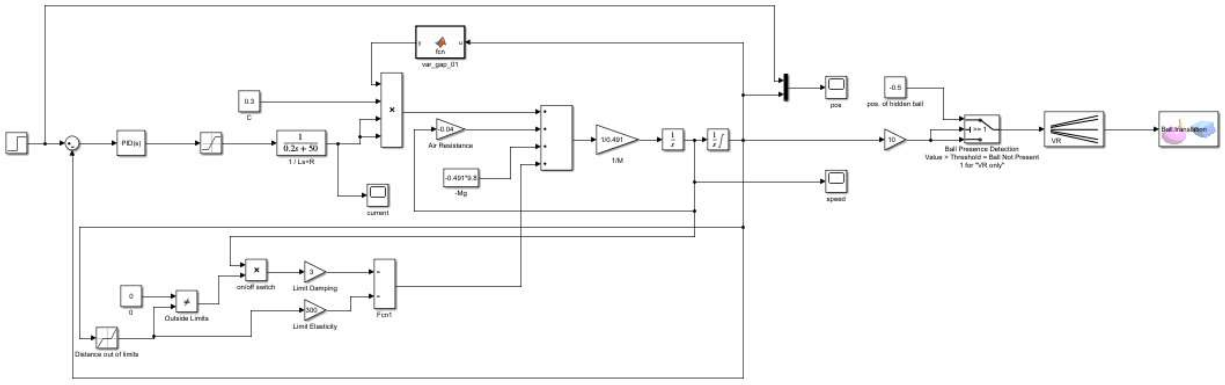

Fig2. Non-Linear Magnetic Levitation System Simulink Model

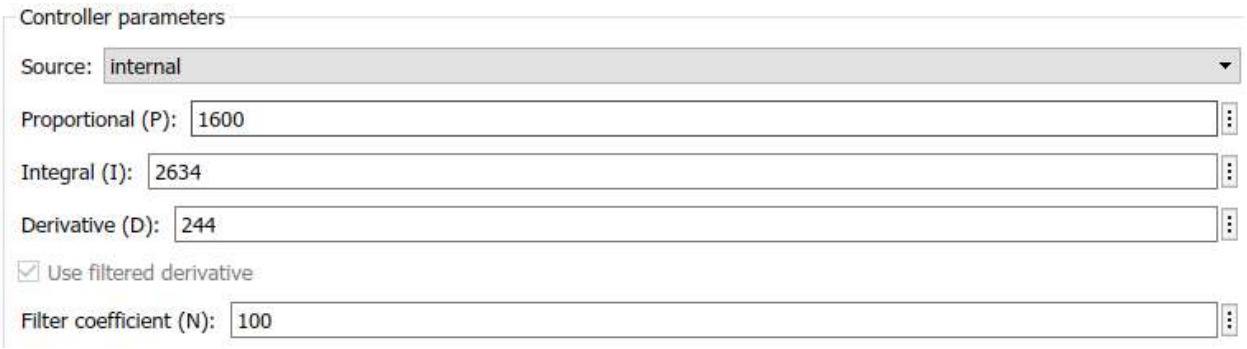

Fig2.1. PID Controllers parameters

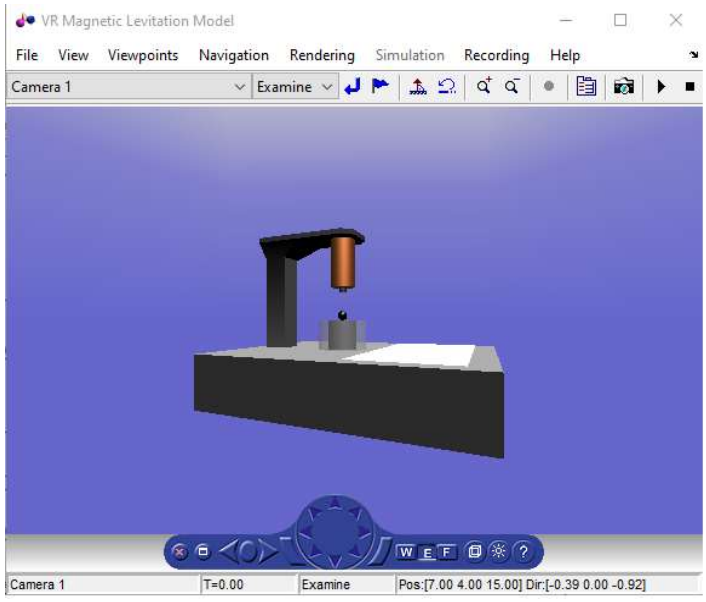

Fig2.2: Ball's Position when T = 0

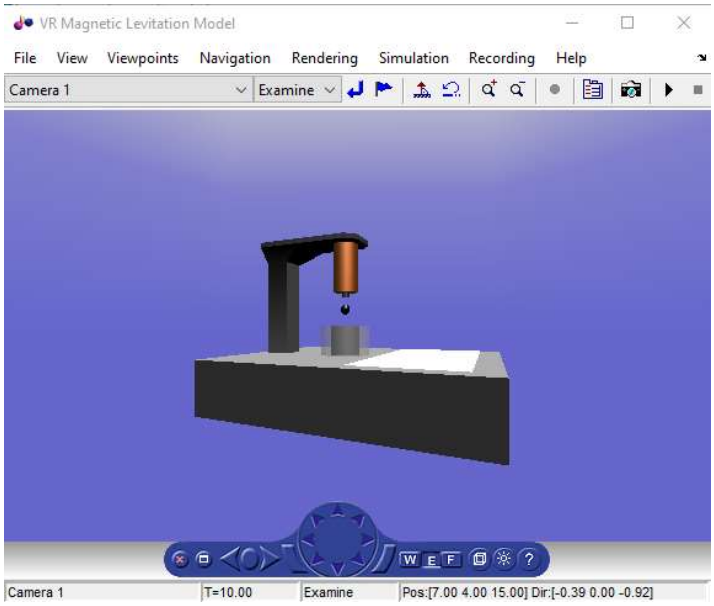

Fig2.3: Ball's Position when T = 10.0

Next, the step responses of position, velocity, and current(states) are shown in Fig2.4, Fig2.5, and Fig2.6 respectively.

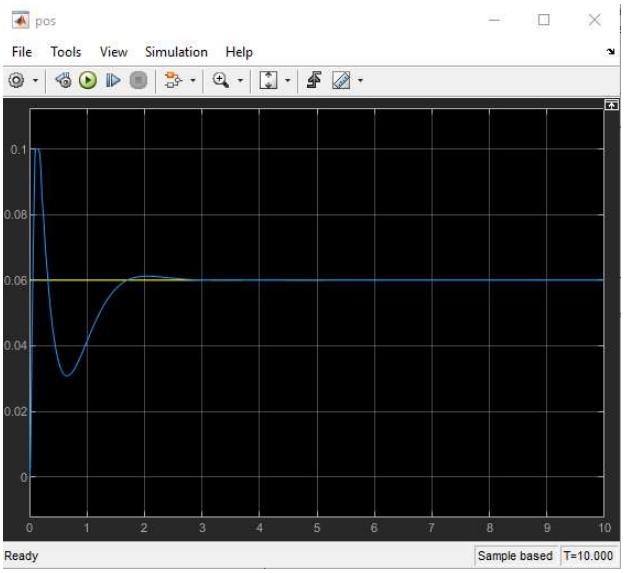

Fig2.4: Step Response of Ball's Position when T = 10.0

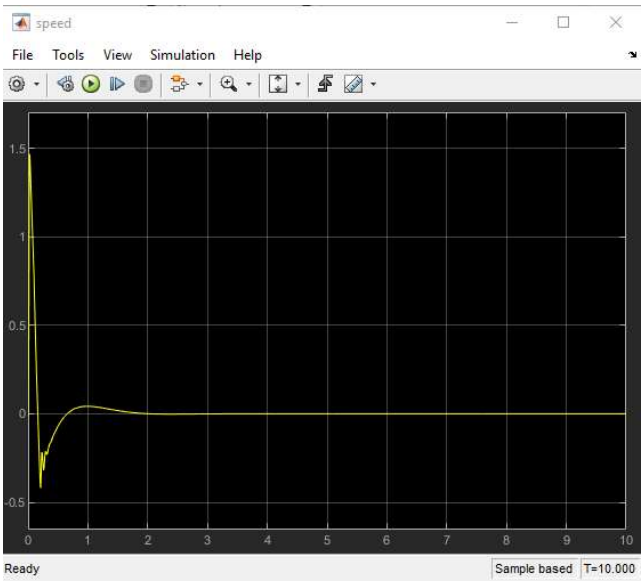

Fig2.5: Step Response of Ball's Velocity when T = 10.0

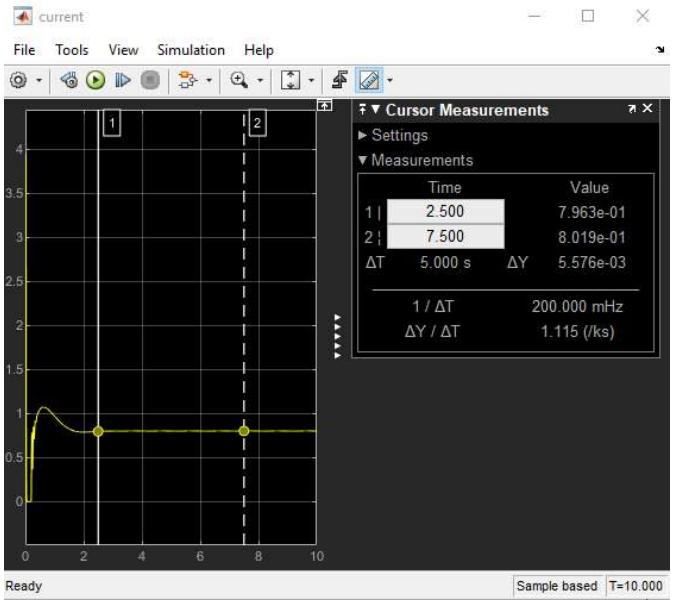

Fig2.6: Step Response of Current when T = 10.0

REFERENCES

[1] https://en.wikipedia.org/wiki/Proportional-integral-derivative_controller

[2] https://circuitcellar.com/resources/quickbits/pid-control/

[3] https://www.elprocus.com/difference-between-open-loop-closed-loop-control-system/

[4] The Simulink® model represents the HUMUSOFT® CE152 Magnetic Levitation educational / presentation scale model.

https://www.mathworks.com/help//releases/R2021a/sl3d/examples/magnetic-levitation-model.html

[5] https://www.researchgate.net/publication/383050008_Modelling_and_Control_of_a_Magnetic_Levitation_System

Appendix A

```
clc;

% Symbolic variables
syms x1 x2 x3 u s tt II AA

%% Equilibrium Points (4)

% Constants
R = 50;       % Resistance
L = 0.2;      % Inductance
g = 9.8;      % Gravitational acceleration
M = 0.491;    % Mass
c = 0.3;      % Damping coefficient
fv = 0.04;    % Friction coefficient

% Equilibrium conditions
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;

% Solve equilibrium equations
stateEqs = [eq1, eq2, eq3, eq4];
sol = solve(stateEqs);

% Equilibrium point solutions
xe1 = sol.x1;
xe2 = sol.x2;
xe3 = sol.x3;

%% Linearization (5)

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

% Compute Jacobian matrix
A = jacobian([f, g, h], [x1, x2, x3]);
A

% Linearize the system at equilibrium points
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)});

% Compute eigenvalues
```

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

%% State-Space Analysis (6)

% Controllability Analysis
% Using ctrb(A, B) or manual calculations
B = [0; 0; 5];  % Input vector

% Compute controllability matrices
A_1B = A_1 * B;
A_2B = A_2 * B;
A_12B = A_1^2 * B;
A_22B = A_2^2 * B;

% Controllability matrices
Phi_c1 = [B A_1B A_12B];
Phi_c2 = [B A_2B A_22B];

% Determinant of controllability matrices
det_Phi_c1 = det(Phi_c1);
det_Phi_c2 = det(Phi_c2);

% Observability Analysis
% Using obsv(A, C) or manual calculations
C = [1 0 0];  % Output vector

% Compute observability matrices
A_1C = C * A_1;
A_2C = C * A_2;
A_12C = C * A_1^2;
A_22C = C * A_2^2;

% Observability matrices
Phi_o1 = [C; A_1C; A_12C];
Phi_o2 = [C; A_2C; A_22C];

% Rank of observability matrices
rank_Phi_o1 = rank(Phi_o1);
rank_Phi_o2 = rank(Phi_o2);

%% State Transition Matrix

% Compute inverse Laplace transform of the state transition matrix
eA1t = ilaplace(inv((s*eye(size(A_1,1)) - A_1)));
eA2t = ilaplace(inv((s*eye(size(A_2,1)) - A_2)));

% Define X matrix for some system analysis (customized)
X = [1 -250 62500 exp(-250*tt); 1 -15.7 246.49 exp(-15.7*tt); 1 15.61 243.67
exp(15.61*tt); II AA AA^2 exp(AA*tt)];

%% Transfer Function

% Define transfer function
```

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

% Solve for poles of the system
poleq = 491*s^3 + 122790*s^2 - 110295*s - 30073750;
solve(poleq, s);

%% PID Controller

% Define transfer function for G1 and G2
s = tf('s');
G1 = -60073.7 / (491*s^3 + 122790*s^2 - 110295*s - 30073750);
G2 = 60073.7 / (491*s^3 + 122790*s^2 - 110295*s - 30073750);

% Plot step response for open-loop G1 and G2
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 Controller
Kp = 1600;
C = pid(Kp);

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

% Plot closed-loop step responses for P controller
figure;
step(T1, t);
title('Step Response Closed Loop Kp.G1');

figure;
step(T2, t);
title('Step Response Closed Loop Kp.G2');

% PD Controller
Kd = 245;
C = pid(Kp, 0, Kd);
T3 = feedback(C*G2, 1);

info_T3 = stepinfo(T3);
disp('Step Info for T3 (PD Controller):');
disp(info_T3);

% Plot closed-loop step response for PD controller
figure;
stepinfo(T3);
step(T3, t);
title('Step Response Closed Loop Kd.Kp.G2');
```

```
% PID Controller
Ki = 2600;
C = pid(Kp, Ki, Kd);
T4 = feedback(C*G2, 1);

info_T4 = stepinfo(T4);
disp('Step Info for T4 (PD Controller):');
disp(info_T4);

% Plot closed-loop step response for PID controller
figure;
stepinfo(T4);
step(T4, t);
title('Step Response Closed Loop Kd.Ki.Kp.G2');
```