

# PROJECT REPORT



---

## CONTROL SYSTEMS

---



**Author :** Madduru Mani Teja

**Date of submission :** 21 july 2024

**Training Program name:** Angstromers Engineering Solution Pvt. Ltd

**Institution:** G Pullaiah College of Engineering and Technology.

# **Table of Contents**

## **1. Introduction: System modelling**

- Introduction
- Learnings
- Use Cases
- Code
- Output

## **2. Cruise Control**

- Introduction
- Learnings
- Use Cases
- Code
- Output

## **3. Motor Speed Control**

- Introduction
- Learnings
- Use Cases
- Code
- Output

## **4. Motor Position Control**

- Introduction
- Learnings
- Use Cases
- Code
- Output

## **5. Suspension System Control**

- Introduction
- Learnings
- Use Cases
- Code
- Output

## **6. Inverted Pendulum Control**

- Introduction
- Learnings
- Use Cases
- Code
- Output

## **7. Aircraft Pitch Control**

- Introduction
- Learnings
- Use Cases
- Code
- Output

## **8. Ball & Beam Control**

- Introduction
- Learnings
- Use Cases
- Code
- Output

# 1.Introduction: System Modelling

## 1.1 Introduction :

### System Modelling Overview :

System modelling involves creating mathematical descriptions of physical systems to understand and predict their behavior. These models can be derived from physical laws or through experimental data. The primary types of models used are:

- **State-Space Representation:** This models systems using vectors and matrices to describe how the system evolves over time.
- **Transfer Function Representation:** This represents systems using ratios of polynomials to describe input-output relationships in the frequency domain.

### MATLAB Commands

MATLAB is a powerful tool for modeling and analyzing these systems. The main commands are:

- **ss** for creating and working with state-space models.
- **tf** for creating and working with transfer function models.

## 1.2 Learnings :

### i) State-Space Representation

- **Concept:** This method uses matrices to describe how system states evolve over time based on inputs. It's represented by:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$$

$$\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$$

- **Learning:** The state-space model is useful for analyzing complex systems with multiple inputs and outputs. It's widely used in modern control systems.

### ii) Transfer Function Representation

- **Concept:** This approach converts time-domain equations into the frequency domain, using the Laplace transform. It's represented by:

$$G(s) = \frac{Y(s)}{U(s)}$$

- **Learning:** Transfer functions simplify the analysis of how systems respond to different frequencies and are helpful for understanding system behavior and stability.

### iii) Mechanical System Modeling

- **Concept:** Mechanical systems, like a mass-spring-damper, are modeled using Newton's laws. For instance:

$$m\ddot{x} + b\dot{x} + kx = F(t)$$

- **Learning:** This helps in analyzing how mechanical systems react to forces, which is key for designing and controlling mechanical systems.

### iv) Electrical System Modeling

- **Concept:** Electrical systems, such as an RLC circuit, are modeled using Kirchhoff's laws:

$$V(t) - Ri - L\frac{di}{dt} - \frac{1}{C} \int i dt = 0$$

- **Learning:** Similar to mechanical systems, this helps in analyzing electrical circuits and designing control systems for them.

### v) System Identification

- **Concept:** When exact system parameters are unknown, system identification uses experimental data to create models.
- **Learning:** This technique is crucial for developing accurate models based on real-world data when theoretical models are impractical.

## 1.3 Use cases :

### i) Control System Design

- **Application:** Used to design controllers for systems like cruise control in cars, ensuring they perform as desired.

### ii) System Optimization

- **Application:** Helps in tuning system parameters for better performance, such as in robotics for precise control.

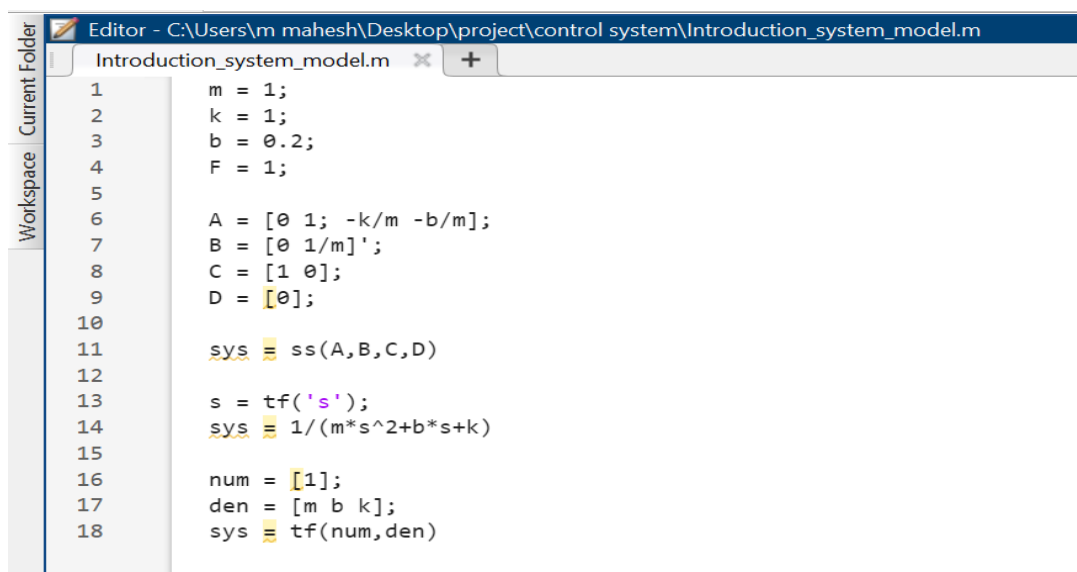
### iii) Fault Diagnosis and Maintenance

- **Application:** Models aid in detecting and diagnosing faults in systems, guiding maintenance efforts.

### iv) Simulation and Testing

- **Application:** MATLAB and other tools use these models for simulating system behavior, helping to test designs before physical implementation.

## 1.4 Code :



```
Editor - C:\Users\m mahesh\Desktop\project\control system\Introduction_system_model.m
Introduction_system_model.m
1      m = 1;
2      k = 1;
3      b = 0.2;
4      F = 1;
5
6      A = [0 1; -k/m -b/m];
7      B = [0 1/m]';
8      C = [1 0];
9      D = [0];
10
11     sys = ss(A,B,C,D)
12
13     s = tf('s');
14     sys = 1/(m*s^2+b*s+k)
15
16     num = [1];
17     den = [m b k];
18     sys = tf(num,den)
```

## Output :

```
>> Introduction_system_model
```

```
sys =
```

```
A =
```

```
      x1      x2
x1      0      1
x2     -1    -0.2
```

```
B =
```

```
      u1
x1      0
x2      1
```

```
C =
```

```
      x1      x2
y1      1      0
```

```
D =
```

```
      u1
y1      0
```

Continuous-time state-space model.

[Model Properties](#)

```
sys =
```

```
      1
-----
s^2 + 0.2 s + 1
```

*fx* Continuous-time transfer function.

```
sys =
```

```
      1
-----
s^2 + 0.2 s + 1
```

Continuous-time transfer function.

[Model Properties](#)

*fx* >> |

## 2. Cruise Control: System Modeling

### 2.1 Introduction :

Automatic cruise control is a feedback control system found in many modern vehicles. Its purpose is to maintain a constant vehicle speed despite external disturbances, such as changes in wind or road grade. This is accomplished by measuring the vehicle speed, comparing it to the desired speed, and automatically adjusting the throttle.

### Physical Setup

We model a vehicle with mass  $m$  acted on by a control force  $u$ , which represents the force generated at the road/tire interface. We assume this force can be controlled directly, neglecting the dynamics of the powertrain and tires. Resistive forces  $bv$ , due to rolling resistance and wind drag, vary linearly with vehicle velocity  $v$  and act in the opposite direction.

### System Equations

The system is modeled as a first-order mass-damper system. Summing forces in the x-direction and applying Newton's 2nd law:

$$m\dot{v} + bv = u$$

The output equation is:  $y=v$

### System Parameters

- Vehicle mass,  $m=1000$  kg
- Damping coefficient,  $b=50$  N.s/m

### State-Space Model

The state-space representation is:

$$\dot{\mathbf{x}} = [\dot{v}] = \begin{bmatrix} -b \\ m \end{bmatrix} [v] + \begin{bmatrix} 1 \\ m \end{bmatrix} [u]$$

$$y = [1][v]$$

### Transfer Function Model

The transfer function of the cruise control system is:

$$P(s) = \frac{V(s)}{U(s)} = \frac{1}{ms + b}$$

## 2.2 Learnings :

- **System Modeling:** Understanding the fundamentals of modeling a physical system using Newton's laws and deriving the corresponding system equations.
- **State-Space Representation:** Learning to represent a dynamic system in state-space form, which is crucial for modern control theory and analysis.
- **Transfer Function:** Deriving the transfer function of a system and understanding its significance in control system analysis.
- **MATLAB Implementation:** Gaining hands-on experience with MATLAB to model and analyze control systems using state-space and transfer function approaches.
- **Feedback Control:** Understanding the principles of feedback control systems and their application in maintaining desired system outputs despite external disturbances.

## 2.3 Use cases :

- **Maintaining Constant Speed:** The primary use of cruise control is to maintain a constant vehicle speed set by the driver, enhancing driving comfort on long trips.
- **Fuel Efficiency:** By maintaining a steady speed, cruise control can help improve fuel efficiency, as it reduces unnecessary acceleration and deceleration.
- **Driver Fatigue Reduction:** Reducing the need for constant speed adjustments can help reduce driver fatigue, making long-distance driving more comfortable and safer.
- **Advanced Driver Assistance Systems (ADAS):** Cruise control systems form the basis for more advanced driver assistance systems, such as adaptive cruise control, which can automatically adjust the vehicle speed based on traffic conditions.
- **Autonomous Vehicles:** Understanding and developing cruise control systems is a step towards the development of fully autonomous vehicles, where maintaining speed and handling disturbances are crucial.

## 2.4 Code



```
Editor - C:\Users\m mahesh\Desktop\project\control system\Cruise_Control_System_Modeling.m
Cruise_Control_System_Modeling.m
m = 1000;
b = 50;

A = -b/m;
B = 1/m;
C = 1;
D = 0;

cruise_ss = ss(A,B,C,D);
|
s = tf('s');
P_cruise = 1/(m*s+b);
```

## Output

```
Command Window
>> Cruise_Control_System_Modeling

cruise_ss =

    A =
        x1
        x1  -0.05

    B =
        u1
        x1  0.001

    C =
        x1
        y1  1

    D =
        u1
        y1  0

Continuous-time state-space model.

P_cruise =

        1
    -----
    1000 s + 50

Continuous-time transfer function.
Model Properties
```

## 3. DC Motor Speed: System Modeling

### 3.1 Introduction:

A DC motor is a common actuator in control systems, providing rotary motion, which can be converted into translational motion with wheels or drums and cables. In this project, we model the speed control of a DC motor, where the input is the voltage applied to the motor's armature, and the output is the rotational speed of the shaft. We assume a viscous friction model where friction torque is proportional to shaft angular velocity.

### Physical Setup

The physical setup involves a DC motor with the following parameters:

- Moment of inertia of the rotor,  $J=0.01 \text{ kg.m}^2$
- Motor viscous friction constant,  $b=0.1 \text{ N.m.s}$
- Electromotive force constant,  $K_e=0.01 \text{ V/rad/sec}$
- Motor torque constant,  $K_t=0.01 \text{ N.m/Amp}$
- Electric resistance,  $R=1 \text{ Ohm}$
- Electric inductance,  $L=0.5 \text{ H}$

### System Equations

The motor torque  $T$  and back emf  $e$  are given by:

$$T = K_t i \quad \text{and} \quad e = K_e \dot{\theta}$$

Using Newton's 2nd law and Kirchhoff's voltage law, the governing equations are:

$$J\ddot{\theta} + b\dot{\theta} = K i \quad \text{and} \quad L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

### Transfer Function

Applying the Laplace transform to the governing equations and eliminating  $I(s)$  :

$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2}$$

## State-Space Model

The state-space representation, choosing rotational speed and electric current as state variables:

$$\frac{d}{dt} \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V$$

$$y = [1 \quad 0] \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix}$$

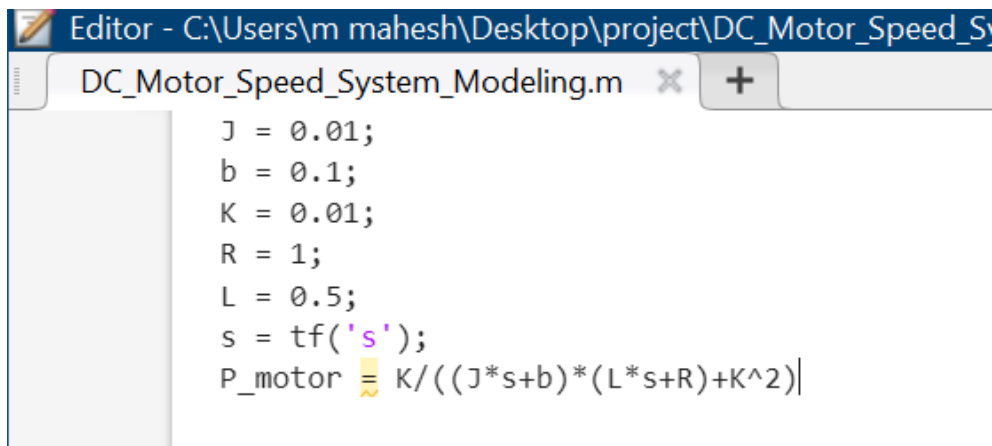
## 3.2 Learnings

1. **System Modeling:** Applying Newton's laws and Kirchhoff's voltage law to model the DC motor.
2. **Transfer Function:** Deriving and understanding the open-loop transfer function.
3. **State-Space Representation:** Representing the system in state-space form for modern control analysis.
4. **MATLAB Implementation:** Utilizing MATLAB for system modeling and analysis.
5. **Control System Design:** Developing and analyzing control systems to meet specific design requirements.

## 3.3 Use Cases

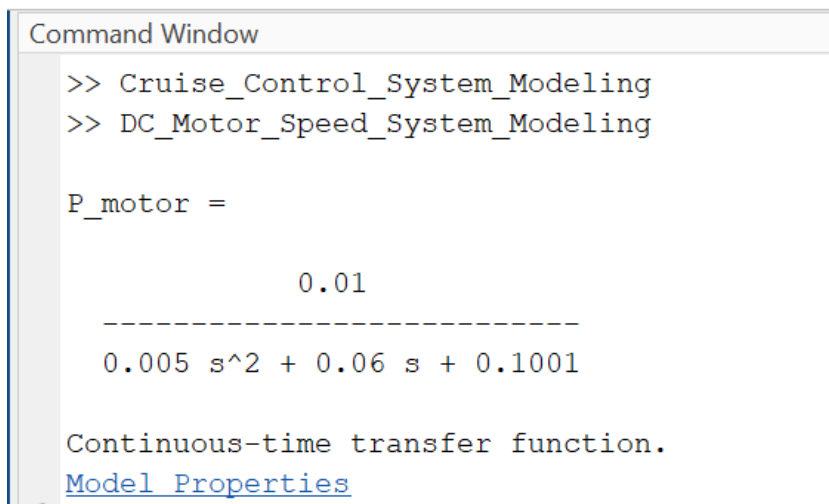
1. **Robotics:** DC motors are widely used in robotic systems for precise control of movement.
2. **Industrial Automation:** Used in conveyors, actuators, and other automated systems requiring speed control.
3. **Electric Vehicles:** Essential in controlling the speed and torque of electric vehicle motors.
4. **Home Appliances:** Found in various household devices like fans, washing machines, and mixers.
5. **Aerospace:** Used in control surfaces and actuators in aircraft and spacecraft for precise control.

### 3.4 Code :



```
Editor - C:\Users\m mahesh\Desktop\project\DC_Motor_Speed_S
DC_Motor_Speed_System_Modeling.m x +
J = 0.01;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;
s = tf('s');
P_motor = K/((J*s+b)*(L*s+R)+K^2)|
```

### Output:



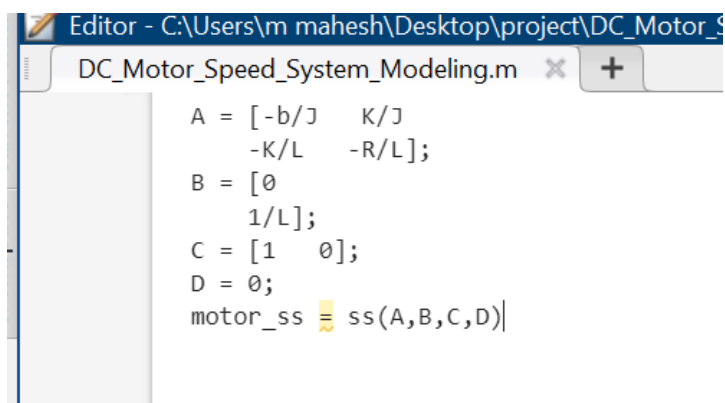
```
Command Window
>> Cruise_Control_System_Modeling
>> DC_Motor_Speed_System_Modeling

P_motor =

          0.01
-----
0.005 s^2 + 0.06 s + 0.1001

Continuous-time transfer function.
Model Properties
```

### Code:



```
Editor - C:\Users\m mahesh\Desktop\project\DC_Motor_S
DC_Motor_Speed_System_Modeling.m x +
A = [-b/J    K/J
      -K/L   -R/L];
B = [0
      1/L];
C = [1    0];
D = 0;
motor_ss = ss(A,B,C,D)|
```

### Output:

```
motor_ss =
```

```
A =
```

	x1	x2
x1	-10	1
x2	-0.02	-2

```
B =
```

	u1
x1	0
x2	2

```
C =
```

	x1	x2
y1	1	0

```
D =
```

	u1
y1	0

Continuous-time state-space model.

[Model Properties](#)

## 4. DC Motor Speed: System Modeling

### 4.1 Introduction :

DC motors are widely used actuators in control systems, providing rotary motion that can be converted into translational motion. This project models the position control of a DC motor, where the input is the voltage applied to the motor's armature and the output is the position of the shaft. We assume a viscous friction model, where the friction torque is proportional to the shaft's angular velocity. This model is based on parameters derived from experiments conducted at Carnegie Mellon's undergraduate controls lab.

### Physical Setup

The physical parameters for the DC motor are as follows:

(J)	moment of inertia of the rotor	3.2284E-6 kg.m <sup>2</sup>
(b)	motor viscous friction constant	3.5077E-6 N.m.s
(K <sub>b</sub> )	electromotive force constant	0.0274 V/rad/sec
(K <sub>t</sub> )	motor torque constant	0.0274 N.m/Amp
(R)	electric resistance	4 Ohm
(L)	electric inductance	2.75E-6H

### System Equations

The torque generated by a DC motor is proportional to the armature current  $i$  and the strength of the magnetic field. Assuming a constant magnetic field, the motor torque  $T$  and back emf  $e$  are given by:

$$T = K_t i \quad \text{and} \quad e = K_b \dot{\theta}$$

Using Newton's 2nd law and Kirchhoff's voltage law, the governing equations are:

$$J\ddot{\theta} + b\dot{\theta} = K i \quad \text{and} \quad L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

## Transfer Function

Applying the Laplace transform to the governing equations and eliminating  $I(s)$ :

$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2}$$

## State-Space Model

Using the motor position  $\theta$ , speed  $\dot{\theta}$ , and current  $i$  as state variables:

$$\frac{d}{dt} \begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{b}{J} & \frac{K}{J} \\ 0 & -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} V$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix}$$

## 4.2 Learnings :

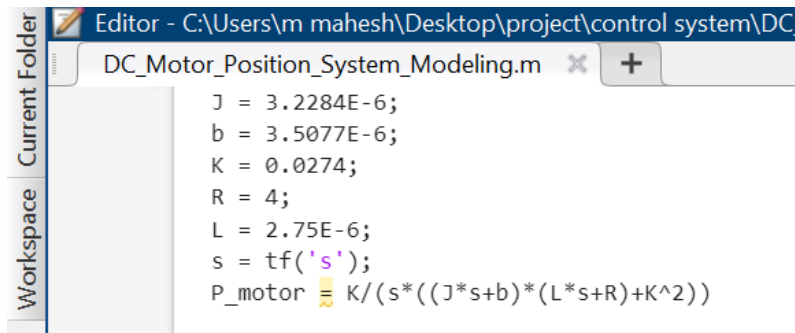
- **System Modeling:** Understanding the dynamics of DC motors and their representation using differential equations.
- **Transfer Function:** Deriving and analyzing the transfer function for motor position control.
- **State-Space Representation:** Expressing the system in state-space form for advanced control analysis.
- **MATLAB Implementation:** Implementing and analyzing the motor model in MATLAB for practical insights.
- **Control Requirements:** Identifying performance requirements for precise control and stability in motor systems.

## 4.3 Use Cases :

- **Robotic Systems:** Precise positioning in robotics for accurate movements.
- **Automated Manufacturing:** Control of actuators in conveyor systems and robotic arms.
- **Electric Vehicles:** Position control in electric vehicle motors for accurate movement.

- **Aerospace:** Control of actuators in spacecraft and aircraft for maneuvering.
- **Consumer Electronics:** Used in devices requiring precise rotational motion, such as camera autofocus mechanisms.

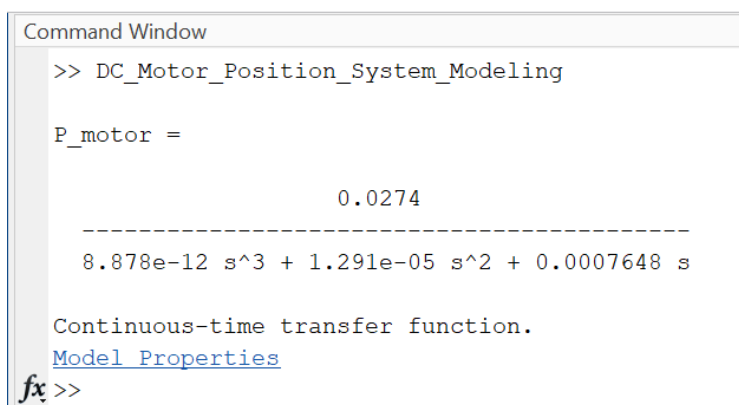
## 4.4 Code:



```

Editor - C:\Users\m mahesh\Desktop\project\control system\DC_
DC_Motor_Position_System_Modeling.m
J = 3.2284E-6;
b = 3.5077E-6;
K = 0.0274;
R = 4;
L = 2.75E-6;
s = tf('s');
P_motor = K/(s*((J*s+b)*(L*s+R)+K^2))
  
```

## Output:



```

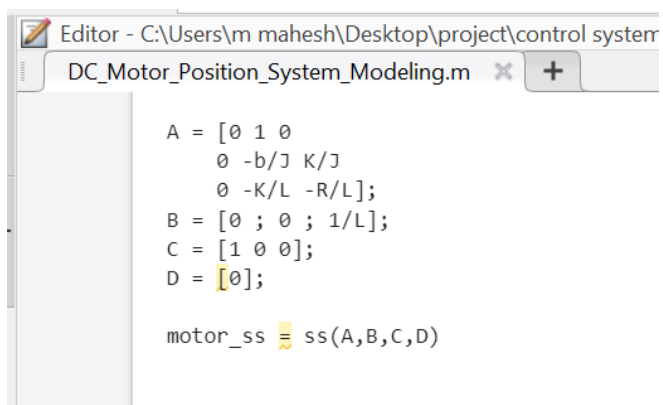
Command Window
>> DC_Motor_Position_System_Modeling

P_motor =

          0.0274
-----
8.878e-12 s^3 + 1.291e-05 s^2 + 0.0007648 s

Continuous-time transfer function.
Model Properties
fx >>
  
```

## Code:



```

Editor - C:\Users\m mahesh\Desktop\project\control system
DC_Motor_Position_System_Modeling.m
A = [0 1 0
      0 -b/J K/J
      0 -K/L -R/L];
B = [0 ; 0 ; 1/L];
C = [1 0 0];
D = [0];

motor_ss = ss(A,B,C,D)
  
```

## Output:



# Command Window

motor\_ss =

A =

	x1	x2	x3
x1	0	1	0
x2	0	-1.087	8487
x3	0	-9964	-1.455e+06

B =

	u1
x1	0
x2	0
x3	3.636e+05

C =

	x1	x2	x3
y1	1	0	0

D =

	u1
y1	0

Continuous-time state-space model.

[Model Properties](#)

## 5.Suspension: System Modeling

### 5.1 Introduction:

Automotive suspension systems are crucial for ensuring vehicle comfort and safety. This project focuses on the modeling of an active suspension system using a simplified 1/4 vehicle model, which represents one wheel of the vehicle. The aim is to analyze the system dynamics and derive transfer functions to understand how the suspension system responds to control inputs and external

### Physical Setup

The suspension system is modeled as a 1-D multiple spring-damper system with the following parameters:

- **1/4 Bus Body Mass (M1):** 2500 kg
- **Suspension Mass (M2):** 320 kg
- **Spring Constant of Suspension System (K1):** 80,000 N/m
- **Spring Constant of Wheel and Tire (K2):** 500,000 N/m
- **Damping Constant of Suspension System (b1):** 350 N.s/m
- **Damping Constant of Wheel and Tire (b2):** 15,020 N.s/m
- **Control Force (U):** Variable control input

### Equations of Motion

The dynamic equations of motion for the suspension system are:

1. For the bus body mass:

$$M_1 \ddot{X}_1 = -b_1(\dot{X}_1 - \dot{X}_2) - K_1(X_1 - X_2) + U$$

2. For the suspension mass:

$$M_2 \ddot{X}_2 = b_1(\dot{X}_1 - \dot{X}_2) + K_1(X_1 - X_2) + b_2(\dot{W} - \dot{X}_2) + K_2(W - X_2) - U$$

Here, X1 is the displacement of the bus body, X2 is the displacement of the suspension mass, and W is the road disturbance input.

## Transfer Function Models

The transfer functions are derived by taking the Laplace Transform of the equations:

**i)Transfer Function  $G_1(s)$ :** This describes the response of the system when only the control force  $U(s)$  is applied:

$$G_1(s) = \frac{X_1(s) - X_2(s)}{U(s)} = \frac{(M_1 + M_2)s^2 + b_2s + K_2}{\Delta}$$

**ii)Transfer Function  $G_2(s)$ :** This describes the response of the system to the road disturbance  $W(s)$  when the control force  $U(s)$  is zero:

$$G_2(s) = \frac{X_1(s) - X_2(s)}{W(s)} = \frac{-M_1b_2s^3 - M_1K_2s^2}{\Delta}$$

**Where  $\Delta$  is the determinant of the coefficient matrix and is given by:**

$$\Delta = (M_1s^2 + b_1s + K_1) \cdot (M_2s^2 + (b_1 + b_2)s + (K_1 + K_2)) - (b_1s + K_1)^2$$

## 5.2 Learnings:

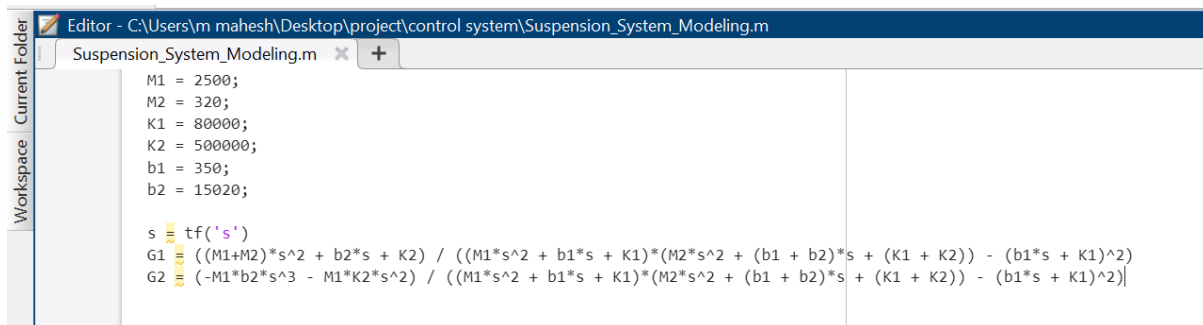
- **System Modeling:** Understanding how to model a suspension system using simplified 1/4 vehicle models and deriving the governing equations of motion.
- **Transfer Function Derivation:** Learning how to derive transfer functions for both control and disturbance inputs.
- **MATLAB Implementation:** Gaining practical experience in implementing mathematical models in MATLAB for analysis and simulation.
- **System Response Analysis:** Analyzing how different inputs affect the suspension system's behavior and response.

## 5.3 Use cases:

- **Vehicle Dynamics:** Improving vehicle comfort and safety by optimizing suspension system design.

- **Automotive Control Systems:** Developing active suspension systems that dynamically adjust to road conditions and driving behavior.
- **Simulation and Testing:** Using MATLAB models to simulate and test suspension system performance before physical implementation.
- **Road Condition Analysis:** Analyzing and compensating for various road disturbances to enhance vehicle stability and ride quality.
- **Design Optimization:** Fine-tuning suspension parameters for better performance in different driving conditions.

## 5.4 Code:



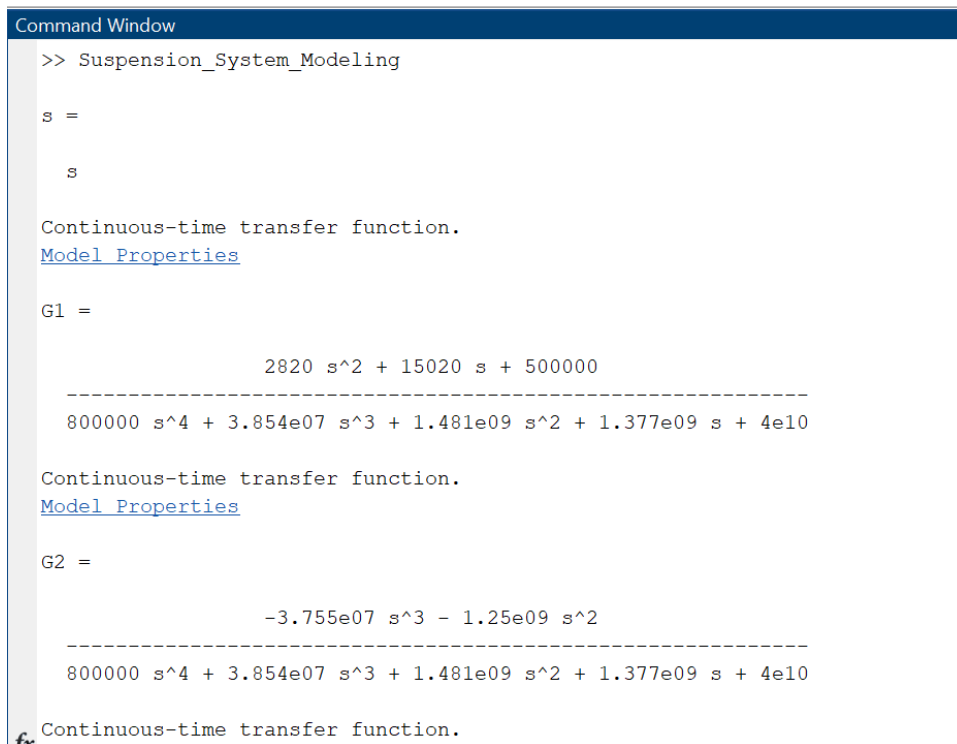
```

Editor - C:\Users\m mahesh\Desktop\project\control system\Suspension_System_Modeling.m
Suspension_System_Modeling.m
M1 = 2500;
M2 = 320;
K1 = 80000;
K2 = 500000;
b1 = 350;
b2 = 15020;

s = tf('s')
G1 = ((M1+M2)*s^2 + b2*s + K2) / ((M1*s^2 + b1*s + K1)*(M2*s^2 + (b1 + b2)*s + (K1 + K2)) - (b1*s + K1)^2)
G2 = (-M1*b2*s^3 - M1*K2*s^2) / ((M1*s^2 + b1*s + K1)*(M2*s^2 + (b1 + b2)*s + (K1 + K2)) - (b1*s + K1)^2)

```

## Output:



```

Command Window
>> Suspension_System_Modeling

s =

s

Continuous-time transfer function.
Model Properties

G1 =

      2820 s^2 + 15020 s + 500000
-----
800000 s^4 + 3.854e07 s^3 + 1.481e09 s^2 + 1.377e09 s + 4e10

Continuous-time transfer function.
Model Properties

G2 =

      -3.755e07 s^3 - 1.25e09 s^2
-----
800000 s^4 + 3.854e07 s^3 + 1.481e09 s^2 + 1.377e09 s + 4e10

Continuous-time transfer function.

```

## 6. Inverted Pendulum: System Modeling

### 6.1 Introduction:

The inverted pendulum is a classic example of a nonlinear dynamic system used in control theory research and practice. This system consists of a pendulum attached to a cart, which moves horizontally. The primary challenge is to keep the pendulum balanced in the upright position by controlling the horizontal movement of the cart. The inverted pendulum is notably unstable in its upright position and requires precise control to maintain balance.

This example is relevant in various practical applications, such as the attitude control of rockets and spacecraft, where precise control of orientation is crucial. The pendulum's dynamics are nonlinear and can be challenging to manage. Linearizing the system around the equilibrium position simplifies the design and analysis of control strategies, allowing us to apply standard control techniques.

### 6.2 Learnings:

- **System Dynamics and Linearization:**
- The inverted pendulum system involves both translational and rotational dynamics, making it a multi-dimensional control problem.
- The system is nonlinear, particularly because of the pendulum's angle relative to the vertical. By linearizing the equations around the upright equilibrium, we simplify the problem into linear equations that are easier to analyze and control.
- Key approximations such as small angle approximations help in deriving linearized equations from the nonlinear dynamics.
- **Transfer Function Derivation:**
- Transfer functions represent the relationship between input and output in the Laplace domain. For the inverted pendulum, two primary transfer functions are derived:
  - **Pendulum Position Transfer Function:** Relates the pendulum's deviation from the vertical (angular position) to the force applied to the cart.

- **Cart Position Transfer Function:** Relates the cart's horizontal position to the applied force.
- These transfer functions are critical for designing and analyzing control systems, as they provide insights into the system's frequency response and stability characteristics.
- **State-Space Representation:**
- State-space models offer a comprehensive way to represent and analyze multi-dimensional systems. For the inverted pendulum, the state-space representation includes:
  - **A matrix:** Describes the system's dynamics.
  - **B matrix:** Represents how the input affects the system.
  - **C matrix:** Relates the system states to the outputs.
  - **D matrix:** Represents any direct feedthrough from the input to the output.
- The state-space model is useful for modern control design techniques, including state feedback and observer design.
- **MATLAB Implementation:**
- MATLAB provides powerful tools for modeling and simulating control systems. The commands for defining transfer functions and state-space models, as well as for converting between them, demonstrate how these theoretical concepts can be applied in practice.
- MATLAB's ability to analyze and visualize system responses helps in designing controllers that meet specific performance criteria.

### 6.3 Use cases:

- **Rocket and Spacecraft Attitude Control:**

Managing the orientation of rockets and spacecraft during launch and flight requires precise control of angular positions. The inverted pendulum's control principles apply directly to these scenarios.

### ➤ Robotic Systems:

Mobile robots and robotic arms often encounter similar stability challenges. Understanding and controlling the dynamics of such systems is crucial for their stable and precise operation.

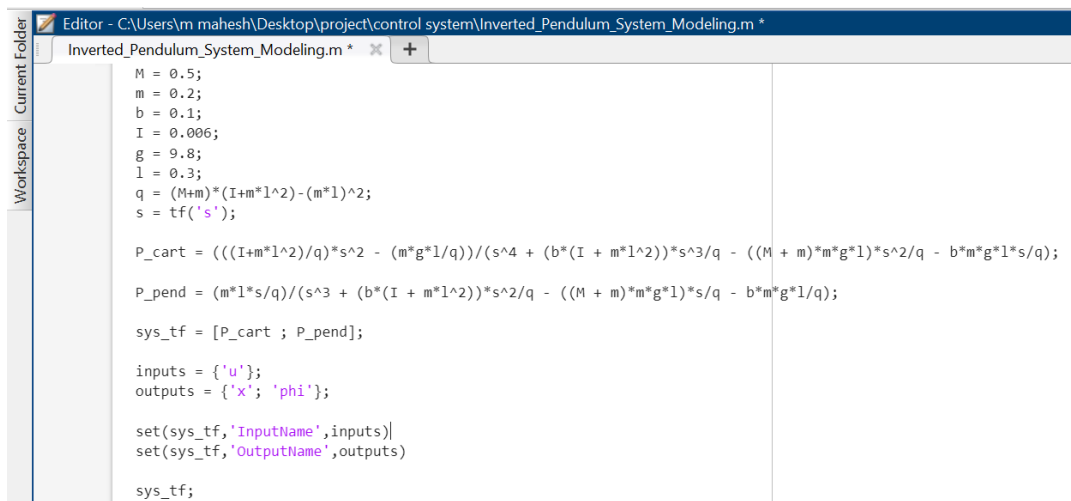
### ➤ Vehicle Suspension Systems:

Active suspension systems in vehicles use principles similar to those in the inverted pendulum. The aim is to maintain vehicle stability and comfort, especially in dynamic conditions.

### ➤ Educational and Research Applications:

The inverted pendulum is a popular testbed for control theory research and educational demonstrations. It provides a practical example of how advanced control techniques can stabilize an inherently unstable system.

## 6.4 Code:



```
Editor - C:\Users\m mahesh\Desktop\project\control system\Inverted_Pendulum_System_Modeling.m *
Inverted_Pendulum_System_Modeling.m * X +

M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;
q = (M+m)*(I+m*l^2)-(m*l)^2;
s = tf('s');

P_cart = (((I+m*l^2)/q)*s^2 - (m*g*l/q))/(s^4 + (b*(I + m*l^2))*s^3/q - ((M + m)*m*g*l)*s^2/q - b*m*g*l*s/q);

P_pend = (m*l*s/q)/(s^3 + (b*(I + m*l^2))*s^2/q - ((M + m)*m*g*l)*s/q - b*m*g*l/q);

sys_tf = [P_cart ; P_pend];

inputs = {'u'};
outputs = {'x'; 'phi'};

set(sys_tf,'InputName',inputs)
set(sys_tf,'OutputName',outputs)

sys_tf;
```

**output:**

```
Command Window

>> Inverted_Pendulum_System_Modeling

sys_tf =

From input "u" to output...
          4.182e-06 s^2 - 0.0001025
x:  -----
    2.3e-06 s^4 + 4.182e-07 s^3 - 7.172e-05 s^2 - 1.025e-05 s

          1.045e-05 s
phi: -----
    2.3e-06 s^3 + 4.182e-07 s^2 - 7.172e-05 s - 1.025e-05

Continuous-time transfer function.
```

## Code:

```
Editor - C:\Users\mahesh\Desktop\project\control system\Inverted_Pendulum_System_Modeling1.m *
Inverted_Pendulum_System_Modeling1.m *

M = .5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;

p = I*(M+m)+M*m*l^2; %denominator for the A and B matrices

A = [0      1      0      0;
     0 -(I+m*l^2)*b/p (m^2*g*l^2)/p 0;
     0      0      0      1;
     0 -(m*l*b)/p      m*g*l*(M+m)/p 0];
B = [0;
     (I+m*l^2)/p;
     0;
     m*l/p];
C = [1 0 0 0;
     0 0 1 0];
D = [0;
     0];

states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'u'};
outputs = {'x' 'phi'};

sys_ss = ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs);
```

## Output:



# Command Window

```
>> Inverted_Pendulum_System_Modeling1
```

```
sys_ss =
```

```
A =
```

	x	x_dot	phi	phi_dot
x	0	1	0	0
x_dot	0	-0.1818	2.673	0
phi	0	0	0	1
phi_dot	0	-0.4545	31.18	0

```
B =
```

	u
x	0
x_dot	1.818
phi	0
phi_dot	4.545

```
C =
```

	x	x_dot	phi	phi_dot
x	1	0	0	0
phi	0	0	1	0

```
D =
```

	u
x	0
phi	0

```
Continuous-time state-space model.
```

## 7. Aircraft Pitch: System Modeling

### 7.1 Introduction:

Aircraft pitch control is crucial for maintaining the desired orientation and stability of an aircraft in flight. The pitch angle, which is the angle between the aircraft's longitudinal axis and the horizontal plane, significantly affects the aircraft's altitude and trajectory. Designing an effective pitch control system ensures that the aircraft can quickly and accurately adjust its pitch to meet various flight conditions and commands.

In this example, we'll focus on a simplified pitch control model of an aircraft, where we aim to design an autopilot system that manages the aircraft's pitch angle. This problem is a classic application in control systems engineering, involving both transfer function and state-space representations.

### Modeling

The dynamics of aircraft pitch can be simplified into longitudinal equations under steady cruise conditions. Here, the pitch angle  $\theta$  is governed by the aircraft's control inputs and its dynamic properties:

1. **Equations of Motion:** The system is described by the following linearized equations:

$$\dot{\alpha} = -0.313\alpha + 56.7q + 0.232\delta$$

$$\dot{q} = -0.0139\alpha - 0.426q + 0.0203\delta$$

a. 
$$\dot{\theta} = 56.7q$$

Here,  $\alpha$  represents the angle of attack,  $q$  is the pitch rate, and  $\delta$  is the elevator deflection angle.

### 2. Transfer Function

By taking the Laplace transform of these equations and solving for the transfer function from the elevator deflection  $\delta$  to the pitch angle  $\theta$ , we obtain:

$$P(s) = \frac{\Theta(s)}{\Delta(s)} = \frac{1.151s + 0.1774}{s^3 + 0.739s^2 + 0.921s}$$

### 3. State-Space Model

The state-space representation of the system is derived from the linearized equations:

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix} \delta$$

The output equation is:

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix}$$

## 7.2 Learnings :

- **Transfer Function Representation:** The transfer function provides a clear understanding of the system's dynamic response to input changes. It describes how input signals (elevator deflection) affect the pitch angle over time, capturing the system's behavior in the frequency domain.
- **State-Space Representation:** The state-space model offers a more detailed view of the system's internal dynamics, including all state variables (angle of attack, pitch rate, and pitch angle). It is particularly useful for designing modern control systems like state feedback and observers.
- **Control Design:** The design criteria for the pitch control system involve ensuring minimal overshoot, quick rise time, and fast settling time, with minimal steady-state error. This demonstrates how control systems can be tailored to meet specific performance requirements, ensuring stability and responsiveness.

## 7.3 Use cases:

- **Autopilot Systems:** The pitch control model is essential in designing autopilot systems for commercial and military aircraft. Accurate pitch control improves flight stability, comfort, and safety.
- **Aircraft Stability Analysis:** Understanding the pitch dynamics helps in analyzing and improving aircraft stability under various flight conditions and disturbances.
- **Control System Design:** Techniques developed for this model can be extended to more complex aircraft systems, including lateral dynamics and full-flight control systems.

- **Training and Simulation:** This simplified model is used in training simulations for pilots and control engineers, offering insights into aircraft behavior and control strategies.

## 7.4 Code :

```
AirCraft_System_model.m ✕ +
s = tf('s');
P_pitch = (1.151*s+0.1774)/(s^3+0.739*s^2+0.921*s);
```

## Output :

```
>> AirCraft_System_model

s =

s

Continuous-time transfer function.
Model Properties

P_pitch =

      1.151 s + 0.1774
-----
      s^3 + 0.739 s^2 + 0.921 s

Continuous-time transfer function.
Model Properties
```

## Code:

```
AirCraft_System_model.m x +
A = [-0.313 56.7 0; -0.0139 -0.426 0; 0 56.7 0];
B = [0.232; 0.0203; 0];
C = [0 0 1];
D = [0];
pitch_ss = ss(A,B,C,D)
```

## Output:

```
Command Window
pitch_ss =

A =
      x1      x2      x3
x1  -0.313    56.7      0
x2  -0.0139  -0.426      0
x3      0     56.7      0

B =
      u1
x1    0.232
x2    0.0203
x3      0

C =
      x1      x2      x3
y1      0      0      1

D =
      u1
y1      0

Continuous-time state-space model.
Model Properties
```

## 8. Ball & Beam: System Modeling

### 8.1 Introduction:

The Ball & Beam system is a classic control problem involving a ball rolling along a beam whose angle can be adjusted by a servo. The objective is to design a controller to manipulate the ball's position on the beam by adjusting the beam's angle. This system serves as a useful example in control theory, demonstrating concepts such as transfer functions, state-space representation, and controller design.

### System Setup and Parameters

(m)	mass of the ball	0.11 kg
(R)	radius of the ball	0.015 m
(d)	lever arm offset	0.03 m
(g)	gravitational acceleration	9.8 m/s <sup>2</sup>
(L)	length of the beam	1.0 m
(J)	ball's moment of inertia	9.99e-6 kg.m <sup>2</sup>
(r)	ball position coordinate	
(alpha)	beam angle coordinate	
(theta)	servo gear angle	

### System Equations

The dynamics of the Ball & Beam system can be described using the Lagrangian mechanics approach. For small angles and linearized conditions, the key equations governing the system are:

#### 1. Lagrangian Equation of Motion:

$$\left( \frac{J}{R^2} + m \right) \ddot{r} = -mg\alpha$$

#### 2. Beam Angle Relation:

$$\alpha = \frac{d}{L}\theta$$

Substituting this into the motion equation:

$$\left(\frac{J}{R^2} + m\right) \ddot{r} = -mg \frac{d}{L} \theta$$

## Transfer Function

To find the transfer function from the servo gear angle ( $\theta$ ) to the ball position ( $r$ ), take the Laplace transform of the linearized equation and solve:

1. Laplace Transform:

$$\left(\frac{J}{R^2} + m\right) s^2 R(s) = -mg \frac{d}{L} \Theta(s)$$

2. Transfer Function:

$$P(s) = \frac{R(s)}{\Theta(s)} = -\frac{mgd}{L \left(\frac{J}{R^2} + m\right)} \frac{1}{s^2}$$

This transfer function represents a double integrator, indicating that the system is marginally stable and requires careful control design.

## 8.2 Learnings :

- **Dynamic System Modeling:** Understand how to model physical systems using simplified assumptions and linearization techniques.
- **Transfer Function Derivation:** Learn to derive transfer functions from system equations using Laplace transforms.
- **State-Space Representation:** Convert system dynamics into state-space form for control design and analysis.
- **Control Design:** Set performance criteria (settling time, overshoot) and design controllers to meet these criteria.
- **MATLAB Utilization:** Use MATLAB for implementing and analyzing transfer function and state-space models.
- **Practical Control Applications:** Apply theoretical knowledge to practical problems in system control, including stability and robustness considerations.

## 8.3 Use cases:

- **Educational Demonstrations:** The Ball & Beam system is often used in educational settings to teach control system concepts and design techniques.

- **Control Systems Research:** Provides a platform for experimenting with control algorithms and understanding system dynamics.
- **Robotic Systems:** Similar principles are applied in robotic arms and other systems requiring precise control of position and orientation.
- **Automated Systems:** Useful in any application where precise positioning and control are required, such as in automated manufacturing and aerospace systems.

## 8.4 Code:

```

Ball_beam_System_model.m
m = 0.111;
R = 0.015;
g = -9.8;
L = 1.0;
d = 0.03;
J = 9.99e-6;

s = tf('s');
P_ball = -m*g*d/L/(J/R^2+m)/s^2;

```

## Output :

```

Command Window

>> Ball_beam_System_model

s =

    s

Continuous-time transfer function.
Model Properties

P_ball =

    0.21
    ----
    s^2

Continuous-time transfer function.
Model Properties

```



## Code:

```
Ball_beam_System_model.m
H = -m*g/(J/(R^2)+m);
A = [0 1 0 0
     0 0 H 0
     0 0 0 1
     0 0 0 0];
B = [0 0 0 1]';
C = [1 0 0 0];
D = 0;
ball_ss = ss(A,B,C,D);
```

## Output:

```
Command Window

ball_ss =

A =
      x1      x2      x3      x4
x1      0      1      0      0
x2      0      0      7      0
x3      0      0      0      1
x4      0      0      0      0

B =
      u1
x1      0
x2      0
x3      0
x4      1

C =
      x1      x2      x3      x4
y1      1      0      0      0

D =
      u1
y1      0

fx Continuous-time state-space model.
```

**9.Referred Link:**

**<https://ctms.engin.umich.edu/CTMS/index.php?aux=Home>**

**10. Github repository :**

**[https://github.com/Madduru-ManiTeja/Control\\_System.git](https://github.com/Madduru-ManiTeja/Control_System.git)**

