

A Report On

Control Systems

by **P.Bhuvaneshwari**

Date of Submission: 21/07/2024.

Training Program Name: Angstromers.

Github Link: [Click Here](#)

List Of Contents

	Pg.no
Section 1: Introduction	1
Section 2: Cruise Control	5
Section 3: Motor Speed	8
Section 4: Motor Position	11
Section 5: Suspension	14
Section 6: Inverted Pendulum	18
Section 7: Aircraft Pitch	22
Section 8: Ball & Beam	25
Section 9: Conclusion	28

SECTION 1

INTRODUCTION

1.1 Theory:

System modeling is the first step in control design. Mathematical models are derived from physical laws or experimental data. State-space and transfer function representations are used for dynamic systems. Physical systems can be described by first-order differential equations. Time-invariant systems have constant parameters, while nonlinear systems can be approximated as linear over a small operating range. Linear time-invariant (LTI) systems can be represented as matrix equations. Control theory is based on LTI assumptions, but feedback control systems are robust against modeling uncertainty.

1.2 Dynamic Systems:

Dynamic systems are systems that change or evolve over time, following a set of rules or laws that describe their behavior. These systems can be physical, biological, economic, or social, and can range from simple to complex.

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$$

Where,

$\mathbf{x}(t)$ = the **state vector**, a set of variables representing the configuration of the system at time t .

$\mathbf{u}(t)$ = the vector of external inputs to the system at time t .

\mathbf{f} = function producing the derivative of state vector.

1.3 State Space Representation:

State-space representation is a mathematical model of a dynamic system, describing its behavior in terms of its internal state and external inputs. It's a fundamental concept in control theory and systems analysis.

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u}$$

Where,

\mathbf{x} = the vector of state variables

$\dot{\mathbf{x}}$ = the time derivative of the state vector

\mathbf{u} = the input or control vector

\mathbf{y} = the output vector

A = the system matrix

B = the input matrix

C = the output matrix

D = the feed forward matrix

1.4 Transfer Function Representation:

Transfer function representation is a mathematical model of a dynamic system, describing its behavior in terms of the relationship between input and output signals. It's a powerful tool for analyzing and designing systems, especially in the frequency domain.

Using the Laplace transform, it is possible to convert a system's time-domain representation into a frequency-domain input/output representation, known as the transfer function. The Laplace transform function is

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^{\infty} e^{-st} f(t) dt$$

Example: Mass-Spring-Damper System

A mass damper system, also known as a tuned mass damper (TMD) or harmonic absorber, is a device that reduces mechanical vibrations in structures by attaching a mass to one or more damped springs.

1.5 Implementation:

```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\introduction.m
+1 introduction.m x cruisecontrol.m x motorspeed.m x motorf
1 % state space model
2 m = 1;
3 k = 1;
4 b = 0.2;
5 F = 1;
6
7 A = [0 1; -k/m -b/m];
8 B = [0 1/m]';
9 C = [1 0];
10 D = [0];
11
12 sys = ss(A,B,C,D)
13
14 % transfer function
15 s = tf('s');
16 sys = 1/(m*s^2+b*s+k)
17
18 num = [1];
19 den = [m b k];
20 sys = tf(num,den)
21
```

Output:

```
C:\Users\DELL 7490\Documents\MATLAB
New to MATLAB? See resources for Getting Started.

workspace Current Folder
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.
```

$$\text{sys} = \frac{1}{s^2 + 0.2s + 1}$$

Continuous-time transfer function.

1.6 Use case of Mass Damper System:

Seismic Isolation in Buildings:

Seismic isolation in buildings is a technique used to reduce the impact of earthquakes on structures. The goal is to protect the building and its occupants from damage and collapse. This is achieved by using base isolators and dampers, which are designed to absorb seismic energy and reduce the transmission of earthquake forces to the structure.

The mass-spring-damper model is used to design these isolators and dampers. The model represents the building as a mass (m), the isolators and dampers as springs (k) and dampers (b), respectively. By tuning the parameters of the model, engineers can design isolators and dampers that can effectively absorb seismic energy and protect the building.

Application:

Using mass-spring-damper models to design isolators and dampers that can absorb seismic energy, thus protecting the structure and its occupants.

Outcome:

The outcomes of seismic isolation are profound: buildings exhibit significantly reduced damage during earthquakes, preserving structural integrity and protecting non-structural elements like internal systems and furnishings. This leads to lower repair costs, enhanced building lifespan, and continued operational functionality, particularly in critical facilities like hospitals and emergency centers.

SECTION 2

CRUISE CONTROL

2.1 Theory:

Cruise control is a feature in vehicles that allows the driver to set and maintain a constant speed without having to press the accelerator pedal. It is also known as speed control or autopilot. The automatic cruise control system uses a feedback control loop to maintain a constant vehicle speed. The system measures the vehicle speed (v) and compares it to the reference speed (v_r), calculating the error (e) between the two. The control law then adjusts the throttle to generate a control force (u) to minimize the error. The vehicle dynamics are modeled by the equation $m\dot{v} = u - bv$, where m is the vehicle mass, \dot{v} is the acceleration, u is the control force, and bv is the resistive force due to rolling resistance and wind drag. The resistive force bv is proportional to the vehicle velocity v and acts in the opposite direction. The system equations are nonlinear, but can be linearized around a steady-state operating point to design a linear control law. The control law is a proportional control, where the control force u is proportional to the error e , with a gain k that determines the response rate.

2.2 System Equations

The system equations represent the dynamics of the vehicle's speed control system.

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

Where,

m = the vehicle's mass

v = the vehicle's speed

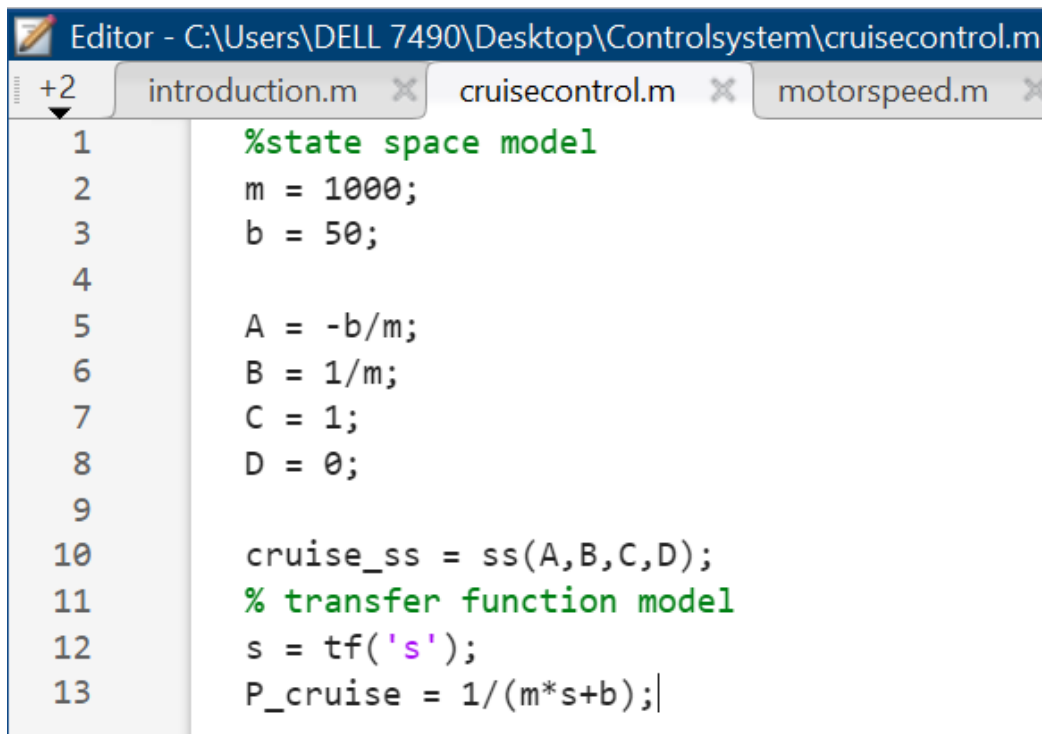
b = damping coefficient represents the resistive forces (rolling resistance and wind drag)

u = the input control force generated by the throttle

\dot{v} = the derivative of velocity (dv/dt)

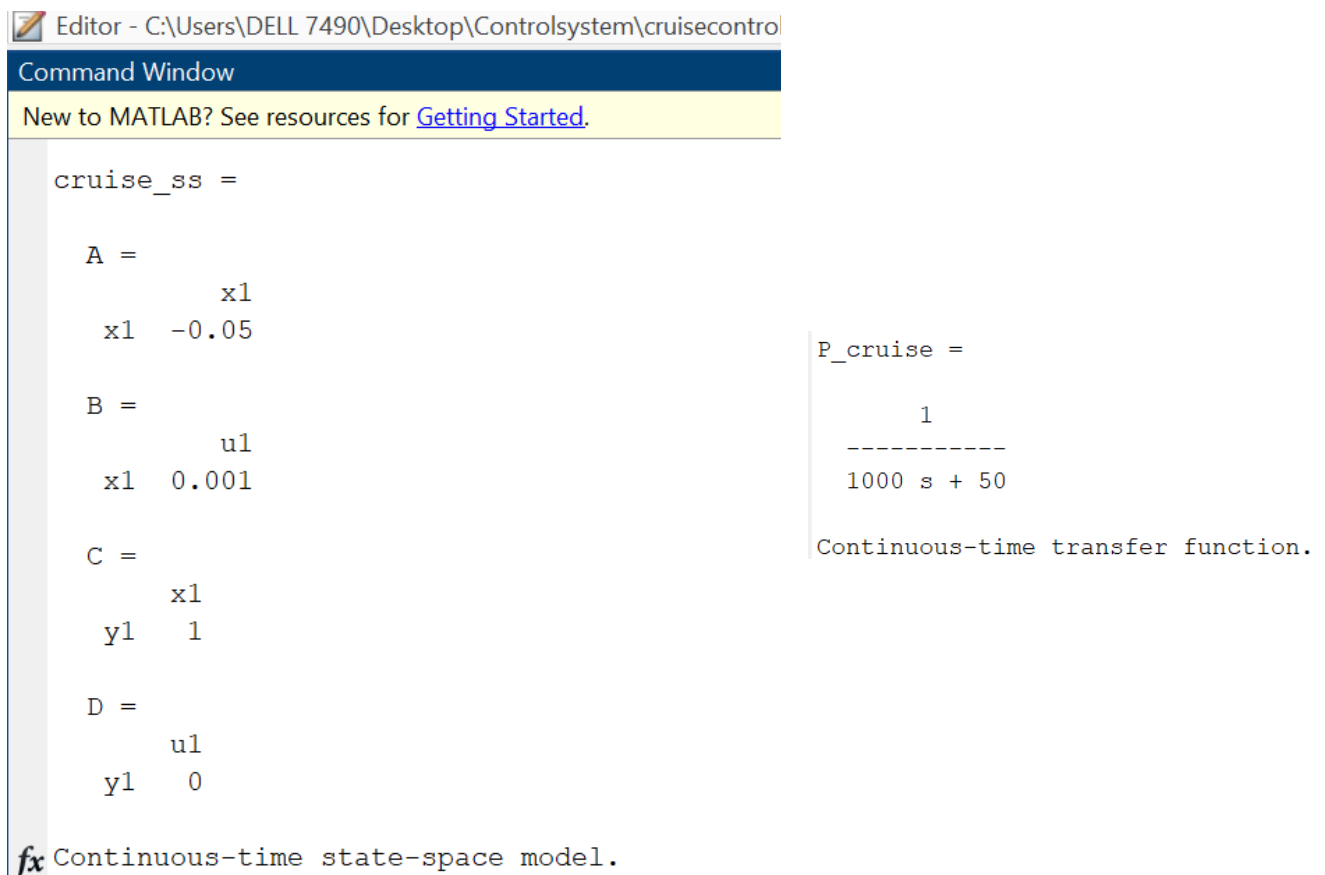
The output equation is, $y = v$

2.3 Implementation:



```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\cruisecontrol.m
+2  introduction.m  x  cruisecontrol.m  x  motorspeed.m  x
1      %state space model
2      m = 1000;
3      b = 50;
4
5      A = -b/m;
6      B = 1/m;
7      C = 1;
8      D = 0;
9
10     cruise_ss = ss(A,B,C,D);
11     % transfer function model
12     s = tf('s');
13     P_cruise = 1/(m*s+b);
```

Output:



```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\cruisecontro
Command Window
New to MATLAB? See resources for Getting Started.

cruise_ss =

    A =
        x1
    x1  -0.05

    B =
        u1
    x1  0.001

    C =
        x1
    y1   1

    D =
        u1
    y1   0

P_cruise =

        1
    -----
    1000 s + 50

Continuous-time transfer function.

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


2.4 Use case of Cruise Control

Adaptive Cruise Control in Autonomous Vehicles

Implementation:

In autonomous vehicles, adaptive cruise control (ACC) extends the basic cruise control functionality by dynamically adjusting the vehicle's speed based on the distance to the car ahead. This system uses radar or lidar sensors to continuously monitor the distance and relative speed of the vehicle in front. The adaptive cruise control system processes this information to modify the throttle and braking commands, ensuring the vehicle maintains a safe following distance while adapting to changes in traffic conditions. The ACC system integrates with other vehicle control systems, such as automatic braking and lane-keeping assist, to enhance overall driving safety and convenience.

Outcome:

The implementation of adaptive cruise control improves driving comfort and safety by reducing the need for manual speed adjustments in varying traffic conditions. Drivers experience less fatigue on long journeys as the system manages speed and distance, while also contributing to smoother traffic flow. The ACC system helps in maintaining a consistent speed and safe distance from other vehicles, reducing the likelihood of collisions and contributing to overall road safety.

SECTION 3

MOTOR SPEED

3.1 Introduction:

The speed of a DC motor is a critical parameter that determines its performance and efficiency. It is a measure of how fast the motor's shaft rotates, typically measured in revolutions per minute (RPM) or radians per second (rad/s). The speed of a DC motor is influenced by various factors, including the voltage applied, load torque, friction, and motor design and construction. Controlling the speed of a DC motor is essential in various applications, such as robotics, CNC machines, power tools, electric vehicles, and industrial automation, where precise control over the motor's operation is necessary for efficient and reliable performance. By regulating the speed of a DC motor, we can achieve optimal performance, reduce energy consumption, and extend the motor's lifespan. As such, understanding and controlling DC motor speed is a fundamental aspect of motor control and system design.

3.2 System Equations:

The torque generated by a DC motor is proportional to the armature current and the strength of the magnetic field.

$$T = K_t i$$

Where,

T: Motor torque (measured in Nm or lb-ft)

K_t: Motor torque constant (measured in Nm/A or lb-ft/A)

i: Armature current (measured in A)

The transfer function is,
$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2} \quad \left[\frac{\text{rad/sec}}{\text{V}} \right]$$

3.3 Implementation:

```
Editor - C:\Users\DELL 7490\Desktop\Controlssystem\motorspeed.m
+2 introduction.m x cruisecontrol.m x motorspeed.m x
1 % transfer function
2 J = 0.01;
3 b = 0.1;
4 K = 0.01;
5 R = 1;
6 L = 0.5;
7 s = tf('s');
8 P_motor = K/((J*s+b)*(L*s+R)+K^2)
9 %state space
10 A = [-b/J    K/J
11      -K/L    -R/L];
12 B = [0
13      1/L];
14 C = [1    0];
15 D = 0;
16 motor_ss = ss(A,B,C,D)
17 %
18 motor_ss = ss(P_motor)
```

Output:

```
Editor - C:\Users\DELL 7490\Desktop\Controlssystem\motorspeed.m
Command Window
New to MATLAB? See resources for Getting Started.
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
```

```
P_motor =

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

Continuous-time transfer function.
```

3.4 Use case of Motor Speed:

Robotic Arm Control

Motor speed control is crucial in robotic arms used for manufacturing and assembly tasks. For instance, a robotic arm equipped with DC motors can perform precise movements to assemble electronic components. In this application, controlling the motor speed is essential for accurate positioning and smooth operation. By adjusting the speed of the motors, the robotic arm can execute delicate tasks, such as inserting components into circuit boards, with high precision. The motor speed control ensures that the arm moves at the correct velocity, avoiding damage to fragile components and improving overall assembly efficiency.

Implementation and Outcome:

In practice, motor speed control for robotic arms can be implemented using a PID (Proportional-Integral-Derivative) controller. The PID controller adjusts the motor voltage based on feedback from encoders that measure the arm's position and speed. For example, if the arm needs to position a component at a specific angle, the PID controller continuously adjusts the motor voltage to correct any deviation from the desired position. This feedback loop allows the robotic arm to maintain accurate speed and position, resulting in high-quality assembly work and increased productivity. The outcome is a reliable and efficient robotic arm capable of performing complex tasks with precision and consistency, reducing manual labor and improving manufacturing quality.

SECTION 4

MOTOR POSITION

4.1 Theory:

The position of a DC motor is a fundamental aspect of its operation and a crucial parameter in control systems. It refers to the angular displacement of the motor's shaft, which determines the motor's rotational position and angular velocity. The DC motor position is a direct result of the interaction between the electrical inputs and the mechanical loads, making it a critical factor in determining the motor's performance and efficiency. In system modeling, the DC motor position is a key output variable that is influenced by various factors, including the input voltage, armature current, torque, and mechanical load. Understanding and controlling the DC motor position is essential in various applications, such as robotics, CNC machines, and power tools, where precise motion control is necessary for achieving desired outcomes.

4.2 System Equations:

The torque generated by a DC motor is proportional to the armature current and the strength of the magnetic field.

$$T = K_t i$$

Where,

T: Motor torque (measured in Nm or lb-ft)

K_t: Motor torque constant (measured in Nm/A or lb-ft/A)

i: Armature current (measured in A)

The state space representation is,

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

(10)

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

4.3 Implementation:

```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\motorposition.m
+2 introduction.m X cruisecontrol.m X motorspeed.m X
1 %transfer function
2 J = 3.2284E-6;
3 b = 3.5077E-6;
4 K = 0.0274;
5 R = 4;
6 L = 2.75E-6;
7 s = tf('s');
8 P_motor = K/(s*((J*s+b)*(L*s+R)+K^2))
9 %state space
10 A = [0 1 0
11      0 -b/J K/J
12      0 -K/L -R/L];
13 B = [0 ; 0 ; 1/L];
14 C = [1 0 0];
15 D = [0];
16
17 motor_ss = ss(A,B,C,D)
18 %
19 motor_ss = ss(P_motor)
```

Output:

```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\motorposition.m
Command Window
New to MATLAB? See resources for Getting Started.

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

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

```
P_motor =

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

Continuous-time transfer function.
```

4.4 Use case of Motor Position:

Automated CNC Machine

Motor position control is essential in automated CNC (Computer Numerical Control) machines used in manufacturing and machining processes. In these systems, precise motor position control is crucial for tasks such as cutting, drilling, and milling, where exact positioning of the machine tool is required. By accurately controlling the position of the CNC machine's spindle or tool, the system can perform intricate and detailed operations with high precision. This control ensures that each cut or hole is made according to the exact specifications provided in the CAD (Computer-Aided Design) files, improving the quality and consistency of the manufactured parts.

Implementation and Outcome:

The implementation of motor position control in CNC machines typically involves using high-resolution encoders and a PID controller. The encoders provide real-time feedback on the tool's position, which is compared to the desired position set by the machine's control software. The PID controller adjusts the motor's input to correct any deviation, ensuring that the tool remains on the intended path. This feedback loop allows for fine-tuned adjustments to motor positioning, resulting in highly accurate machining operations. The outcome is a CNC machine capable of producing complex parts with tight tolerances, enhancing manufacturing efficiency and reducing material waste. This precision also minimizes the need for manual adjustments and rework, leading to increased productivity and lower operational costs.

SECTION 5

SUSPENSION

5.1 Theory:

Suspension refers to the system of components that connect a vehicle's wheels to its frame or body, allowing for movement and shock absorption. It plays a crucial role in maintaining vehicle stability, handling, and ride comfort by absorbing bumps and vibrations from the road. The suspension system consists of various components, including springs, shock absorbers, struts, and control arms, which work together to maintain tire contact with the road and regulate vehicle motion. By dampening oscillations and resisting deflections, the suspension ensures a smooth ride, precise handling, and optimal vehicle performance. In addition, it helps to maintain proper tire alignment and prevent excessive wear, making it a vital component of a vehicle's overall safety and reliability.

5.2 System Equations:

The automotive suspension system is a complex control problem, which is simplified by using a 1/4 model, representing one wheel and the corresponding suspension components. This 1-D model consists of multiple spring-damper systems, including the wheel and tire, suspension spring, and damper, as well as an actuator that provides a control force, U , to regulate the motion of the vehicle body. The active suspension system aims to optimize ride comfort, handling, and stability by controlling the force transmitted to the vehicle body, mitigating the effects of road disturbances and oscillations. By actively managing the suspension motion, the system can improve vehicle performance, safety, and passenger comfort, making it an essential aspect of modern vehicle design.

Equation of motion is:

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

M_2 = Mass of the second object.

\ddot{X}_2 = Acceleration of the second object (second derivative of X_2 with respect to time)

b_1 = Damping coefficient related to the relative velocity between the first object and the second object.

\dot{X}_1 = Velocity of the first object (first derivative of X_1 with respect to time).

\dot{X}_2 = Velocity of the second object.

K_1 = Spring constant related to the relative position between the first object and the second object.

X_1 = Position of the first object.

X_2 = Position of the second object.

U = An external force or control input applied to the system. The negative sign indicates the direction of this force relative to the motion of M_2

To understand how the dynamic equations of a system can be transformed into transfer functions using the Laplace Transform, let's break down the process step-by-step. In control systems, transfer functions are used to describe the relationship between inputs and outputs in the Laplace domain, which simplifies the analysis of system dynamics.

The first transfer function is,

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

The second transfer function is,

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

5.3 Implementation:

```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\suspension.m
+1  | cruisecontrol.m | motorspeed.m | motorposition.m | suspension.m | invertedpendulum.m | aircraftpitch.m | ballbear
1   | %transfer function model
2   | M1 = 2500;
3   | M2 = 320;
4   | K1 = 80000;
5   | K2 = 500000;
6   | b1 = 350;
7   | b2 = 15020;
8   |
9   | s = tf('s');
10  | 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)*(b1*s+K1));
11  | 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)*(b1*s+K1));
```

Output:

```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\suspension.m
Command Window
New to MATLAB? See resources for Getting Started.

Trial License -- for use to evaluate programs for possible purchase as an end-user only.

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.
```

5.4 Use case of Suspension:

Active Suspension Control for Off-Road Vehicles

Implementation:

In an off-road vehicle, an active suspension control system is integrated to enhance the vehicle's performance over rough terrain. This system employs active suspension actuators that dynamically adjust the suspension stiffness and damping in response to real-time feedback from accelerometers and gyroscopes mounted on the vehicle. The control strategy is designed using a fuzzy logic controller that adapts to varying terrain conditions, such as rocks, mud, and uneven surfaces. The system actively adjusts the ride height and suspension settings to maintain optimal wheel contact and vehicle stability. The control inputs are derived from a combination of terrain sensors and vehicle speed data, ensuring that the suspension reacts promptly to changes in the driving environment.

Outcome:

The active suspension system significantly improves the off-road vehicle's capability by providing superior traction and stability on challenging terrains. The vehicle exhibits enhanced maneuverability and reduced bounce and sway, allowing it to navigate through rough and uneven surfaces more effectively. The real-time adjustments help maintain better wheel contact with the ground, improving overall vehicle control and safety. As a result, the vehicle can handle extreme off-road conditions with greater ease, offering a more comfortable and controlled ride for passengers. The active suspension system thus enhances the vehicle's off-road performance, making it more versatile and reliable in rugged environments.

SECTION 6

INVERTED PENDULUM

6.1 Theory:

The inverted pendulum system consists of a cart and a pendulum attached to it, with the objective of balancing the pendulum by applying a force to the cart. The system is unstable and nonlinear, making it a challenging control problem. The design requirements include settling time, rise time, and steady-state error constraints for both the pendulum's angle and the cart's position. The system equations are derived from force analysis and linearized about the vertically upward equilibrium position, resulting in two linearized equations of motion. These equations describe the dynamics of the system and will be used to design a control system to balance the pendulum and control the cart's position.

6.2 System Equations:

The given transfer function $P_{cart}(s)=X(s)/U(s)$ represents the relationship between the output $X(s)$ (the Laplace transform of the cart's position) and the input $U(s)$ (the Laplace transform of the applied force) for a cart-pendulum system. Let's break down the terms and variables involved in the transfer function:

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \quad \left[\frac{m}{N}\right]$$

M=Mass of the cart.

m=Mass of the pendulum.

l=Length from the pivot to the center of mass of the pendulum.

I=Moment of inertia of the pendulum about its center of mass.

b= Damping coefficient (friction) of the cart.

g=Acceleration due to gravity.

q: A term that combines the system parameters, defined as $q=I(M+m)+Mml^2$

The linearized equations of motion from above can also be represented in state-space form if they are rearranged into a series of first order differential equations. Since the equations are linear, they can then be put into the standard matrix form shown below.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

6.3 Implementation:

```

Editor - C:\Users\DELL 7490\Desktop\ControlSystem\invertedpendulum.m
+1 | cruisecontrol.m | motorspeed.m | motorposition.m | suspension.m | invertedpendulum.m | aircraftpitch.m | ballbeam.m |
1 | %transfer function
2 | M = 0.5;
3 | m = 0.2;
4 | b = 0.1;
5 | I = 0.006;
6 | g = 9.8;
7 | l = 0.3;
8 | q = (M+m)*(I+m*l^2)-(m*l)^2;
9 | s = tf('s');
10 |
11 | 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);
12 |
13 | 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);
14 |
15 | sys_tf = [P_cart ; P_pend];
16 |
17 | inputs = {'u'};
18 | outputs = {'x'; 'phi'};
19 |
20 | set(sys_tf, 'InputName', inputs)
21 | set(sys_tf, 'OutputName', outputs)
22 |
23 | sys_tf

```

Output:

```

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.

```

```

Editor - C:\Users\DELL 7490\Desktop\Controlsystem\invertedpendulum.m *
+1  | cruisecontrol.m | motorspeed.m | motorposition.m | suspension.m | invertedpendulum.m * |
1   | M = .5;
2   | m = 0.2;
3   | b = 0.1;
4   | I = 0.006;
5   | g = 9.8;
6   | l = 0.3;
7   | p = I*(M+m)+M*m*l^2; %denominator for the A and B matrices
8
9   | A = [0      1      0      0;
10  |      0 -(I+m*l^2)*b/p (m^2*g*l^2)/p 0;
11  |      0      0      0      1;
12  |      0 -(m*l*b)/p      m*g*l*(M+m)/p 0];
13  | B = [
14  |      0;
15  |      (I+m*l^2)/p;
16  |      0;
17  |      m*l/p];
18  | C = [1 0 0 0;
19  |      0 0 1 0];
20  | D = [0;
21  |      0];
22  | states = {'x' 'x_dot' 'phi' 'phi_dot'};
23  | inputs = {'u'};
24  | outputs = {'x'; 'phi'};
25  | sys_ss = ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs)

```

Output:

```

Editor - C:\Users\DELL 7490\Desktop\Controlsystem\invertedpendulum.m
Command Window
New to MATLAB? See resources for Getting Started.
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
fx

```

```

sys_tf =

From input "u" to output...
      1.818 s^2 + 4.845e-15 s - 44.55
x:  -----
      s^4 + 0.1818 s^3 - 31.18 s^2 - 4.455 s

      4.545 s - 7.774e-16
phi: -----
      s^3 + 0.1818 s^2 - 31.18 s - 4.455

Continuous-time transfer function.

```

6.4 Use case of Inverted Pendulum:

Balancing a Segway

The inverted pendulum model is crucial in designing the control system for a Segway, a two-wheeled, self-balancing personal transporter. The primary challenge in a Segway's operation is maintaining its balance, similar to balancing an inverted pendulum. The control system must constantly adjust the wheels' speed and direction to keep the platform upright, compensating for shifts in the rider's weight and external disturbances.

Implementation and Outcome:

To implement this, gyroscopic sensors and accelerometers are used to measure the tilt angle and angular velocity of the Segway. These measurements are fed into a control algorithm, often a PID controller or a more sophisticated state-space controller, which calculates the necessary adjustments to the wheels' speed and direction to maintain balance. The control system continuously monitors the Segway's orientation and applies corrective forces through the wheels to counteract any tilting.

SECTION 7

AIRCRAFT PITCH

7.1 Theory:

Aircraft pitch refers to the rotation of an aircraft around its lateral axis, which runs from wingtip to wingtip. It is a fundamental aspect of an aircraft's orientation and movement. Pitching motion involves the nose of the aircraft moving up or down, resulting in a change in the aircraft's angle of attack and altitude. The aircraft pitch control system is a complex problem, but by making certain assumptions, the equations of motion can be simplified and decoupled into longitudinal and lateral dynamics. The longitudinal dynamics govern the pitch of the aircraft, which is the focus of this example. The assumptions include steady-cruise at constant altitude and velocity, and a change in pitch angle not affecting the aircraft's speed. Under these assumptions, the longitudinal equations of motion are derived, resulting in two nonlinear coupled differential equations.

7.2 System Equations:

The transfer function for aircraft pitch is,

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

Recognizing the fact that the modeling equations above are already in the state-variable form, we can rewrite them as matrices as shown below.

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

Since our output is pitch angle, the output equation is the following.

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

7.3 Implementation:

```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\aircraftpitch.m
+1  | cruisecontrol.m | motorspeed.m | motorposition.m | suspensi
1   | %transfer function
2   | s = tf('s');
3   | P_pitch = (1.151*s+0.1774)/(s^3+0.739*s^2+0.921*s)
4   | %state space
5   | A = [-0.313 56.7 0; -0.0139 -0.426 0; 0 56.7 0];
6   | B = [0.232; 0.0203; 0];
7   | C = [0 0 1];
8   | D = [0];
9   | pitch_ss = ss(A,B,C,D)
```

Output:

```
P_pitch =

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

Continuous-time transfer function.
```

```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\aircraftpitch.m
Command Window
New to MATLAB? See resources for Getting Started.

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

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

7.4 Use case of Aircraft Pitch:

Maneuvering During Turbulence

A vital use case of aircraft pitch control is managing the aircraft's response to turbulence. Turbulence can cause sudden and unpredictable changes in the aircraft's altitude and attitude, leading to discomfort for passengers and potential safety hazards. An advanced pitch control system can help mitigate these effects by automatically adjusting the aircraft's pitch to counteract the disturbances caused by turbulent air. This system uses real-time data from accelerometers and gyroscopes to detect changes in pitch angle and vertical acceleration. By rapidly responding to these changes, the control system can smooth out the aircraft's trajectory and maintain a more stable flight path.

This use case is often implemented using a robust control approach, such as H-infinity (H_∞) control. H_∞ control is designed to handle uncertainties and disturbances within the system, making it well-suited for turbulence management. The control system continuously monitors the aircraft's motion and adjusts the pitch to minimize the impact of turbulence. For example, if the aircraft experiences an upward gust, the control system will command a slight nose-down adjustment to counteract the lift increase. The outcome is a reduction in the aircraft's vertical acceleration and oscillations, leading to a smoother ride for passengers and reduced structural stress on the aircraft. By effectively managing pitch during turbulence, this control system enhances passenger comfort and safety, ensuring a more pleasant and secure flying experience.

SECTION 8

BALL & BEAM

8.1 Theory:

The ball on the beam system is a classic control problem where a ball is allowed to roll along a beam with one degree of freedom. The beam's angle is controlled by a servo gear and a lever arm, which changes the angle of the beam, causing the ball to roll due to gravity. The system's parameters include the ball's mass, radius, and moment of inertia, as well as the lever arm offset, gravitational acceleration, beam length, and servo gear angle. The design criteria for the controller are to achieve a settling time of less than 3 seconds and an overshoot of less than 5%. This means that the controller should be able to move the ball to its desired position within 3 seconds, without overshooting the target position by more than 5%. The controller will manipulate the servo gear angle to change the beam's angle, which in turn will control the ball's position. By meeting these design criteria, the controller will be able to precisely control the ball's movement along the beam.

8.2 System Equations:

The equation represents the dynamics of the ball on the beam system, taking into account both the rotational inertia of the ball and its linear motion

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

Where,

R^2 : Rotational inertia of the ball adjusted for its radius.

m : Mass of the ball.

\ddot{r} : Linear acceleration of the ball.

$mgd/L\theta$: The component of gravitational force causing the ball to move along the tilted beam.

The transfer function is as follows:

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

8.3 Implementation:

```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\ballbeam.m
+1 | cruisecontrol.m | motorspeed.m | motorposition.m
1  %transfer function
2  m = 0.111;
3  R = 0.015;
4  g = -9.8;
5  L = 1.0;
6  d = 0.03;
7  J = 9.99e-6;
8
9  s = tf('s');
10 P_ball = -m*g*d/L/(J/R^2+m)/s^2
11 %state space
12 H = -m*g/(J/(R^2)+m);
13 A = [0 1 0 0
14      0 0 H 0
15      0 0 0 1
16      0 0 0 0];
17 B = [0 0 0 1]';
18 C = [1 0 0 0];
19 D = [0];
20 ball_ss = ss(A,B,C,D)
```

Output:

```
Editor - C:\Users\DELL 7490\Desktop\Controlsystem\ballbeam.m
```

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
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

```
P_ball =
```

```
0.21
----
s^2
```

```
Continuous-time transfer function.
```

fx

8.4 Use case of Ball & Beam:

Automatic Tilt Control for Solar Panels

Implementation:

In the Automatic Tilt Control for Solar Panels system, a PID-controlled servo motor adjusts the tilt of solar panels based on real-time data from sunlight sensors. The system continuously tracks the sun's position and optimizes the panel angle to maximize solar energy capture throughout the day. By automating this adjustment, the system enhances energy efficiency, reduces manual intervention, and improves overall performance, ensuring the panels are always oriented for optimal sunlight exposure.

Outcome:

The Automatic Tilt Control for Solar Panels system significantly boosts energy efficiency by increasing solar energy capture by up to 20-30% compared to fixed-panel systems. The automation of tilt adjustments minimizes manual labor, reduces maintenance, and ensures optimal panel orientation throughout the day. This results in enhanced overall performance and greater energy output, contributing to more efficient and reliable solar power generation.

SECTION 9

CONCLUSION

In conclusion, control systems, each application demonstrates the fundamental principles of control theory—feedback, stability, and precision—in unique contexts:

- **Cruise Control:** Enhances driving comfort and safety by maintaining a vehicle's speed with minimal driver intervention. It adjusts the throttle to keep the vehicle at a set speed despite changes in road conditions or incline.
- **Motor Speed and Position:** Control systems regulate the speed and position of motors with high accuracy, crucial for applications ranging from industrial machinery to robotics. They ensure smooth operation and precise control in various mechanical systems.
- **Inverted Pendulum:** Serves as a classic example of balancing and stabilization in dynamic systems. Control systems maintain the pendulum in an upright position, demonstrating advanced techniques in feedback control and dynamic stability.
- **Suspension Systems:** Improve vehicle ride quality and handling by adjusting the suspension settings in real-time. Control systems optimize comfort and performance by responding to road conditions and driving dynamics.
- **Aircraft Pitch:** Ensures stable and controlled flight by managing the pitch angle of the aircraft. Control systems adjust the control surfaces to maintain the desired trajectory and balance during various flight phases.
- **Ball and Beam:** Illustrates fundamental control concepts by adjusting the angle of the beam to control the position of a ball. This example highlights the application of dynamic modeling and feedback control in simple yet effective systems.

Overall, these systems showcase the versatility and importance of control theory in enhancing performance, stability, and automation across different fields. By applying sophisticated control strategies, engineers and technologists achieve more efficient, reliable, and precise outcomes in various real-world applications.