INTERNSHIP REPORT ON

CONTROL SYSTEMS

DATE OF SUBMISSION : 19-07-2024

BY : MANGALI RAMESH

Table of Contents for Control Systems Report

Summary:

This report covers the basics of control systems, which are essential for managing and regulating various dynamic systems in engineering. It explains key concepts, mathematical models, and practical applications.

The main goals are to understand how control systems work, create models using state-space and transfer functions, and implement these models with tools like MATLAB and Simulink.

1. Introduction

1.1 Background

Control systems are essential in modern engineering and technology. They automate processes, ensure stability, and enhance efficiency in various applications, from household appliances to industrial machinery and aerospace systems.

1.2 Importance of Control Systems

Control systems are critical for achieving desired performance, maintaining stability, and ensuring safety in engineering applications. They enable the precise regulation of system variables, enhancing efficiency and reliability.

1.3 Objectives of the Report

The primary objective of this report is to explore various control systems, including their modeling, implementation, and practical applications. We aim to provide a comprehensive understanding of control systems' principles and their real-world applications.

2. System Modelling

2.1 Overview of System Modelling

System modeling involves creating mathematical representations of physical systems to analyze and predict their behavior. These models can be used to simulate system dynamics and design appropriate control strategies. This section provides an overview of key concepts in system modeling, including state-space representation and transfer functions.

2.2 Mathematical Modelling

Mathematical modeling is the process of developing equations that describe the behavior of a system. Two primary methods for mathematical modeling in control systems are state-space representation and transfer functions.

2.3 State-Space Representation

State-space representation is a mathematical model that describes a system's dynamics using a set of first-order differential (or difference) equations. It is particularly useful for modeling multi-input multi-output (MIMO) systems.

The state-space model consists of two equations: the state equation and the output equation.

State Equation: $x(t) = Ax(t) + Bu(t)$

Output Equation: $y(t) = Cx(t) + Du(t)$

Where:

- X(t) is the state vector
- u(t) is the input vector
- y(t) is the output vector
- A is the system matrix
- B is the input matrix
- C is the output matrix
- D is the feedthrough (or direct transmission) matrix

2.4 Transfer Functions

A transfer function is a mathematical representation of the relationship between the input and output of a linear time-invariant system. It is expressed as a ratio of polynomials in the Laplace transform domain.

The transfer function G(s) of a system is defined as:

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

Where:

- Y(s) is the Laplace transform of the output y(t)
- U(s) is the Laplace transform of the input u(t)
- N(s) is the numerator polynomial
- D(s) is the denominator polynomial

Code:

introduction_1.m

```
m = 1;
k = 1;
b = 0.2;
F = 1;
A = [0 1; -k/m -b/m];
B = [0 1/m]';
C = [1 0];
D = [0];
sys = ss(A,B,C,D)

s = tf('s');
sys = 1/(m*s^2+b*s+k)

num = [1];
den = [m b k];
sys = tf(num,den)
```

Output:

```
>> introduction_1

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

Continuous-time transfer function.

Model Properties

sys =

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

Continuous-time transfer function.
```

3. Cruise Control

3.1 Introduction to Cruise Control Systems

Cruise control systems maintain a vehicle's speed by automatically adjusting the throttle. This section introduces the basic concepts and components of cruise control systems.

3.2 Modelling of Cruise Control Systems

This subsection covers the mathematical modeling of cruise control systems, including the development of transfer functions and state-space representations.

3.3 Implementation and Results

Details the implementation process of a cruise control system model in MATLAB/Simulink and presents the simulation results.

Code:

```
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:

```
>> cruise_control

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.
Model Properties

P_cruise =

       1
  -----------
  1000 s + 50

Continuous-time transfer function.
```

3.4 Use Case: Automotive Industry

Discusses how cruise control systems are applied in the automotive industry to enhance driver comfort and safety.

4. Motor Speed Control

4.1 Introduction to Motor Speed Control

Motor speed control systems regulate the speed of electric motors in various applications. This section introduces the principles and importance of motor speed control.

4.2 Modelling and Simulation

Covers the modeling and simulation of motor speed control systems, including the development of control algorithms.

4.3 Implementation and Results

Details the implementation process and presents the results of the motor speed control system.

Code:

```
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)

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)

motor_ss = ss(P_motor)
```

Output:

```
>> cruise_control

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.
Model Properties

P_cruise =

       1
  -----------
  1000 s + 50

Continuous-time transfer function.
```

Model Properties

```
motor_ss =

  A =
           x1       x2
   x1     -12  -5.005
   x2       4        0

  B =
        ul
   x1  0.5
   x2    0

  C =
       x1  x2
   y1   0   1

  D =
       ul
   y1   0

Continuous-time state-space model.
```

4.4 Use Case: Industrial Automation

Explores how motor speed control systems are utilized in industrial automation to improve process efficiency and precision.

5. Motor Position Control

5.1 Introduction to Motor Position Control

Motor position control systems regulate the position of motors, essential in robotics and manufacturing. This section introduces the concepts and applications of motor position control.

5.2 Modelling and Simulation

Covers the modeling and simulation techniques for motor position control systems.

5.3 Implementation and Results

Details the implementation process and presents the simulation results.

Code:

```
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))

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)
motor_ss = ss(P_motor)
```

Output:

```
>> motor_position

P_motor =

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

Continuous-time transfer function.
Model Properties

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

motor_ss =

  A =
              x1           x2          x3
   x1  -1.455e+06  -1.052e+04           0
   x2        8192           0           0
   x3           0           1           0

  B =
        u1
   x1  512
   x2    0
   x3    0

  C =
         x1     x2     x3
   y1     0      0  735.8

  D =
       u1
   y1   0

Continuous-time state-space model.
```

5.4 Use Case: Robotics

Discusses the application of motor position control systems in robotics for precise movement and positioning.

6. Suspension Control

6.1 Introduction to Suspension Systems

Suspension systems improve vehicle stability and comfort by absorbing shocks. This section introduces the basics of suspension control systems.

6.2 Modelling and Analysis

Covers the modeling and analysis of suspension systems, including the development of dynamic models.

6.3 Implementation and Results

Details the implementation process and presents the results of the suspension control system model.

Code:

```
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)*(b1*s+K1))
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:

```
>> suspension

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.
```

6.4 Use Case: Automotive Suspension Systems

Explores how suspension control systems are applied in the automotive industry to enhance ride quality and vehicle handling.

7. Inverted Pendulum Control

7.1 Introduction to Inverted Pendulum

The inverted pendulum is a classic control problem that involves balancing a pendulum in an upright position. This section introduces the principles and challenges of inverted pendulum control.

7.2 Modelling and Analysis

Covers the modeling and analysis techniques for the inverted pendulum system.

7.3 Implementation and Results

Details the implementation process and presents the results of the inverted pendulum control system.

Code:

```
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
```

```
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)
sys_tf = tf(sys_ss)
```

Output:

```
>> inverted_pendulum

q =

    0.0132


P_cart =

                  4.182e-06 s^2 - 0.0001025
  ------------------------------------------------------------
  2.3e-06 s^4 + 4.182e-07 s^3 - 7.172e-05 s^2 - 1.025e-05 s

Continuous-time transfer function.
Model Properties

P_pend =

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

Continuous-time transfer function.
Model Properties

sys_tf =

  From input to output...
                        4.182e-06 s^2 - 0.0001025
   1:  ------------------------------------------------------------
       2.3e-06 s^4 + 4.182e-07 s^3 - 7.172e-05 s^2 - 1.025e-05 s


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

Continuous-time transfer function.
Model Properties

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.
Model Properties
>> inverted_pendulum_transfer

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.
Model Properties

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.
```

7.4 Use Case: Educational Tools

Discusses how the inverted pendulum system is used as an educational tool to teach control system concepts.

8. Aircraft Pitch Control

8.1 Introduction to Aircraft Pitch Control

Aircraft pitch control systems regulate the pitch angle of an aircraft to ensure stable flight. This section introduces the principles of aircraft pitch control.

8.2 Modelling and Simulation

Covers the modeling and simulation techniques for aircraft pitch control systems.

8.3 Implementation and Results

Details the implementation process and presents the simulation results.

Code:

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

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:

```
>> aircraft_pitch

P_pitch =

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

Continuous-time transfer function.
Model Properties

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.
```

8.4 Use Case: Aerospace Engineering

Explores the application of pitch control systems in aerospace engineering to enhance flight stability and performance.

9. Ball and Beam Control

9.1 Introduction to Ball and Beam System

The ball and beam system involves balancing a ball on a beam and is a classic control problem. This section introduces the principles of ball and beam control.

9.2 Modelling and Analysis

Covers the modeling and analysis techniques for the ball and beam system.

9.3 Implementation and Results

Details the implementation process and presents the results of the ball and beam control system.

Code:

```
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


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:

```
P_ball =

  0.21
  ----
  s^2

Continuous-time transfer function.
Model Properties

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

Continuous-time state-space model.
```

9.4 Use Case: Control System Education

Discusses how the ball and beam system is used in control system education to demonstrate fundamental control concepts.

10. Conclusion

This report has explained the basics of control systems and their importance in engineering. We looked at how control systems help manage and regulate various processes. By studying state-space and transfer function models, we saw how these tools help analyze and design control systems.

Using MATLAB and Simulink, we showed how to implement these models in real-world applications. We learned how different control strategies affect system performance and why accurate modeling is crucial.

Control systems are essential in many areas, from cars to factories. They improve efficiency, stability, and performance. The practical examples and video demonstrations highlighted their real-world benefits.

In summary, control systems are vital for managing complex systems. This report provides a foundation for further study and application, encouraging ongoing learning and innovation in control systems. Future work can explore advanced control techniques and new technologies to improve control systems even more.