Mechatronics Lab - Exercise 1 : Mathematical Modelling of a Pendulum over a Cart

Contents

Creating the Modelling Parameters

```
M = 0.5; %Mass of the Cart
m = 0.2; %Mass o fthe Pendulum
b = 0.1; %Damping Coef
g = 9.8; %Gravity
I = 0.006; %Inertia
l = 0.3; %Length of Pendulum
disp("Created the Modelling Parameters")
```

```
Created the Modelling Parameters
```

Task 1: Deriving the Transfer Function and State Space model and representing in MATLAB

Task 1.1: Transfer Functions Model

```
s = tf('s'); %Creating the 's' variable required in a transfer function
q = (M+m)*(I+m*l^2)-(m*l)^2; %Creating a 'q' variable for easy calculations

%Delcaring the Transfer Functions of the Inverted Pendulum System

%Transfer Function of the Cart
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*q*l*s/q);

%Transfer function of the Pendulum
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);

%Visulalising the Transfer Function
sys_tf=[P_cart;P_pend];
inputs={'u'};
outputs={'x';'phi'};
set(sys_tf,'InputName',inputs);
set(sys_tf,'OutputName',outputs);
disp("The complete TF Equation is : ");
sys_tf
```

```
The complete TF Equation is :

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 - 1.38e-08 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.
```

Task 1.2: State Space Model

```
p=I*(M+m)+M*m*l^2; %Creating a 'p' variable to replace all denominators in State Matrix.

%State Equation
```

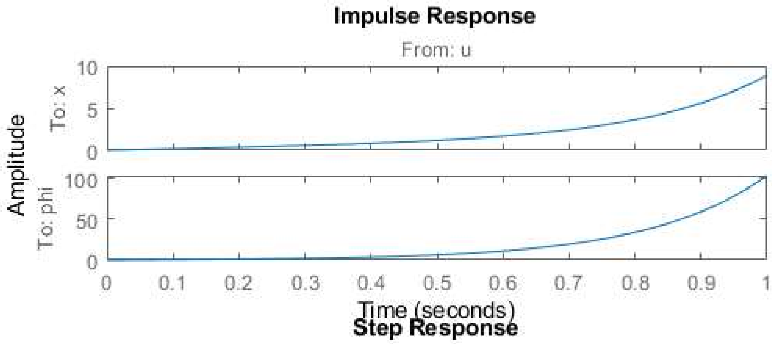
```
A=[0 1 0 0;0 -((I+m*l^2)*b)/p (m^2*g*l^2)/p 0;0 0 0 1;0 -(m*l*b)/p (m*g*l*(M+m))/p 0]; %State Equation
B=[0;(I+m*l^2)/p;0;(m*l)/p]; %Input Equation

%Output Equation
C=[1 0 0 0;0 0 1 0];
D=[0;0];

states = {'x' 'x_dot' 'phi' 'phi_dot'}; %Where, x = Position
                                        %       x_dot = velocity
                                        %       phi = Angle of Pendulum
                                        %       phi_dot = derivative of phi

% Visualising the State Space Equations
inputs = {'u'};
outputs = {'x'; 'phi'};
sys_ss = ss(A,B,C,D,'statename', states,'InputName', inputs,'OutputName', outputs);
disp("The state space model is:")
sys_ss
```

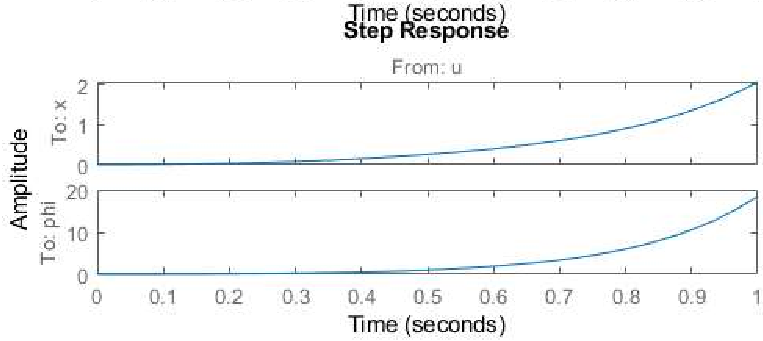
```
The state space model is:

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.
```

Task 2: Open Loop Impulse Response and Step Response of the System

```
t=0:0.01:1; %Specifying Time steps

disp("Plotting the responses")
%Impulse Response
figure('Name','Open Loop Response of Inverted Pendulum');
subplot(2,1,1);
impulse(sys_tf,t)
%Step Response
subplot(2,1,2);
step(sys_tf,t)

%{ Inference : From the graphs shown we can clearly understand that the system is not stable. Hence we go for a Closed Loop PID control. %}
```

```
Plotting the responses
```

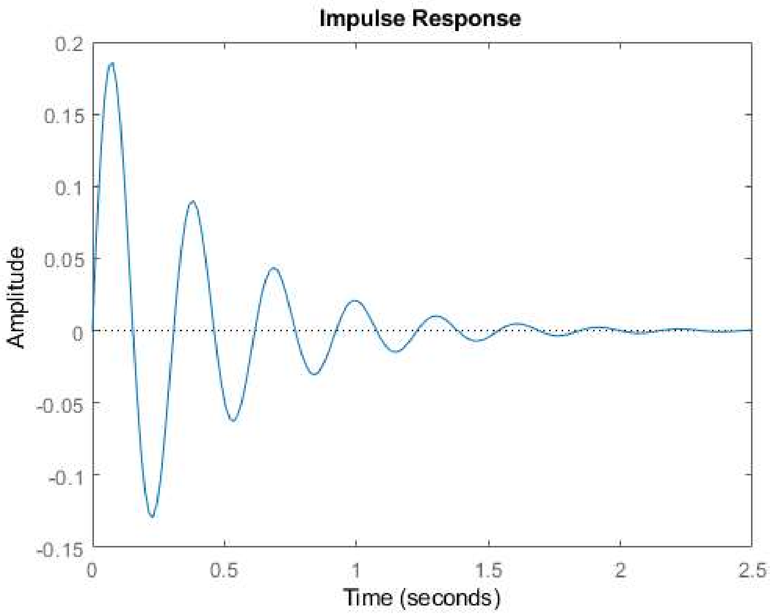




Task 3: PID Controller Design

```
%Initialise the Parameters.
disp("Implementing the PID Controller")
%kp = input('Enter the Proportional Constant Value:');
%ki = input('Enter the Integral Constant Values:');
%kd = input('Enter the Derivative Constant Value:');
kp = 100;
ki = 1;
kd = 1;
%Visualise the PID Controller.
cntr = pid(kp,ki,kd)

%Checking the stability of the system with given PID control Parametrs.
clsys= feedback(P_pend,cntr)

figure('Name','Response after Implementing PID Control');
impulse(clsys)

%{Inference : We can say that the system (now closed loop) after the implementation of a PID controller is stable. %}
disp("End")
```

```
Implementing the PID Controller

cntr =

             1
  Kp + Ki * --- + Kd * s
             s

  with Kp = 100, Ki = 1, Kd = 1

Continuous-time PID controller in parallel form.


clsys =

                        1.045e-05 s^2
  ---------------------------------------------------------
  2.3e-06 s^4 + 1.087e-05 s^3 + 0.0009737 s^2 + 2.091e-07 s

Continuous-time transfer function.

End
```


Impulse Response
Amplitude
Time (seconds)
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
0.5
1
1.5
2
2.5