SELF- BALANCING ROBOT

PRAMOD KESHAV PES1201700582
AJAY VICTOR PES1201701170
PRASHANTH B PES1201701729

ABSTRACT

A Self Balancing robot is a 2 wheeled robot that balances itself using a closed loop feedback system. This concept combines concepts of both robotics as well as control systems to realise a system capable of maintaining its balance on a surface in real time. In our approach we have used the concepts of An Inverted pendulum setup placed on a cart to estimate the equivalent kinematic equations, a transfer function/state space model and Matlab/Simulink and Simscape based simulations in which a PID controller is designed in order to make the robot successfully maintain its balance.
In addition to the PID Controller design, we have explained another method of control called State-Feedback control which can be done through Ackermann's formula and LQR control.

The MATLAB code written has been modularised entirely with suitable functions. Subsystems have been developed for Simulink and Simscape based simulations for better understanding.

Keywords: Self-Balancing robot, Closed loop system, Inverted Pendulum, Transfer function,State Space model, Modularisation,Subsystems,PID Controller, State-Feedback, Ackermann's formula, LQR Control,Matlab,Simulink,Simscape;

INTRODUCTION

As the term 'self balancing' implies the robot must be able to maintain both its position and orientation in an upright position constant over a given period of time without losing balance and toppling over.
In our study we have considered a two wheeled robot. Two wheeled robots achieve better mobility and rotation in confined or narrow spaces compared to humanoid robots.For this reason these bots are widely used in mobile robot platforms. However to maintain its balance a two wheeled robot must use the movement of its wheels and tilting its body to compensate for any external force that it might incur in its normal working environment. If the external force exceeds the response capability of the robot, then the robot loses balance.
Thus, to ensure optimum working of a two wheeled robot we must design and realise a system that would account for all the external factors that the robot might face in a dynamic real-time environment to ensure that it does not lose balance and topple over.

Inverted Pendulum-Cart System

An inverted pendulum is a pendulum which has its centre of mass(CoM) at the

pivot point. It is unstable and will fall over without the addition of a control over it.
It is a classic automation problem that has numerous theoretical approaches and several practical applications.
The main objective of using this system will be to control or stabilize the pendulum angle and velocity as well as analysing the effect of control of the pendulum on the cart position.

In this report, the concept of the inverted pendulum on a cart will be exploited to simulate the working of a small-scale self-balancing robot.

PID

It's well known to us that the PID controller is a very reliable control technique according to many characteristics such as the very satisfying performance with the tuning methods with any linear-system, low cost, dealing with it is simpler than other techniques and very limited maintenance. Due to its simplicity, robustness and successful practical application, PID (Proportional-Integral-Derivative) controllers have become the most widely used controller in the industry. The problem which arises with the PID technique is the non-linear systems, the problem of affecting the speed after adding any additional loads, suffering from changing dynamics after a long time operation which will be very difficult to be covered with a fixed PID controller.

Main Advantages of using PID Controllers:

1. They improve the steady-state accuracy by decreasing the steady state error.
2. As the steady-state accuracy improves, the stability also improves.
3. PID controllers also help in reducing the unwanted offsets produced by the system.
4. They can control the maximum overshoot of the system by adding the derivative gain to the system.
5. They can help in reducing the noise signals produced by the system.
6. They can also help to speed up the slow response of an overdamped system.

This paper will utilise the PID controller as a method of control design to stabilize the plant i.e the inverted pendulum-cart system. PID Controller has been designed and the effect of the control over the pendulum angle and the cart position has been analysed thoroughly through the help of softwares like MATLAB, Simulink and Simscape.

SIMSCAPE

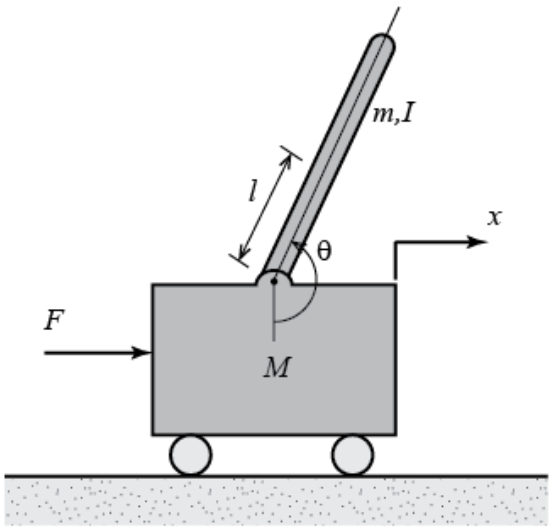
Simscape is an add-on present in SIMULINK which enables the user to create complex models of physical systems.With Simscape, you build physical component models based on physical connections that directly integrate with block diagrams and other

modeling paradigms. It helped us develop the control systems aspect of our project and test the system-level performance.

Objective of the Project

We have already established that the inverted pendulum is unstable without control and will fall over,unless we make the cart move to balance it.
Additionally, the dynamics of the system here is non-linear. Hence, a suitable dynamic model with equations will be derived and modelled.
The objective of the control system modelled, will be to balance the inverted pendulum by applying a force to the cart that the pendulum is attached to.

Design Criteria and Requirements



This is a two-dimensional problem where the pendulum is constrained to move in the vertical plane as shown in the figure.
For this system, the control input is Force F, which moves the cart horizontally, and the outputs are the angular position of the pendulum Θ and the horizontal position of the cart *x*.
The quantities of the constants used in the system are:

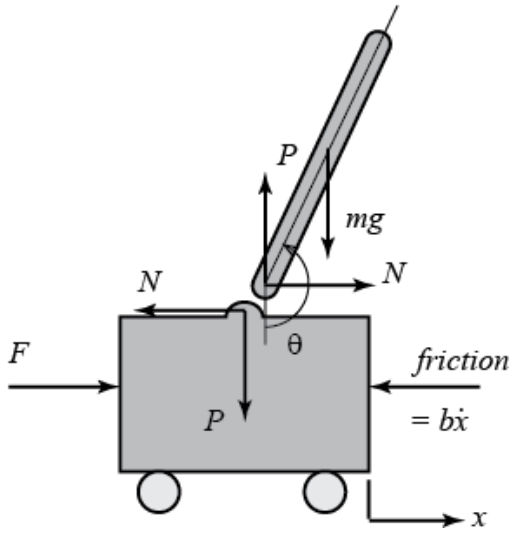
M	mass of cart	0.5 kg
m	mass of the pendulum	0.2kg
b	coefficient of friction of the cart	0.1 N/m/sec
l	length to pendulum centre of mass	0.3 m
I	Mass moment of inertia of pendulum	0.006 kgm^2

The pendulum will initially begin moving from the upward equilibrium point, Θ= π.
The design requirements for the system are:
1. Settling time for Θ less than 5 seconds.
2. Pendulum angle: never more than 0.05 radians from the vertical→ Maximum overshoot.

NOTE: For the control aspect design of the system, only the control of the pendulum's position and velocity will be dealt with. This is because the techniques used for the control are best suited for Single input Single output(SISO) systems. Hence, none of the design criteria deal with the cart's position. However, the controller's effect on the cart position after the controller design has been analysed.

System Modelling: Force Analysis and System Equations

The free-body diagram(FBD) of the two components of the inverted pendulum system is as follows:



Summing the forces in the FBD of the cart in the horizontal direction, we get:

$$M\ddot{x} + b\dot{x} + N = F \longrightarrow (1)$$

Summing the forces in the FBD of the pendulum in the horizontal direction, reaction force N is given by:

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \longrightarrow (2)$$

From equations 1 and 2, we get one of the two main governing equations of the system:

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \longrightarrow (3)$$

To get the second equation of motion of the system, summing the forces in the perpendicular direction, we get:

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \longrightarrow (4)$$

Summing the moments about the centroid of the pendulum, we get:

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \longrightarrow (5)$$

Combining these two equations, we get the final equation as:

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \longrightarrow (6)$$

These equations are nonlinear and since the control design is w.r.t linear systems, we have linearised the system.

Steps to linearize the following equations:

1. Linearize about the vertical equilibrium position $\Theta = \pi$.
2. Let Φ represent the deviation of the pendulum's position from the equilibrium, given by: $\Theta = \pi + \Phi$.
3. Using the small angle approximations of the nonlinear functions in our system equations:

$$\theta = \pi + \Phi$$
$$\cos\theta = \cos(\pi + \Phi) = -1$$
$$\sin\theta = \sin(\pi + \Phi) = -\Phi$$
$$\left(\frac{d\theta}{dt}\right)^2 = 0;$$

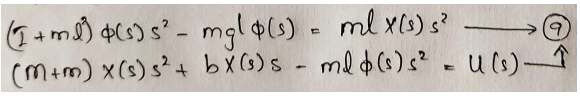
After substituting these approximations into our nonlinear equations of motion(eq. 3 and 6), we get:

$$(I + ml^2)\ddot{\phi} + (-mgl\phi) = ml\ddot{x} \longrightarrow (7)$$
$$(m+m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \longrightarrow (8)$$

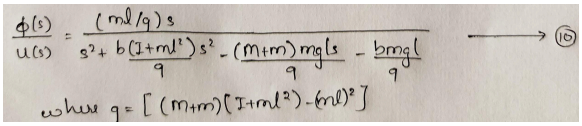
These are the linear equations of motion.

Transfer function Model

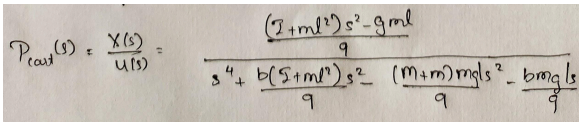
Taking Laplace transform of the linear equations of motion, assuming zero-initial conditions, will yield:



After solving for a single-input[U(s)] and single-output[Φ(s)], the transfer function for the pendulum is given by:



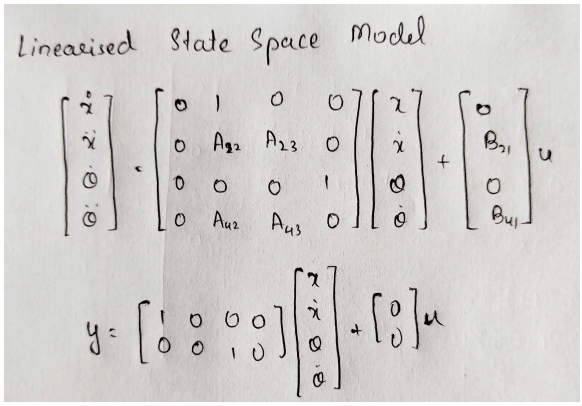
Similarly solving for the cart position[X(s)], we get the transfer function for the cart as follows:

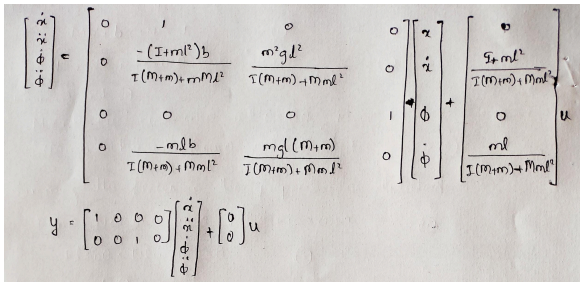


This concludes the Transfer function modelling.

State-Space Modelling

The linearized equations of motion from the previous section, can also be represented in the form of state-space after rearranging them in a series of first-order differential equations. The equations can be put in matrix form as follows:





Matlab Representation of Transfer function and State-space models

The entire code has been modularized completely, for efficiency and better understanding.

The user defined function **transfer_function** will give the same output as obtained before:

```
function [P_cart, P_pend, sys_tf] = transfer_function(M, m, b, I, g, l)
    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 in transfer function form:

```
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-0

                               1.045e-05 s
   phi:  ----------------------------------------------------
         2.3e-06 s^3 + 4.182e-07 s^2 - 7.172e-05 s - 1.025e-0

Continuous-time transfer function.
```

Similarly, a user-defined function called **State_space_model** was written which gives the same output in state-space form:

```
function [A, B, C, D, sys_tf] = State_Space_model(M, m, b, I, g, l)
    q = (M+m)*(I+m*l^2)-(m*l)^2;
    s = tf('s');
     A = [0      1              0           0;
      0 -(I+m*l^2)*b/q   (m^2*g*l^2)/q   0;
      0      0              0           1;
      0 -(m*l*b)/q       m*g*l*(M+m)/q  0];

    B = [     0;
          (I+m*l^2)/q;
               0;
            m*l/q];
    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',o
    sys_tf=tf(sys_ss)
end
```

Output in State-space form:

```
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.
```

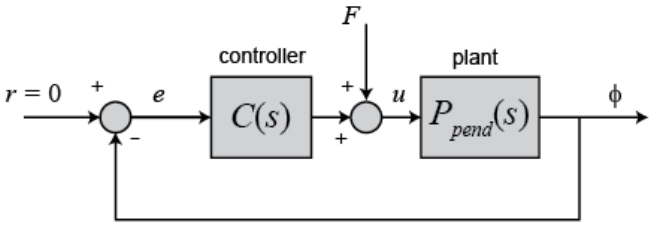
The next section describes the Controller design for the system shown here:

Controller Design

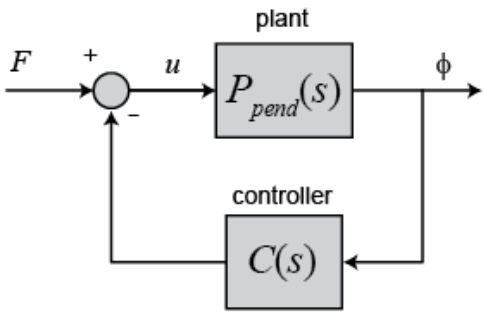
In this section, a PID controller has been designed for the inverted-pendulum, considering a SISO system, without controlling the cart's position.

The structure of the controller for this model is different because we are attempting to control the pendulum's position,which should return to the vertical after the initial disturbance, so the reference signal to be tracked is zero. This type of situation is often called a Regulator problem. The external force applied to the cart is considered as an impulse disturbance.

The block diagram for this system is as follows:



For easier analysis and design of the system, the modified block diagram is as follows:



The resulting transfer function T(s) for the closed-loop system from an input force(F) to an output of pendulum angle(Φ) is given as:

$$T(s)=\frac{\Phi(s)}{F(s)}=\frac{Ppend(s)}{1+C(s)Ppend(s)}$$

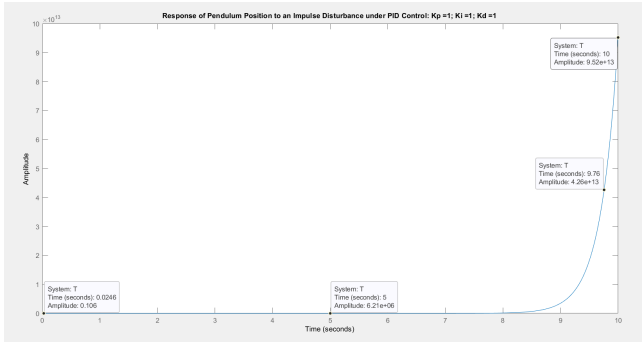
Matlab Code for PID Controller

A user defined function called **calculate_pid** was written which plots the response of the pendulum system to different sets of PID i.e Kp,Ki and Kd values.

```
function [] = calculate_pid(Kp, Ki, Kd)
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_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);

    if (length(Kp) == length(Ki) && length(Ki) == length(Kd))
        n = length(Kp);
        for i = 1:n
            X = [' Case: ',num2str(i)];
            disp(X);
            C = pid(Kp(i),Ki(i),Kd(i));
            T=feedback(P_pend,C);
            figure(i);
            t=0:0.01:10;
            impulse(T,t);grid on;
            title(['Response of Pendulum Position to an Impulse Disturbance',' under PID Control:
                num2str(Kp(i)), '; Ki =', num2str(Ki(i)), '; Kd =', num2str(Kd(i))]);
        end
    else
        disp('Invalid input');
    end
```

The values for Kp,Ki and Kd were manually tuned in MATLAB through trial-and-error. Outputs for three specific cases obtained through the code have been put up here with relevant explanation. The outputs are the responses of the pendulum after applying an impulsive force on the cart.
The impulse response was plotted using the command: **impulse(T,t);**

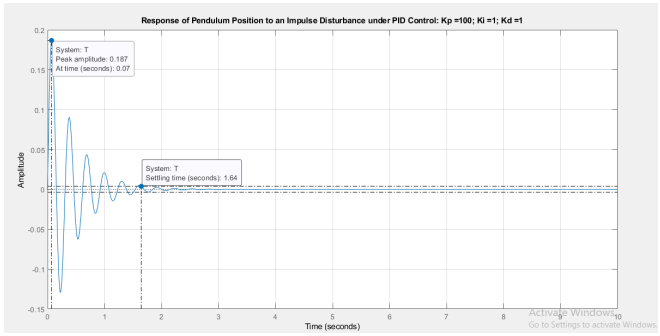
Case 1:Kp=1;Ki=1;Kd=1



This was the first set of values used to control the system. As seen from the graph, the system is not yet stabilized i.e it is still unstable.

So, the proportional gain Kp was modified to alter the response. After some modifications, we arrived at this particular case.

Case 2: Kp=100;Ki=1;Kd=1

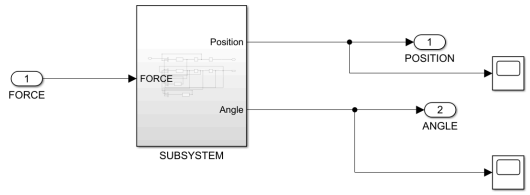


As can be observed from the graph, the system is now stabilised.
The settling time for the pendulum is 1.64 seconds which is less than the required 5 seconds criterion.

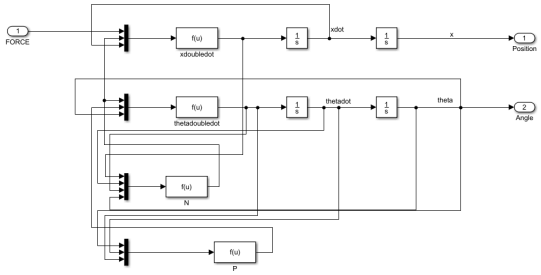
But the max overshoot or the deviation of the pendulum from the vertical is at 0.187 amplitude which is much more than the required 0.05 value.

So, to reduce the maximum overshoot, the derivative gain Kd was increased.

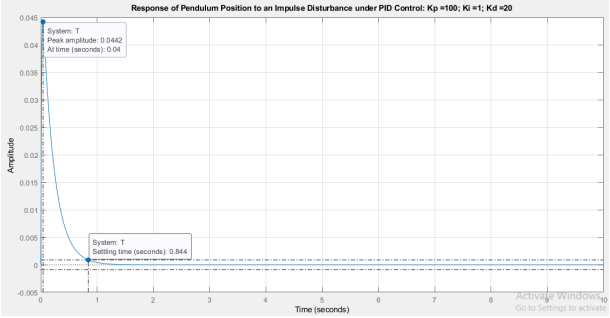
The output for the final case is as follows:



Subsystem:



Case 3: Kp=100;Ki=1;Kd=20

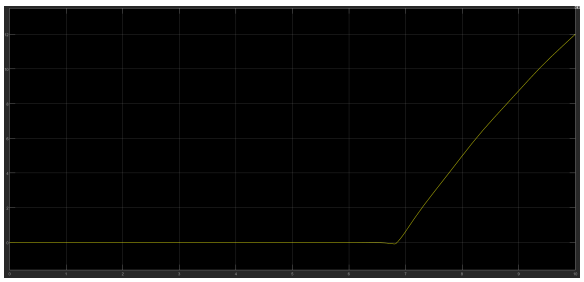


This graph shows the optimum response of the pendulum position. As can be seen from the characteristics, the maximum overshoot is limited to 0.0442 radians which satisfies the criterion. The settling time remains the same at 1.64 seconds, thus satisfying all the required design criteria.

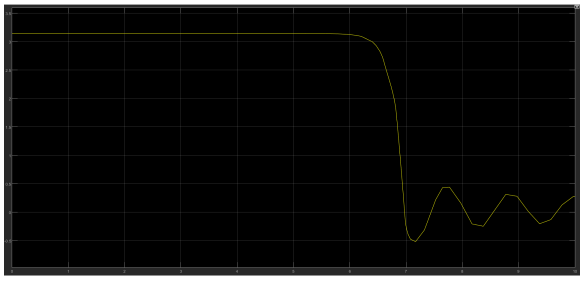
This ends the section on controller design using MATLAB.

Output:

Position



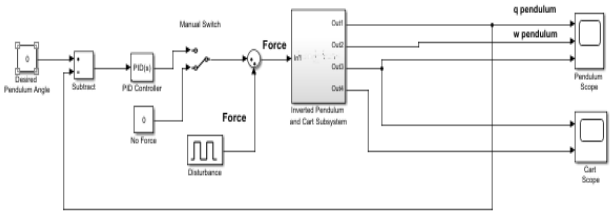
Angle



SIMSCAPE model of the System with and without Controller:

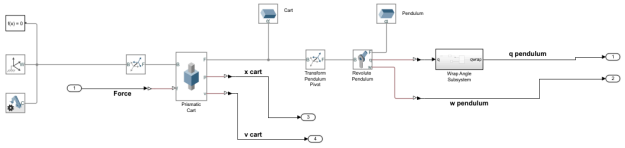
As has already been established in the Introduction section, we have used Simscape

to model the inverted pendulum-cart system using physical blocks.
The overall system is as shown below:



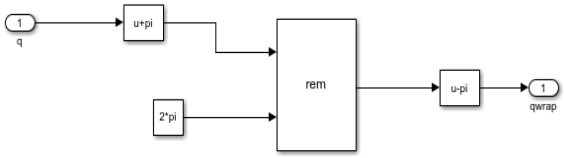
1. The desired position angle is represented through a constant block with constant fixed at 0.
2. The Subtract block is present as a means of subtracting the feedback from the desired angle to result in the error signal which is in turn given to the PID block, which is the inbuilt block present in the SIMULINK library browser.
    a. The in-built PID block actually has 4 parameters namely the Kp,Ki and Kd values along with a coefficient called filter coefficient.
    b. If the PID function is observed in the block, the derivative part of the function is associated with a low pass filter.This is a note-worthy point.

A manual switch was used to toggle between open-loop and a closed-loop system with a controller.
The pulse generator represents the impulsive force acting on the cart.
The **inverted Pendulum and Cart subsystem** shows the intricacies of the physical model created. It is as shown below:



1. The solver configuration was kept to auto mode.
2. Blocks representing the prismatic movement of the cart and the rotating motion of the pendulum were added and the parameters altered suitably.
3. Blocks representing masses of the pendulum and cart are represented by the same names.

Another subsystem called **Wrap Angle** subsystem was created because the measured angle of the pendulum had to be limited to -π to π radians.
This subsystem is as shown:

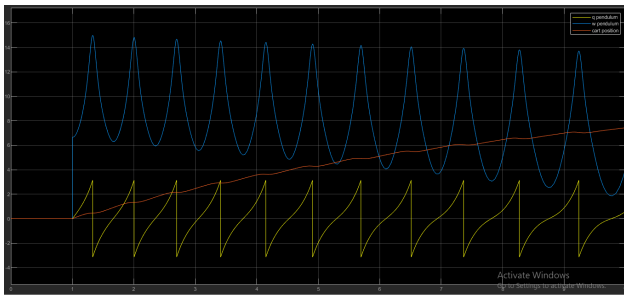


To limit the pendulum angle as mentioned above, π radians was added to the measurement using the bias block present in the diagram. Then the remainder was found out by dividing this sum by 2π and the remainder was passed through another bias block, where it was subtracted by π radians, thereby achieving our objective for this subsystem.

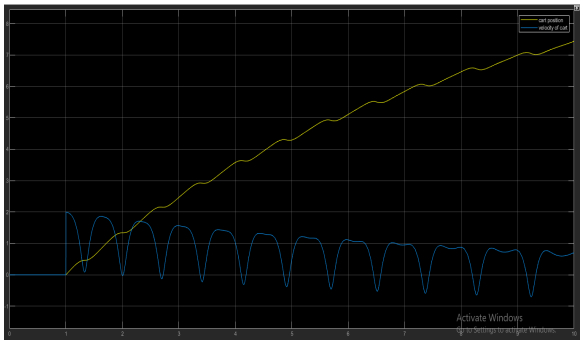
RESULTS for SimScape:
This section presents the results obtained through graphs and the animation

screenshots obtained from the SimMechanics Explorer.

Open-Loop Behaviour



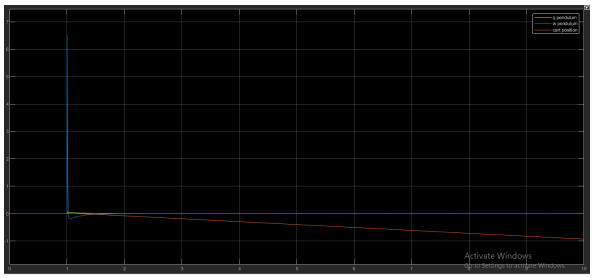
This graph shows the outputs of the pendulum i.e q pendulum→ yellow graph shows the position of the pendulum; w pendulum→ blue graph shows the angular velocity of the pendulum; cart position→ red graph shows the cart position, to compare the cart position with that of the pendulum. There is a fluctuation in the q and w pendulum, meaning, that the pendulum is trying to balance itself on the cart, but the position of the cart is increasing gradually indicating that it is moving in the positive direction i.e towards the right. Thus, the system is not getting balanced. This shows the open-loop behaviour of the system.



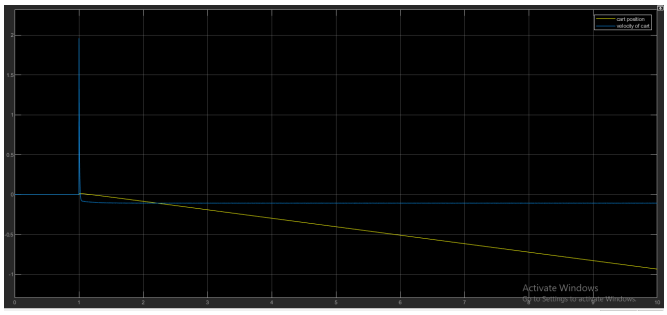
This graph shows the cart velocity → blue graph and the cart position→ yellow graph,which is the same as explained in the previous graph. The velocity of the cart is increasing and decreasing as can be observed in the graph, and not balancing itself at the end of the simulation time which was 10 seconds.

Closed-Loop Behaviour with PID

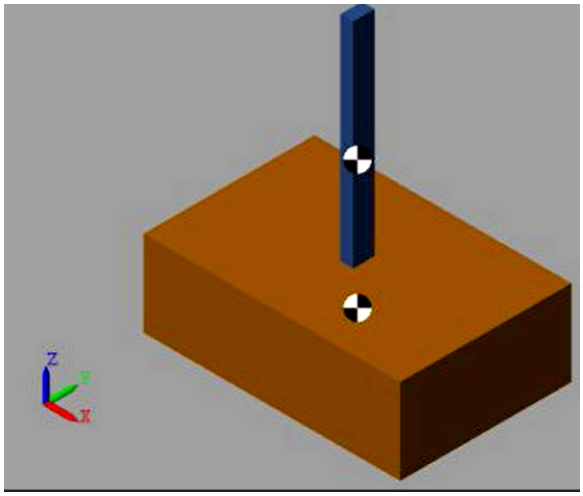
Running the simulation with the manual switch switched to the PID controller, these two graphs were obtained.



The notation for the graph is the same as that of the open-loop. But it can be observed here that there are no fluctuations like before. Instead, the system is balanced. After the initial impact,the controller was able to quickly bring down the pendulum angle to zero and the pendulum velocity is zero indicating the pendulum is balanced.
The position of the cart indicates that the cart is moving slowly with constant velocity in the negative X direction to keep the pendulum balanced.



This graph indicates the position and velocity of the cart. The cart position remains the same as before and the velocity of the cart is increasing at the start due to the impulse disturbance, but later is made constant.



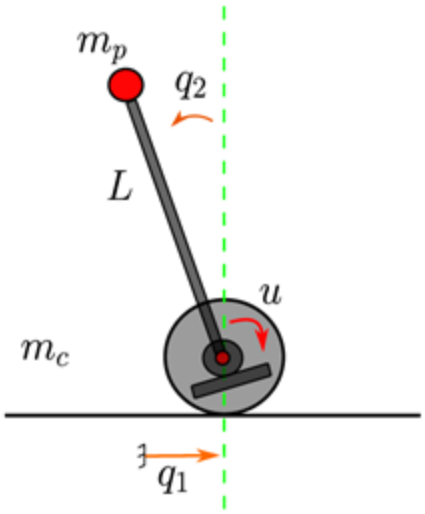
This screenshot shows the balanced state of the pendulum-cart system with their CoM indicated in the figure. The animation videos for the same can be accessed through the link provided at the start of the document.

This ends the section on the PID controller design along with the respective results obtained.

In addition to the traditional PID Control design, another kind of control known as Linear Quadratic Regulator(LQR) was designed with the help of a course assignment from **Technische Universitat Kaiserslautern(TUK)**.
We took resources from this assignment, tried to control the Inverted pendulum-cart system. This has been presented as a proof-of-concept for our project.

State Space Controller
Credits: Alen Turnwlad (ETI TUK)

Here we consider a simplified model of the Pendulum-Cart system as shown below where the point mass $\mathbf{m_p}$ is connected to the cart with the mass $\mathbf{m_c}$ via a massless arm with the length **L**. $\mathbf{q_1}$ is the displacement of the cart and $\mathbf{q_2}$ is the angle of the pendulum. As input, u is the force applied to the system.



Here we will handle the stabilization problem around the upper position of the pendulum. We have only considered the linearized model.

Dynamic Model of the System

The Equations of motion of the above system can be derived using methods such as Lagrange-Formulations or Newton-Euler Dynamic equations.

From this we obtain

$$M(q)\,\ddot{q} + h(q,\dot{q}) = g_q\,u$$

with

$$M(q) = \begin{pmatrix} m_c + m_p & -L\,m_p\cos q_2 \\ -L\,m_p\cos q_2 & L^2\,m_p \end{pmatrix}$$

$$h(q,\dot{q}) = \begin{pmatrix} L\,m_p\,\dot{q}_2^2\sin q_2 \\ -L\,g\,m_p\sin q_2 \end{pmatrix} + \begin{pmatrix} d_1\,\dot{q}_1 \\ d_2\,\dot{q}_2 \end{pmatrix},\quad g_q = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Here d1 and d2 are known as damping factors representing friction in both the cart velocity and angular joint velocity respectively.

We consider a state representation as

$$x = \begin{pmatrix} q \\ \dot{q} \end{pmatrix}$$

Here q = is q1 and q2 which is position and angular displacement; q1˙,q2˙ are the linear and angular velocities.

$$\dot{x} = F(x) + G(x)\,u = \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ -M^{-1}(q)\,h(q,\dot{q}) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ M^{-1}(q \end{pmatrix}$$

Linearizing about q1*,q2* = 0 we get

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

with

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{g\,m_p}{m_c} & -\frac{d_1}{m_c} & -\frac{d_2}{L\,m_c} \\ 0 & \frac{g\,(m_c+m_p)}{L\,m_c} & -\frac{d_1}{L\,m_c} & -\frac{d_2\,(m_c+m_p)}{L^2\,m_c\,m_p} \end{pmatrix},\quad B = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{m_c} \\ \frac{1}{L\,m_c} \end{pmatrix}$$

MATLAB modelling

For the above system we use the following constant values as shown

```
%% Define the parameters
mc = 1.5; % Mass of cart
mp = 0.5; % Mass of Pendulum
g = 9.8; % Earth gravity
L = 1; % length of pendulum
d1 = 1e-2; % damping of cart displacement
d2 = 1e-2; % damping in joint
```

State Space model as given below

```
%% Define the matrices

A = [0                0                    1                          0;
     0                0                    0                          1;
     0           (g*mp)/mc                -d1/mc                  -d2/(L*mc);
     0      (g*(mc + mp))/(L*mc)          -d1/(L*mc)       -(d2*mc + d2*mp)/(L^2*mc*mp)];

B = [     0;       0;       1/mc;       1/(L*mc)];
```

Output considered

```
%% Output
%C = [0;1;0;0]; % q_2 as output, not Observable
C = [1;0;0;0]; % q_1 as output
D = 0;
```

As seen from above, for the angular displacement as output, the System is not Observable. Thus we consider position displacement as output so as to design a controller for a system, it will make the system to be both Controllable and Observable.

Building the System

Using inbuilt MATLAB function ‘ss’ we can enter the A, B, C and D matrices of a state space model to create a state space model of a given system.

```
%% Build System
sys = ss(A,B,C',D)
x0 = [0; 5*pi/180; 0;0];
```

The Controllability, Observability and the location of the poles of the system can be tested by using inbuilt function .

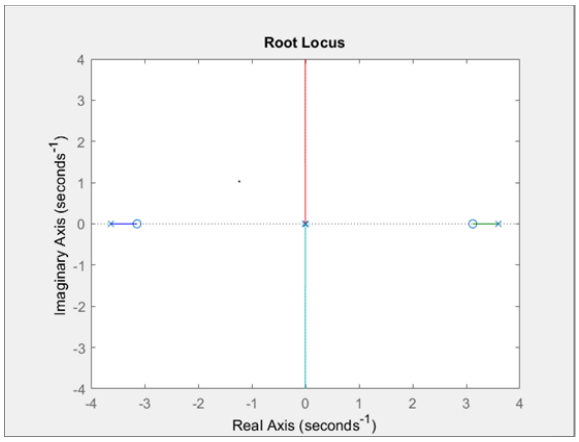
```
Sc = ctrb(sys)
So = obsv(sys)
rlocus(sys)
```

Using above commands, we obtain Controllability and Observability Matrix,

```
Sc =

         0    0.6667   -0.0089    2.1780
         0    0.6667   -0.0222    8.7118
    0.6667   -0.0089    2.1780   -0.1452
    0.6667   -0.0222    8.7118   -0.5372


So =

    1.0000         0         0         0
         0         0    1.0000         0
         0    3.2667   -0.0067   -0.0067
         0   -0.1089    0.0001    3.2669
```

respectively as:

The root locus plot is as shown below:



From this plot we see that there is always a pole on the right half of the s- plane at any given point of time due to which a standard P controller alone will not stabilize the system. Thus we make use of a State Feedback Controller here.

State Feedback Controller

A State Feedback Controller can be designed based on our preferred placement of poles. There are 2 different approaches we can use to design said feedback system. The state feedback system is a negative feedback system.

1) Ackermann's Formula

MATLAB provides an inbuilt function acker that performs pole placement on the basis of Ackermann's formula given the A, B matrices of the State Space system.

It calculates a gain matrix k such that the state feedback u = - kx where x is the state representation would place the poles at the locations as per the user's choice.

(Here the Eigenvalue of A – bk matches the value of p)

```
des_pole = [-3;-3;-3;-3];
K = acker(A,B,des_pole)
```

O/P:

```
K =

  -12.3980  113.3291  -16.5659   34.5159
```

2) LQR (Linear Quadratic Regulator)

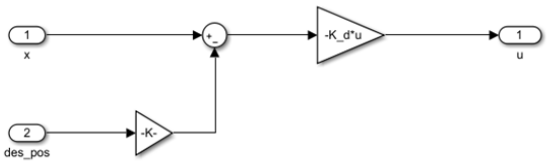
LQR is another approach to calculate the gain matrix k for a state feedback system for a system using weight matrices Q and R which are the weight matrices for states and inputs respectively. The optimum pole values are calculated based on the weight values provided which in turn provide an optimal Controller.

```
Q = 10*eye(4);
R = 0.1;
K_lqr = lqr(A,B,Q,R)
```

O/P:

```
K_lqr =

  -10.0000  133.8075  -18.2604   42.0920
```

Desired trajectory for the system can be obtained by providing the position as an input to the state feedback system. This can be shown via the Simulink model of the controller.



The state feedback controller is used to estimate the force to be applied to move the robot from one point to another while maintaining its balance.

Here the input vector *x* gives us the current position of the bot and *des_pos* give the desired final position, the difference of these two gives the distance by which the bot is to be moved, which when multiplied by gain matrix K of the designed controller, gives the force u to be applied on the bot.

In practice there are no continuous time controllers, hence we discretize the designed system using the MATLAB function **'c2d'** which converts a continuous time system to a discrete time system, provided with a sampling time Ts in seconds.

```
%% Descrete time
Ts = 0.1;
sys_d = c2d(sys,Ts)
Ad = sys_d.a;
Bd = sys_d.b;
Cd = sys_d.c;
Dd = sys_d.d;
```

Here sys_d is the discrete time system, Ad, Bd, Cd, Dd are the discrete time matrices for the State Space model of the system.

The Controller is now redesigned for a discrete system using either of the 2 methods mentioned above. As the optimal value of Q and R were provided by the paper of reference, we decided to use the LQR method to design said controller.

```
Q = 300*diag([1,10,1,1]);
R = 0.02;
N = zeros(length(B),1);

[K_d,~,~] = dlqr(Ad,Bd,Q,R,N)
[Ob,~,~] = dlqr(Ad',Cd',Q,R,N)
```

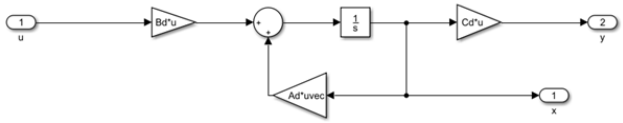
Here Ob is the observer gain used to design a discrete time observer.

State Observer

A good controller allows access to all the states of the system. This is not always the case, as what we obtain from the controller and the system is only the output of the system. Using the obtained output of the system we can reconstruct the states of the system.

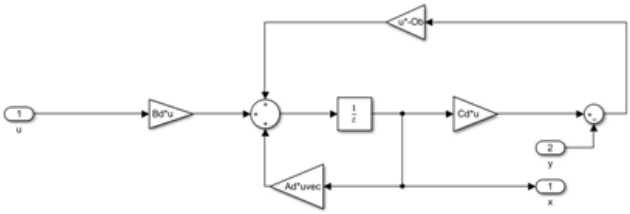
As we have seen above, the system can be designed using the state equation

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{B}\,u$$



The system above is a continuous time system which uses x0 as its initial conditions.

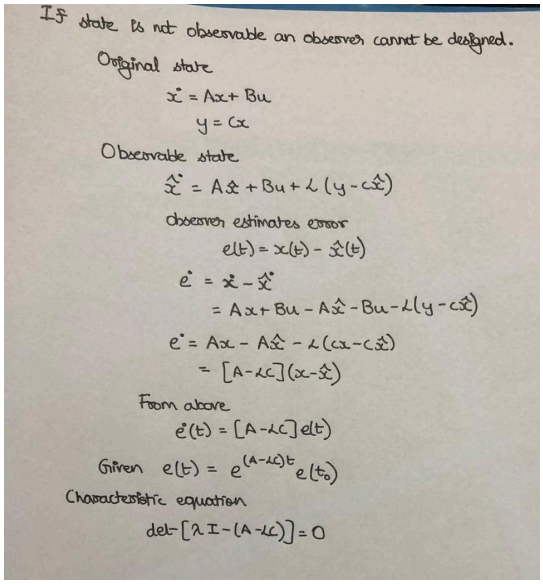
The various states of the system can be observed within a system with the use of a State Observer. A State observer is essentially a reproduction of the system, with the output y and the force u as inputs to reproduce the states x.



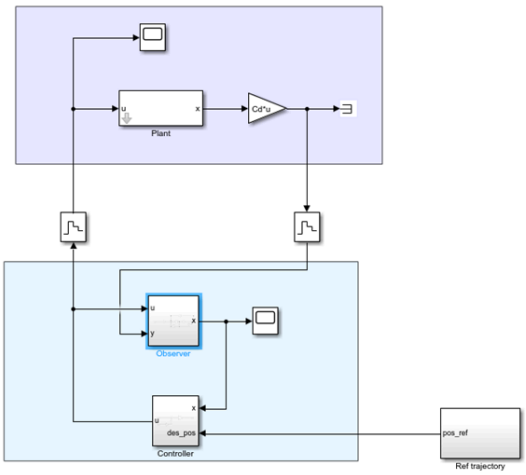
Here we see that the integrator is replaced with a time delay, as this is a discrete time system. The continuous variables of the system are replaced by the discrete time variables A_d, B_d, C_d, D_d as obtained above.

The output of the model is then compared with the output of the observer system and is feedback with an observer gain as designed above. Thus, the observer as designed reproduces the state vector of the system.

This is seen from the equations:



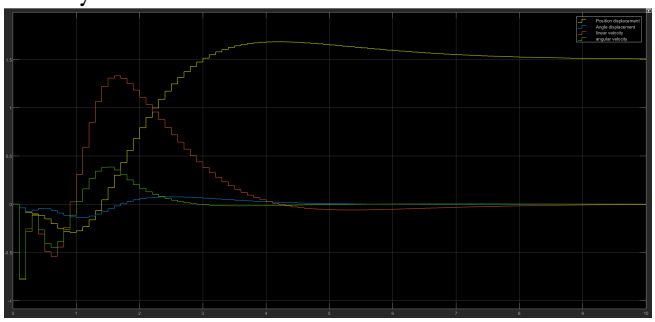
Final Model



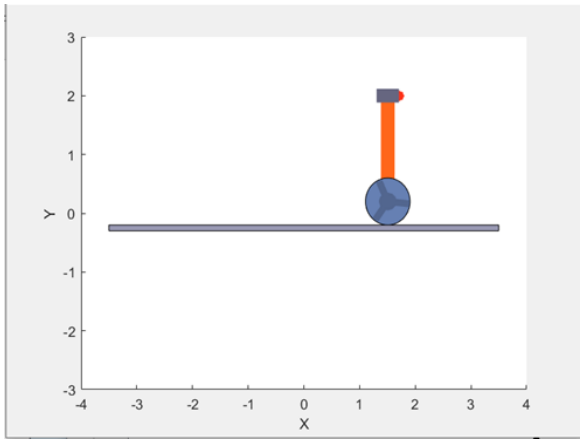
Plant and ref trajectory are blocks provided by the git repository to plot the self-balancing robot with variable motion on a flat surface.

Output

The output of the state variable within 10 seconds as obtained from the observer are given above.
Here Y is position displacement, B is angular displacement, R is linear velocity, G is angular velocity:



We also obtained a real time of the behaviour of the self-balancing robot as designed by the University with the help of the Plant and another function call_plot.m provided with the documentation for the State Space Modelling.



CONCLUSION

The dynamic mathematical model for the Inverted Pendulum-Cart system was modelled and simulated systematically.

Entire MATLAB code was modularized completely so that it was more efficient and understandable.

All the simulations obtained in different simulation environments were validated and verified and were shown to satisfy the design criteria.

Due to time constraint and other unprecedented events, we could not undergo the hardware aspect of the project which included adaptive control, reinforcement learning and such.

Finally, the self-balancing robot was simulated and illustrated as a proof of concept,which has various applications in the practical field.

INDIVIDUAL CONTRIBUTIONS

1. Initial Dynamic model and System Equations: All members of the team
2. MATLAB code framework: Pramod Keshav
3. Modularisation of the code: Ajay Victor
4. SIMULINK model and simulations:Prashanth B
5. SIMSCAPE simulations: Pramod Keshav
6. State-Feedback Control Design: Ajay Victor

Ajay Victor - MEDIUM
Pramod Keshav- HIGH
Prashanth B - MEDIUM

REFERENCES

1. Self-balancing robot implementing the inverted pendulum concept- IEEE Paper
1. Autonomous Dual Wheel Self Balancing Robot Based on Microcontroller- Journal of Basic and Applied Scientific Research
2. https://in.mathworks.com/help/slcontrol/gs/automated-tuning-of-simulink-pid-controller-block.html
3. https://in.mathworks.com/products/simscape.html
4. https://www.electrical4u.com/types-of-controllers-proportional-integral-derivative-controllers/
5. CAE in der Regelungstechnik: Exercises from University of Kaiserslautern. (Instructor: Alen Turnwlad (ETI TUK))
6. Self-Balancing robot-'DIRK'- University of Twente