DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY ROPAR

RUPNAGAR-140001, INDIA

IIT Ropar

MACHINE DESIGN (ME306) PROJECT REPORT

For

Simulink Model of a Self-Balancing Robot Using PID Control

Submitted by

Mehboob Alam (2022MEB1323)

Sahil Mhapsekar (2022MEB1324)

Mili Prajapati (2022MEB1325)

Mohit Sharma (2022MEB1326)

Palakpreet Kaur (2022MEB1327)

Lab-Group: Mon (Group-C)

Supervised By

Dr. Srikant Shekhar Padhee

Dr. Jitendra Prasad

Report Submitted On: 15-05-2025

Simulink Model of a Self-Balancing Robot Using PID Control

Objective:

The objective of this project is to design and simulate a self-balancing robot using MATLAB Simulink. The focus is on modelling the dynamics of the system and implementing a PID controller to maintain the robot's balance. The project aims to understand the principles of control systems, particularly how feedback control can be used to stabilize an inherently unstable system. Through this simulation, we seek to analyse the response of the robot to disturbances and evaluate the effectiveness of the PID controller in achieving system stability.

Problem Statement:

Balancing a robot on two wheels is a difficult task because it can easily fall over. Most robots use extra wheels or support to stay upright, but that makes them less flexible and harder to move around. A self-balancing robot can solve this problem, but it needs a smart control system to keep its balance on its own. In this project, we try to build and test a self-balancing robot using MATLAB Simulink. We use a PID controller to help the robot stay upright and check how well it works when the robot faces small pushes or changes.

System Modelling in Simscape Multibody:

To simulate the behaviour of a self-balancing robot, we created a detailed 3D model using **Simscape Multibody** in MATLAB. This toolbox helps in visualizing the physical structure and movement of mechanical systems using realistic joints, bodies, and constraints.

- **Structure Description**

The robot consists of the following physical parts:

- **Three horizontal plates**:
  These plates are coloured red in the model. The **bottom plate** is connected to the wheels and acts as the base. The **middle plate** is where we imagine placing the sensors and control circuit in a real prototype. The **top plate** is used to simulate the vertical structure or load, which acts like the "inverted pendulum".
- **Four vertical rods**:
  These are modelled as long cylindrical bars connecting the plates. They represent the **rigid body frame** of the robot and help maintain vertical alignment. These rods simulate the real-life mechanical frame that holds the robot together.
- **Wheels**:
  Two black circular wheels are connected to the bottom plate. These wheels are actuated

using motors in simulation. Their rotation is controlled to balance the robot upright using feedback from the control system.

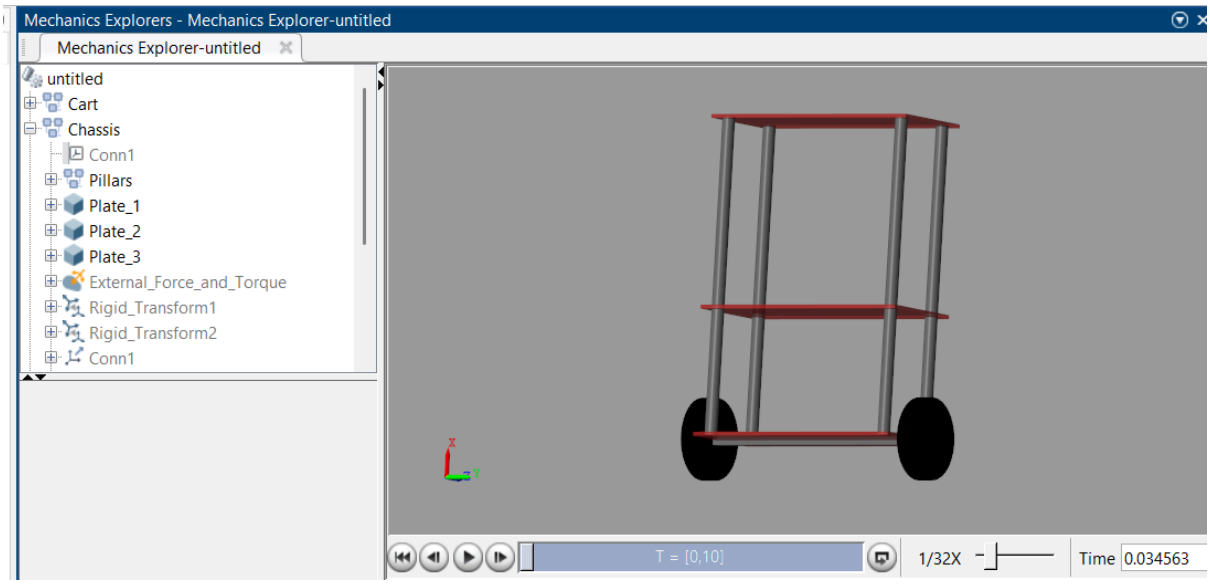

Figure 1: 3D Simscape Multibody Model of the Self-Balancing Robot
This figure shows the complete 3D model of the self-balancing robot designed using Simscape Multibody. It includes three horizontal plates, vertical rods forming the frame, and two wheels at the bottom for movement and balance.

Project Description

Overview

This project focuses on modelling and simulating a **self-balancing robot system** using MATLAB Simulink and Simscape Multibody tools. The robot is designed as a pendulum mounted on a cart that can move horizontally. The primary objective is to maintain the robot's balance by automatically controlling the position of the cart so that the pendulum remains upright even when disturbed.
Self-balancing robots are a common example of an inverted pendulum problem, which is a classic challenge in control systems and robotics. The robot is unstable by nature, so it requires a feedback control mechanism that senses its tilt and applies the right corrective force to balance itself.
The entire system is modelled in Simulink using blocks that represent mechanical parts, control systems, and simulation configurations. The system is simulated over a specified time period to observe how effectively it maintains balance.

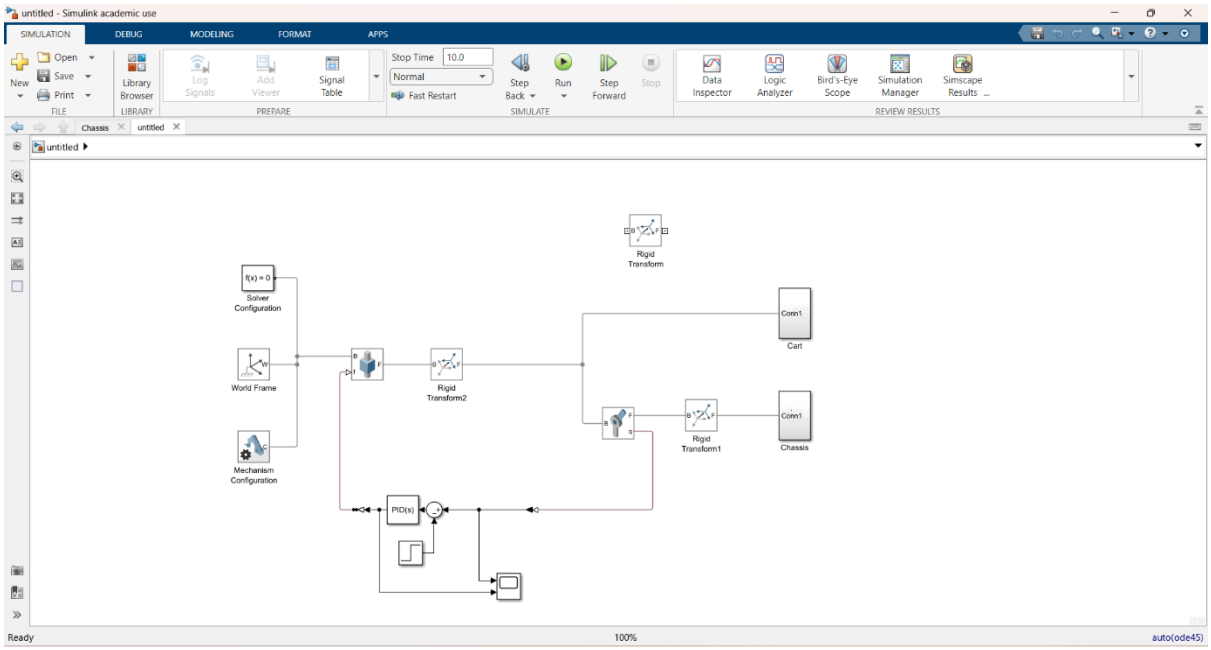

Fig 2 Complete Simulink model of the self-balancing robot showing all mechanical, control, and simulation blocks.

Inputs to the System

The input to this system is provided by the **Step block**, which simulates a sudden disturbance or desired setpoint change in the pendulum angle. It models a tilt or pushes to the robot at the start of the simulation.

- The step input simulates an initial deviation from the upright position.
- The goal of the controller is to counteract this deviation by moving the cart in such a way that the pendulum returns to the upright position (angle = 0 degrees).
- The **desired setpoint angle** is zero, meaning the robot should ideally remain perfectly vertical.

This input acts as the starting point for the system to test its balancing ability.

Mechanical System Modelling (Process)

The mechanical system, representing the physical parts of the robot, is modelled using **Simscape Multibody blocks**. These blocks allow us to simulate real-world rigid body dynamics, joints, and transforms.

- **Cart block** and **Chassis block**: These subsystems represent the base on which the pendulum is mounted and the supporting structure of the robot. The cart moves horizontally to balance the pendulum.
- **Rigid Transform blocks (Rigid Transform1 and Rigid Transform2)**: These define precise position and orientation relationships between the different mechanical parts. They ensure the pendulum and cart are connected correctly in space.
- **Revolute Joint block**: This joint allows rotation of the pendulum relative to the cart along one axis. This mimics the real pivot point where the pendulum swings.

- The **Prismatic Joint block** allows the **Cart** to move linearly along the horizontal axis. This sliding motion is essential to adjust the base position of the robot when it detects a tilt, helping in restoring balance.
- **World Frame block**: It sets a global reference coordinate system for the entire simulation, defining the fixed ground or environment.
- **Mechanism Configuration block**: It defines important physical and simulation parameters like gravity direction and solver settings for the mechanical system.
- **Solver Configuration block**: This block specifies the numerical solver used to simulate the dynamics over time, ensuring accurate and stable simulation results.

Together, these components create a realistic model of the robot's physical structure and movements.

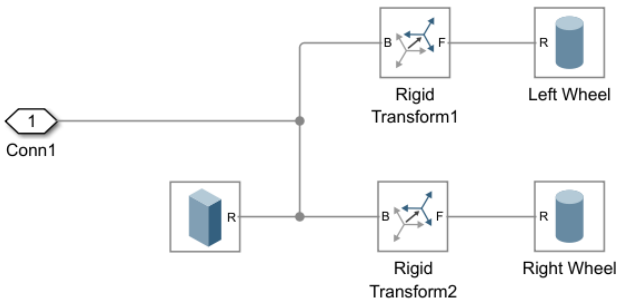

Fig 3 Components of cart block

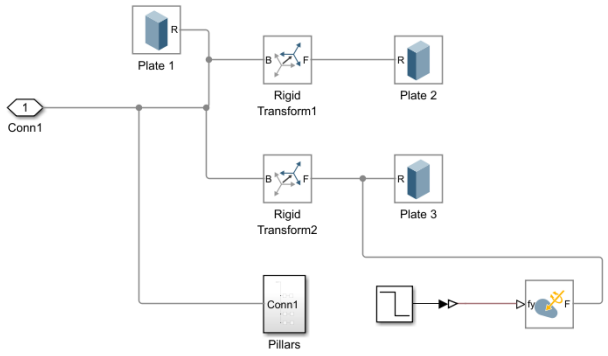

Fig 4 Components of chassis block

Fig 5 Prismatic Joint block

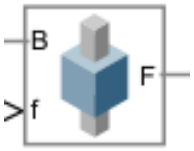

Fig 6 Revolute Joint block

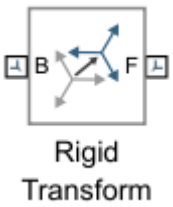

Fig 7 Rigid Transform

Control System (Process)

The control system is the core of the self-balancing mechanism. It senses the pendulum angle, compares it with the desired angle (usually zero), and then calculates the force needed to keep the pendulum balanced.

- The **Sum block** calculates the error by subtracting the actual pendulum angle from the desired angle.
- This error is fed into the **PID Controller block**, which is a type of feedback controller that continuously adjusts the control input to minimize this error. The PID controller uses three terms:
    - Proportional (P) term that reacts to the current error,
    - Integral (I) term that accounts for past errors, and
    - Derivative (D) term that predicts future errors.

- The output of the PID controller is a corrective force signal.
- This force signal is applied to the **Cart subsystem**, moving it left or right to counterbalance the pendulum.
- The control loop is closed by continuously measuring the pendulum angle and adjusting the force accordingly.

The controller ensures that any disturbance to the robot causes the cart to move in the right direction and magnitude to bring the pendulum back upright.

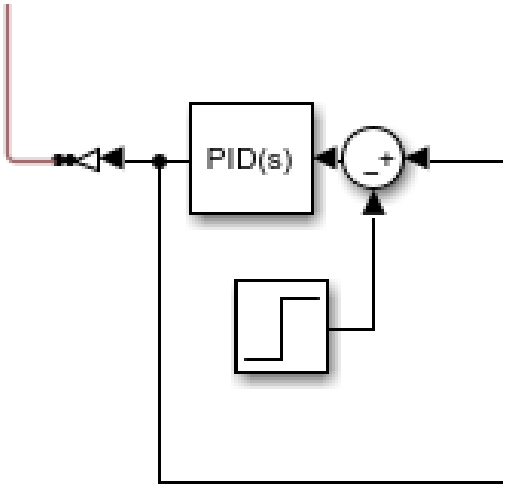

Figure 8: Control system components showing the PID feedback controller, error summation, and step disturbance input.

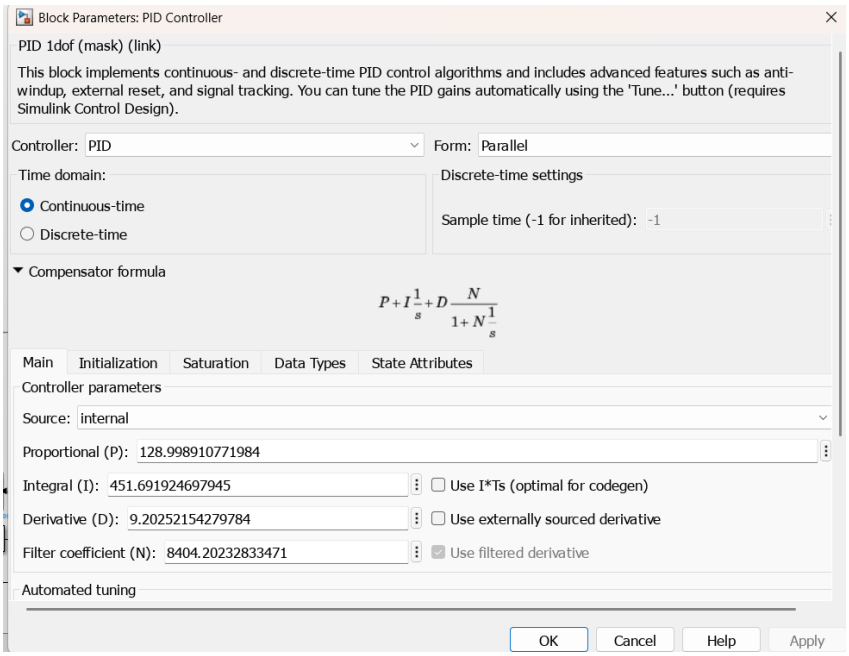

Figure 9: PID Controller Block Parameters Configuration

This dialog box displays the configuration of the PID Controller block used in the Simulink model. The controller is set to operate in continuous time with the parallel form. The tuned gain values are: Proportional Gain (P) = 128.9989, Integral Gain (I) = 451.6919, Derivative Gain (D) = 9.2025, These values govern the corrective force applied to the cart to maintain the balance of the inverted pendulum.

Outputs of the System

The outputs are observed and analysed through the **Scope block**, which records signals during simulation.

- The primary output is the **pendulum angle over time**, showing how it reacts to disturbances.
- Another output is the **cart position**, showing how the cart moves to balance the pendulum.
- From the simulation results, we expect to see the pendulum angle start at a disturbed value and then gradually return to zero as the controller acts.
- The cart position changes dynamically, moving back and forth to stabilize the robot.

These outputs verify if the system works as intended and successfully balances the robot.

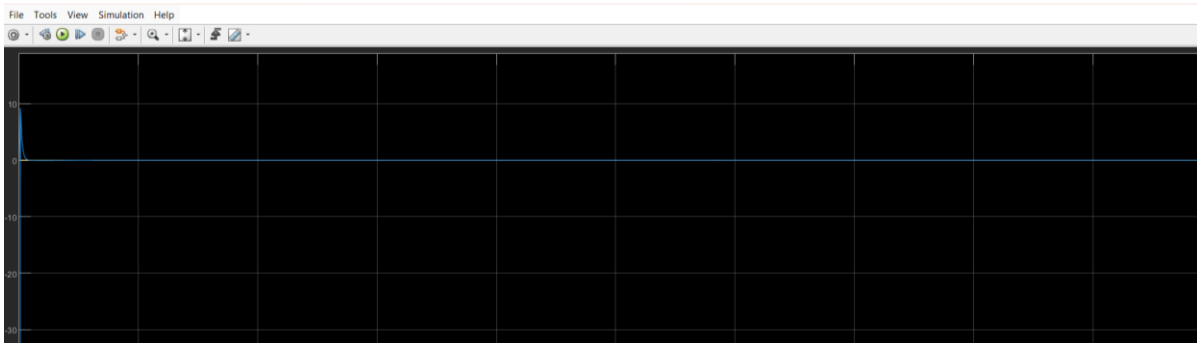

Figure 10: Time (x axis) vs Angular Position (θ) (y axis) plot in degrees from the Scope block, showing the pendulum angle returning to zero, indicating successful self-balancing by the control system.

Summary of Simulation Process

1. The simulation begins with the robot at rest.
2. The step input provides an initial tilt to the pendulum.
3. The sensors measure the angle and feed it back to the controller.
4. The PID controller calculates the necessary force to apply.
5. The cart moves in response to the control signal.
6. The pendulum swings back toward the vertical position.
7. Over time, the pendulum angle stabilizes near zero, showing successful balancing.
8. The simulation runs until the stop time (10 seconds in our model).

Conclusion

In this project, we built a model of a self-balancing robot using MATLAB Simulink and Simscape Multibody. The robot uses a PID controller to keep itself balanced by continuously adjusting its position. Our simulation showed that the robot can maintain stability and quickly respond to any disturbances.

This project helped us understand how to model mechanical parts like the cart and chassis, and how to use control systems to keep the robot stable. It also showed the importance of setting the right PID values for good performance. Overall, this work gives a good base to develop more advanced robots in the future.