INTELLIGENT CONTROL OF TWO WHEEL SELF BALANCING ROBOT

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

OF

MASTER OF TECHNOLOGY

IN

CONTROL & INSTRUMENTATION

Submitted by:

SHIVAM

2K20/C&I/09

Under the supervision of

Prof. BHARAT BHUSHAN

Dr. BHAVNESH JAINT

DEPARTMENT OF ELECTRICAL ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi – 110042

MAY, 2022

DEPARTMENT OF ELECTRICAL ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi – 110042

CANDIDATE'S DECLARATION

I, SHIVAM, Roll No. 2K20/C&I/09 of M. Tech (Control & Instrumentation), hereby declare that the Dissertation titled "INTELLIGENT CONTROL OF TWO WHEEL SELF BALANCING ROBOT" which is submitted by me to the Department of Electrical Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or similar title or recognition.

Place: Delhi SHIVAM

Date:

DEPARTMENT OF ELECTRICAL ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi – 110042

CERTIFICATE

I hereby certify that the Project Dissertation titled " INTELLIGENT CONTROL OF TWO WHEEL SELF BALANCING ROBOT " which is submitted by SHIVAM, Roll No. 2K20/C&I/09 of Electrical Engineering Department, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date:

Prof. Bharat Bhushan
(Professor)
Supervisor
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

Dr. Bhavnesh Jaint
(Assistant Professor)
Co-Supervisor
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

ACKNOWLEDGEMENT

I have taken efforts in and dedication this Thesis. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincerethanks to all of them.

I am highly indebted to Prof. Bharat Bhushan and Dr. Bhavnesh Jaint for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents & member of Delhi Technological University for their kind co-operation and encouragement which help me in completion of this Project.

ABSTRACT

The self-propelled robot is based on the principle of the Modified pendulum, which is a two wheel-drive vehicle with a vertical position vertically. Contains both hardware and software implementation. A machine model based on the design of the cart circuit space, the pendulum system. To find its stable distorted position, I used a standard feedback controller (i.e., a PID controller). By nature, we have to control both the angel of the pendulum and the position of the chariot. The Simulink model applied in this project uses PID controller to provide the necessary control action required for the cart to remain stable. For the control parameters for PID I have used different tuning methods. These tuning methods are encoded in MATLAB and the results from there are used in the Simulink model. For extra comparison of controllers, I have used LQR controller to observe the difference I control techniques in linear and nonlinear control models. These parameters are considered as the system parameters and determine the external power required to measure the robot upwards.

It will be prevented from falling by giving acceleration to the wheels according to their inclination. If the board is tilted at an angle to the wheels; the robot's weight center will receive artificial power that will use torque opposite the slope.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1
INTRODUCTION

1.1 Introduction To Two Wheel Self Balancing Robot

For constructing a 2-D robot, it is important to solve the problem of the distorted pendulum or the distorted pendulum in the cart[1]. The calculations of the inverted pendulum system and also the expressions have a high level of complexity. The motive is just to adjust the wheel's position to keep the angle of the inverted pendulum remains upright i.e., it should be ninety degrees from the ground. By any chance when the pendulum rotates in any direction, the two wheels below the cart moves in the direction so that the angle of the robot remains constant. What I observed from the simulation is that when the movement of the cart is slow when the disturbance of the angle is less and the movement of the cart is fast when the disturbance is angle is more.[2][3][4]

For simpler understanding, the movement of the robot is confined to only two axes. The speed of both the wheels will be similar. Thus, for the purpose of calculation we only have to consider that the cart will move ahead or back.[5]. While viewing in MATLAB, it will be shown us to as the cart is moving sideways. For practical implementation every individual wheel will have a separate control system. But there is no need to go into that type of complexity from now. When the movement of cart is restricted to a single plane the equations of the calculation become much easier to operate on.[6]

1.2 Control System

The best control systems examples are that of the traffic lights and also washing machines. There are many other examples like motor vehicles, mobile phones etc.

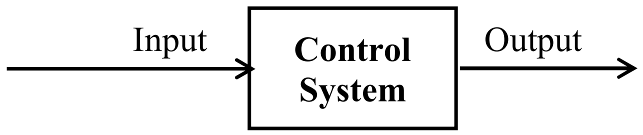


Figure 1.1 Block Diagram of Control System

There are many types of control system but some main classifications are the Single Input Single output control system and the Multiple Input Multiple Output Control System. Here the type of control system I am going to apply is Multiple Input Multiple Output Type.

1.3 Classification of Control System

The Based on the Feed Back Path the classification of control system is mainly of two types i.e., Open Loop control system and Closed Loop control system.

In open loop control system, there is no output fed back to the controller. The control action is not dependent on the desired output.
The figure 1.2 represents the Block Diagram of open loop control system.

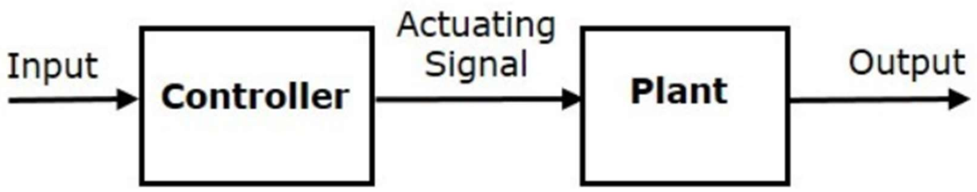


Figure 1.2 Open Loop Control System

Closed loop control system consists of a feedback element which sends the output to the controller where this signal is compared with the desired reference signal so that the output can be changed as per the requirements automatically. The error detector present in figure 1.2 does the comparison between the output and the reference signal. The error is then analyzed by the controller and the necessary control action is taken.

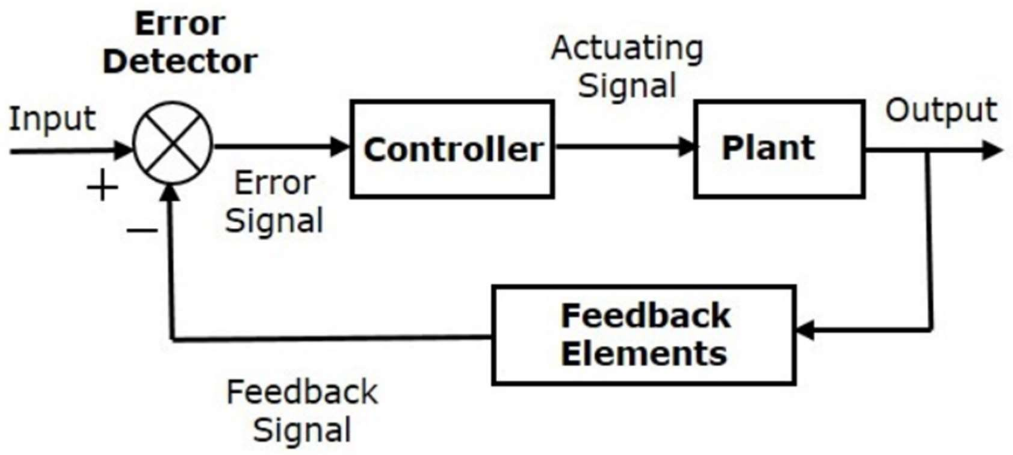


Figure 1.3 Closed Loop Control System

1.4 Introduction to MATLAB SIMUINK

I have used MATLAB software for the building of my project and implementing the control system. MATLAB which is the abbreviation of MATrix LABoraory is a multiplatform programming language developed by MathWorks. It provides us with a numeric computing environment.

It provides us the ability to perform matrix multiplication, create user interfaces, help us in implementation of different algorithms, plotting graphs etc. SIMULINK helps us with the model-based design approach for embedded systems and dynamic systems. The complete development of this project is based on MATLAB and SIMULINK.

Today many projects are built on MATLAB before it’s practical implementation in the real world. Because of the availability of this software the ability to see the performance of a system before hand is increased. The pros an cons of a system can be observed and its feasibility in the real world can also be studied. Just for example in this project it would have taken me much more time, effort and capital to build a successful model which can compare all the algorithms and deriving the results from them would also be a big challenge. But due to this software it was a seamless process to build this system and implement different algorithms and get the results.

1.5 Objective:

- Designing a Simulink model of the two-wheel self-balancing robot in MATLAB.

- To compare different control methods that are responsible for balancing the two-wheel self-balancing robot.

The main motive of the project is balancing the pendulum i.e., the inclination angle is the basis of the whole modelling. As soon as the angle begins to displace from its desired position, the movement in the cart happens. We need to get the data from the gyroscope and accelerometer to maintain the inclination angle in the practical application.[1]

- **Mathematical analysis of approximate model**- The reference of the inverted pendulum on a cart has been taken and with the help of the force balancing equations I have derived the transfer function.
- **System Analysis**- The parameters responsible for control action and the required output is analyzed.
- **Simulation-** In this part I built the model of the two-wheel self-balancing robot model in SIMULINK. After that I applied the controller to the model so that the control action can balance the model and I can get desired results.
- **Optimization-** The control methods need to be optimized so that we can get he results which are practically possible to implement in the real world. The main Optimization control method used here is the LQR controller which compares all the input parameters and gives the desired result. Further discussion is there in the following chapters.
- **Comparison-** at last I have done the comparison of the control systems I have used and the conclusion shows that which control system is more effective than others and what parameters are responsible for those results.

1.6 Motivation

The Two Wheel Self Balancing Robot is an extremely extensive system which contains many areas of the complete control system subject. The development of the model in SIMULINK taught me the model building process in SIMULINK. The designing of control system also contains the benefits of the exploration to the implementation of control systems to different processes.

Moreover, the development of the reusable rockets named Falcon of SpaceX moved me to think how it would be possible to balance any system upright in midair and making it land safely on the ground and that too at the accurate position. When I studied through different sources, I came to know that the inverter pendulum system is the base of this engineering marvel.

CHAPTER 2
LITERAURE REVIEW

2.1 Two Wheel Self Balancing Robot

From this project report I got the basic idea of the two wheel self-balancing robot and the dynamic equations which helped in deriving the transfer function of my model. I also got the basic idea of control system responsible for the control system of the robot from this report.[1] Laplace transform, Equations of free body diagram was taken from here. Also, an insight of LQR control was taken. This report provides very deep learning of the practical implementation of Two Wheel self-balancing robot.[2]

The idea for the basic structure shown in the MATLAB model is taken from here. This paper is giving us the idea for the microcontroller-based controller for Two-wheel self-balancing robot. It also gives light on the PID controller used in project.[3] This paper provides an insight to the mathematical model of the robot which is derived by the differential equation method using the state space modelling procedure. Also, a brief idea of using PID controller is provided in this.[4]

For coming up with a model which can be solved mathematically and implemented on Simulink, it was very necessary to get the knowledge of the basic structure of two-wheel self-balancing robot. The papers from [41]–[50] give the necessary knowledge to help me able to build a feasible model of the robot. These papers also gives the future scope of this system in the field of robotics.

2.2 MATLAB Simulation model of the Two Wheel Self Balancing Robot

Representation of closed loop control system and the MATLAB Simulink model design is taken from this paper. Also the state space representation for LQR based controller is taken from this. Response of open loop transfer function and closed loop transfer function is given in this.[5] This paper provides the insight in the kinematics model of a two wheeled self balancing robot. It mainly focuses on the control of wheels for the control of the motion.[6] Here an optimal PID controller is designed with the help of MATLAB

Simulink. The basic idea of transfer function of the robot is also provided here.[7]

The dynamic model, equations of motion of the robot, the equation for conservation of energy, free body diagram is given in this paper. This paper provide to many conference papers and journals in their research. It provides very deep mathematical analysis of the two-wheel self-balancing robot.[8]

2.3 Two Wheel Self Balancing Robot with PID controller

Working of PID based two-wheel elf balancing robot and the programming flow chart of the self-balancing robot is provided here. This paper provides the idea of implementation of the two wheeled self-balancing robot using Arduino microcontroller.[9] Robot's controller block diagram and PID tuning is provide here. In addition to that here the practical implementation of the control system using complementary filters is provided here.[10] The basic idea of PID controller is taken from here. This provides the deep analysis od PID controller used in the system and fuzzy logic method is also used as a controller for the two-wheeled self-balancing robot.[11] The state space analysis of the robotic system is provided here with extreme clarity. This paper helped very much in calculating the transfer function and MATLAB coding in the initial stages of the project. Also, the control loop for LQR controller and the gain matrix for the LQR controller is given in here.[12]

It provides the idea of type of PID controller to be used on the basis of the disturbance rejection and system robustness and then the performance of PID controller is checked. It also provides idea for tuning of PID controllers.[13] It provides a view of classical and modern approaches used for PID tuning methods and their usage in various areas. Most of the systems used today still use PID controllers but their tuning methods are improving constantly. As mentioned, it provides the insight on tuning methods of PID controller.[14]

The papers referring [23]–[27] have provided with some important details about how to use the PID controller, what are the control parameters required for using PID controller, and how to implement it on different systems. PID controller tuning with algorithms is also explained here.

2.4 The Concept of Genetic Algorithm and GA based PID controller

The basic idea of Genetic Algorithm is given in here. The flow chart of the functioning of genetic algorithm is also provided here for the better understanding of the algorithm.[15] The flow chart of genetic algorithm is provided in here. This paper also compares the difference between PSO and genetic algorithm. It also provides an insight on how to implement PSO and Genetic Algorithm in practical applications.[16]

The basic idea of Genetic Algorithm based PID tuning is shown here. GA based PID tuning is successfully applied for the optimum adaptive control.[17] The basic aim of this research paper was to implement the PSO based PID tuning. Here a model of DC motor in a plant is tuned with both Ziegler-Nichols method and PSO method and the results of both are compared hence forth.[18]

Genetic Algorithms is very popular for obtaining optimal or accurate parameters for different applications. But first the explanation of Genetic Algorithms is very important. The papers from [28]–[32] helps in the explanation of genetic algorithms to me. The very vast area of research present in these papers adds to the knowledge of the area.

2.5 Particle Swarm Optimization and PSO based PID Controller

This paper is about the particle swarm optimization and support vector machines to improve the accuracy of the datamining system where this system was implemented. It gives an insight to PSO and the effectiveness of it's optimization techniques.[19] This paper represents Arduino based experimental two wheeled self-balancing robot which is more focused on economical building of the model and implementation. With the help of computer programming and modelling the authors were able to achieve the implementation of the project successfully.[20]

There are many types of other nature inspired algorithms which are in use for many optimization and estimation problems. But the Particle Swarm Optimization technique is relatively simple and highly effective to implement. I gained the knowledge about it from the papers [33]–[37].

2.6 Optimal Control System and LQR Controller

No matter how quickly a system responds but it will be of no use if the control system is not economical. For example, if we apply hard braking on motor vehicles, the vehicle stops quickly but the quality of the parts of the vehicle gets compromised. That's why we apply brakes in a manner that the parts are safe. Same applies for the controller's design. The controller should provide quick and smooth response so. In control systems it is called optimal control of a system. The papers [37]–[41] provides the knowledge about both Optimal Control Systems and LQR controller.

This paper is based on the control of unstable and non-linear two-wheeled self-balancing robot using the linear control techniques like LQR and LQG methods. This paper also shows us that these control methods has the ability to Reject the disturbances in a much effective way.[21]

For coming up with a model which can be solved mathematically and implemented on Simulink, it was very necessary to get the knowledge of the basic structure of two-wheel self-balancing robot. The papers from [41]–[50] give the necessary knowledge to help me able to build a feasible model of the robot. These papers also gives the future scope of this system in the field of robotics.

CHAPTER-3
MATHEMATICAL MODELING OF AN APPROXIMATE MODEL

3.1 Problem Setup and Design Requirements

As mentioned above the model used here is the inverted pendulum on a cart. Mathematical analysis of this model has to be done. This type of model is very common in the research papers or conferences of inverted pendulum mounted on a cart. The main reason for its popularity is that without a suitable control system, the cart is extremely unstable. We need control action for each and every working second to maintain the pendulum in desired position. Moreover, the dynamics of the system are not very simple. The main motive of the project is that as soon as the angle begins to displace from its desired position, the movement in the cart happens and the pendulum obtains a stable upright position. The Falcon rocket which is able to land back on the ground for its reusability is an excellent example of this type of system.[7]

Here I have considered a system in which the pendulum moves in a particular direction which will be provided by some predetermined reference values and as shown in the figure below. In this inverted pendulum system, the control action moves the cart in the x direction towards the inclination angle. [8]

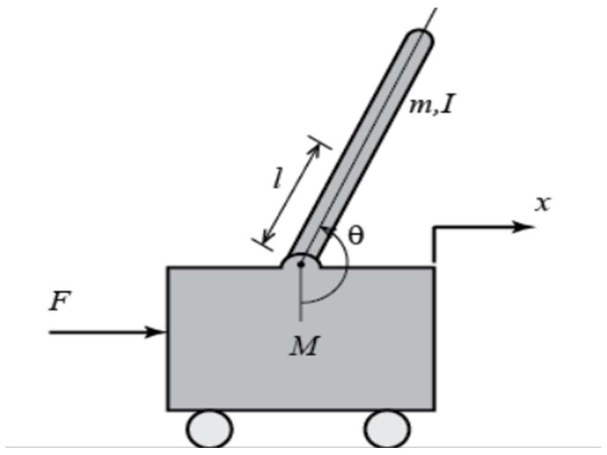


Figure 3.1 A model of an inverted cart pendulum system

3.2 Mathematical Analysis of The System

The free-body diagrams of the pendulum and the cart of the inverted pendulum system are shown below.

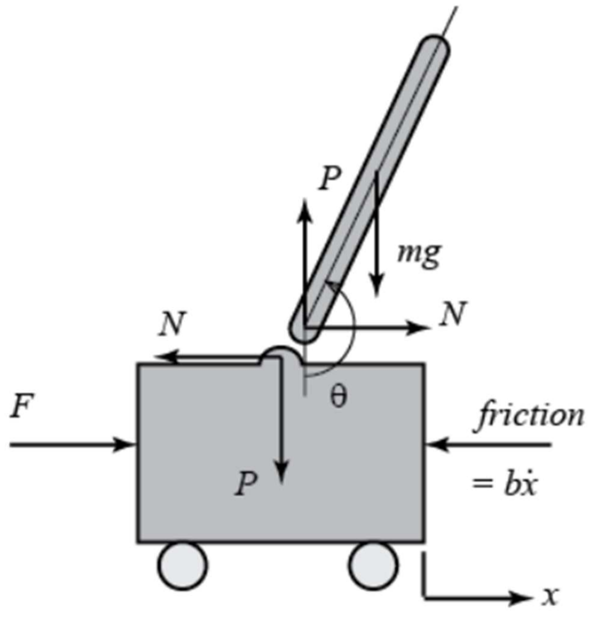


Figure 3.2 Free body diagram for inverted pendulum model

Summing the forces in the free-body diagram of the cart in the horizontal direction, I get the following equation of motion.

$$M\ddot{x} + b\dot{x} + N = F \tag{1}$$

It is to be noted that I can also take a sum of the forces in the direction vertical to the cart, but it will be of no use.

If I sum the forces in horizontal direction of the free-body diagram of the pendulum in, I get the following expression for the reaction force N.

$$N = m\ddot{x} + ml\ddot{\theta}\cos\,\theta - ml\dot{\theta}^2\sin\,\theta \tag{2}$$

If I substitute this equation in the 1st equation, I get one of the two equations which governs this system.

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

To get the 2nd equation of motion, I need to take the sum of the force's perpendicular to the pendulum. By solving the equations along the perpendicular axis simplifies the maths. I should get the following equation.

$$P\sin\,\theta+N\cos\,\theta-mg\sin\,\theta=ml\ddot{\theta}+m\ddot{x}\cos\,\theta \tag{4}$$

To remove P and N terms in the equation above, I need to sum the moments about the centroid of the pendulum to get the following equation.

$$-Pl\sin\,\theta-Nl\cos\,\theta=I\ddot{\theta} \tag{5}$$

Combining these last two expressions, I get the 2nd governing equation.

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

Since the analysis and control design techniques I'll be employing during this example apply only to linear systems, this set of equations has to be linearized. Specifically, I'll linearize the equations about the vertically upward equilibrium position, $\theta=\pi$, and can assume that the system stays within a tiny low neighborhood of this equilibrium. This assumption should be reasonably valid since in check I desire that the pendulum not deviate over 20 degrees from the vertically upward position. Let ϕ represent the deviation of the pendulum's position from equilibrium, that is, $\theta=\pi+\phi$. Again, presuming a tiny low deviation (ϕ) from equilibrium, I will use the subsequent small angle approximations of the nonlinear functions in our system equations:

$$\cos\,\theta=\cos\,(\pi+\phi)\approx-1 \tag{3.7}$$

$$\sin\,\theta=\sin\,(\pi+\phi)\approx-\phi \tag{3.8}$$

$$\sin\,\theta=\sin\,(\pi+\phi)\approx-\phi \tag{3.9}$$

After substituting the above approximated values into our nonlinear equations, I reach at the 2 linearized equations of motion. Note *u* has been substituted for the input F.

$$(I+ml^2)\ddot{\phi}-mgl\phi=ml\ddot{x} \tag{3.10}$$

$$(I+ml^2)\ddot{\phi}-mgl\phi=ml\ddot{x} \tag{3.11}$$

3.3 Transfer Function

Laplace transform should be taken of the linearized system equations to obtain the transfer function assuming the initial conditions as zero. The Laplace transforms that we get are shown below.

$$(I + ml^2)\Phi(s)s^2 - mgl\Phi(s) = mlX(s)s^2 \quad (3.12)$$

$$(M + m)X(s)s^2 + bX(s)s - ml\Phi(s)s^2 = U(s) \quad (3.13)$$

The relationship between a single input and a single output is represented by a Transfer function at a time. For finding the first transfer function for the output $\phi(s)$ and an input of *U(s)*, *X(s)* needs to be eliminated from the above equations. Solving the 1st equation for *X(s)*.

$$X(s) = \left[\frac{I+ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s) \quad (3.14)$$

Then substitute the above into the second equation.

$$(M + m)\left[\frac{I+ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s^2 + b\left[\frac{I+ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s - ml\Phi(s)s^2 = U(s) \quad (3.15)$$

Rearranging, the transfer function is then the following

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4+\frac{b(I+ml^2)}{q}s^3-\frac{(M+m)mgl}{q}s^2-\frac{bmgl}{q}s} \quad (3.16)$$

where,

$$q = [(M + m)(I + ml^2) - (ml)^2] \quad (3.17)$$

From the transfer function above I can be see that there is both a pole and a zero at the origin. These can be cancelled and the transfer function becomes the following.

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3+\frac{b(I+ml^2)}{q}s^2-\frac{(M+m)mgl}{q}s-\frac{bmgl}{q}} \left[\frac{rad}{N}\right] \quad (3.18)$$

Second, the transfer function with the cart $X(s)$ position as the output can be derived in a similar manner to arrive at the following.

$$P_{\text{cart}}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2-gm}{q}}{s^4+\frac{b(I+ml^2)}{q}s^3-\frac{(M+m)mgl}{q}s^2-\frac{bmgl}{q}s} \left[\frac{m}{N}\right] \quad (3.19)$$

3.4 State-Space Model

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.

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\phi}\\ \ddot{\phi}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix}+\begin{bmatrix}0\\ \frac{I+ml^2}{I(M+m)+Mml^2}\\ 0\\ \frac{ml}{I(M+m)+Mml^2}\end{bmatrix}u \qquad (3.20)$$

$$\mathbf{y}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix}u \qquad (3.21)$$

The *C* matrix has 2 rows because both the cart's position and the pendulum's position are part of the output. Specifically, the cart's position is the first element of the output *y* and the pendulum's deviation from its equilibrium position is the second element of *y*.

Robot Parameters used by us:

M = 1	mass of the chassis
m = 0.2	mass of the wheels and shaft
b = 0.1	estimate of viscous friction coefficient (N-m-s)
I = 0.0005	moment of inertia of the pendulum
g = 9.8	acceleration due to gravity (m/s^2)
l = 0.125	length to pendulum center of mass

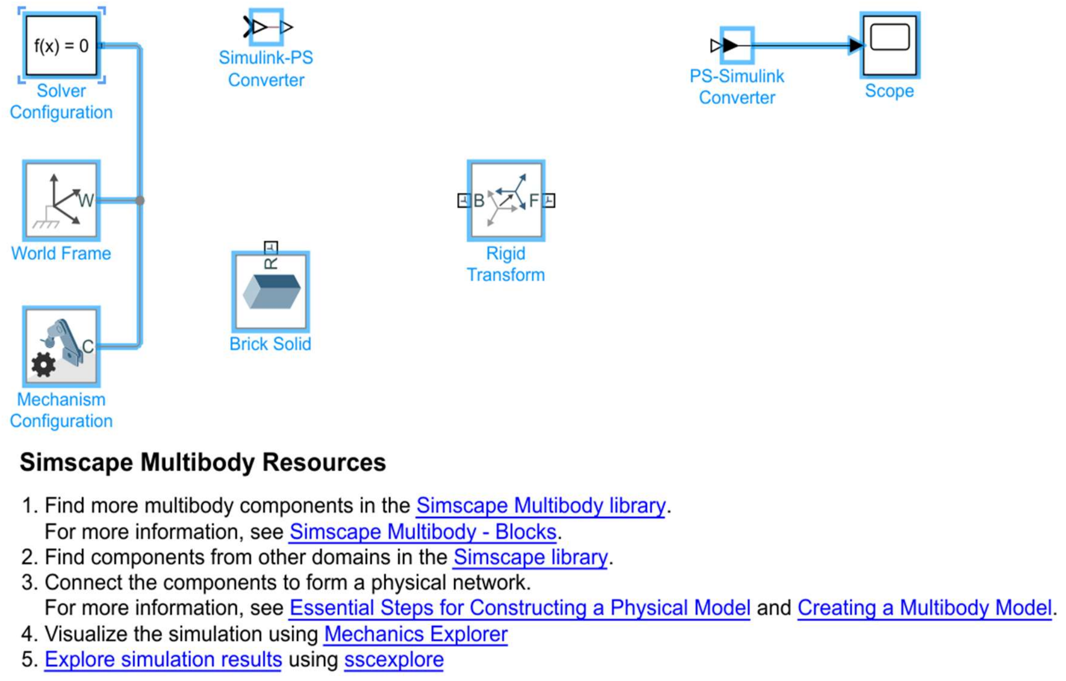
The transfer function of the Robot Assembled by us in MATLAB is-

$$\mathbf{\frac{0.81s}{s^3+0.01s^2-9.60s-0.80}}$$

CHAPTER 4
BUILDING THE MODEL

4.1 World Frame

Open the new Simscape Multi body model. For this I have to type smnew in the MATLAB command window. The new model opens, as shown below, with a few commonly used blocks already in the model. The PS-Simulink and Simulink-PS blocks def ne the boundary between the Simulink input / output models where the blocks are sequent ally tested and the Simscape models where the figures are simultaneously tested. The Solver Configuration is responsible for the calculation, the World Frame is responsible for providing the earth-linking axis to the model, and the Pathway Setting where can determine all the gravitational forces of each model.



Simscape Multibody Resources

1. Find more multibody components in the Simscape Multibody library.
   For more information, see Simscape Multibody - Blocks.
2. Find components from other domains in the Simscape library.
3. Connect the components to form a physical network.
   For more information, see Essential Steps for Constructing a Physical Model and Creating a Multibody Model.
4. Visualize the simulation using Mechanics Explorer
5. Explore simulation results using sscexplore

Figure 4.1 Simscape Multibody Model

To configure the basic settings in the model, do the following:

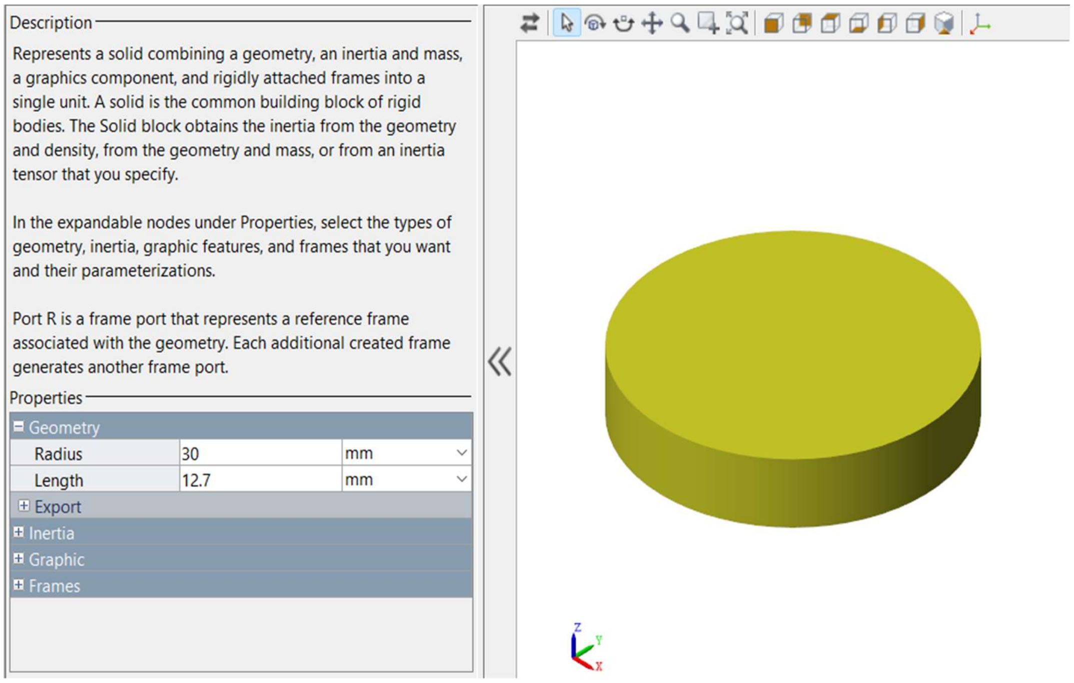
- Double-click on the Mechanism configuration block and set the gravitational force to “[0 0 -9.81]”, this represents acceleration due to the 9.8 m / s ^ 2 gravitational force.
- Open the Solver Configuration block and make sure the Use local solution checkbox is not selected.
- Type CTRL-E to open the Configuration Parameters box.
- In the Solver window, make sure Type is set to “Variable-Step” and that the Solver is set to “default”, and set the stop time to “10”.

4.2 Building The Robot Model

4.2.1 Making the Wheels

For wheels I will follow the following instructions- *Simulink library > Simscape > Multibody > Body Elements > Cylindrical Solid.* Now double click on the cylinder and specify the measurements.

Wheel Body:



Description

Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

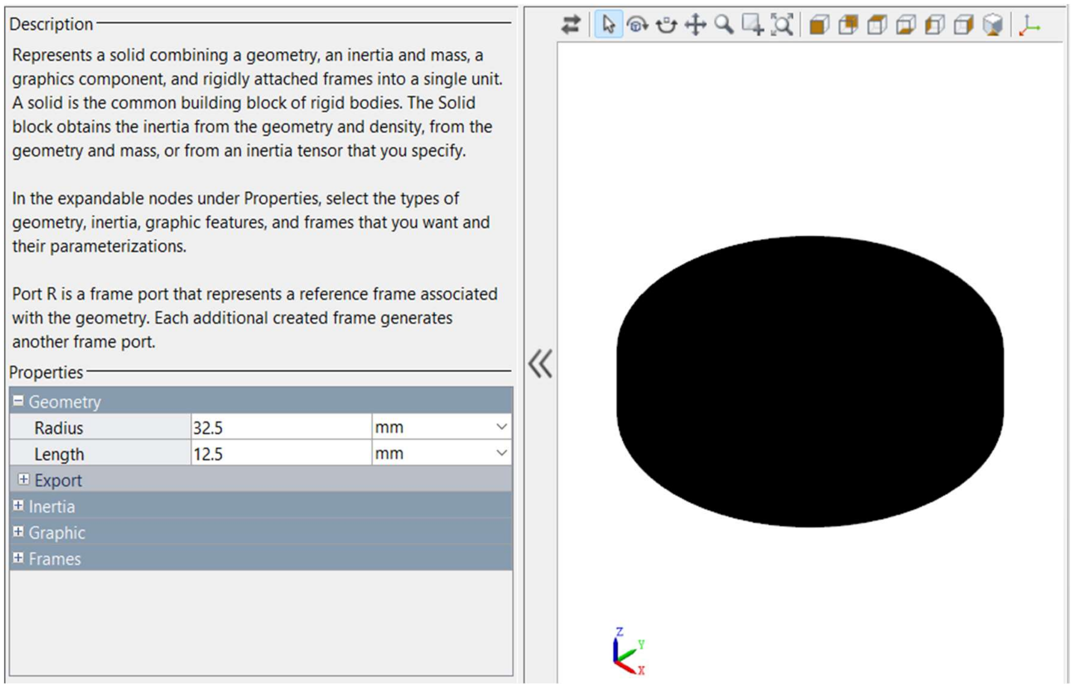
Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

Properties

⊟ Geometry		
Radius	30	mm
Length	12.7	mm
⊞ Export		
⊞ Inertia		
⊞ Graphic		
⊞ Frames		

Figure 4.2 Dimensions of Wheels

Wheel Tyre:



Description

Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

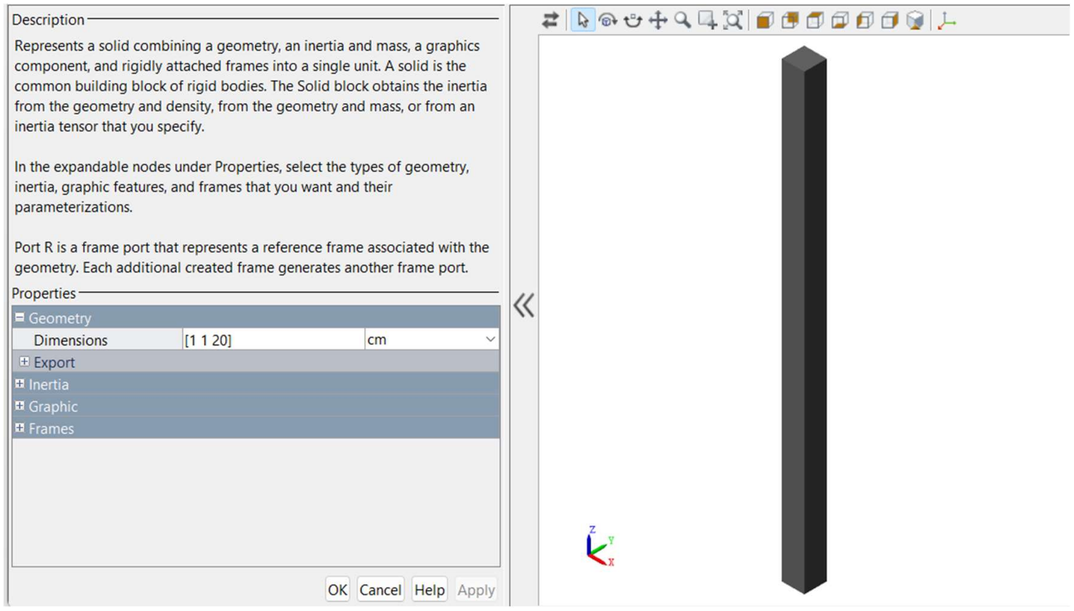
Properties

Geometry		
Radius	32.5	mm
Length	12.5	mm
Export		
Inertia		
Graphic		
Frames		

Figure 4.3 Dimension of wheel tyre

After this model is built just make a copy of the same and name it as Right Wheel. Now after doing that all, I have to do is to connect these wheels with a shaft. So, I can use the shaft option from the Simscape multibody model and define the parameters.

4.2.2 Shaft for connecting the wheels:



Description

Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

Properties

Geometry		
Dimensions	[1 1 20]	cm
Export		
Inertia		
Graphic		
Frames		

Figure 4.4 Dimensions of shaft connecting the wheels

The connection of all these components will complete the cart formation of our model.

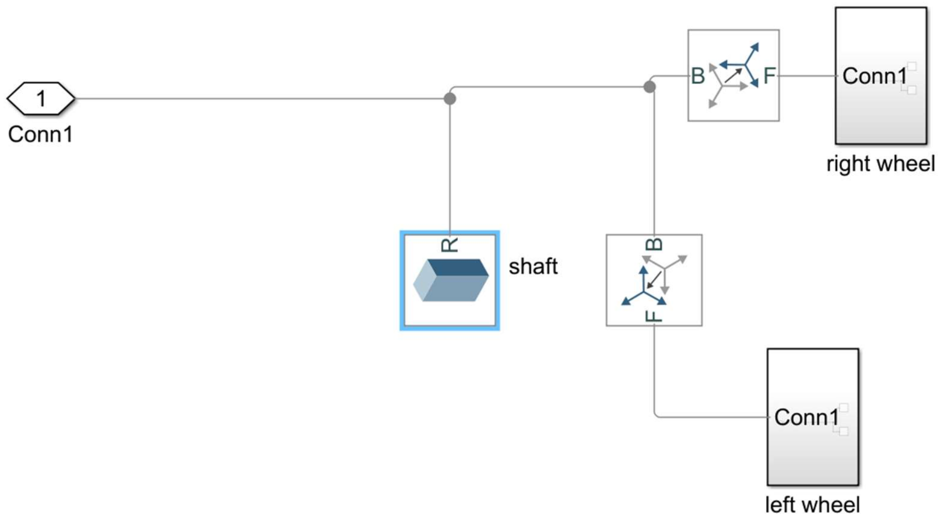
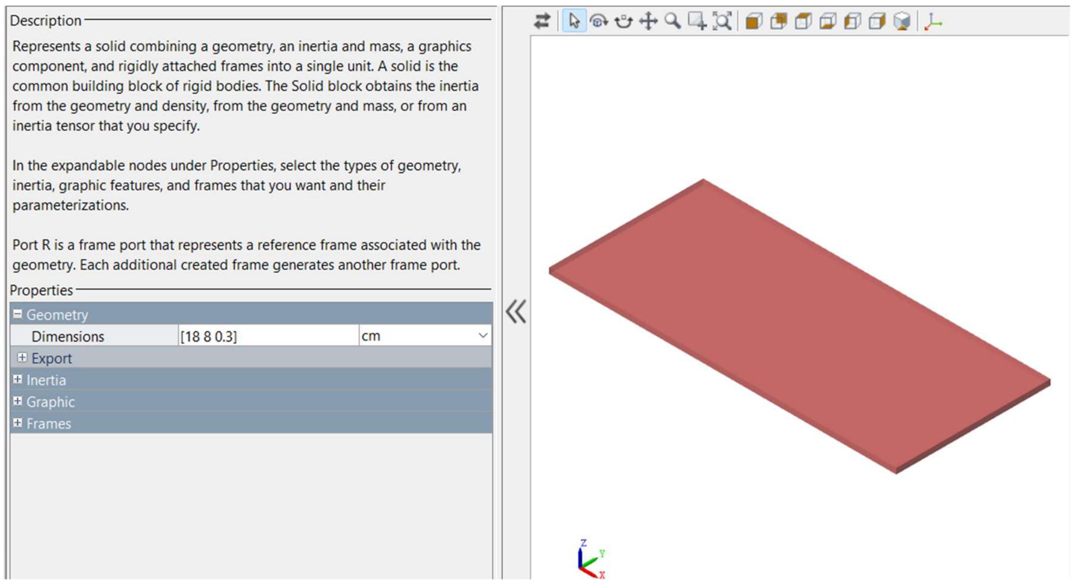


Figure:4.5 Connection of wheels and shaft for cart formation

4.2.3 Making the upper body:

With the help of Brick Solid Block in the Simscape Multibody I will form the plates by defining proper dimensions.



Description

Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

Properties

Geometry

Dimensions [18 8 0.3] cm

Export

Inertia

Graphic

Frames

Figure 4.6 Dimensions of plates for upper body

For the other two plates I just have to copy this model to form a new one. Then I will connect these using **Rigid Transform** Block. This Rigid Transform block connects two frames in a time- invariant transformation ().

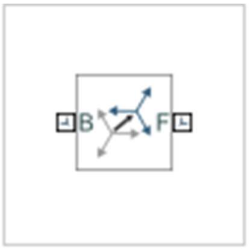
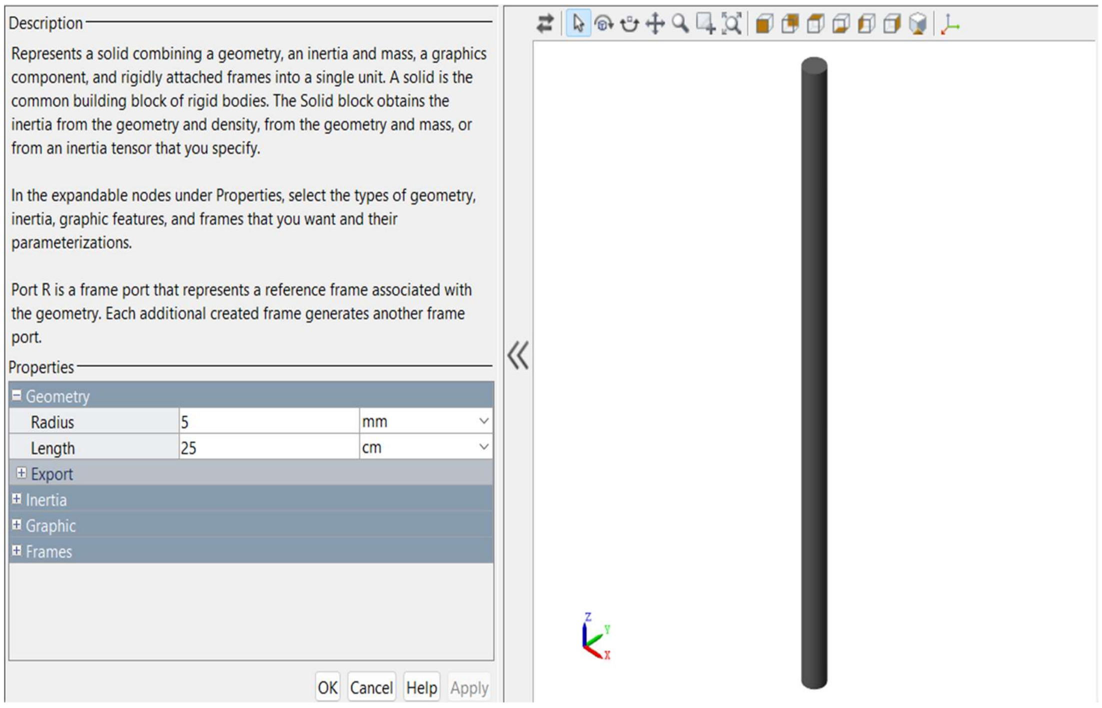


Figure 4.7 Rigid Transformation Block

I will have to connect the plates with the help of cylindrical pillars so I will have to define the dimensions of that also.



Description

Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandable nodes under Properties, select the types of geometry, inertia, graphic features, and frames that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry. Each additional created frame generates another frame port.

Properties

Geometry		
Radius	5	mm
Length	25	cm

Figure 4.8 Dimensions of Pillars connecting plates

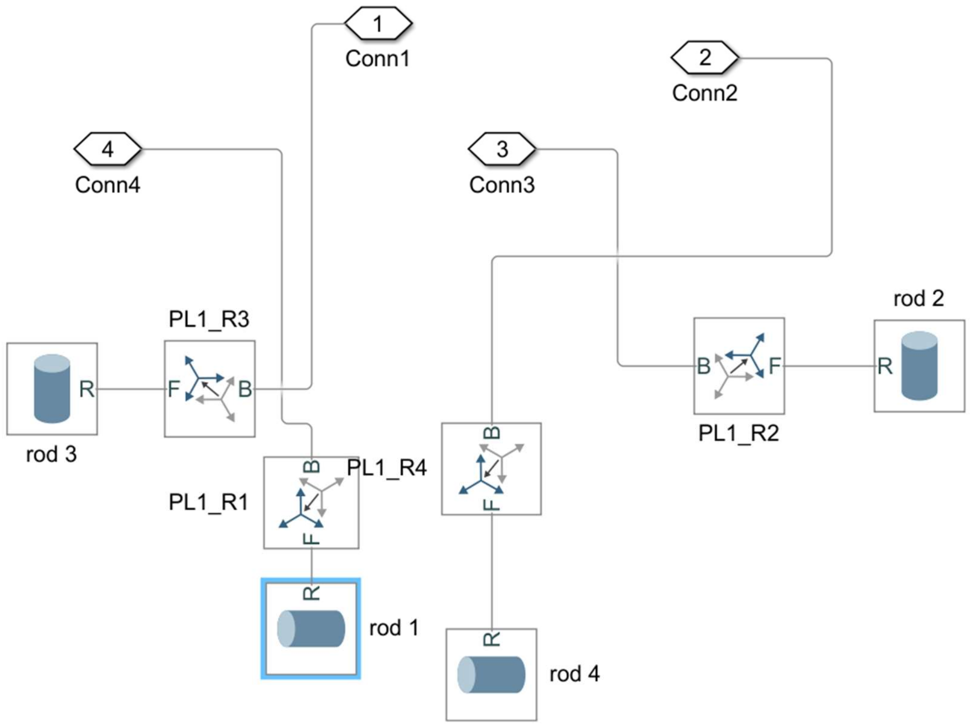


Figure 4.9 Connections of Pillars

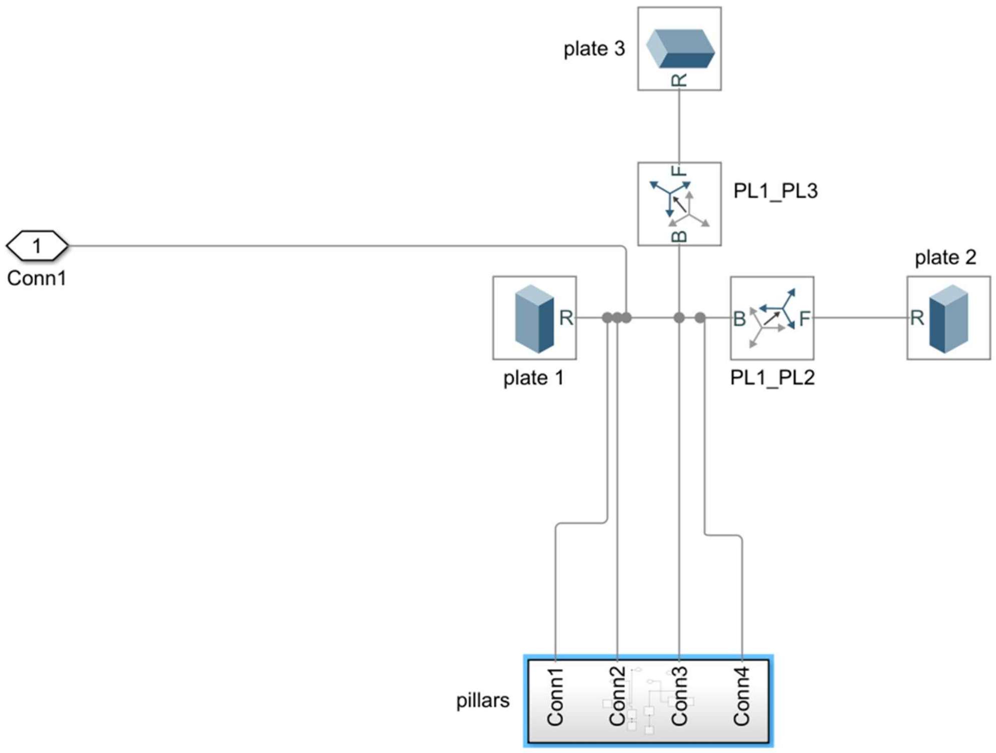


Figure 4.10 Connection of Chassis (upper body)

The two extra connections required for completing our model is the combination of prismatic joints and revolute joints. I know that prismatic joints are joints which can move in a translational motion but not in rotational motion. The connection between the cart and the chassis will be of revolute joint as due to the wheel, the cart will rotate. And the force applied to the cart will be of translational type. This force will balance the cart so that it is upright.

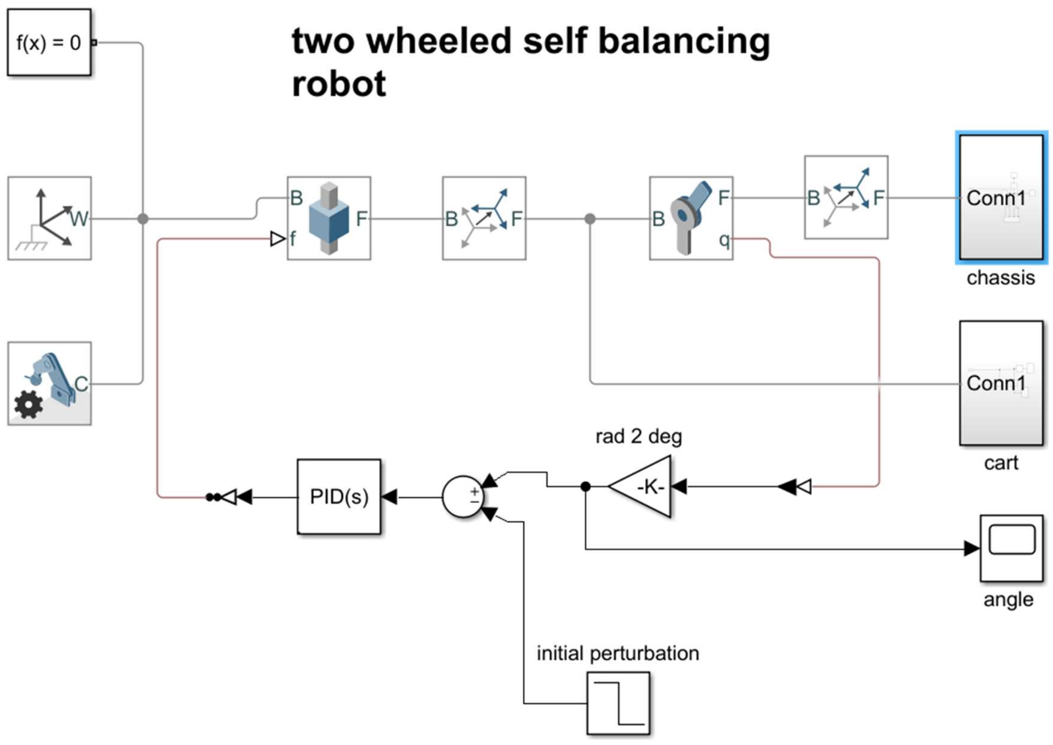


Figure 4.11 The complete Simulink model of Two Wheel Self Balancing Robot with PID controller.

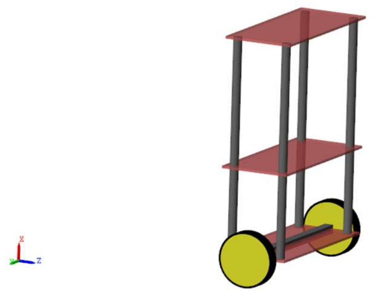


Figure 4.12 The final model after being assembled

CHAPTER 5
BUILDING THE CONTROL SYSTEM

5.1 Basic Control System Design

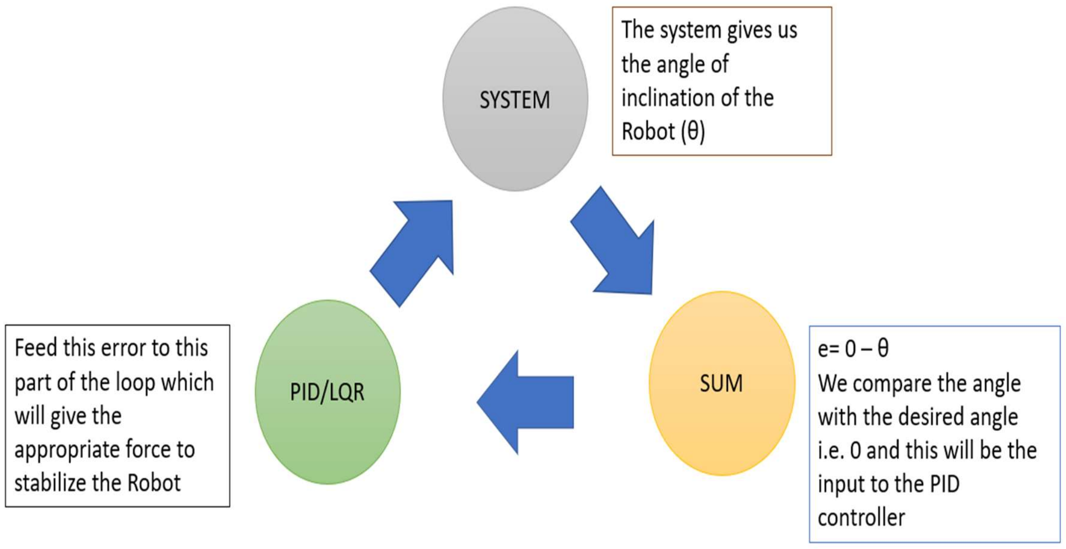

SYSTEM
The system gives us the angle of inclination of the Robot (θ)
SUM
e= 0 – θ
We compare the angle with the desired angle i.e. 0 and this will be the input to the PID controller
PID/LQR
Feed this error to this part of the loop which will give the appropriate force to stabilize the Robot

Figure 5.1 Control system design for Two wheel Self Balancing Robot

I can see that for building the control system I need to obtain the input which I get as the angle of inclination from the system as angle (θ). Now I just need to compare this angle with the desired angle which I want to be maintained by the system. I can see that our desired angle is 0, so I will obtain an error value by subtracting the angle obtained from the system from 0 and feed this input to the controller, namely PID or LQR.[8]

Now the controller will feed this error to the loop which will give the appropriate force to stabilize the Robot.[9]

5.2 PID Controller Overview

A proportional-integral-derivative controller is a standard response control. The PID controller considers "error" as the difference between the output and the given references I wish and attempts to reduce the error by adjusting the control parameters.[1]

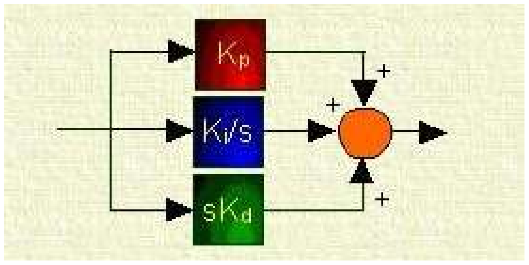


Figure 5.2 Control Parameters for PID controller

PID control parameters:

In the time-domain analysis, the output of a PID controller, which is proportional to control input is given by:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt} \tag{4.1}$$

To check how the PID controller works in a closed loop system using system variables. ‘e’ represents a system error due to both system sound and measurement noise, the difference between the output I want and the actual output. This error signal is provided by the PID controller, and the controller determines both the output and the value of the error. Plant inputs should be the sum of the recurring outflows with the outflow error, the equals of constant proportional error and significant periods of total error.

The transfer function of a PID controller is found by taking the Laplace transform of Eq

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \tag{4.2}$$

Where,

Kp = Proportional gain

Ki = Integral gain

Kd = Derivative gain

The structure of a plant:

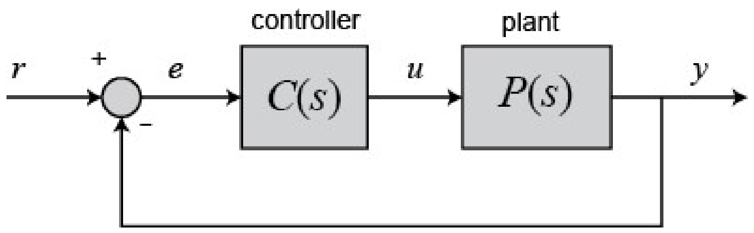


Figure 5.3 system with PID controller

5.3 Effects Of Control Parameters On The Close Loop System

Due to proportional controller, I will have reduced the rise time but no effect on steady state error. An integral control (K_i) reduces the steady-state error for step input, butnegative effect on rise time. A derivative increases the stability of the system as well as reduces the overshoot.

Table 4.1 PID parameter effect comparison

Kp	**Decrease**	**Increase**	**Small Change**	**Decreas e**
Ki	**Decrease**	**Increase**	**Increase**	**elimina te**
Kd	**Small Change**	**Decrease**	**Decrease**	**No change**

5.3 Pid Controller Design With State Space

The easiest method one should try to make the pendulum balanced is to rotate the wheels within the inclined direction until the inclination angle approaches to zero where the pendulum is in balance. Basically, the rotation speed of the wheel should be proportional to the angle of inclination (e.g. move faster when the inclination is more and vice versa) in order that the robot move with a greater settle time. this can be called the best PID control with neglecting both the I and also the D terms. [10]

Proceeding to the subsequent step within the design process, I've to search out state-feedback control gains represented during a vector assuming that I are cognizant (i.e. can measure) all the state variables (four state variables are there). There are various methods to try to it. If I recognize the specified closed-loop pole locations, I will use the advanced control theory. I am able to also use the "LQR" command which returns the optimal controller gain by heat and trial method for a linear plant, cost function must be power of two at the most and initial conditions must up to zero. [11]

I've got to test that the system is controllable before design a controller. By meeting all of this property of controllability implies that I am able to set the state of the system anywhere within the controllable region (under the physical constraints of the system). The system to fulfil all the conditions to be completely state controllable, the rank of the controllability is that the number of independent rows (or columns).

$$\mathcal{C} = [A|AB|A^2B|\cdots|A^{n-1}B]$$

The controllability matrix of the system is shown by the above equation. The quantity of powers indicates to the amount of state variables of the system. Addition of terms to the controllability matrix with higher powers of the matrix cannot increase the rank of the matrix because they're linear combination of each other.

Controllability matrix is consisting of 4 variables; the rank of the matrix should be 4 to be controllable. By using the command ctrl in MATLAB to get the controllability matrix. Likewise using rank command, I am able to find the rank. So, I am going to test in simulation chapter. [12]

5.4 Pre-Compensation

The designed controller meets our transient requirements so, but now I should always focus upon the steady-state error. With relation to the opposite design methods, where I feedback the output and compare it to the reference input to compute a slip, with a full-state feedback controller I are feeding back all of the states. I'd like to compute what the steady-state value of the states should be, multiply that by the chosen gain, and use a brand-new value as our "reference" for computing the input. I will mate by adding a continuing gain after the reference input.[13]

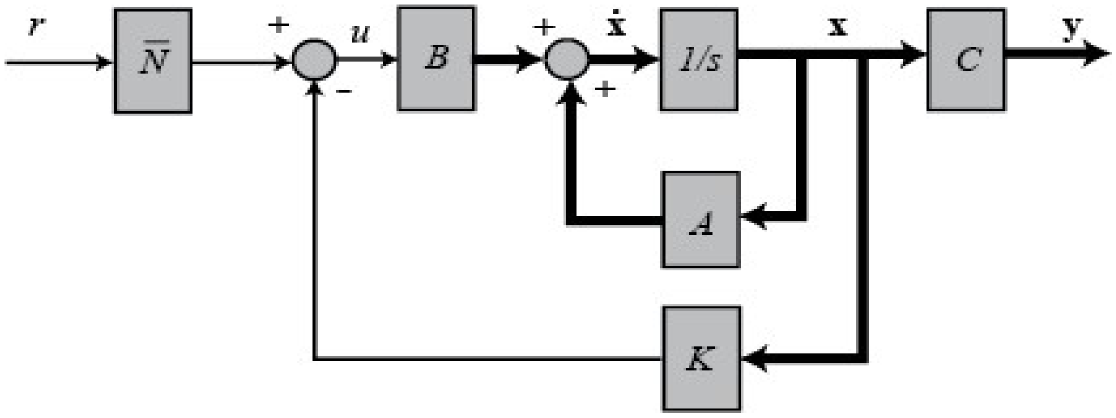


Figure 5.4 The Cart Pendulum System

CHAPTER 6
TUNING OF PID CONTROLLER

6.1 Controller Tuning

When a mathematical model of a system is obtainable, the parameters of the controller may be explicitly determined. However, due to the unavailability of a mathematical model, experimental determination of parameters is a must. Controller tuning is that the process of determining the controller parameters which produce the specified output. Controller tuning allows for optimization of a process and minimizes the error between the variable of the method and its point.[14][15]

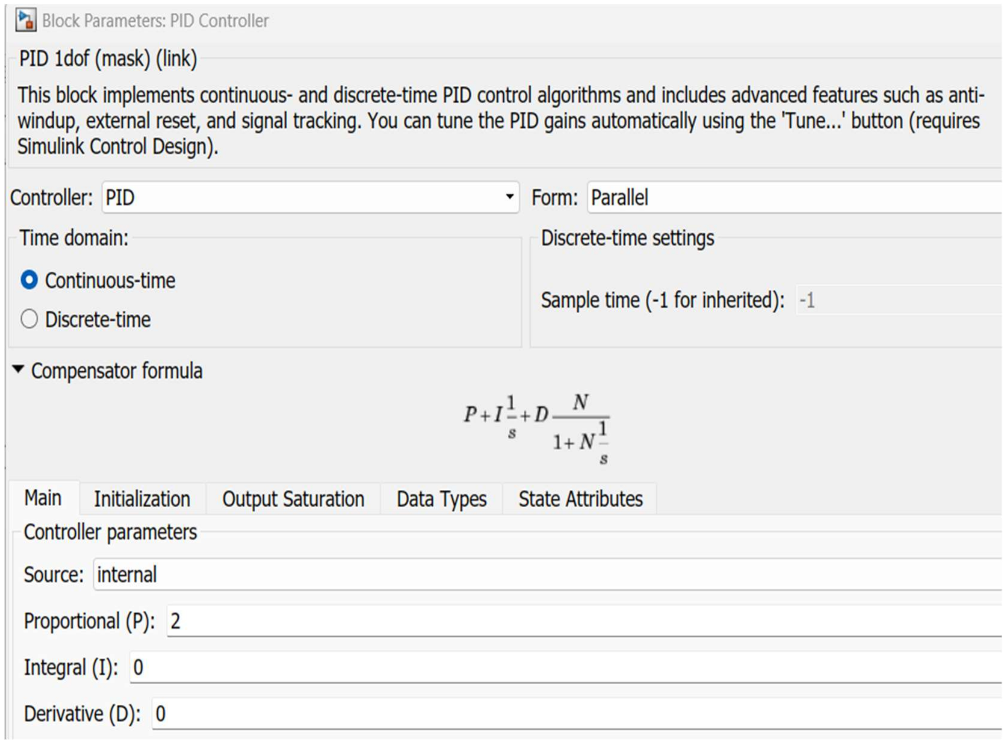
6.2 Tuning Methods

- MATLAB PID Auto Tuner Application (For Reference Values)
- Particle Swarm Optimization based Tuning of PID controller
- Genetic Algorithm based Tuning of PID controller

I also use hit and trial method for observing the controller reaction of Robot to the given control values. For example, I will set the value of Kp=2, Kd=0 and Ki=0. For changing these values, I will double click on the PID block on the Simulink model and change the values.

The figure 5.1 represents the controller parameters with random values. First, I have used random parameters to observe the control action and the result of what happens to the cart after applying these values. After that I have used actual tuning methods so that the accurate control action can be observed and the correct controller parameter rage can be determined.

The controller responds differently with different parameters obviously and sine there are three values to be determined, I have to be very careful in choosing the values with the hit and trial method. The result of choosing a not suitable value of the controller parameters can be perfectly seen in the figure 5.1.


Block Parameters: PID Controller

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID

Form: Parallel

Time domain:

- [✓ Continuous-time]
- Discrete-time

Discrete-time settings

Sample time (-1 for inherited): -1

Compensator formula

$$P+I\frac{1}{s}+D\frac{N}{1+N\frac{1}{s}}$$

Controller parameters

Source: internal

Proportional (P): 2

Integral (I): 0

Derivative (D): 0

Figure 6.1 Random values of controller parameters

Now I will see the reaction of the cart to these values.

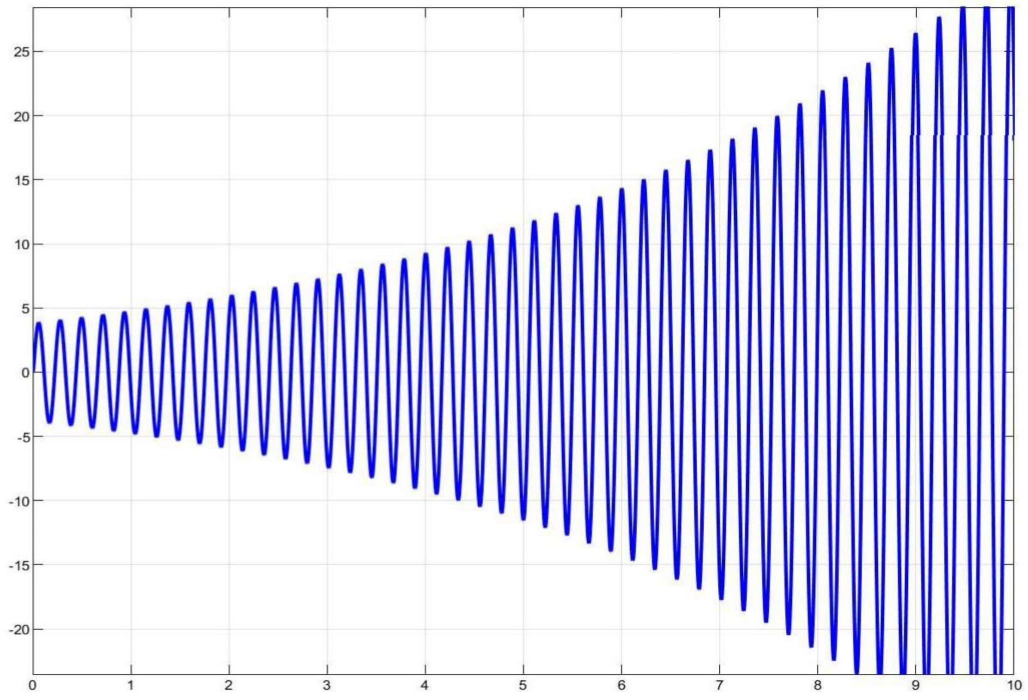


Figure 6.2 Angle with reference to the given control parameters

I can see that the disturbance of the angle keeps on increasing with respect to time and this shows us that the system is obviously not stable. So, I will try some different control parameters.

Now I will take the value of Kp=2, Kd=0.01 and Ki=3 and see the response of robot to these control parameters.

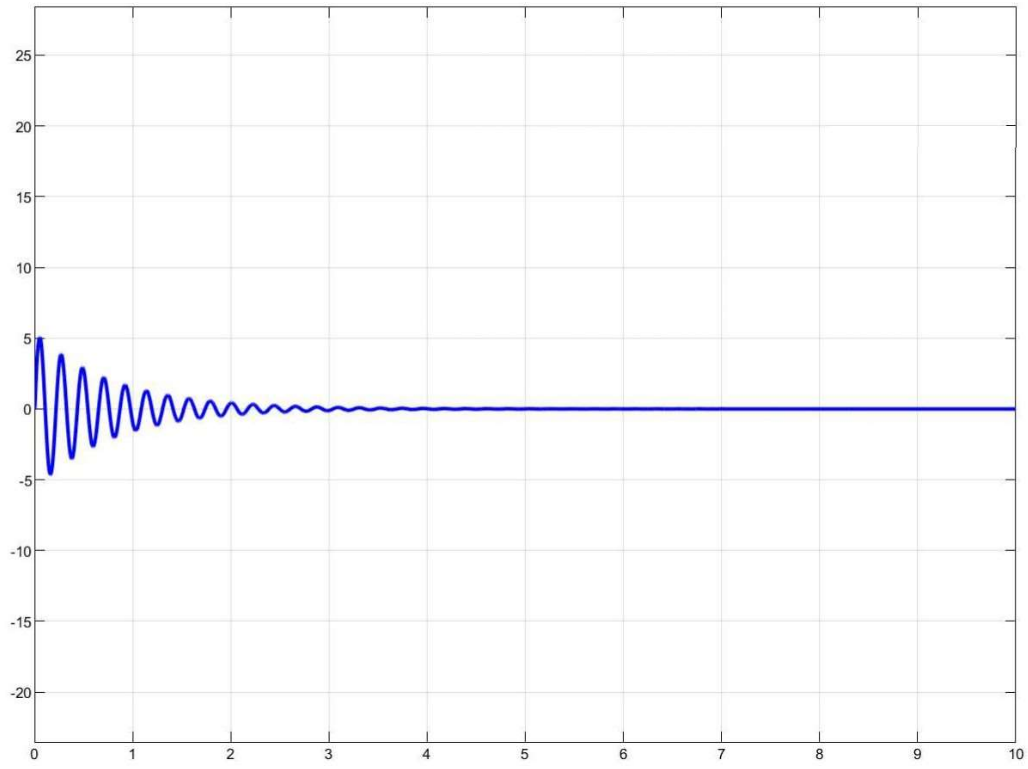
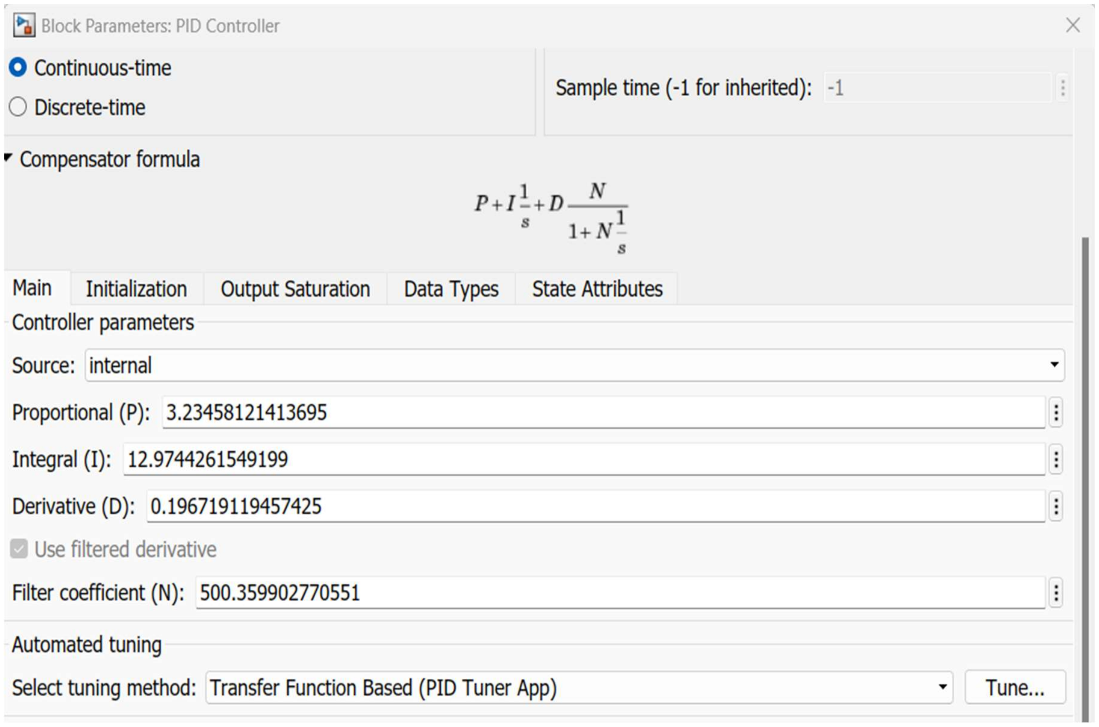


Figure 6.3 Angle disturbance w.r.t. Kp=2, Kd=0.01 and Ki=3

I can see that with new parameters I are able to achieve stabilization of the robot after some time. But I cannot always rely on hit and trial as it does not ensure the desired results after many trials also.

So, from now onwards I will try different tuning parameters of PID so that I can get ensured control parameters which will give us the best results.
For the first tuning method I will look forward to the MATLAB PID Tuner Application present in the Simulink. For this I have to double click on the PID block in the Simulink model, select the tuning method as **Transfer Function Based (PID Tuner App)** and

click on **Tune.**



Continuous-time [✓]
Discrete-time

Sample time (-1 for inherited): -1

$$P+I\frac{1}{s}+D\frac{N}{1+N\frac{1}{s}}$$

Controller parameters

Source: internal

Proportional (P): 3.23458121413695

Integral (I): 12.9744261549199

Derivative (D): 0.196719119457425

Use filtered derivative [✓]

Filter coefficient (N): 500.359902770551

Automated tuning

Select tuning method: Transfer Function Based (PID Tuner App)

Figure 6.4 PID control parameters with PID tuner app

6.3 Tuning Of PID With MATLAB Autotuner App

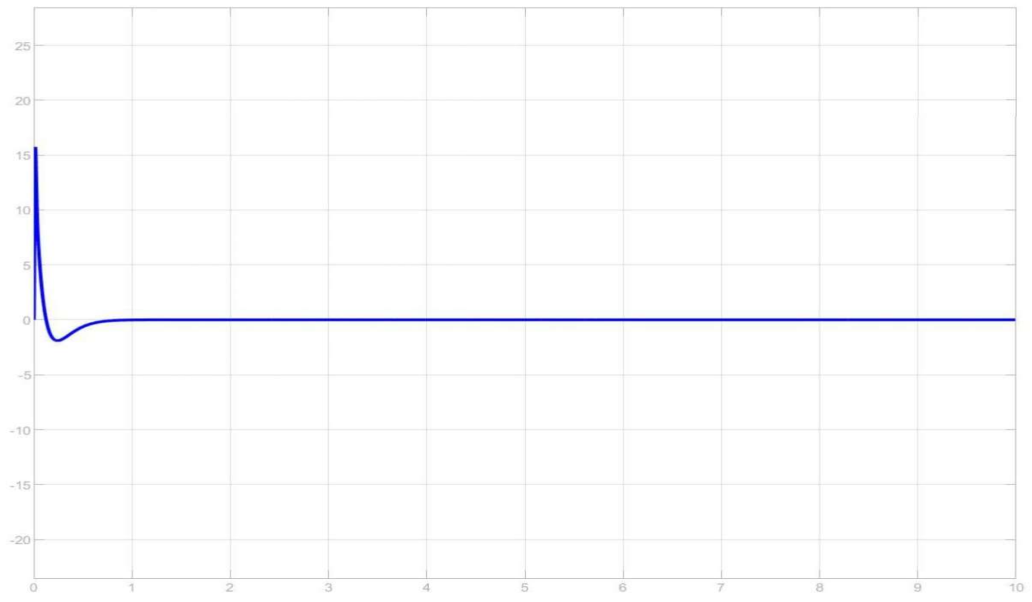


Figure 6.5 Angle disturbance with MATLAB autotuner PID parameters

The values obtained are, Kp= 3.245, Kd=12.9744 and Ki=0.1967

Now I can see that with the PID tuner app the disturbance or oscillation in the angle of the robot after initial disturbance is very less and the time taken to achieve the stability is extremely less as compared to the previous hit and trial method. Sine this application is inbuilt in the MATLAB I will consider these values as the standard values of Kp, Kd, Ki and compare these values to the other tuning parameters namely, PSO based PID tuning and GA based PID tuning.

CHAPTER 7
GA BASED TUNING OF PID CONTROLLER

7.1 Introduction To GA

Genetic Algorithms are a family of computer models inspired by evolution. These algorithms incorporate a potential solution to a specific problem in a simple data structure such as a chromosome and use regenerative operators in these structures to store sensitive information. Genetic algorithms are often regarded as operating systems, although the scope of the problems with which the genetic algorithms are used is much broader.[16]

The implementation of a genetic algorithm begins with the human population (usually random) of chromosomes. Man, then examines these structures and assigns them to reproduction in such a way that those chromosomes represent a better solution to the intended problem given more opportunities for "reproduction" than those chromosomes are the worst solutions. The "advantage" of a solution is usually defined in terms of the current population.[16]

This particular definition of a genetic algorithm is deliberately absurd because in a sense, the word genetic algorithm has two meanings. In solid translation, the genetic algorithm refers to a model that was developed and researched by John Holland (1975) and Holland students (e.g., DeJong. 1975). Yet most of the existing theory of genetic algorithms applies only to or primarily to the model presented by Holland, as Ill as the variance in what will be referred to in this paper as a canonical genetic algorithm. Recent theoretical developments in genetic algorithms are particularly applicable to the canonical genetic algorithm (overall, 1993).

In the broader application of the term, the genetic algorithm has any human-based model that uses selecting and reassembling operators to generate new sample points in the search field. Many models of the genetic algorithm have been developed by highly functional researchers from a experimental point of view. Many of these researchers are prone to use and are often interested in genetic algorithms as development tools.

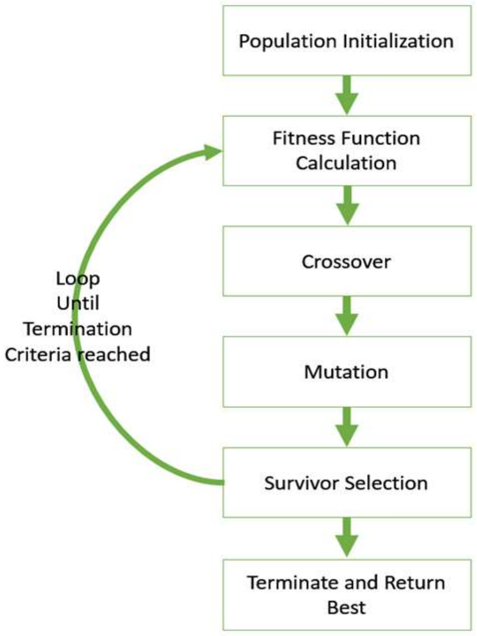

Population Initialization
Fitness Function Calculation
Crossover
Mutation
Survivor Selection
Terminate and Return Best
Loop Until Termination Criteria reached

Figure 7.1 Flow Chart of Genetic Algorithm[17]

7.2 Implementation Of GA For Tuning PID In MATLAB [18]

Here the Survivor Selection is nothing but the objective function which decides the fitness of the value. Here objective function is also known as cost function.

So, I will define the cost function in MATLAB and use the Global optimization toolbox for implementing the Genetic Algorithm solver to tune the parameters of the PID with the help of the transfer function obtained in the first chapter.
Now I will open the optimization toolbox and define the function which will be used as cost function.

Then the GA solver will come into play and run several iterations and then stabilize at some particular range of values. I can stop after 5 iterations as it is enough for getting the desired values and note the values of Kp, Kd, and Ki.

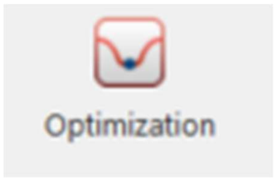
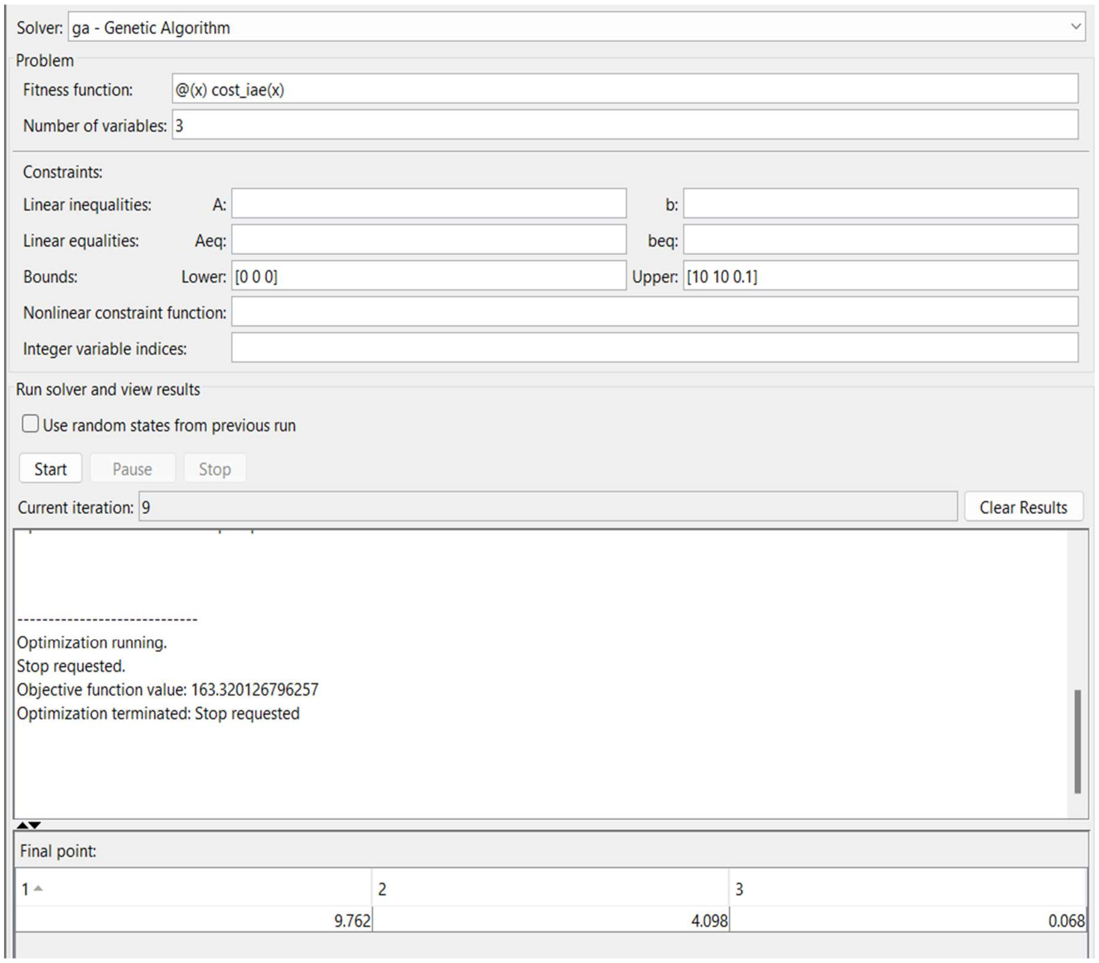


Figure 7.2 Optimization Toolbox


Solver: ga - Genetic Algorithm
Problem
Constraints:
Linear inequalities: A: b:
Linear equalities: Aeq: beq:
Nonlinear constraint function:
Integer variable indices:
Run solver and view results
Use random states from previous run
Start
Pause
Stop
Clear Results
Final point:

Fitness function: @(x) cost_iae(x)
Number of variables: 3
Bounds: Lower: [0 0 0] Upper: [10 10 0.1]
Current iteration: 9

```
----------------------------
Optimization running.
Stop requested.
Objective function value: 163.320126796257
Optimization terminated: Stop requested
```

1	2	3
9.762	4.098	0.068

Figure 7.3 Values of Control parameters obtained after 8 iterations

As shown above, I get the values of control parameters as, Ki=9.762, Kp=4.098 and Kd=0.068.

Now I will put these PID parameter values in the PID block in our Simulink model of the Robot. After that I will run the model and observe the results.
The x-axis in the graph shows time and the y axis in the graph shows angle. Figure 6.5 represents the result of the GA based PID controller method. We can observe that the initial rise in angle is lesser in comparison to the MATLAB Autotuner App.

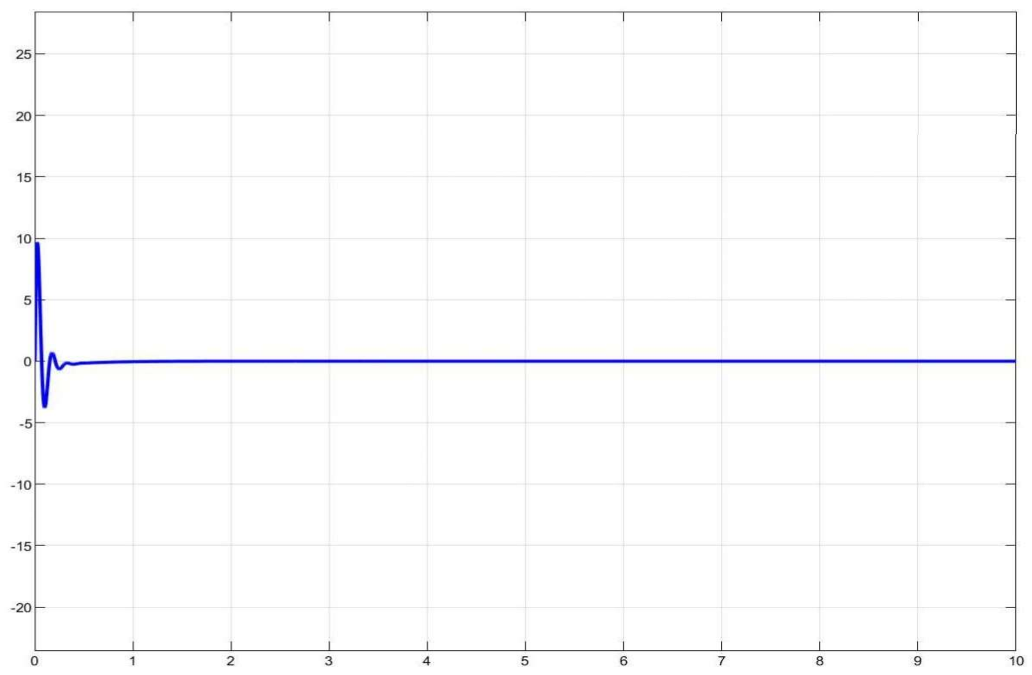


Figure 7.4 Angle disturbance in TWR due to Control Parameters by GA

CHAPTER 8
PSO BASED TUNING OF PID

8.1 Introduction To PSO [19]

I am sure that each of us in our lives has heard from those who wish the best, “Have a good relationship. It helps I to cultivate a positive attitude.” When I talk about ‘good company,’ I are talking about the unequal distribution of good qualities among team members in order to achieve the same better goal. That’s why I always say ‘Work as a Team.’ The Particle Swarm Optimization (PSO) Algorithm is based on that. In 1995, Kennedy and Eberhart wrote a research paper based on the social behavior of animal groups, in which they argued that sharing information between groups increased the survival rate. Just as a bird seeks its prey at random, it can enhance its search when working with a herd. The benefit of the operation is to share the best information, which can help the herd to find the best hunting ground.[20]

In computational science, particle swarm optimization (PSO) is a calculation method that makes the problem over and over again try to improve the candidate's solution for a certain level of quality. It solves the problem by having a number of candidate solutions, here called particles, and moving these particles to the search according to a simple mathematical formula for particle structure and speed. The movement of each particle is influenced by its known location, but is also directed to the most Ill-known areas in the search field, which are updated as better areas are found by other particles. This is expected to advance the best solutions.

In short, the PSO is encouraged to seek food and social morality. It was initially suggested that there should be continuous indirect activities. The PSO is developed using two methods, Artificial Life which mimics a herd of birds, fish learning, and swarm theory and the other is the Evolutionary Computation.

The group seeks food in a collaborative way and each member of the group learns experience with them and other members by changing the search pattern to find food. The

PSO is developed using simple concepts and older operators. The PSO is mathematically inexpensive in both memory and speed, and can be easily implemented using a computer program. The PSO starts by launching a random population like GA. Unlike GA operators, solutions are provided at a random speed to check the search space. Each solution in the PSO is called a particle.

Three distinct features of PSO

- Best fitness of each particle
- Best fitness of swarm
- Velocity and position update of each particle

pbest i: the best solution (fitness) achieved so far by particle i

gbest: the best solution (fitness) achieved so far by any particle in the swarm

Velocity and position update: for exploring and exploiting the search space to locate the optimal solution.

8.2 Position And Velocity Calculation [20]

Position of particle (i) is adjusted as

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \tag{7.1}$$

Velocity of particle (i) is updated as follows:

$$v_i^{(t+1)} = wv_i^{(t)} + c_1 r_1 \left(p_{(i,lb)}^{(t)} - x_i^{(t)}\right) + c_2 r_2 \left(p_{gb}^{(t)} - x_i^{(t)}\right) \tag{7.2}$$

- i is the $i-th$ particle.
- c_1 and c_2 are the acceleration coefficients.
- t is the generation counter
- r_1 and r_2 are random numbers $\in [0,1]$.
- $v_i^{(0)}$ set randomly
- $p_{(i,lb)}^{(t)}$ is the local best of $i-th$ particle
- w adds to the inertia of the particle
- $p_{gb}^{(t)}$ is the global best.

Momentum part, $wv_i^{(t)}$

- Inertia component
- Memory of previous flight direction
- Prevents particle from drastically changing direction

Cognitive Part, $c_1 r_1 \left(p_{(i,lb)}^{(t)} - x_i^{(t)}\right)$:

- Quantifies performance relative to past performances
- Memory of previous best position
- Nostalgia

Social Part, $c_2 r_2 \left(p_{gb}^{(t)} - x_i^{(t)}\right)$:

- Quantifies performance relative to neighbors
- Envy

Geometrical Illustration of Velocity Components

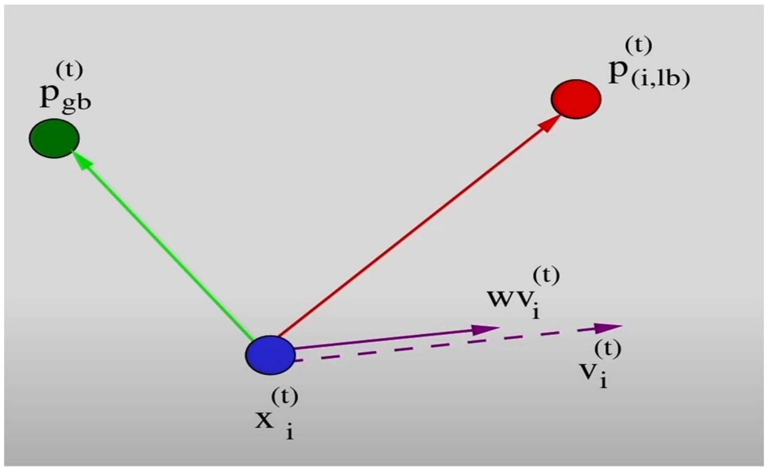


Figure 8.1 Momentum part, $wv_i^{(t)}$

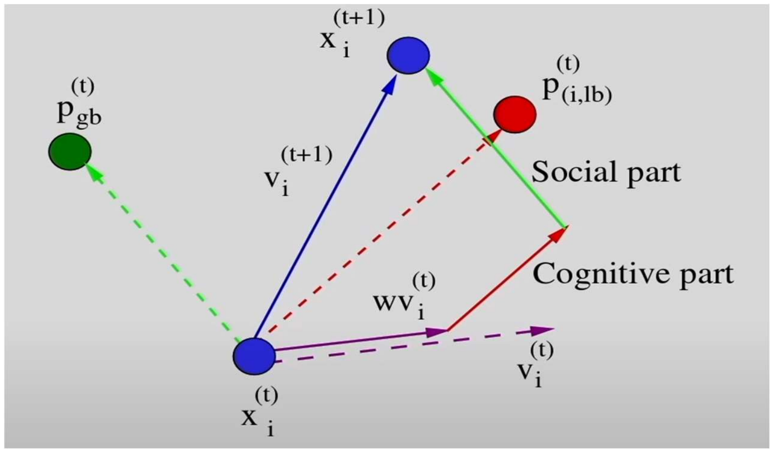


Figure 8.2 Cognitive Part, $c_1 r_1 \left(p_{(i,lb)}^{(t)} - x_i^{(t)}\right)$ Social Part, $c_2 r_2 \left(p_{gb}^{(t)} - x_i^{(t)}\right)$

$p_{(i,lb)}^{(t)}$ is the personal best position of i – th particle in t+ generation.

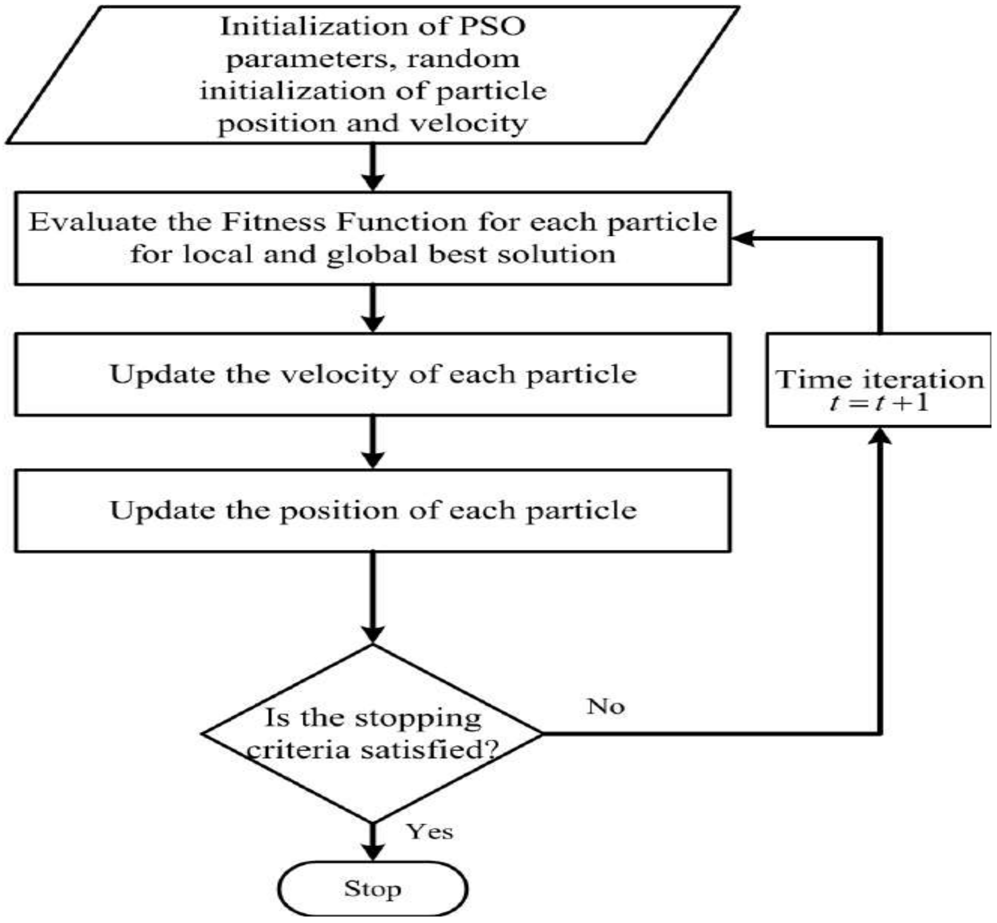

Initialization of PSO parameters, random initialization of particle position and velocity
Evaluate the Fitness Function for each particle for local and global best solution
Update the velocity of each particle
Update the position of each particle
Is the stopping criteria satisfied?
No
Yes
Stop
Time iteration $t = t + 1$

Figure 8.3 Flow Chart of PSO algorithm

8.3 Tuning Of PID With PSO Algorithm Using MATLAB:[19]

Some key steps in coding the PSO algorithm to tune PID are-

1- Choose number of variables as 3
2- Use the objective function as used earlier
3- The codes like 'particleswarm','MaxIterations',20,'SwarmSize',50,'PlotFcns', are already present in MATLAB Simulink and is the part of Global optimization toolbox.

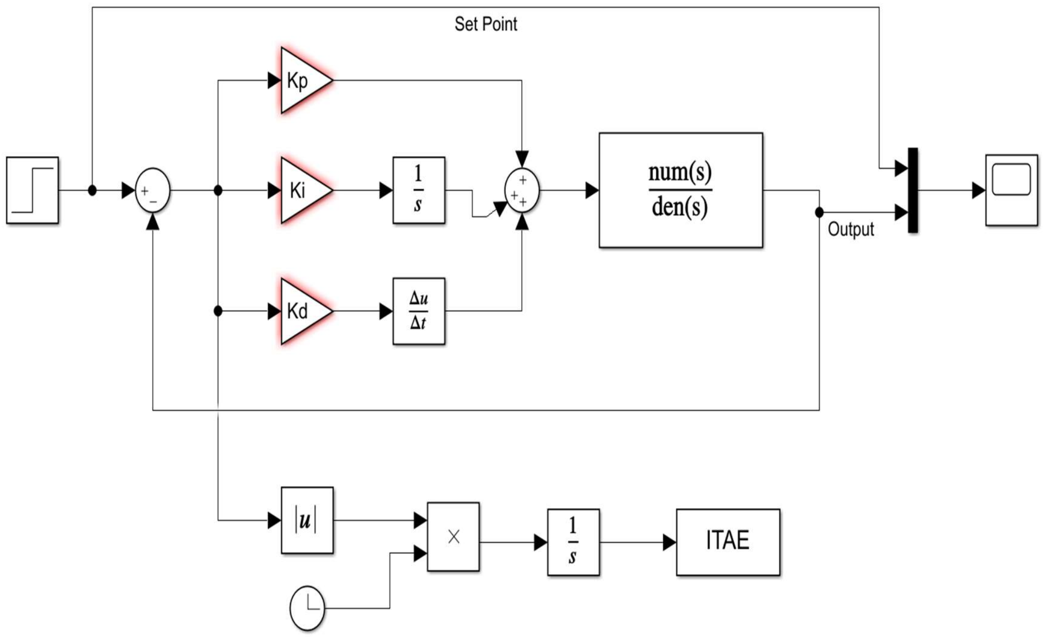
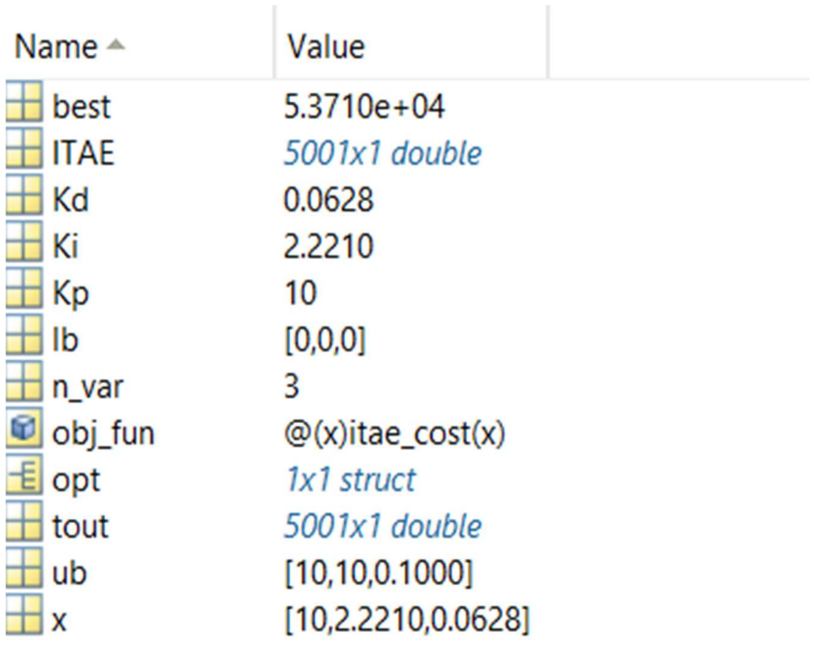


Figure 8.4 SIMULINK model for obtaining the tuning parameters with the transfer function.



Name	Value
best	5.3710e+04
ITAE	5001x1 double
Kd	0.0628
Ki	2.2210
Kp	10
lb	[0,0,0]
n_var	3
obj_fun	@(x)itae_cost(x)
opt	1x1 struct
tout	5001x1 double
ub	[10,10,0.1000]
x	[10,2.2210,0.0628]

Figure 8.5 The parameters obtained for PID are Ki=10, Kd=0.0628 and Kp= 2.2210

Now I will update these values of Kp, Kd and Ki in the Two Wheel Self Balancing model and trace the response w.r.t these control parameters.

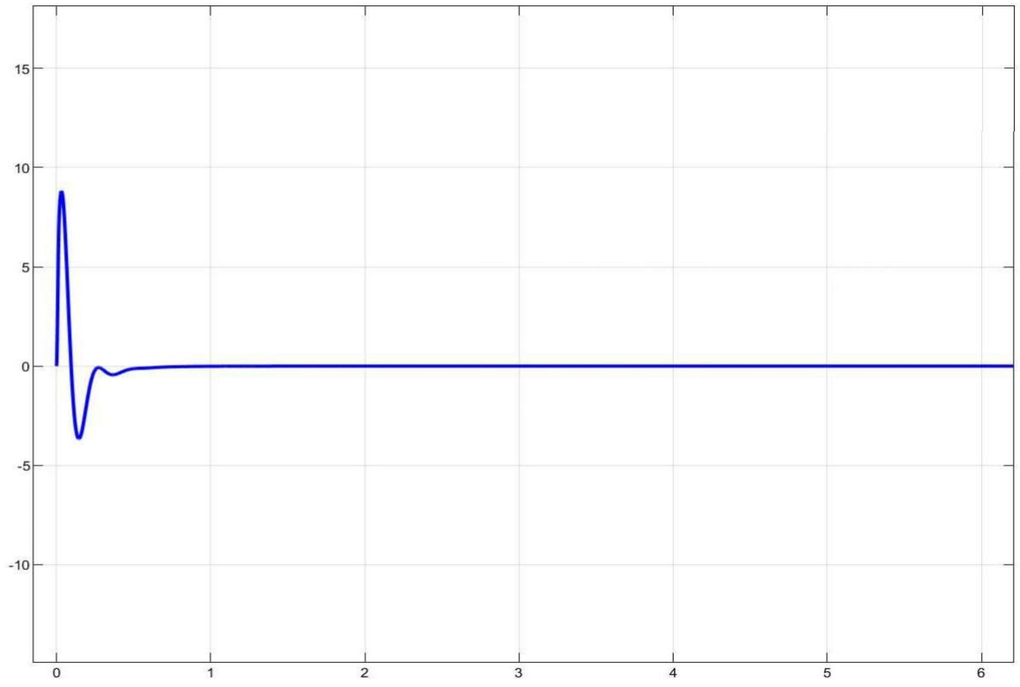


Figure 8.6 Angle disturbance in TWR due to Control Parameters by PSO

CHAPTER 9
LQR CONTROL OF TWO WHEEL SELF BALANCING ROBOT

9.1 Introduction To LQR Controller [13]

Complete control theory is about flexible system performance at low cost. A case in which the dynamics of a system is defined by a set of dividing line calculations and the cost is defined by a quadratic function is called the LQ problem. One of the main implications of this theory is that the solution is provided by a linear-quadratic regulator (LQR), a response control with the numbers given below. LQR is an integral part of the solution of the LQG (linear-quadratic – Gaussian) problem. Like the LQR problem itself, the LQG problem is one of the basic problems in control theory.

The settings of the controller (controller) that controls the machine or process (such as a plane or chemical reactor) are obtained using a mathematical algorithm that reduces the cost of work by the weight features provided by the person (engineer). Cost work is often defined as the sum of the basic measurements, such as height or processing temperature, from the desired values. The algorithm thus detects those control settings that minimize unwanted deviations. The magnitude of the regulatory action itself can also be incorporated into the cost function.[21]

The LQR algorithm reduces the amount of work done by the control system engineer to improve control. However, the engineer still needs to specify cost work parameters, and compare the results with the specific design goals. This usually means that the construction of the controller will be a repetitive process in which the engineer judges the "correct" simulated controls and then adjusts the parameters to produce a controller that best complies with the design principles.

The LQR algorithm is actually the default way to find the right state response controller. Thus, it is not uncommon for control engineers to choose alternatives, such as a complete country response, also known as pole placement, where there is a clear relationship between control parameters and control behavior. Difficulty in obtaining appropriate weight characteristics limits the use of LQR based controls.

LQR is one of the ways to design control systems in the region. Compared to how to place the poles, this method gives us a way to get the best poles for the system. In addition, this method is able to maintain a balance between the performance of the poles in providing the maximum response and power applied to the actuators.

9.2 Cost Function For LQR Controller [22]

The quadratic cost function responsible for measuring is:

$$J = \int_0^\infty x^T Qx + u^T Ru \tag{8.1}$$

The optimal feedback control law is:

$$u = -K_r x \tag{8.2}$$

Where K_r is the optimal feedback gain matrix obtained as:

$$K_r = R^{-1}B^T P \tag{8.3}$$

P is a real symmetric matrix, the solution to Riccati equation:

$$PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{8.4}$$

In order to design LQR controller in MATLAB first of all I should examine the system to check whether it is controllable. If the system is controllable, the poles are placed where the system is stable. Then, LQR controller is designed by determining the optimal state feedback gain matrix K_r using the following function in MATLAB:

$$K_r = lqr(A, B, Q, R) \tag{8.5}$$

The values of Q and R have been chosen by trial in simulation.

9.3 Implementation Of LQR Controller

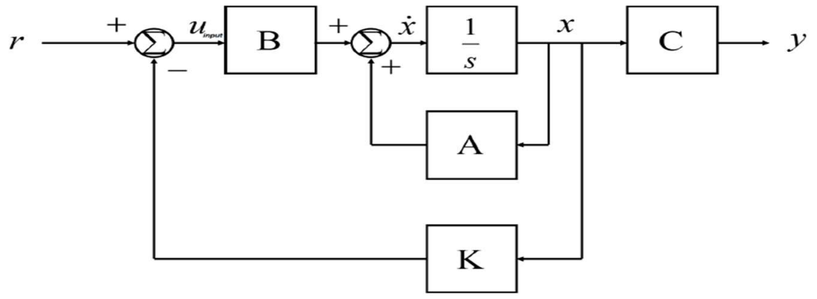


Figure 9.1 Block representation of state space system showing the system matrices A, B and C as well as the gain matrix K.[22]

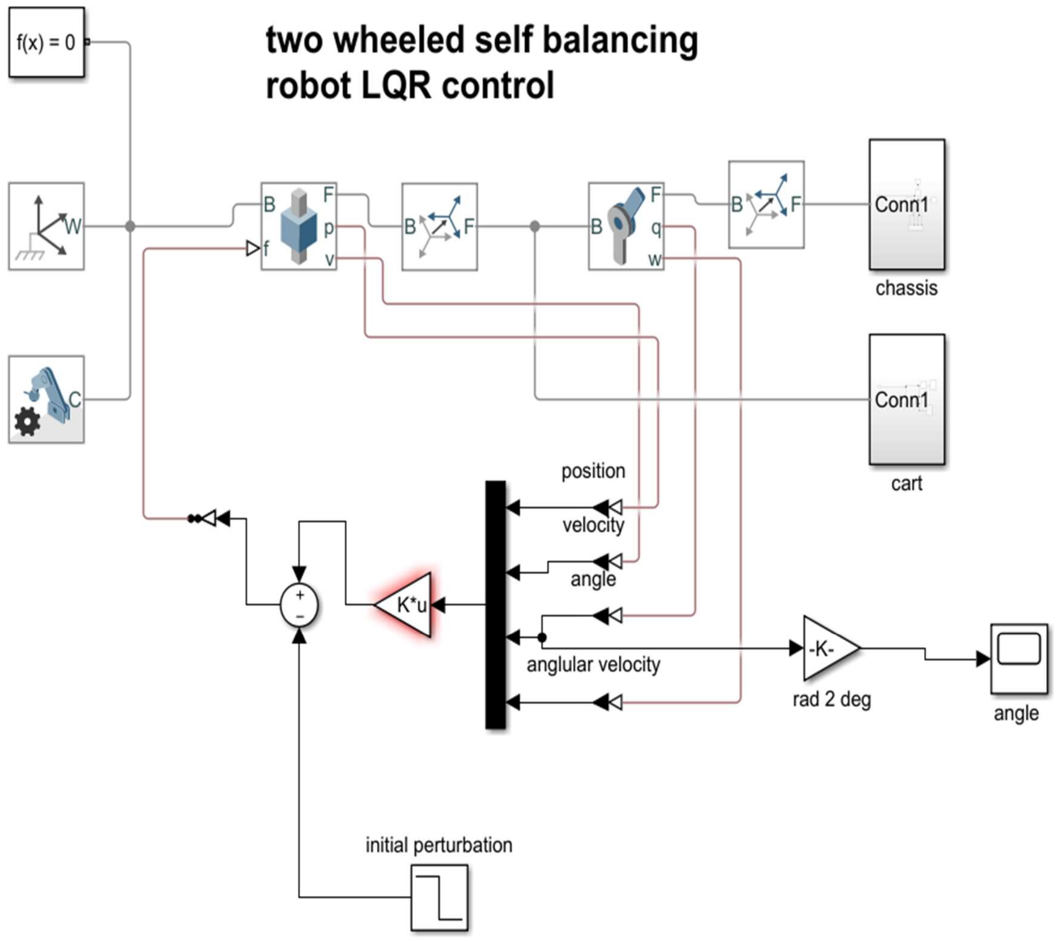


Figure 9.2 Simulink model for Two wheeled self-balancing robot using LQR control

The LQR controller first linearizes the model and then provides the control parameters to the system. The value of gain matrix obtained in the MATLAB script code is directly fed to the Simulink model and henceforth the control action is provided by the controller.

The figure 8.4 gives the result of the disturbance of angle faced by the system in the LQR controller. We can evidently observe that the oscillations in the angle is very less and the stabilization graph of the system is very smooth. Obviously the time taken by the system to stabilize itself is more but the change in angle is very less.

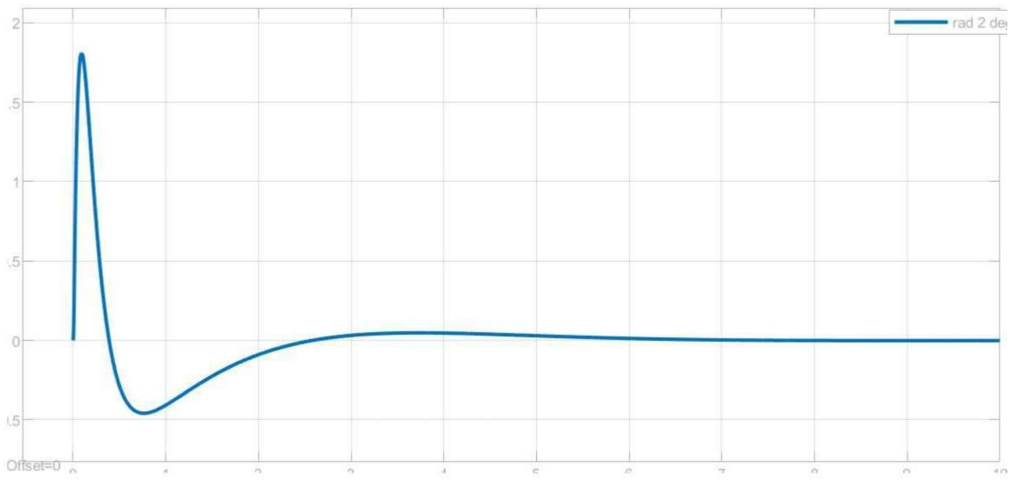


Figure 9.3 Angle disturbance with the LQR control method

CHAPTER 10

CONCLUSION

10.1 Theoretical Comparison Of PSO And GA

Particle Swarm Optimization (PSO) is a relatively recent heuristic algorithm which is based on the behavior of swarming characteristics of living organisms. PSO is closely similar to the GA as these two are efficient search methods which means that PSO and the GA change from one set of points to another set of points within each iteration with remarkable improvement from the previous data using some probabilistic and deterministic rules. Conversely, the GA is a well-established and popular algorithm with many applications and different versions.

Although both GA and PSO are an important part of the evolutionary optimization algorithms, they do have some disadvantages which limits their usage to only a few problems. In order to solve these problems a combination of both GA and PSO can be used to improve the overall performance. Combining these two algorithms together can lead to create a strong algorithm that has practical values and combines the advantages of PSO and GA. So, a hybrid algorithm of GA and PSO can be a good topic for future research.

10.2 Practical Comparison Of GA Based PID And PSO Based PID Controller

First of all, lets take a look at the control parameters i.e., Kp, Kd and Ki obtained from different Tuning methods used above.

Table 10.1 Comparison of Kp, Kd, and Ki

	Kp	Kd	Ki
MATLAB AUTO TUNER APP	*3.235*	*12.974*	*0.1967*
GA based PID Tuning	4.363	9.610	0.083
PSO based PID Tuning	2.221	10	0.062

I are taking the MATLAB auto tuner app as the reference and hence I are comparing the GA based PID and PSO based PID results to see that which method can be used for effective results.

As the results shows, Both the algorithms namely GA and PSO are able to produce results close enough to the MATLAB auto tuner app. So, this motivates us to tell that both the algorithms can be effectively used to perform search operations or optimization problems effectively with right parameters known to us.

There is one more parameter which I can consider, which is the time our cart took to achieve stability. In the table below I can see the time taken by each method to help the cart gain stability.

Table 10.2 Comparison of Angle and time taken to gain stability

	Time taken to Gain stability (Approximately)	Maximum Amplitude of Angle reached (degrees)
MATLAB AUTO TUNER APP	*0.75 seconds*	*15*
GA based PID Tuning	0.5 seconds	12.5
PSO based PID Tuning	0.65 seconds	12

Though I can see that GA and PSO tuning methods are able to produce better results but oscillations present in the MATLAB Auto tuner app is extremely less compared to both the algorithm. Nevertheless, it is some sort of achievement of this project that I am able to produce better results than the reference at some aspects.

10.3 Comparison Of PID Controller With LQR Technique

It is known that PID controller does not linearizes the model and gives the control action with the help of control parameters which we are able to provide the system. The PID controller may give a result which is quick or which makes the response of the system fast. But the result may not be optimal. In practical world I need optimal solutions because the cost of reducing the time may be a lot more.

That’s why at some places I use optimal control methods to control the system. The reason for using LQR exclusively in this project is only to check how the results will be affected. LQR uses the state space representation of the model and generates the gain matrix K. which is the used in the Simulink model to provide the control action to our model.

Table 10.3 Comparison of PID with LQR

	Time taken to Gain stability (Approximately)	Maximum Amplitude of Angle reached (degrees)
MATLAB AUTO TUNER APP	0.75 seconds	15
LQR	5 seconds	1.75

I can clearly see that LQR control method takes significant amount of time to stabilize the robot but the Angle disturbance is a lot less. If implemented practically it will be very useful in long run as the low the angle disturbance will be, the physical toll on the body of the robot will also be very less and the maintenance cost would be a lot less. Due to this the life of the model will also increase significantly. So, in practical implementation I need to look at the aspect of optimal control.

REFERENCES

[1] P. Kumar Tripathy, "SELF-BALANCING BOT USING CONCEPT OF INVERTED PENDULUM." [Online]. Available: www.nitrkl.ac.in [2013] Project Report

[2] H. Hellman, H. Sunnerman, and M. E. Grimheden, "Two-Wheeled Self-Balancing Robot Design and control based on the concept of an inverted pendulum," 2015.

[3] A. v. Putov, E. v. Ilatovskaya, and M. M. Kopichev, "Self-balancing Robot Autonomous Control System," Jun. 2021. doi: 10.1109/MECO52532.2021.9459720.

[4] U. Adeel, K. Saleem Alimgeer, and A. Hameed, "Autonomous Dual Wheel Self Balancing Robot Based on Microcontroller Robotics View project Designing MIMO antenna with reduced coupling View project," 2013. [Online]. Available: www.textroad.com

[5] L. Kakinada and K. Singh, "Modelling and analysis of two-wheeled self balanced robot," May 2021. doi: 10.1109/INCET51464.2021.9456255. [2021] "2nd International Conference for Emerging Technology (INCET) Belgaum, India. May 21-23, 2021"

[6] F. Jeremic, Lecture Notes "Background Inverted Pendulum Visualization Derivation Without Oscillator Derivation With Oscillator Derivation of Equations of Motion for Inverted Pendulum Problem Background Inverted Pendulum Visualization Derivation Without Oscillator Derivation With Oscillator Kinetic Energy Definition The energy which an object possesses due to its motion Background," 2012.

[7] H. Bin, L. W. Zhen, and L. H. Feng, "The kinematics model of a two-wheeled self-balancing autonomous mobile robot and its simulation," in *2010 2nd International Conference on Computer Engineering and Applications, ICCEA 2010*, 2010, vol. 2, pp. 64–68. doi: 10.1109/ICCEA.2010.169.

[8] A. Mathew, R. Ananthu, P. Binsy, A. Vahid, C. Thomas, and S. Sidharthan, "Design and control of a two-wheel self-balancing robot," *IOP Conference Series: Materials Science and Engineering*, vol. 1114, no. 1, p. 012058, Mar. 2021, doi: 10.1088/1757-899x/1114/1/012058.

[9] Y. Zhuang, Z. Hu, and Y. Yao, “Two-wheeled self-balancing robot dynamic model and controller design,” in *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, Mar. 2015, vol. 2015-March, no. March, pp. 1935–1939. doi: 10.1109/WCICA.2014.7053016.

[10] T. Nikita and K. T. Prajwal, “PID Controller Based Two Wheeled Self Balancing Robot,” in *Proceedings of the 5th International Conference on Trends in Electronics and Informatics, ICOEI 2021*, Jun. 2021, pp. 1–4. doi: 10.1109/ICOEI51242.2021.9453091.

[11] F. F. Rabbany, A. Qurthobi, and A. Suhendi, “Design of Self-Balancing Virtual Reality Robot Using PID Control Method and Complementary Filter,” in *Proceedings - 2021 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology, IAICT 2021*, Jul. 2021, pp. 15–19. doi: 10.1109/IAICT52856.2021.9532576.

[12] A. S. Wardoyo, S. Hendi, D. Sebayang, I. Hidayat, and A. Adriansyah, “An investigation on the application of fuzzy and PID algorithm in the two wheeled robot with self balancing system using microcontroller,” in *Proceedings - 2015 International Conference on Control, Automation and Robotics, ICCAR 2015*, Jul. 2015, pp. 64–68. doi: 10.1109/ICCAR.2015.7166003.

[13] J. Dabbagh and I. H. Altas, “Nonlinear Two-Wheeled Self-Balancing Robot Control Using LQR and LQG Controllers.” .”[2019 11th International Conference on Electrical and Electronics Engineering (ELECO)]

[14] W. Tan, J. Liu, T. Chen, and H. J. Marquez, “Comparison of some Ill-known PID tuning formulas,” *Computers and Chemical Engineering*, vol. 30, no. 9, pp. 1416–1423, Jul. 2006, doi: 10.1016/j.compchemeng.2006.04.001.

[15] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar, “A review of PID control, tuning methods and applications,” *International Journal of Dynamics and Control*. Springer, 2020. doi: 10.1007/s40435-020-00665-4.

[16] D. Whitley, “A Genetic Algorithm Tutorial.” 1994, Springer

[17] S. Shabir and R. Singla Professor, “A Comparative Study of Genetic Algorithm and the Particle Swarm Optimization,” 2016. [Online]. Available: http://www.irphouse.com International Journal of Electrical Engineering.ISSN 0974-2158 Volume 9, Number 2 (2016)

[18] D. S. Pereira and J. O. P. Pinto, “Genetic Algorithm based system identification and PID tuning for optimum adaptive control,” in *IEEE/ASME International*

Conference on Advanced Intelligent Mechatronics, AIM, 2005, vol. 1, pp. 801–806. doi: 10.1109/aim.2005.1511081.

[19] M. I. Solihin, L. F. Tack, and M. L. Kean, "Tuning of PID Controller Using Particle Swarm Optimization (PSO)," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 1, no. 4, p. 458, 2011, doi: 10.18517/ijaseit.1.4.93.

[20] C. L. Huang and J. F. Dun, "A distributed PSO-SVM hybrid system with feature selection and parameter optimization," *Applied Soft Computing Journal*, vol. 8, no. 4, pp. 1381–1391, Sep. 2008, doi: 10.1016/j.asoc.2007.10.007.

[21] C. Gonzalez, I. Alvarado, and D. M. la Peña, "Low cost two-wheels self-balancing robot for control education," in *IFAC-PapersOnLine*, Jul. 2017, vol. 50, no. 1, pp. 9174–9179. doi: 10.1016/j.ifacol.2017.08.1729.

[22] J. Dabbagh and I. H. Altas, "Nonlinear Two-Wheeled Self-Balancing Robot Control Using LQR and LQG Controllers."[2019 11th International Conference on Electrical and Electronics Engineering (ELECO)]

[23] S. Skogestad, "Probably the best simple PID tuning rules in the world."

[24] Q. G. Wang, T. H. Lee, H. W. Fung, Q. Bi, and Y. Zhang, "PID tuning for improved performance," *IEEE Transactions on Control Systems Technology*, vol. 7, no. 4, pp. 457–465, Jul. 1999, doi: 10.1109/87.772161.

[25] R. C. Panda, C.-C. Yu, and H.-P. Huang, "PID tuning rules for SOPDT systems: Review and some new results," 2004.

[26] O. A. Somefun, K. Akingbade, and F. Dahunsi, "The dilemma of PID tuning," *Annual Reviews in Control*, vol. 52. Elsevier Ltd, pp. 65–74, Jan. 01, 2021. doi: 10.1016/j.arcontrol.2021.05.002.

[27] H. A. Varol and Z. Bingul, "A new PID tuning technique using ant algorithm," in *Proceedings of the American Control Conference*, 2004, vol. 3, pp. 2154–2159. doi: 10.23919/acc.2004.1383780.

[28] J. Stender, "Introduction to Genetic Algorithms."

[29] K. De, "Learning with Genetic Algorithms: An Overview," 1988.

[30] M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," *Computer (Long Beach Calif)*, vol. 27, no. 6, pp. 17–26, 1994, doi: 10.1109/2.294849.

[31] M. A. S. Barbosa and M. M. Gouvêa, "Access point design with a genetic algorithm," in *Proceedings - 2012 6th International Conference on Genetic and Evolutionary Computing, ICGEC 2012*, 2012, pp. 119–123. doi:

10.1109/ICGEC.2012.39.

[32] M. Chen and Z. Yao, "Classification techniques of neural networks using improved genetic algorithms," in *Proceedings - 2nd International Conference on Genetic and Evolutionary Computing, WGEC 2008*, 2008, pp. 115–119. doi: 10.1109/WGEC.2008.23.

[33] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.

[34] R. C. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, 2001, vol. 1, pp. 81–86. doi: 10.1109/cec.2001.934374.

[35] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 1999, vol. 3, pp. 1945–1950. doi: 10.1109/CEC.1999.785511.

[36] C. L. Huang and J. F. Dun, "A distributed PSO-SVM hybrid system with feature selection and parameter optimization," *Applied Soft Computing Journal*, vol. 8, no. 4, pp. 1381–1391, Sep. 2008, doi: 10.1016/j.asoc.2007.10.007.

[37] M. I. Solihin, L. F. Tack, and M. L. Kean, "Tuning of PID Controller Using Particle Swarm Optimization (PSO)," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 1, no. 4, p. 458, 2011, doi: 10.18517/ijaseit.1.4.93.

[38] D. Masti, M. Zanon, and A. Bemporad, "Tuning LQR Controllers: A Sensitivity-Based Approach," *IEEE Control Systems Letters*, vol. 6, pp. 932–937, 2022, doi: 10.1109/LCSYS.2021.3087556.

[39] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar, "PID, LQR and LQR-PID on a quadcopter platform," 2013. doi: 10.1109/ICIEV.2013.6572698.

[40] Y. M. Sam, M. R. H. A. Ghani, and N. Ahmad, "LQR controller for active car suspension," in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2000, vol. 1. doi: 10.1109/tencon.2000.893707.

[41] H. Shousong and Z. Qixin, "Stochastic optimal control and analysis of stability of networked control systems with long delay," *Automatica*, vol. 39, no. 11, pp. 1877–1884, Nov. 2003, doi: 10.1016/S0005-1098(03)00196-1.

[42] E. Vinodh Kumar and J. Jerome, "Robust LQR controller design for stabilizing

and trajectory tracking of inverted pendulum," in *Procedia Engineering*, 2013, vol. 64, pp. 169–178. doi: 10.1016/j.proeng.2013.09.088.

[43] R. Khusainov[1], I. Afanasyev[1], L. Sabirova[1], and E. Magid[2], "Bipedal robot locomotion modelling with virtual height inverted pendulum and preview control approaches in Simulink environment."

[44] M. Jibril, A. Tadese, and M. Tadese, "Robust Control Theory Based Performance Investigation of an Inverted Pendulum System using Simulink," 2020. [Online]. Available: www.ijariie.com808

[45] S. J. Huang and C. lo Huang, "Control of an inverted pendulum using grey prediction model," *IEEE Transactions on Industry Applications*, vol. 36, no. 2, pp. 452–458, 2000, doi: 10.1109/28.833761.

[46] E. Vinodh Kumar and J. Jerome, "Robust LQR controller design for stabilizing and trajectory tracking of inverted pendulum," in *Procedia Engineering*, 2013, vol. 64, pp. 169–178. doi: 10.1016/j.proeng.2013.09.088.

[47] O. Boubaker, "The inverted pendulum benchmark in nonlinear control theory: A survey," *International Journal of Advanced Robotic Systems*, vol. 10. May 07, 2013. doi: 10.5772/55058.

[48] S. Irfan, A. Mehmood, M. T. Razzaq, and J. Iqbal, "Advanced sliding mode control techniques for Inverted Pendulum: Modelling and simulation," *Engineering Science and Technology, an International Journal*, vol. 21, no. 4, pp. 753–759, Aug. 2018, doi: 10.1016/j.jestch.2018.06.010.

[49] A. I. Roose, S. Yahya, and H. Al-Rizzo, "Fuzzy-logic control of an inverted pendulum on a cart," *Computers and Electrical Engineering*, vol. 61, pp. 31–47, Jul. 2017, doi: 10.1016/j.compeleceng.2017.05.016.

[50] J. J. Wang, "Simulation studies of inverted pendulum based on PID controllers," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 440–449, Jan. 2011, doi: 10.1016/j.simpat.2010.08.003.

RESUME

Name: -	SHIVAM
Date of Birth: -	26-12-1996
Qualification: -	B.Tech in Electrical Engineering from Government Engineering College Ajmer M.Tech in Control and Instrumentation from Delhi Technological University
Email: -	ss4274009@gmail.com

PAPER NAME

Intlligent Control Of Two Wheel Self Bala ncing Robot-.pdf

WORD COUNT	CHARACTER COUNT
8515 Words	**44631 Characters**
PAGE COUNT	FILE SIZE
48 Pages	**3.3MB**
SUBMISSION DATE	REPORT DATE
May 31, 2022 11:41 AM GMT+5:30	**May 31, 2022 11:42 AM GMT+5:30**

● **14% Overall Similarity**

The combined total of all matches, including overlapping sources, for each database.

- 7% Internet database
- 4% Publications database
- Crossref database
- Crossref Posted Content database
- 12% Submitted Works database