Delhi Technological University,
Shahbad Daulatpur, Bawana Road, Delhi -110042

A Project Report On

Simulation of Quadruped Robot

This project report is a compilation of the simulation of Quadruped Robot.

Mentor
Dr. Vikas Rastogi
Professor
Department of Mechanical Engineering, DTU

Submitted by:
Abhishek Kr Mishra 2k17/ME/11
Akash Chauhan 2k17/ME/24
Umakant Vashishtha 2k17/ME/244

Certificate

This is to certify that the project titled “Simulation of Quadruped Robot” submitted by Abhishek Kr. Mishra (ME/011),Umakant Vashishtha (ME/244) and Akash Chauhan (ME/024) in fulfilment for the B.Tech Project-1 (ME401) of Bachelor of Technology in Mechanical Engineering from Delhi Technological University (Formerly Delhi College of Engineering) is an authentic work carried out by them under my supervision and guidance.

Dr. Vikas Rastogi
Professor
Department of Mechanical Engineering
Delhi Technological University, Delhi

Acknowledgement

We would like to express our deep gratitude to our research mentor, Dr. Vikas Rastogi, Professor, Dept. of Mechanical Engineering, Delhi Technological University for giving us the opportunity to do this project and for the invaluable guidance throughout this project. His dynamic personality, clear vision, sincerity and motivation, all have inspired us a lot. It was a great privilege and honour to work and study under his guidance. We express our gratitude for all that he has offered us.

We extend special thanks to the Hon'ble Vice Chancellor Prof. Yogesh Singh, Delhi Technological University and Prof. Vipin, HoD Dept. of Mechanical Engineering, Delhi Technological University for providing us this platform to explore new avenues in life and carry out projects and researches. Our sincere thanks goes to all the people, researchers whose works and teachings have helped us sail through the project and complete the project work directly or indirectly.

Lastly, we praise and thank the God, the almighty, for showering his blessings and guiding us throughout this project and to complete this project and the project report lucratively.

Content

1.0.0 Introduction

1.1.0 Quadruped Robot

Quadruped robots can mimic animal walking gait and they have certain advantages like walking on terrain and extremely rough surfaces. Obstacles can impede the movement of wheeled vehicles, where a quadruped can adapt to avoid obstacles by adjusting its height.
Quadruped Robots have four legs or limbs and follow the gait patterns of quadruped animals. They are faster and more stable than biped robots. Depending on their leg structure, they can be broadly classified into two categories, Mammal-type and Sprawling-type.

1.2.0 Need of Quadruped Robots

The necessities of mobile robots for complex and dangerous environments have initiated the development of dynamic quadruped machines, which exploit the potential advantages of the legged locomotion and enhanced mobility in unstructured terrains. Such a versatile system with high maneuverability should be labor efficient, cost-effective, and indispensable in industries. As its name suggests, Quadruped Robots have four legs or limbs and follow the gait patterns of quadruped animals. They are faster and more stable than biped robots.

1.3.0 Recent Advancements In This Robotics Field

Mobile robots have an extensive area of applications in various fields like space exploration, military application, industrial use, and many more. Hence, the design and development of a mobile robot is a crucial part of the above application. Among all the mobile robots, the quadrupedal robot is a legged robot, which is superior to wheeled and tracked robots due to its potential to explore in all the terrain like the human and animal. In this paper, the survey concentrates on various design and development approaches for the quadrupedal robot, and environment perception techniques are discussed. Besides, Spot is one of the most advanced and intelligent quadrupedal robots.

2.0.0 Objective

The objective of this project was to analyse the different components of quadruped robot and to understand its locomotion and its variation based on kinematics analysis of quadruped robot in simulation.
We have simulated different walking behaviors of a simple quadruped robot in MatLab to study the complexity in its physical implementation. Our design is rather primitive for the initial step to reduce the unnecessary complexity.
The idea is to understand the various requirements and test the feasibility before we move to the physical model. We tried to create a model that is flexible enough to generate different walking patterns, although we have kept the degrees of freedom lesser than what a fully fledged quadruped robot would have.

3.0.0 Implementation Summary

We have used a simulation-based approach to understand the working of quadruped robots. We imported our CAD models of the robot in a simulink module named Simscape multibody. Simulink generated a block diagram for the shared assembly and then we added actuation and sensing features in the revolute joint block of the block diagram. Control the joint with the PID controller. This constitutes the mechanical model of the quadruped robot.

We carried out inverse kinematics calculation of the robot leg and the resulting calculations and equations are used to construct a simulink block named inverse kinematics. We replicated inverse kinematics blocks for all the four legs which had their own coordinate system.

After that we selected an elliptical trajectory that each leg should follow in vertical plane order to take a step and derived a parametric equation of this elliptical trajectory and then modified it in such a way that with increasing time the points that are fed to the inverse kinematics are progressive in nature and results in taking the step of the leg.

Once inverse kinematics and trajectory planning blocks are set up we can control the phase and speed of each leg in order to simulate any kind of gait that it should perform in a given situation.

4.0.0 Tools Used

4.1.0 SolidWorks

SolidWorks is a solid modeling computer-aided design and computer-aided engineering computer program. SolidWorks is published by Dassault Systèmes. It is a paid software that runs primarily on Microsoft Windows.
We used it to export our 3D model in xml format which is compatible with the Simscape import feature.

4.2.0 OnShape

Onshape is a computer-aided design (CAD) software system, delivered over the Internet via a Software as a Service (SaaS) model. It makes extensive use of cloud computing, with compute-intensive processing and rendering performed on Internet-based servers, and users are able to interact with the system via a web browser or the iOS and Android apps.
We used OnShape for building our 3D model

4.3.0 MatLab - SimuLink

SimuLink helps combine textual and graphical programming to design systems in a simulation environment. We can use SimuLink to design and simulate mechanical systems before moving to hardware. It gives us three main benefits:

- Model and Simulate Your System
- Test Early and Often
- Automatically Generate Code

We can import our 3D model exported from SolidWorks and generated in OnShape. And then we can define which joints we wish to control and sense.

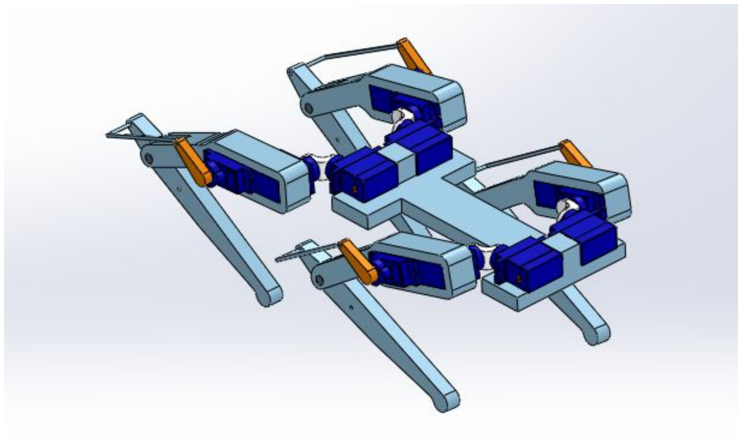
5.0.0 Methodology

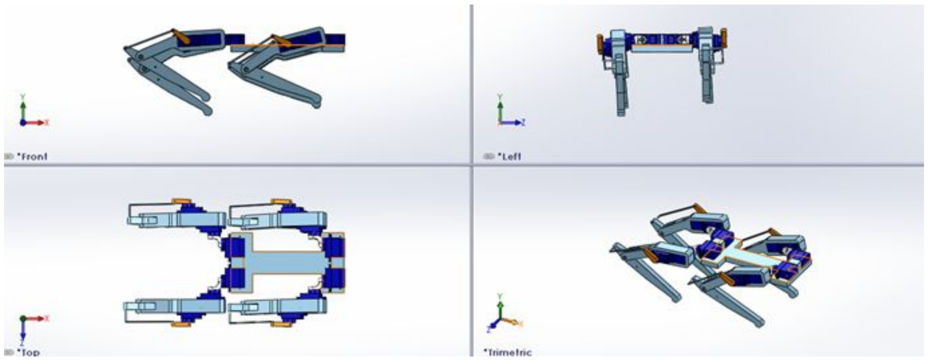
This Section will explain the overall methodology that we have used to create this project. It is divided into four major blocks as mentioned below

5.1.0 Mechanical Model

5.1.1 Computer Aided Design

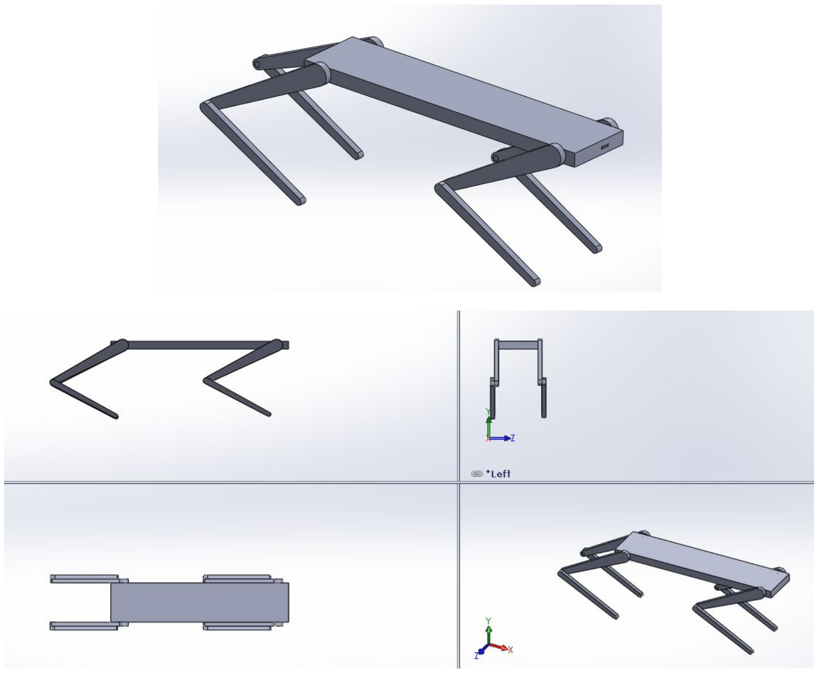
We created two designs using solidworks and onshape one of which was used for testing and simulation while the other was used for 3D printing and prototype creation of the robot. The design that we used for the simulation is a simplified version of the actual design which both the designs have the same parameters like link length and co-ordinate system.





Design to be used for prototype development

Since the minimum number of degrees of freedom required in a robot leg to perform normal walking is 2. In the simulation we have used only two degrees of freedom in one leg of the robot. For the actual Robot that will be used for prototyping we have designed 3 degrees of freedom which can enable it to perform additional motion like yawing, pitching and rolling along all the 3 axes.



Design used for simulation

5.1.2 Importing CAD in simulink

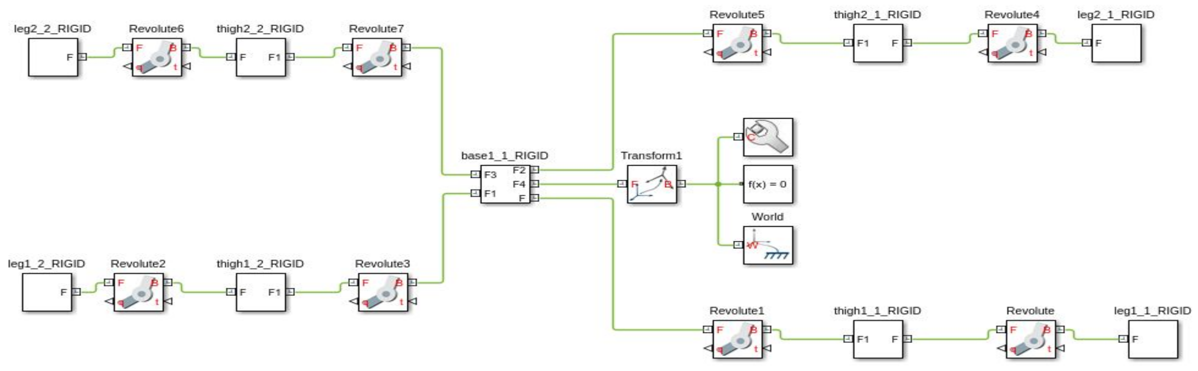
After the completion of CAD design we can import the model in simulink for simulation Import the model in simulink we need a plugin in solidworks called symlink multibody which generates the following files:

- Assembly.XML
  It contains all the information of the robot assembly and its mate relations and constraints. It also contains the dynamic properties of the CAD parts like weight, centre of mass and co-ordinate system information.
- STEP files
  Step files of each of the components in the Assembly which are used by the simscape module to generate simulation using those parts.

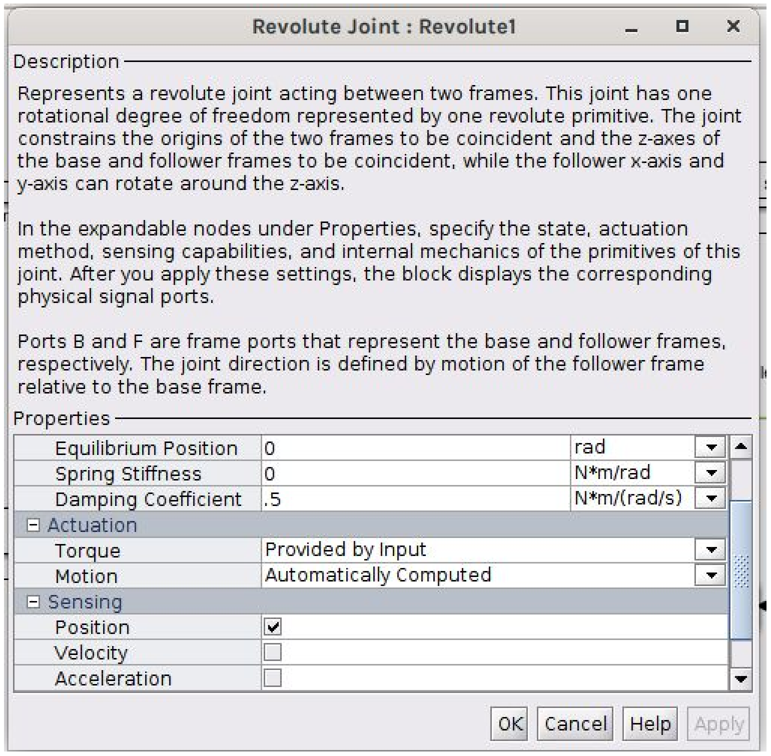
Now these files can be used to import the CAD assembly in Simscape multibody by executing the following command in the MatLab command window.

```
>> smimport("Assembly.XML");
```

This will generate a simulink file Which will contain a connected block representing the original CAD assembly in form of block diagram. This block diagram consists of links connected by joints in this case we had only revolute joints.



Now this revolute joint block can be modified and its parameter can be changed accordingly.
Following changes were carried out in the revolute block in order to make it behave like an actuator:



Revolute Joint : Revolute1

Description

Represents a revolute joint acting between two frames. This joint has one rotational degree of freedom represented by one revolute primitive. The joint constrains the origins of the two frames to be coincident and the z-axes of the base and follower frames to be coincident, while the follower x-axis and y-axis can rotate around the z-axis.

In the expandable nodes under Properties, specify the state, actuation method, sensing capabilities, and internal mechanics of the primitives of this joint. After you apply these settings, the block displays the corresponding physical signal ports.

Ports B and F are frame ports that represent the base and follower frames, respectively. The joint direction is defined by motion of the follower frame relative to the base frame.

Properties

Equilibrium Position	0	rad
Spring Stiffness	0	N*m/rad
Damping Coefficient	.5	N*m/(rad/s)
⊟ Actuation		
Torque	Provided by Input	
Motion	Automatically Computed	
⊟ Sensing		
Position	✓	
Velocity	☐	
Acceleration	☐	

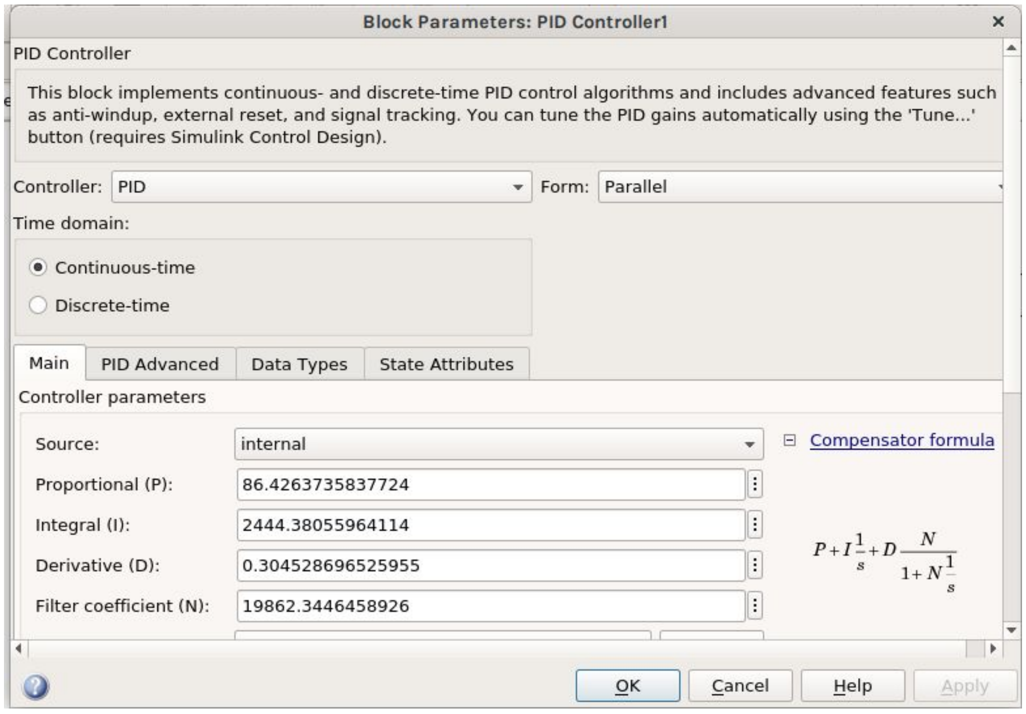
- In the Actuation option select torque provided by input and motion computed automatically.
- In sensing, check the position option because we want to know the position of the revolute joint after it has moved for the feedback to the close loop system.

5.1.3 Controlling the joints using PID

PID stands for proportional integral and derivative control which is a robust dynamic and versatile controller which is used in various reference tracking applications.
Now we can control the revolute blocks using PID controllers. Symlink has a PID block to control the system. This block takes input as the difference between the desired input and the observed feedback which in this case is the angle. The desired input angle will be given by the inverse kinematics and the feedback will be given by the sensor in the revolute block that was enabled in the previous section. The output of the PID block is Torque value which is used to control the revolute joint.
PID controller has three gains, proportional gain integral gain and derivative gain, these gains are to be changed according to the application The process of tweaking the gains is known as tuning the PID controller. There are various methods that are available to tune the PID controller but it is done by trial and error method more frequently.
In simulink there is a PID tuner app that tunes the PID values to optimise response of the system.

5.1.4 Tuning the PID gains using PID tuner app

The pic by opening the PID block and clicking “tune”. What PID tuner app does is linearize the system and returns the best possible gains given the required response speed and robustness.
Figure below shows the optimal gains that we have obtained after tuning the PID blocks.


Block Parameters: PID Controller1

PID Controller

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

Main | PID Advanced | Data Types | State Attributes

Controller parameters

Source:	internal
Proportional (P):	86.4263735837724
Integral (I):	2444.38055964114
Derivative (D):	0.304528696525955
Filter coefficient (N):	19862.3446458926

Compensator formula

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

5.2.0 Inverse Kinematics

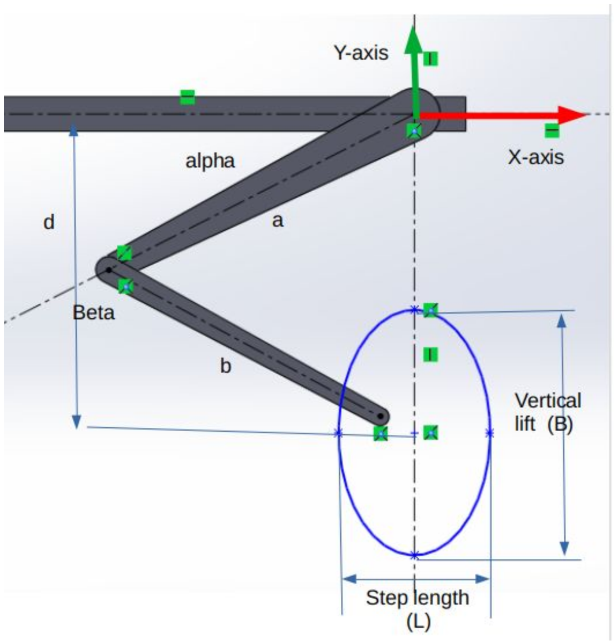
This section describes the methodologies and equations that we have obtained after performing inverse Kinematics of the robot leg.

5.2.1 What is inverse kinematics

Inverse kinematics is the process of computing links and joint parameters given the desired state in the space that a robot has to achieve.
In our case we had two degrees of freedom robot leg so we had two parameters alpha and beta as shown in the figure to compute given X and Y coordinates of the tip of the leg. We took a trigonometrical approach to carry out the inverse kinematics.

5.2.2 Coordinate systems

Each leg has its own coordinate system located at the centre of the 1st revolute joint which is at the thigh of the robot. The x-coordinate of this coordinate system is oriented along the forward of the robot as shown in the figure and the y-coordinate of this coordinate system is along the vertical direction.



5.2.3 Calculations and equations

After carrying out the calculation we got the following equations for the alpha and beta in terms of X and Y coordinate of the robot leg.
We added additional checks to make sure that the leg handles the input location that is impossible for it to reach. In that case the robot will try to reach the closest possible location along that point.

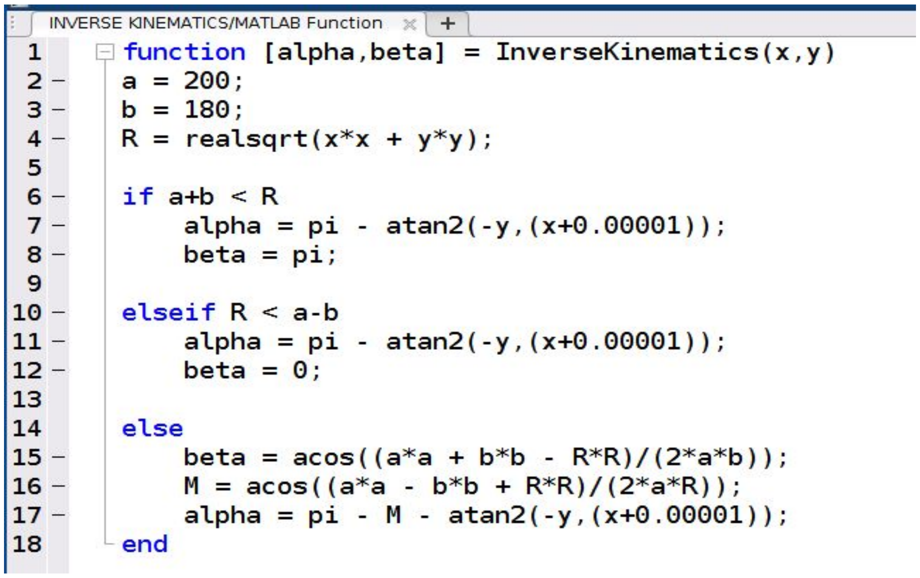
$$\beta = acos(\frac{a^2 + b^2 - R^2}{2ab})$$

$$M = acos(\frac{a^2 - b^2 + R^2}{2aR})$$

$$\alpha = \pi - M - atan(\frac{-y}{x})$$

5.2.4 Code of the inverse kinematics block

Figure below shows the inverse kinematics block definition in MatLab.


INVERSE KINEMATICS/MATLAB Function

```
function [alpha,beta] = InverseKinematics(x,y)
a = 200;
b = 180;
R = realsqrt(x*x + y*y);

if a+b < R
    alpha = pi - atan2(-y,(x+0.00001));
    beta = pi;

elseif R < a-b
    alpha = pi - atan2(-y,(x+0.00001));
    beta = 0;

else
    beta = acos((a*a + b*b - R*R)/(2*a*b));
    M = acos((a*a - b*b + R*R)/(2*a*R));
    alpha = pi - M - atan2(-y,(x+0.00001));
end
```

All the checks and conditions have been implemented in form of link length **a** and **b** and it is made sure that this inverse kinematics block does not fail even if the input is not in the scope of the robot leg.

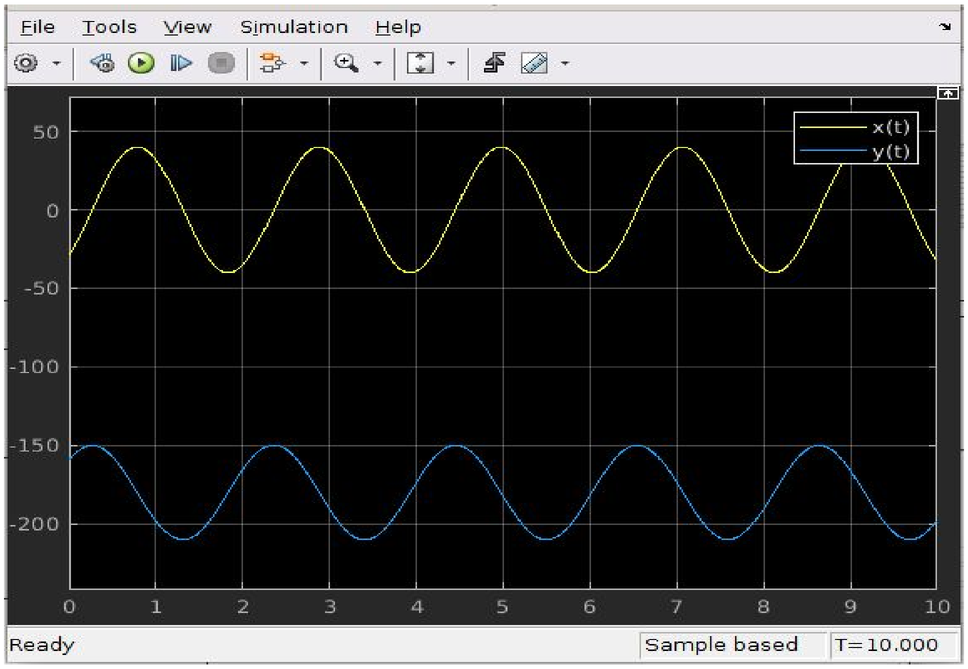
5.3.0 Trajectory Generation

Trajectory generation is the process of generating a trajectory that an articulated Joint Robot should follow in order to achieve some task. In our case the trajectory that we have chosen is an ellipse in the plane of robot leg motion which is in vertical plane. Now the variables of this ellipse are semi major axis and semi minor axis. These variables can be made a function of surface roughness and obstacles in the surface. In this way we can make it drive autonomously on slippery surfaces and surfaces with lots of pebbles and granules.

5.3.1 Working out the trajectory equation

The advantages that we have with elliptical trajectory is that it can be implemented as a single function and it is simple. It can also be easily modified depending upon the roughness of the surface it offers and is an easier solution with an extended application.

We use the parametric equation of the ellipse and transform it such that with increasing time it outputs X and Y coordinates of the ellipse which when followed by the robot leg produces a forward moving step. Equations below show the X and Y coordinate of the ellipse and its dependence on the time.

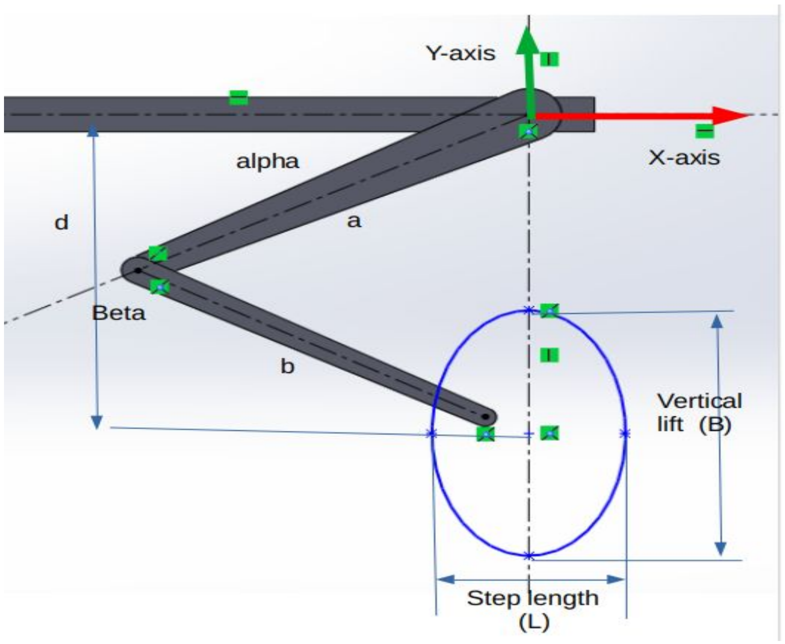


Elliptical parametric equation in terms of time t for x and y is:

$$X(\theta) = (L/2) \times cos(\theta + \pi)$$

$$Y(\theta) = -B \times sin(\theta + \pi) - d$$

Where L = Step length and B = step height and $\theta = \omega t + \phi$



5.3.2 Changing Phase and Velocity

In the parameter θ to the above mentioned equations is given as:

$$\theta = \omega \times t + \phi$$

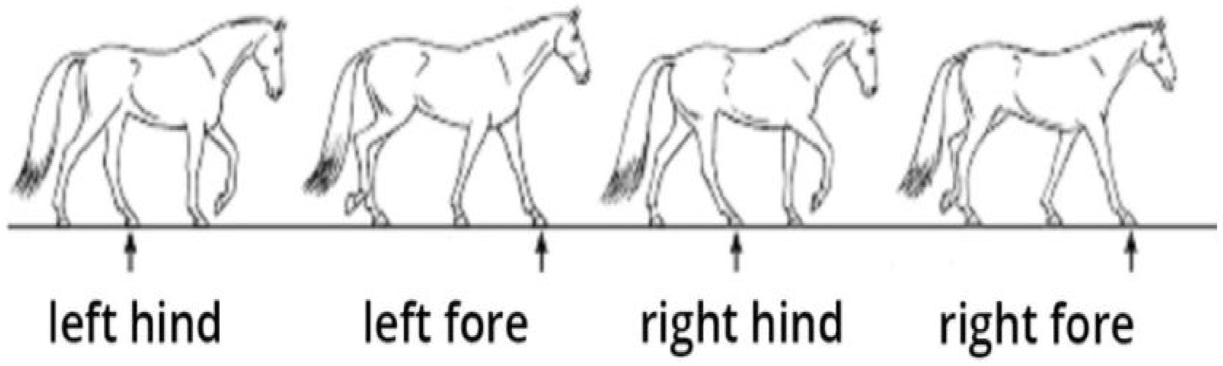
Where ω is the measure of how frequently trajectory planning blocks will give x,y commands to the inverse kinematics block.
When ω is higher the robot leg will move faster it will take one step quicker than the other legs.
ϕ is the phase difference between different legs. Which is used to tune the relative motion between different legs. For example if we want the front right leg and back left leg of the robot to move together in sync then the ϕ should be equal to zero. And if we want the front right leg to move first and then the back left leg should move in that case we can set the phase of the back left leg to π radians.

5.4.0 Simulating Different Gait

Different Gaits can be simulated by changing the ω and ϕ parameters (discussed in previous section) of each leg.

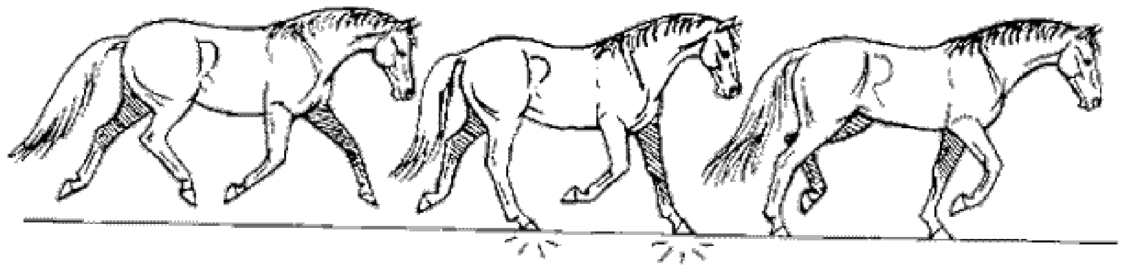
5.4.1 Static Walking

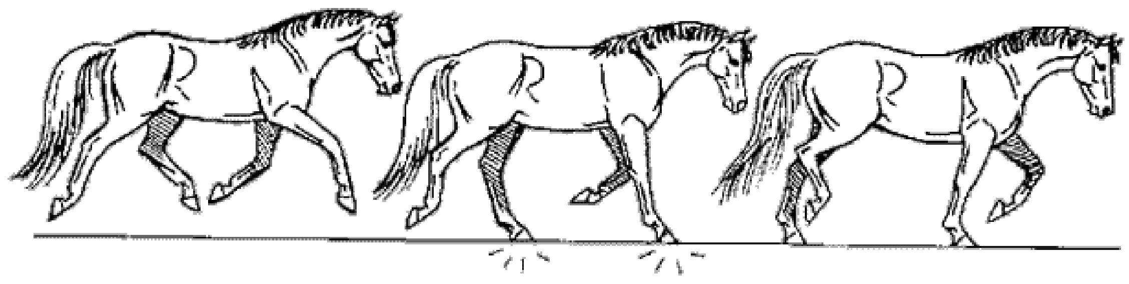
In static walking gait implementation the quadruped robot moves in such a way that at any instant at least 3 legs are on the ground. In this configuration robot moves one leg at a time and its centre of mass is located within the triangle formed by joining the tips of the 3 legs that are on the ground. So it shifts its weight in the triangle every time it takes a step.
Static walking gait is generally performed by four legged animals when they maneuver over hard to walk surfaces like ice, rocky mountains.



5.4.2 Trotting

Trotting gait is seen when a robot runs at a lower speed. In this at any instant less than 3 legs are on the ground. Generally in trotting gait implementation the diagonal legs of the robot move simultaneously. This comes under Dynamic gait because the motion is not stable if it is stopped. The center of mass of the robot is on the line joining the two diagonal legs.
To implement trotting the phase difference between diagonal legs is set to 0 radians and the phase difference between the adjacent legs should be π radians.


Suspension (in the air)
First beat: left front and right hind together (left diagonal)
Pushing off


Suspension (in the air)
Second beat: right front and left hind together (right diagonal)
Pushing off

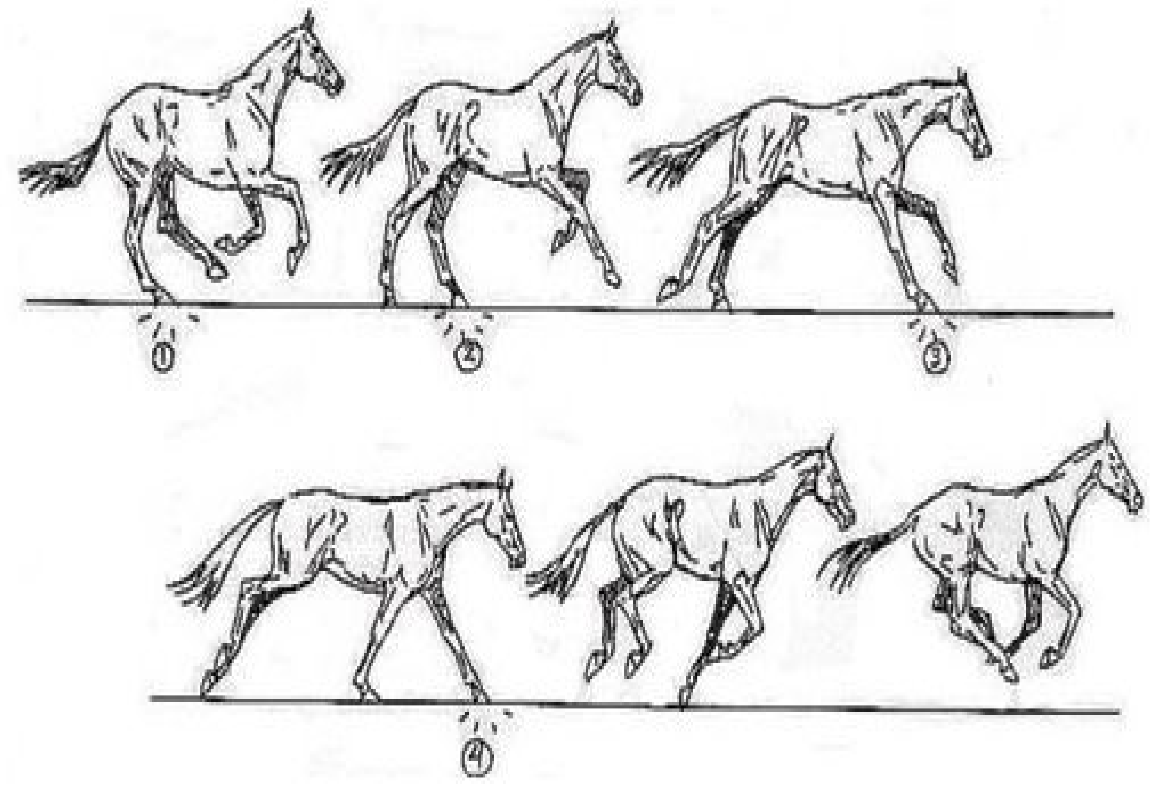
5.4.3 Galloping

In galloping cat implementation at any instant there are less than two legs on the ground. This makes it highly dynamic in nature and is most unstable. This is generally performed by quadruped animals when they are running very fast. We have simulated the most commonly observed galloping technique called rotary galloping. In rotary galloping the phase difference between all the four legs is $\pi/2$

radians. The order in which the legs take steps are circular in nature. For example the order which is being followed in the simulation is as follows:

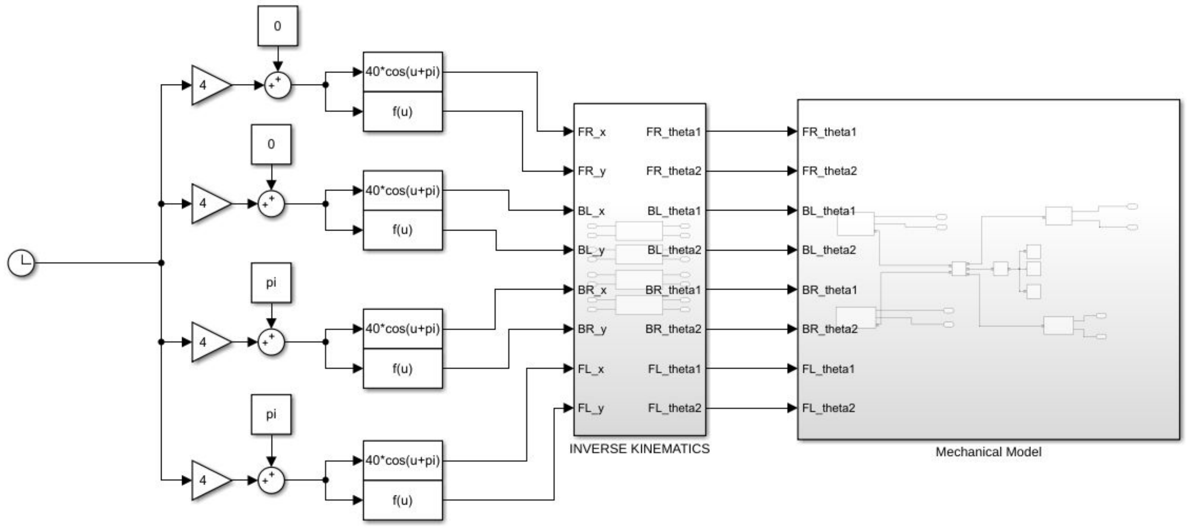
Front right leg → Front Left Leg → Back Left Leg → Back Right Leg → Front right Leg

The general gait pattern observed during galloping is shown below.



6.0.0 Result's

Following Results have been obtained by performing the simulation of the quadruped robot.



6.1.0 Walk

In static Walking configuration the following gait patterns were generated.

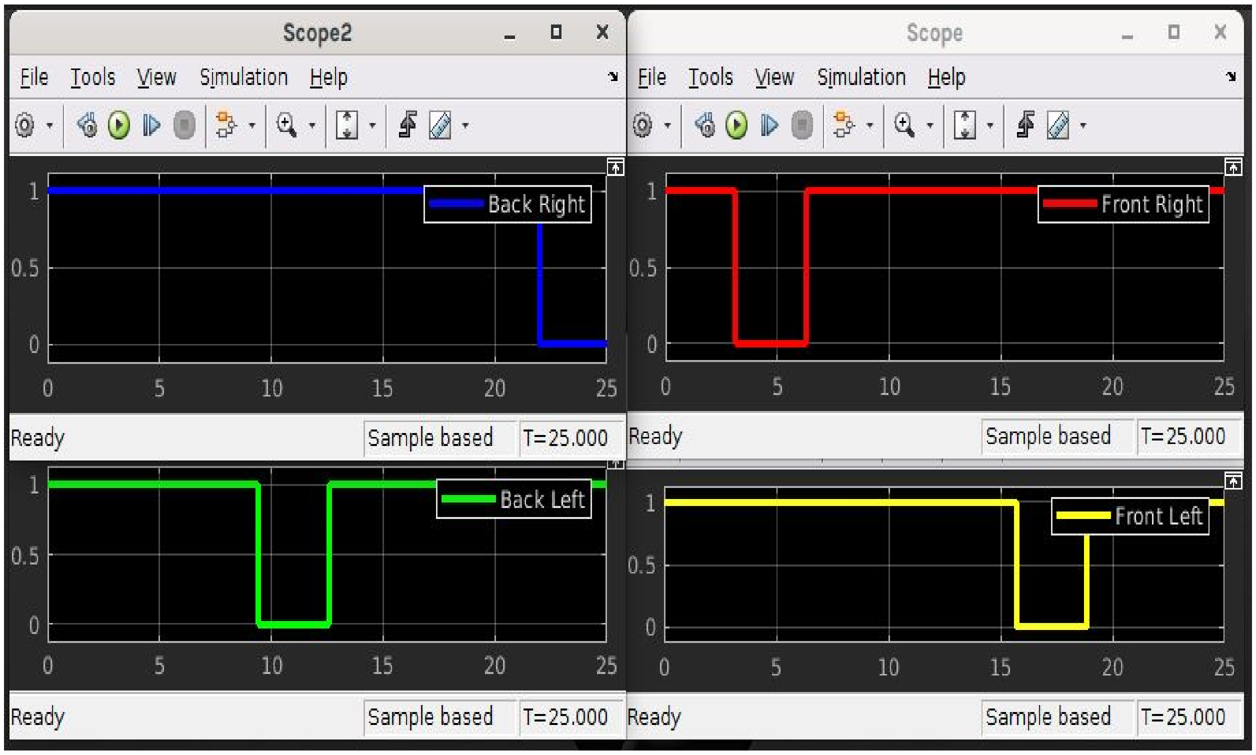
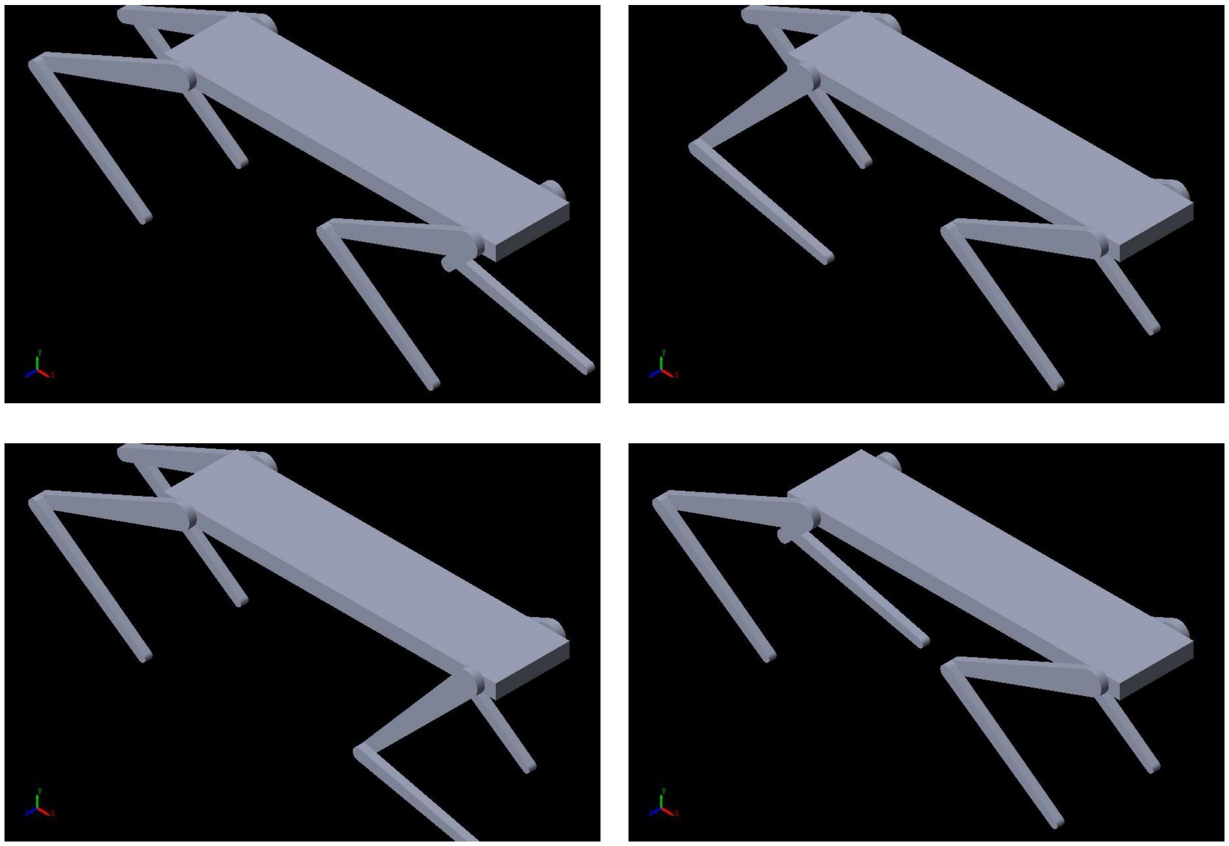


Figure above shows the gait patterns obtained in static walking configuration. The values along the x axis represent the simulation time. Value along y axis is either 1 or 0. If the leg is on the ground it is represented as 1 and 0 if it is not on the ground. It is easier to visualise if we consider that value is 1 if there is normal reaction from the ground and 0, if there is no normal reaction force acting on the leg.

In the figure we see that the left front leg (red) moves up first and then comes back to ground; this is represented as a dip in the graph. Then the right hind leg (green) takes a step followed by the right front leg (yellow) and at last the left hind leg (blue) takes a step. This whole cycle is repeated throughout the walking duration.

Frames below show the visual confirmation of the behaviour.

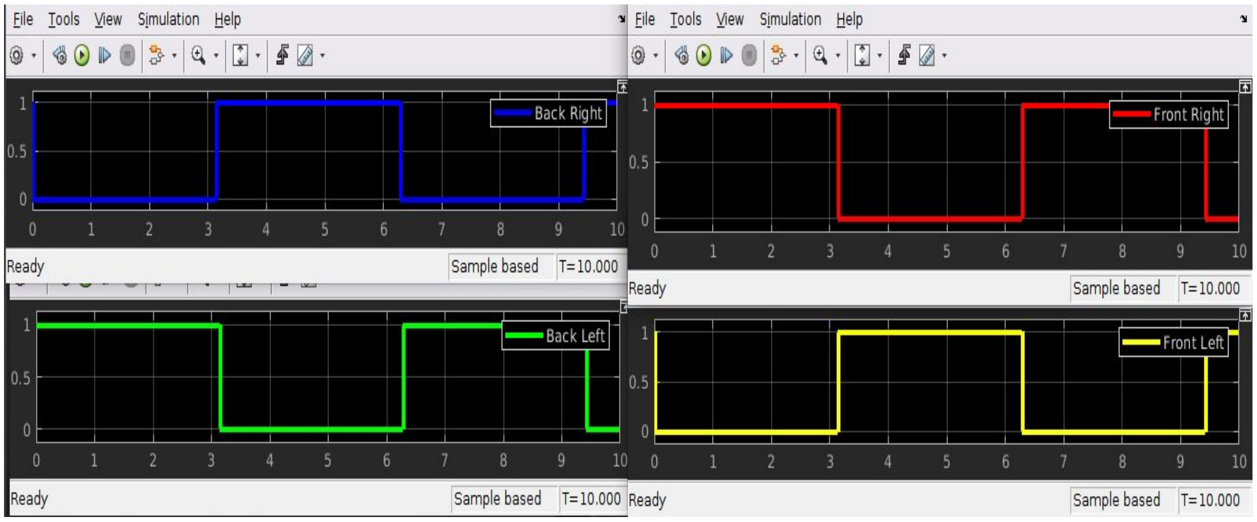


Leg moving order

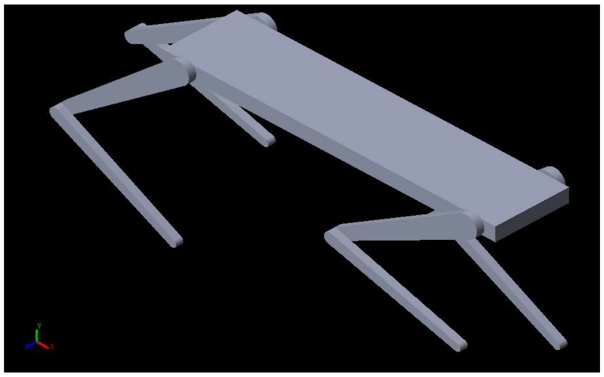
Left Front Leg → Right Hind Leg → Right Front Leg → Left Hind Leg → Left Front Leg (Left is inside the screen, right is coming out of the screen)

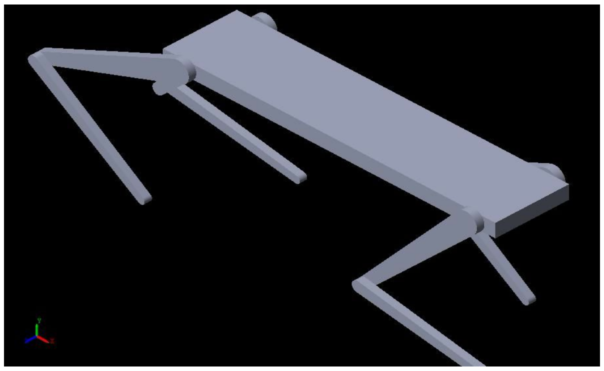
6.2.0 Trot

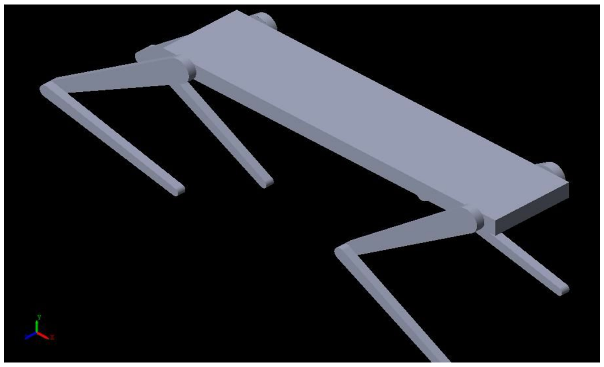
The following gait patterns were obtained in trotting configuration

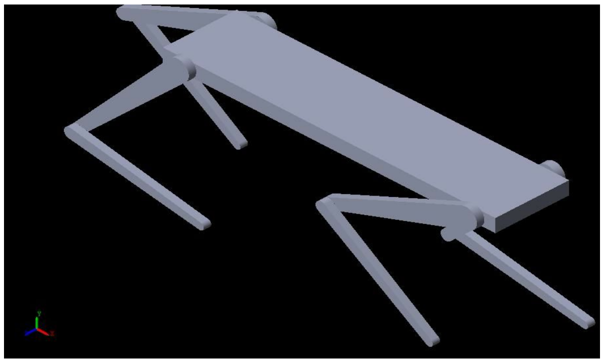


At time t = 0, front right leg and back left leg are on the ground and are in phase with each other while the other two legs are in the air and these two legs differ from the other diagonal legs in phase by $\pi/2$. Due to this they behave opposite to the legs on other diagonal. At t = 0, both legs (red and green markers) lift the legs simultaneously. While the legs on the other diagonal steps on the ground. Refer to frames below for clarity.

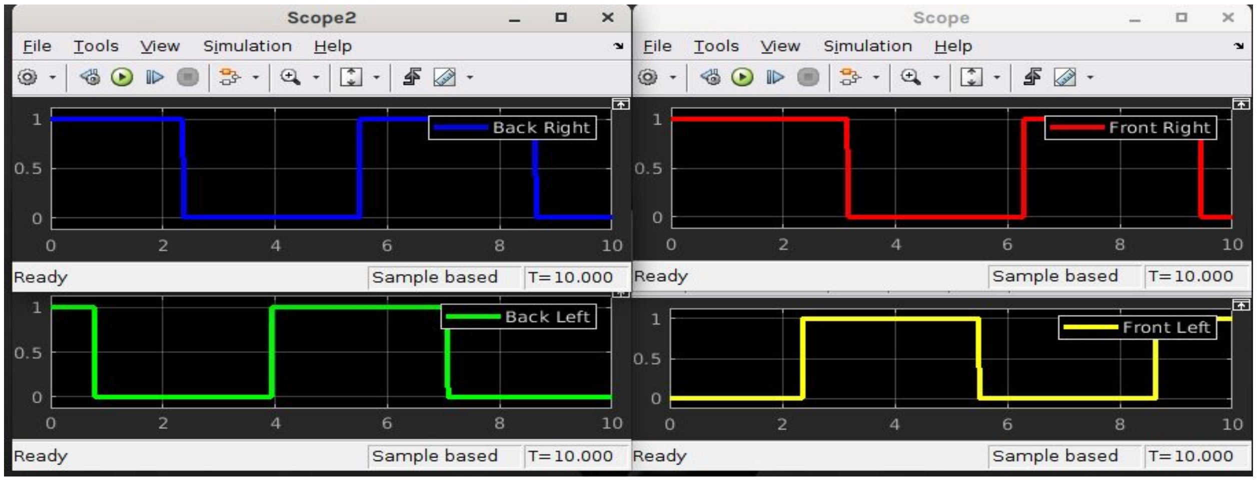








6.3.0 Gallop

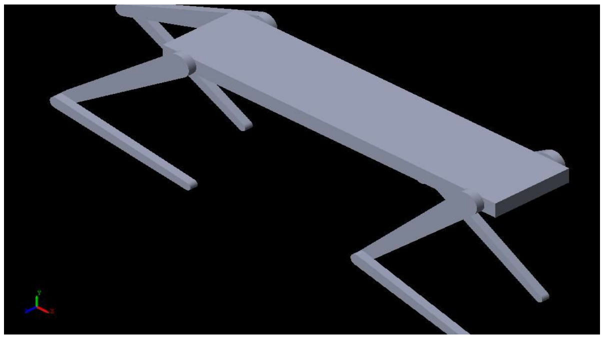


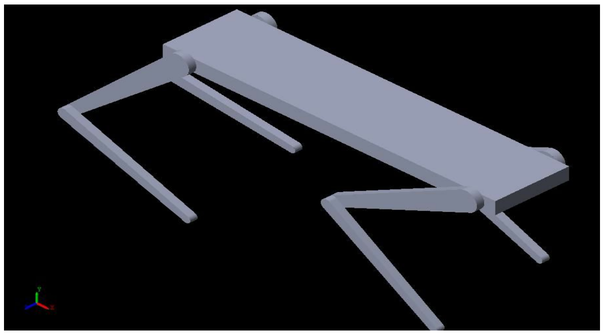
In this configuration, we implemented circular gallop which means that legs take steps in circular fashion as opposed to the diagonal fashion. Legs take steps in following order

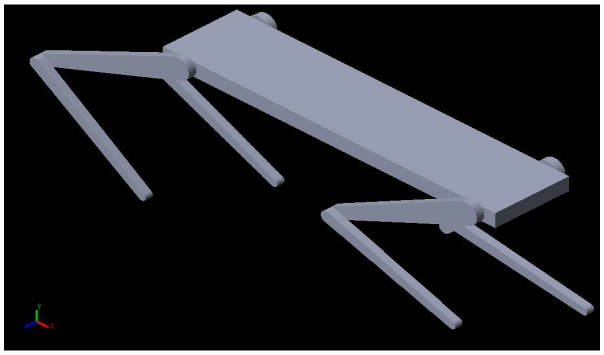
Left Hind Leg → Right Hind Leg → Right Front Leg → Left Front Leg → Left Hind Leg (Left is inside the screen, right is coming out of the screen)

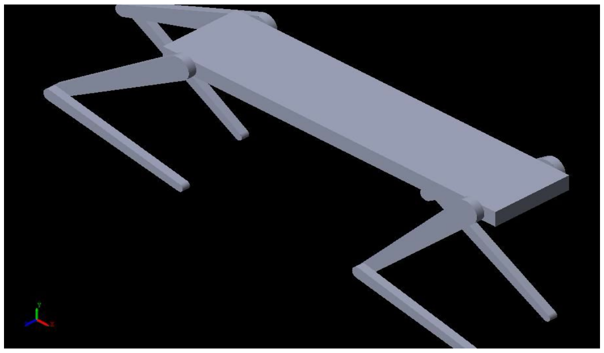
Initially the blue, red and green markers show that corresponding legs are on ground and Front left leg (yellow) is in the air.

In this order the phase difference between two legs is $\pi/2$. So at any instant less than 2 legs are on the ground. Figures below show the stepping order.









7.0.0 External Links and resources

Kindly visit the following links to the generated animation of the above mentioned motion configurations.

- Walking
  https://drive.google.com/file/d/1raHgYWAi5a1sCB3BcNhs8fvAesmnw2po/view?usp=sharing
- Trotting
  https://drive.google.com/file/d/1ZafKNBinkDyKhg6FRXCUUFvHNG9-Qxn7/view?usp=sharing
- Galloping
  https://drive.google.com/file/d/1z63L7GgCdYBF6zR7IB71WD_ffGjTzKIU/view?usp=sharing

8.0.0 Conclusion

This project was very educational for us, and required a lot of patience whenever we faced challenges. We understood a lot of factors that are required in consideration before we move on to the physical implementation.
We got the simulation to work perfectly. We analyzed different walking behaviours for the Quadruped Robots with two degrees of freedom. We also designed another model with higher degrees of freedom to extend the movements later on.
This project can find applications for many other people who are planning to build Quadruped Robots in analysing their models and sensing and troubleshooting.

9.0.0 References

https://en.wikipedia.org/wiki/Gait
https://grahameq.org/2020/06/5347/
Legged Locomotion
A simple rule for quadrupedal gait generation determined by leg loading feedback: a modeling study
Gait Analysis of Quadruped Robot Using the Equivalent Mechanism Concept Based on Metamorphosis