Simulation & Control of Quadrupedal Robot

Project By:

Bhavik Mukesh Panchal (bmp7963)

TABLE OF CONTENT

Introduction

Mobile Robotics has been evolving as one of the most promising domains in the field of Robotics. The ability of these robots to explore and maneuver in complex environments without human intervention attracts the attention of researchers across the globe. The mobile robots are classified into three different areas viz. **wheeled robots, tracked robots, and legged robots.** Robot locomotion system is an essential characteristic of mobile design, which depends not only on **working space** but also on technical measures like **maneuverability, controllability, terrain condition, efficiency, and stability.** Applications involving locomotion over rough terrains or disaster management where the robot is needed to access the remote areas within the debris demand the use of legged robots. Legged robots are further classified depending on the number of legs the robot has. Hence the types of legged robots are pogo-stick robots or one-legged robots, bi-pedal or two-legged robots, **quadrupedal or four-legged robots**, six-legged or hexapod robots, and eight-legged robots. Each of the types has unique applications and special locomotion mechanisms.

The GAIT behavior of the quadrupedal robots is inspired by the quadruped animals like horses, dogs, etc. This project is focused on the Simulation & Control of a Quadrupedal Robot, using trajectory generation for the locomotion and describing three types of GAIT behaviors for the robot, viz. **Walking, Trotting and Galloping**. These are based on the speed and leg-movement patterns of the robots. These behaviors can be transitioned depending on the application and the terrain pattern. All the mechanisms are designed and simulated in MATLAB, and Simulink.

Background and History

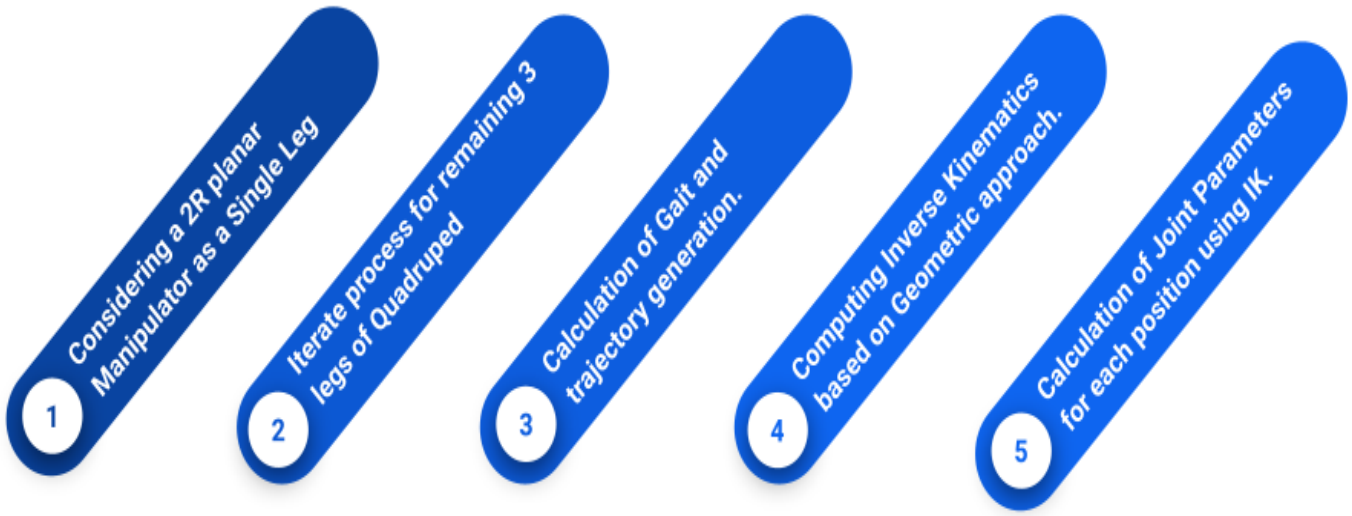
The aspects of mobile robots such as design, development, and planning of the motion are now of new interest in the field of robotics. The application field of areas such as space industry, military applications, commercial and scientific development. The quadrupedal-legged walking robot is more powerful, dynamic, and versatile than wheeled and tracked robots, as it can perform tasks that are relative to humans and animals. Walking robots are more likely to replace wheeled motion.

The main advantage of using legged robots for motion depends heavily on postures, leg number, and function of legs. As inferred, the robot which has wheels or belt tracks for motion is not operational in dangerous situations, rough surfaces, and also inappropriate backgrounds. The mechanism of the leg has easy access to almost every surface present on earth, which includes marshy, sandy, rough, high friction, sloping lands, as we can comfortably glide through everything, which was taken into consideration for demonstrating agility, smoothness, and reliability. The ability to change the motion and make it more efficient can also be achieved. Technically, the stability and efficiency gait, number of links(legs) are more suitable.

Different legs of a robot allow different gaits and their movement is useful in robot transporting objects. The analysis parts include walking, step length, cadence, symmetry, and joints. They play a key role in the research of biped robots as well.

Mechanism design, analysis, and optimization, as well as kinematic and dynamic simulation of legged walking robots, are the main concerns for making an efficient, robust, and reliable legged walking robot. The degrees of freedom and actuation system efficiency are also decided by legs moving motions. We have made the model of a four-legged robot and it is simulated in Matlab software, the parameters such as kinematic and dynamic are also taken and fed into the software for completion and resemble it to real life. We used Matlab since it is a widely used software package. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

Methodology



Step 1: Considering 2R planar Manipulator as a Single leg

The first move is to establish the limbs of the Walking Robot and for this, we are considering a 2R revolute planer manipulator to form a single leg. Each leg is connected with the main body i.e. **Torso** by a revolute joint. The angle between Torso and 1st link is called **Hip Joint** while the angle between the legs is called **Knee Joint**.

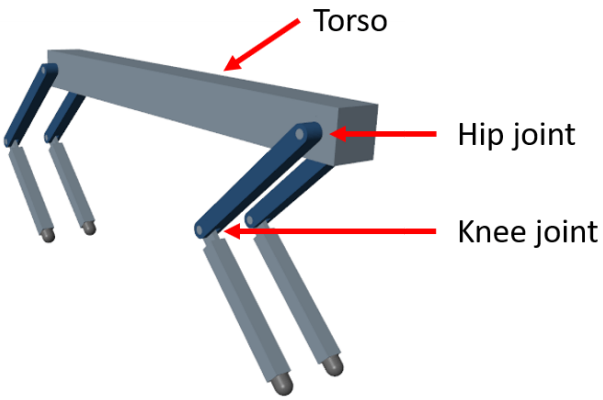


Fig. A quadrupedal robot 3D model

Step 2: Iterate process for remaining 3 legs of Quadruped

With the exact same dimensions and specifications, we multiplied and provided the robot with three more legs. After successfully establishing all the legs, a Torso is used to connect all of them in between and make it imply a rigid body. Each leg has its coordinate system located at the center of the first revolute joint. The center of the same joint is located at the thigh of the robot. We have considered the positive X-axis of the robot in the forward direction.

Step 3: Calculation of Gait & Trajectory generation

To perform the locomotion of the robot we need to find the trajectory of each leg with respect to the others so it will perform forward and backward motion. This trajectory can be computed by Gait analysis of quadrupedal animals. For our reference, we are considering a horse gait for walking, throttling, and galloping action. Once we got Gait for the robot then we created an elliptical trajectory for each motion and each leg. After calculating the trajectory for each leg we will get the start point, endpoint, and multiple via points for the 2R manipulator leg.

Step 4: Computing Inverse Kinematics based on Geometric approach

Once we got all points i.e. start, end, and via points, now it is time to find joint angle values for each position. To find joint values we use the DH parameter method with forward and inverse

kinematics. Here we converted points from cartesian space to joint space. At each instant we will get joint angles for each robot leg.

Step 5: Calculation of Joint Parameters for each position using IK

The most important part of simulation and animation is carried out via Simscape by feeding the output of Inverse Kinematics generated by trajectory generation. For the smoothness in the trajectory, the constant acceleration is used where each and every via point is covered in the path with a constant time duration at each point. The position parameters are fed as input to Inverse Kinematics which is calculated for a single leg for the initial simulation of the model. The process is the same and is repeated for the other legs involved in the workspace and is thus animated. In the last step, on tuning the hyperparameters of the simscape model we can attain different gait manipulation like walking, galloping, trotting.

Implementation Summary

We used a **simulation-based approach** to understand the working of quadruped robots. The first step implemented was to import the CAD model of the robot in a Simulink module named **SimScape Multibody**. Once the CAD file is imported, integration of actuation and sensing features in the revolute joint block was done. Controlled the joint with PID Controller. This overall feature constitutes the mechanical model of a quadruped robot.

We carried out an inverse kinematics calculation of the robot leg and the resulting calculation and equation were used by having a geometric approach to construct a Simulink block named Inverse kinematics blocks for all four legs which had their own coordinate system.

Now coming to the 3rd step in Methodology that is trajectory generation. We selected an elliptical trajectory that each leg should follow in the vertical plane in order to take steps and derived a parametric equation to compute the inverse kinematics using the X and Y position value of the elliptical trajectory. In order to define the gait system these via points (X and Y) are fed into the inverse kinematics block progressively as a function of time which results in taking the step of the leg.

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

Tools Used

1. **SolidWorks**

   SolidWorks is a modeling computer-aided design and computer-aided engineering software. We used it to create the solid model of the quadruped robot and used it to export our 3D model to Simulink blocks

2. **Matlab and Simulink**

   Simulink and Matlab programming were used to define modal properties and set up the simulation environment for our quadrupedal robot. The main benefits of using Matlab and Simulink were:

   - Better Integration of the system

- Easy to Visualize
- Automatically generate code

We can also define the suitable sensing method required in order to simulate our robot with a proper feedback loop.

Simulink and Working

1. Importing Cad in Simulink

After completion of CAD design in SolidWorks, we can directly import our 3D model in Simulink as STEP files.

- **STEP Files:** Contains 3D geometry and features 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

>> Assem4_DataFile.m

Running this command will generate and set all the modal parameters required to set up our simulation environment.

Now, this revolute joint block can be modified and its parameter can be changed in order to have position sensing and carry out the PID Tuning. Following changes were carried out in the revolute block in order to make it behave like an actuator. In sense, check the position option because we want to control the position of our quadrupedal robot using a PID closed-loop control system.

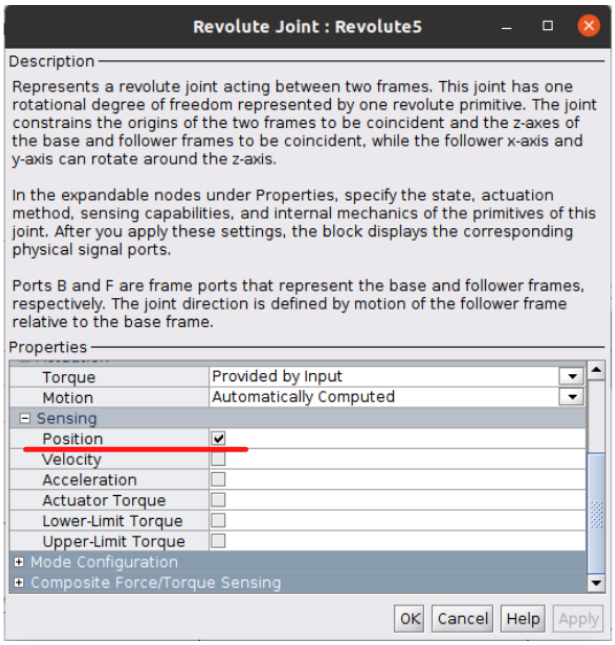


Fig. Revolute Joint Property window

2. Controlling the joints using PID controller

PID stands for **proportional integral and derivative** control which is a robust dynamic. and a 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 the 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 optimize the response of the system.

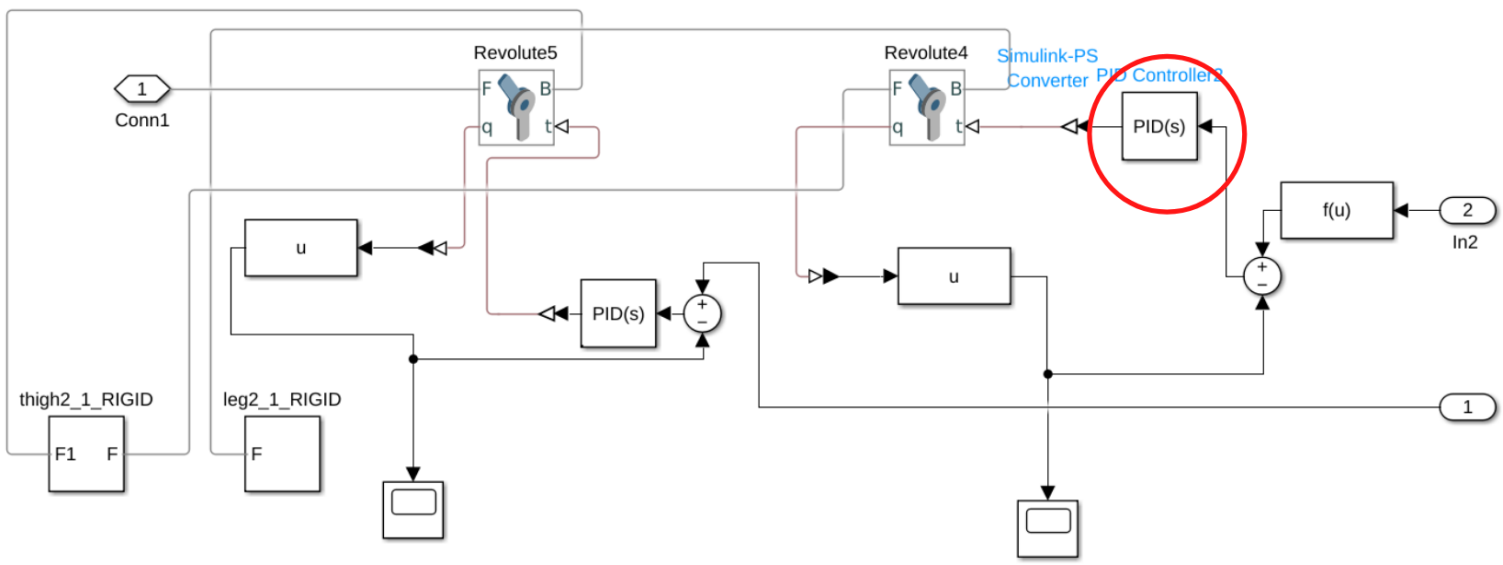


Fig. **Front Right** Block with PID Controller

The PID controller was tuned to have a proper closed-loop feedback system to compensate for any error in the system.

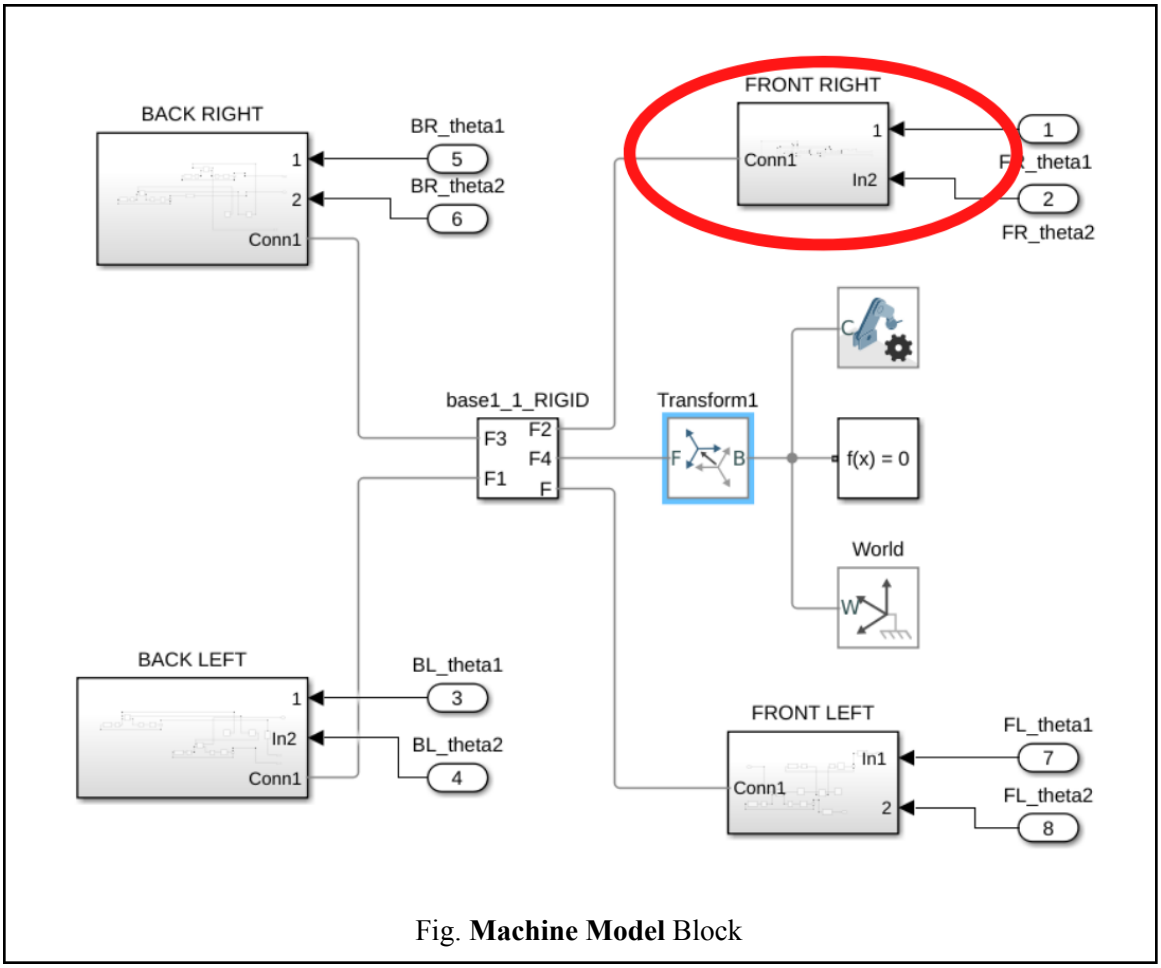


Fig. **Machine Model** Block

3. Inverse Kinematics

This section describes the methodologies and equations that were obtained after performing inverse kinematics of the robot leg.

In this case, the quadrupedal had two degrees of freedom in the robot leg so it had two parameters **alpha** and **beta** as shown in the figure to compute given **X** and **Y** coordinates of the tip of the leg. The trigonometric approach is used to carry out inverse kinematics.

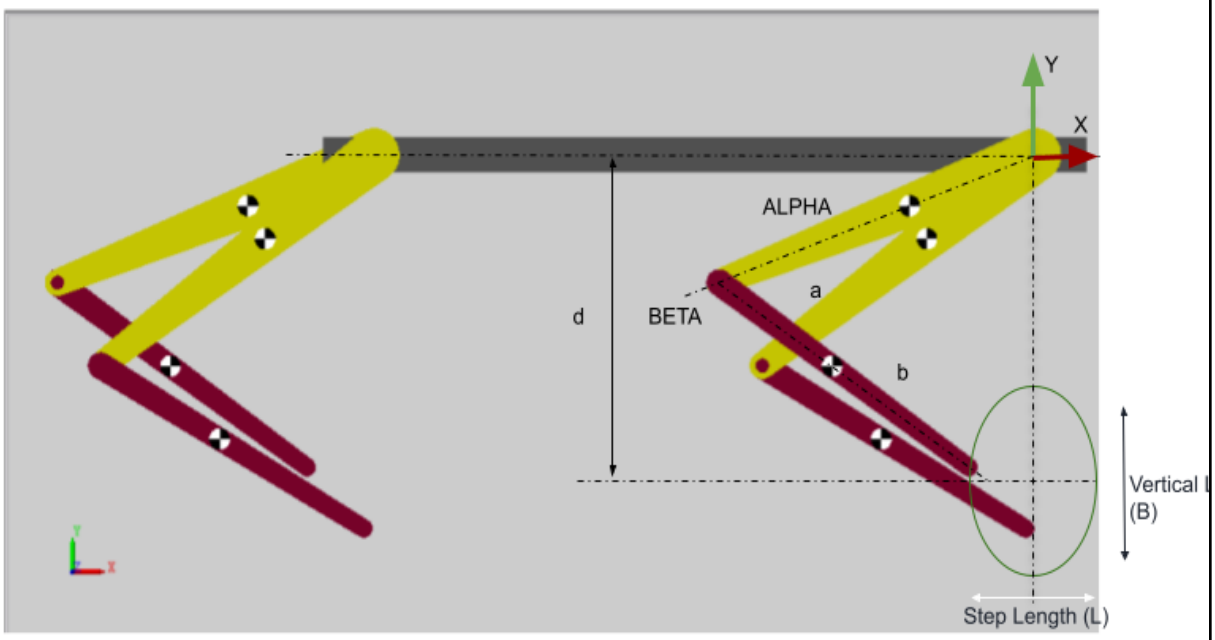


Fig. Model of Quadrupedal Robot with parameters

Coordinate System

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

Calculation and Equation

After carrying out the calculation we got the following equation for the alpha and beta in terms of the X and y coordinate of the robot leg.

$$beta = acos((a*a + b*b - R*R)/(2*a*b));$$

$$M = acos((a*a - b*b + R*R)/(2*a*R));$$

$$alpha = pi - M - atan2(-y,(x+0.00001));$$

All the possibilities have been checked in order to avoid failure of the inverse kinematics block 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.

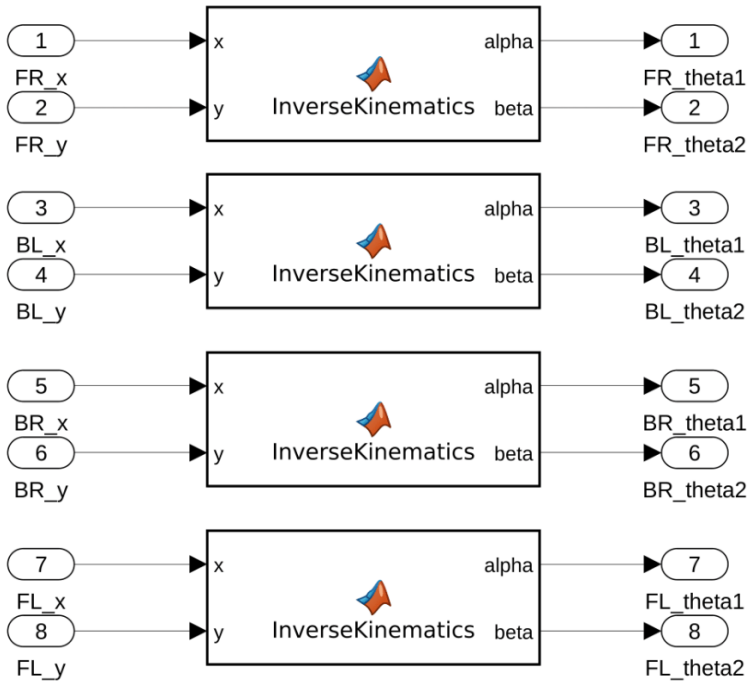


Fig. Inverse Kinematics Block

```
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
```

Fig. InverseKinematics Function

4. Trajectory Generation

Now coming to the third step in the methodology. Trajectory generation is the process of generating a trajectory that an articulated joint robot should follow in order to achieve some task. In the case of quadrupedal, the trajectory is an ellipse because it can be implemented as a function of time and it is simple.

Generating Trajectory Equation

The parametric equation of the ellipse is used and transformed 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. The equations below show the X and Y coordinate of the ellipse and its dependence on time.

Elliptical parametric equation in terms of time t for X and Y is:

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

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

$$L = \text{Step length and } B = \text{step height and } \theta = \omega t + \phi$$

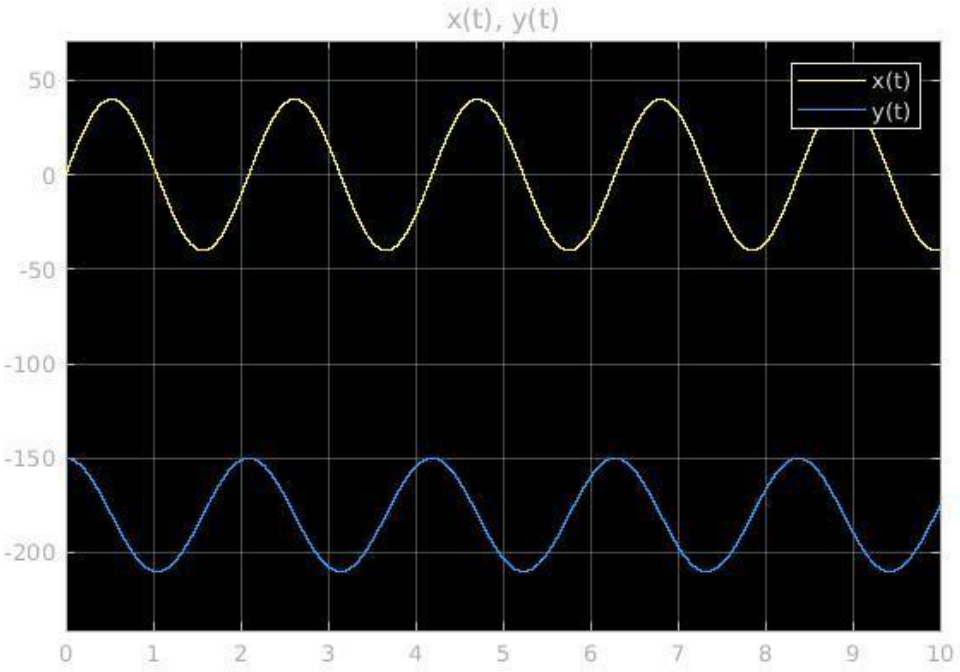


Fig. Trajectory Generation for X and Y positions

Changing the phase and velocity

The parameter θ to the above-mentioned equations is given as:

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

ω = 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 = 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.

Simulating Different gait

Different Gaits can be simulated by changing the ω and ϕ parameters of each leg. Gait can be simulated are :

- Static

- Trotting
- Galloping

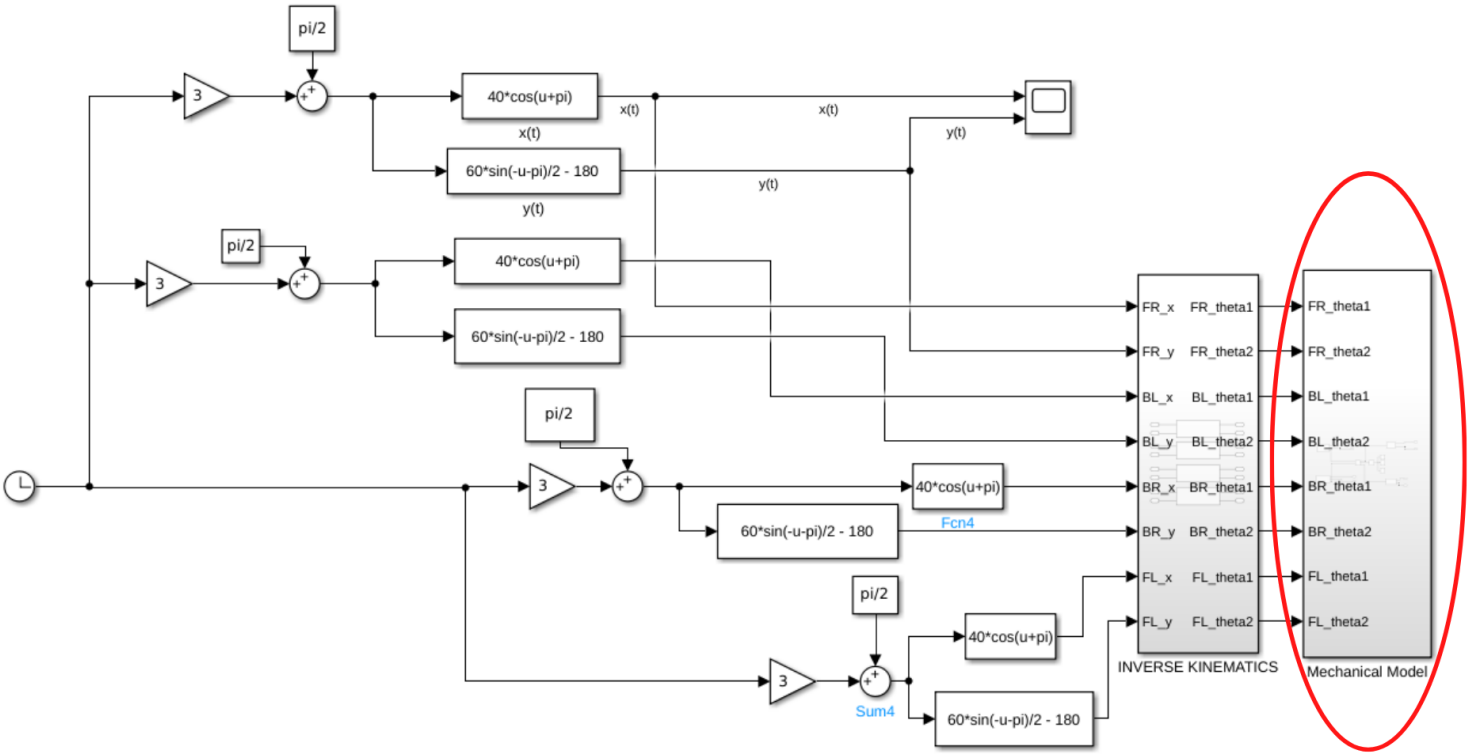


Fig. Simulink Model for Quadrupedal Robot

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, the robot moves one leg at a time and its center 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.

https://youtu.be/I4A3sWSK03k

Trotting

A trotting gait is seen when a robot runs at a slower 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.

https://youtu.be/d4mzMAfDF9s

Galloping

In galloping at 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 π /2radians. The order in which the legs take steps is circular in nature.

In terms of legs position the order in the simulation is as follows:

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

https://youtu.be/AcMc5FlZrD8

Conclusion

- Despite years of development, there is still a huge gap between the behavior of quadrupedal robots and four-footed animals. However, these robots have proved to be helpful while performing practical applications like mine inspection, space exploration, fire-fighting, or where navigation is required in an unconstrained environment.
- The selection of the quadrupeds is based on the type of the synchronized pattern of leg movements i.e. walking, trotting, galloping, etc, according to the speed.
- The speed for galloping is higher as compared to walking and throttling. The comparison in terms of speed and energy consumption for each motion is represented in the following figure.

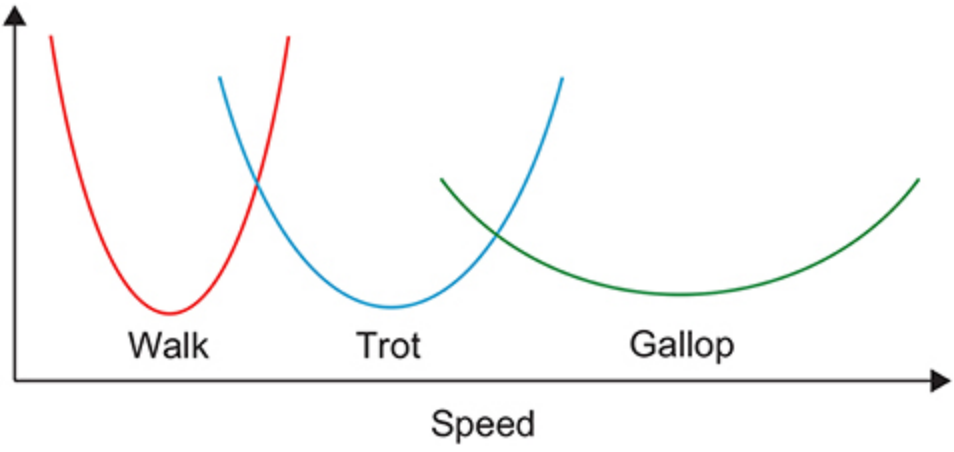


Fig. Speed Vs Energy Consumption for Walking, Throttling and Galloping

- A quadrupedal robot is simulated in MATLAB and Simulink by embedding the CAD design of the robot. A PID control system is implemented to make the robot leg follow the elliptical trajectory defined by the equation.

- The transition between different GAITs is possible between all these three modes in a single simulation by changing the phase angle of the trajectory equation. Locomotion for different quadrupedal robots can be implemented using a similar algorithm and trajectory equation.

Future Developments

The leg-movement pattern of the quadrupedal robot in this project is based on the equation of the trajectory. This makes the robot leg follow a certain movement pattern without sensing the terrain. Different types of terrains demand a different leg movement, which is expected for the robot to adapt to. Hence, Reinforcement learning can be implemented in order to make robots generate the trajectories by sensing the terrain data and then generating the trajectory for the locomotion. Also, force sensors can be attached to the joints in order to compute the dynamic forces acting on the joints. Later on, force damping systems can be implemented in order to preserve the joints and the other parts of the robot from damages. Values beyond the threshold can be damped and smooth trajectories can be generated to stabilize the movement.

References

1. Biswal, P. and Mohanty, P.K., 2021. Development of quadruped walking robots: A review. Ain Shams Engineering Journal, 12(2), pp.2017-2031.
2. Bo, Z., Minxiu, K. and Weidong, W., Gait Planning and Motion Simulation of a Biped Walking Robot. In 2008 IEEE Conference on Robotics, Automation, and Mechatronics.
3. Curran, A., Colpritt, K., Sullivan, M., and Moffat, S.M., 2018. Humanoid walking robot.

4. Liang, C., Ceccarelli, M. and Carbone, G., 2011. Design and simulation of legged walking robots in the MATLAB environment. MATLAB for Engineers-Applications in Control, Electrical Engineering, IT, and Robotics.