Design & Simulation of 6DOF & 3DOF Stewart Platform using Linear Actuators

Author Monojit Layek[1]

*IIT Kanpur, Robotics & Automation, Kanpur, India
*e-mail: mlayek@icloud.com

1 Overview

The Stewart Platform is a sophisticated type of parallel manipulator that boasts an impressive six degrees of freedom, allowing for a wide range of precise movements. This innovative platform is constructed of a rigid base, a remarkably agile moving platform, and six legs that serve as a connection between the two. By employing linear actuators in each of the legs, the platform can move seamlessly in three translational and three rotational directions.

1.1 CAD Model Details

- **Base:** The base of the model is made of aluminium material.
- **Platform:** The platform of the model is also made of aluminium material.
- **Cylinder Material:** The cylinder material is made of aluminium.
- **Universal Joint Pin:** The universal joint pin is made of steel.
- **Piston Rod:** The piston rod is made of titanium.
- **Model Weight:** The total weight of the model is 36.5 kg.
- **Base and Platform Radius:** The base and platform have a radius of 200 mm.
- **Base to Platform Height:** The distance between the base and platform is 257.5 mm.

Figure 1. 3D Fusion Model of Stewart Platform

1.2 URDF Details

- This repository includes a URDF file that describes the Stewart Platform model. The URDF file contains information on the robot's links, joints, and sensors, as well as its visualization and collision properties. However, to create the URDF of the parallel mechanism, we first need to convert it into an open chain and then compile it as a URDF.

- Once the URDF file is created, the model can be assembled in a physics simulator by adding constraints. The simulator allows us to simulate the motion and behaviour of the platform in real-world physics.

- In the case of the Stewart Platform model, it is necessary to ensure that all the joints are properly connected to form a parallel chain. The image below shows an example of the joint connections in the Stewart Platform model.

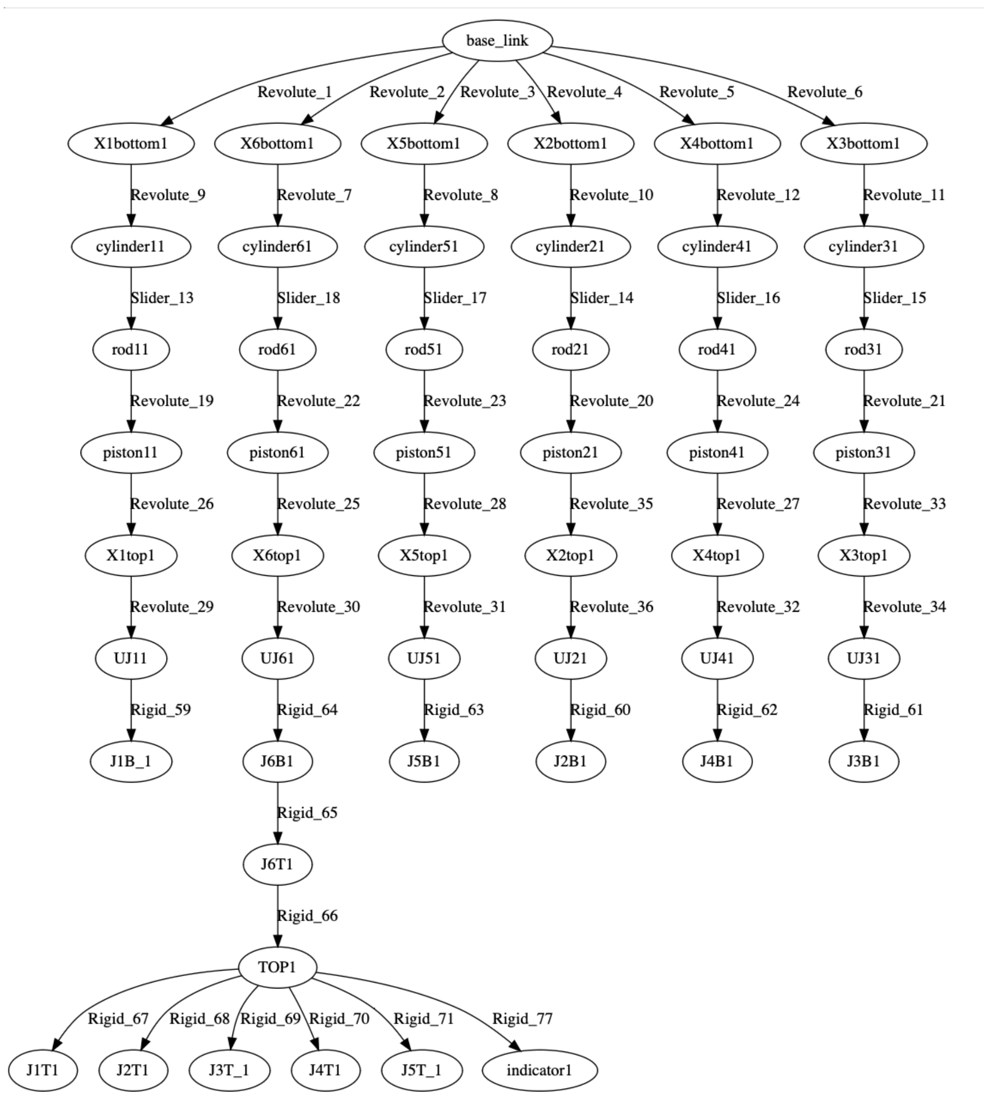

Figure 2. Openchain URDF Joint Connections

2 Inverse Kinematics of Stewart Platform

The inverse kinematics of a Stewart Platform (1) is the process of determining the joint angles required to position the platform in a specific orientation. Since the platform has six degrees of freedom, six equations are required to determine the joint angles. The inverse kinematics of the Stewart Platform can be solved using geometric, analytical, or numerical methods. The solution to the inverse kinematics problem is important for precise control of the platform, which is essential in applications such as flight simulators, motion platforms, and virtual reality systems. However, for certain applications that require the platform to have a reduced degree of freedom, such as RPR 3dof (2), we can restrict the translational motion of the platform. This simplifies the inverse kinematics problem and allows for precise control of the platform with fewer degrees of freedom.

2.1 Base and Platform Anchors

Standard notation (3) for the fundamental parameters that determine the mechanical configuration is

- $r_B \rightarrow$ Radius of Base (Bottom)
- $r_P \rightarrow$ Radius of Platform (Top)
- $\gamma_B \rightarrow$ Half of the angle between two anchors on the base
- $\gamma_P \rightarrow$ Half of the angle between two anchors on the platform

We may define $\psi_B \in R^{6\times 1}$ & $\psi_P \in R^{6\times 1}$ and the polar coordinates of the anchors on a unit circle radius using these γ_B & γ_P. These are derived from the gamma values of B and P.

If we have r_B and r_P, then we may define as the coordinates of the anchors in their respective local frames in Cartesian space, which are $B \in R^{6\times 3}$ and $P \in R^{6\times 3}$. For instance, an illustration of the anchor points on base B may be found below.

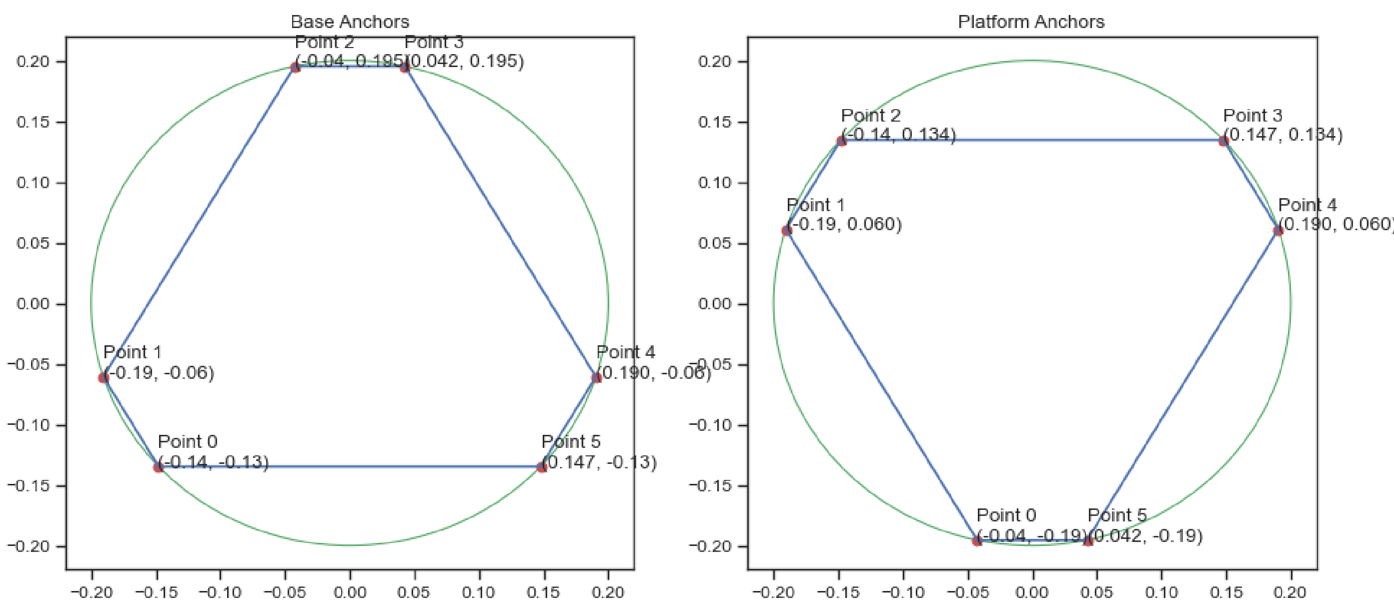

Figure 3. Geometry of the Stewart Platform: base and platform attachment points

2.2 Positioning Oneself at Home

The gap between the base and the platform at the starting point must then be specified. Your resting linear actuator length is. Let's say it's the base plate radius. Using the usual notation, we must additionally define the rotation matrices.

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

2.3 Determine Linear Actuation from Inverse Kinematics

- We may now begin working on the inverse kinematics problem, using the target translation vectors $T = (t_x, t_y, t_z)^T$ and the rotation vector $\theta = (\theta_x, \theta_y, \theta_z)^T$, determine the required leg length(4).
- After the plate has been rotated and translated as desired, all that remains is to determine the new locations of the various anchors.
- Given that each leg's job is to establish a connection between the base and the platform's anchor (5), the required vector (direction and length) for each leg is simply the leg's location in 3D space with respect to its corresponding base anchor (6).

  $$l = T + H + R(\theta)^T \times P - B$$

  Where, T and H are in $R^{3\times1}$ replicated 6 times to have dimensions $R^{3\times6}$ to facilitate matrix calculations. It's possible to interpret this as,

  $$l_{Actuated} = T_{Desired\ Translation} + H_{displacement(Base\ Center, Home\ Pos)} + R_{Coordinate\ Rotation(global frame)}$$

  A leg's length is simply the leg vector's magnitude $|l| = (l_{k,x}^2 + l_{k,y}^2 + l_{k,z}^2)^{0.5}$

- Simply adding the displacement of each leg's anchor at the ground yields the leg's position relative to the global frame's centre of the base.

And that's only to figure out the inverse kinematics of linear actuator-driven Stewart platforms.

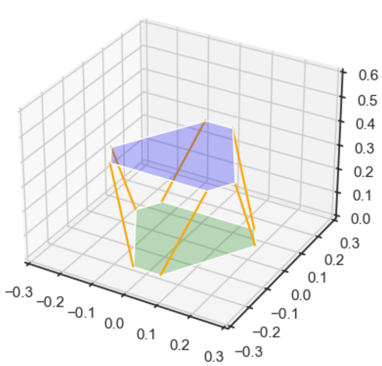

Figure 4. Initial Position

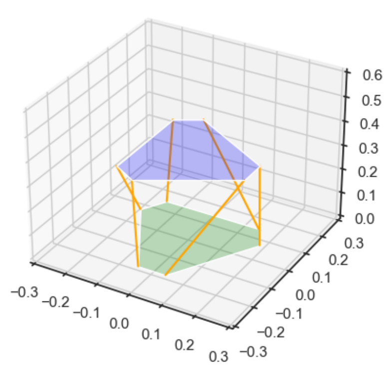

Figure 5. $Rot_Z(30°)$

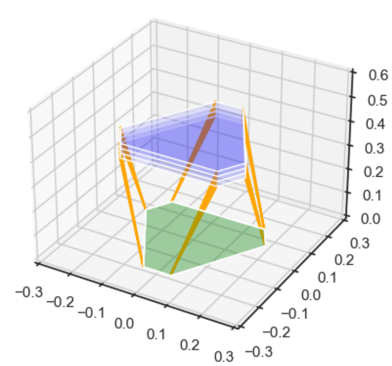

Figure 6. $Trans_Z(100mm)$

Figure 7. Line Diagram: Inverse Kinematics Analysis for Platform Position Variations

3 Linear Actuators

Our model consists of six linear actuators that are connected to six individual servo motors using a lead screw mechanism. The lead screw mechanism converts the rotational motion of the servo motor into linear motion to actuate the linear actuator. We control the servo motors using PWM (Pulse Width Modulation) technique, which allows us to control the position and speed of each actuator with high precision.

By using a lead screw mechanism, we are able to achieve a high mechanical advantage that allows us to move large loads with high accuracy and precision. The PWM (7) technique provides us with fine-grained control over the position and speed of each actuator, allowing us to achieve smooth and precise motion.

3.1 Actuator Force Calculation

To calculate the force required for a linear actuator to move a 20 kg mass to a full 0.19 m actuation within a time limit of 20 seconds, we would need to consider four elements: friction, acceleration, gravity, and applied force.

In the model, the actuator is making a 63.9° angle with the base and the coefficient of friction is 0.4. Now we can calculate the force required using the following equation:

$$F_{total} = F_{friction} + F_{accleration} + F_{load} + F_{applied}$$

$$F_{total} = \mu F_N \cos(\phi) + \frac{mv}{t} + mg\sin(\theta) + F_{applied}$$

Where,

- $m \rightarrow m_{Top} + m_{Load}$ Mass applied = 34 kg
  - Platform can carry a max load of 20kg
  - Top weight is 14 kg approx
- $g \rightarrow$ Earth gravity ($9.81 m/s^2$)
- $\theta \rightarrow$ Inclination angle (63.94°)
- $v \rightarrow V_{max}$ of actuation ($19 mm/s$)
- $t \rightarrow$ Time to actuate full length (20 seconds)
- $\mu \rightarrow$ Frictional coefficient of lead screw (0.4)
- $\phi \rightarrow$ Angle of Normal Load (90°)
  - Let's assume the load make a 90° angle
- $F_N \rightarrow$ Normal Force applied to the actuators
- $F_{applied} \rightarrow$ Dynamic force ($\pm 10N$)

After calculation $F_{total} \approx 360N$ is subjected to all six actuators i.e. each actuator is carrying a maximum load of 60 N.

Now, The servo torque required for a 60N force in a linear actuator will depend on the mechanical advantage of the actuator system, which is determined by the lead screw pitch and the diameter of the lead screw. Assuming that the actuator is a simple lead screw system and neglecting any losses due to friction, the torque required can be calculated using the equation: $T = \frac{Fp}{2\pi}$

- $T \rightarrow$ Torque required in each servo (Nm)
- $F \rightarrow$ Force applied to each actuators (60N)
- $p \rightarrow$ Lead screw pitch (5mm)

If the lead screw pitch is 5mm (0.005m) and the force required is 60N, then the torque required would be $\approx 0.024Nm$. [1]

[1] **Note:** Ideal conditions are assumed for the lead screw, but practical systems require a motor with a torque margin and a factor of safety of 4 to overcome friction, and inertia, and ensure reliability.

3.2 Pulse-Width Modulation

The linear actuator function converts servo motion to linear actuation using PWM signals. PWM stands for Pulse-Width Modulation, which is a technique for controlling the amount of power delivered to a device by rapidly turning the power on and off. By varying the duty cycle (the proportion of time that the power is on) and the frequency of the pulses, we can control the position, speed, and force of the linear actuator.

The linear actuator function uses the following formula to calculate the actuation step size:

```
frequency = 50
max_force = 250
steps = int(actuation_duration * frequency)
pwm_period = 1.0 / frequency
duty_cycle = 0.5
pwm_high_time = pwm_period * duty_cycle
actuation_step = [l / steps for l in linear_distance]
pwm_signal = i * pwm_period % pwm_period < pwm_high_time
```

Where,

- $'linear\ distance'$ $\rightarrow$ is an array of the desired linear distances to be covered by each actuator,
- $'steps'$ $\rightarrow$ is the number of PWM steps required to cover the distance in the given actuation duration,
- $'actuation\ step'$ $\rightarrow$ Step size for each actuator,
- $'max\ force'$ $\rightarrow$ Maximum linear actuation force.

The function then generates PWM signals with a frequency of 50 Hz and a duty cycle of 50% and actuates the linear actuators according to the desired distance and duration. The function also applies a maximum force of 250 N to each actuator to ensure stability and safety.

By using this function, users can easily convert servo motion to linear actuation and control the position, speed, and force of the linear actuators in their applications.

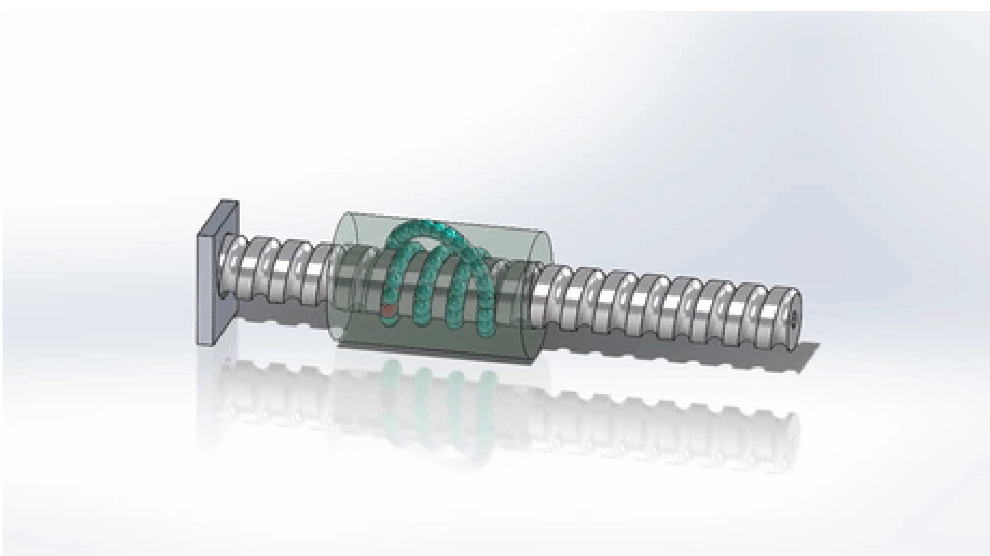

Figure 8. Linear Actuator Lead Screw Mechanism Diagram

4 Algorithm

Algorithm 1 Inverse Kinematic computing function

function INV-KINEMATICS(T, θ) ▷ $T \rightarrow$ translation and $\theta \rightarrow$ rotation in x,y,z
▷ Predefined Parameters
  $R_B, R_P, \gamma_B, \gamma_P, H$ ▷ Where R - Radius, γ - half angle, H - home position of platform
  $B = R_B \times \phi_B(\gamma_B)$ ▷ Calculate the coordinate point of base
  $P = R_P \times \phi_P(\gamma_P)$ ▷ Calculate the coordinate point of platform
  $rot(\theta_x, \theta_y, \theta_z) = rot_x \times rot_y \times rot_z$ ▷ Get rotation matrix of platform
  $l = T(x,y,z) + H + R(\theta) \times P - B$ ▷ Get Coordinate for each leg after actuation
  $L = (l_{k,x}^2 + l_{k,y}^2 + l_{k,z}^2)^{0.5}$ ▷ Leg lengths required to achieve the desired position of platform
end function

Algorithm 2 Linear Actuator: Motor driver function

function LINEAR-ACTUATOR(L, t) ▷ $L \rightarrow$ linear actuation distance and $t \rightarrow$ actuation duration
  $L_{prev} \rightarrow zeros|previousstate$
  $f = 50$ ▷ Frequency of 50HZ
  $F_{max} = 250$ ▷ 250 N linear actuation force
  $N = \lfloor t \times f \rfloor$ ▷ pwm no of steps
  $period_{pwm} = \frac{1.0}{f}$ ▷ pwm period
  $dutycycle = 0.5$ ▷ 50% duty cycle
  $t_{pwm} = period_{pwm} * dutycycle$
  $step_{actuate} = \frac{L}{steps}$ ▷ actuation step
  for $i = 0 \rightarrow length(N)$ **do**
    if $(i * period_{pwm} \% period_{pwm} < t_{pwm})$ **then** ▷ pwm signal is high
      $L_i = step_{actuate} * i + L_{prev}$ ▷ Target Position of the linear actuator
    else ▷ pwm signal is low
      $L_i = L_{i-1}$ ▷ Hold the previous position
    end if
    Move linear actuator to position L ▷ Activate linear actuators to the desired position L.
    $Sleep(\frac{1}{f})$
  end for
  $L_{prev} = Li$
end function

Algorithm 3 Simulation: Motor driver function

function SIMULATION($[L, \theta, t]$) ▷ $L \rightarrow$ translation, $\theta \rightarrow$ Rotation and $t \rightarrow$ time duration
  Load URDF, set the Physics environment
  Constrained the URDF to make it Parallel chain
  Start Simulation
  L = INV-KINEMATICS(L, θ)
  LINEAR-ACTUATOR(L, t)
  Stop simulation, Deactivate Physics environment
end function

5 Result & Discussion

5.1 Test 1

To simulate RPR 3DOF motion, we used a 6-DOF Stewart platform with the platform locked to a 100mm displacement towards the Z-axis with respect to the platform reference frame. We then simulated a sequence of motions as follows:

- 30 degrees yaw within 4 seconds.
- 20 degrees pitch within 3 seconds.
- 15 degrees roll within 2 seconds.
- Return back to home position within 1 second.

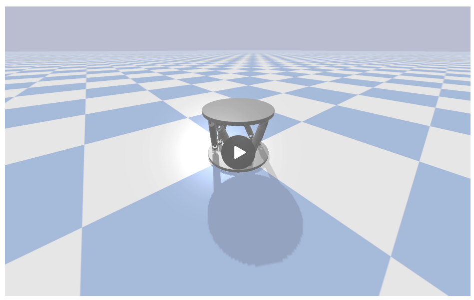

Figure 9. Simulation 1

5.2 Test 2

Simulated a sequence of motions as follows:

- 30 degrees yaw and return back to platform position within 4 seconds
- 20 degrees pitch return back to platform position within 3 seconds
- 15 degrees roll return back to platform position within 2 seconds
- Return back to home position within 1 second.

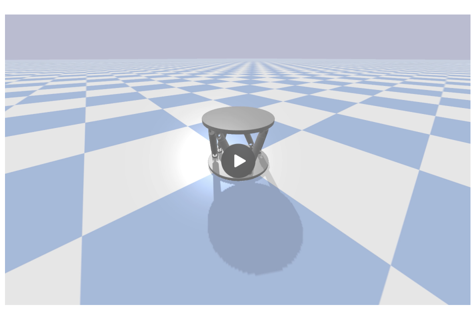

Figure 10. Simulation 2

5.3 Test 3

In the 6DOF simulation, we designed a custom environment and utilized a custom function to generate a spiral trajectory for the platform. Custom Python scripts were used to generate forces and torques for each actuator to achieve accurate motion. This simulation highlights the versatility and precision of the Stewart platform for robotics and automation. [2]

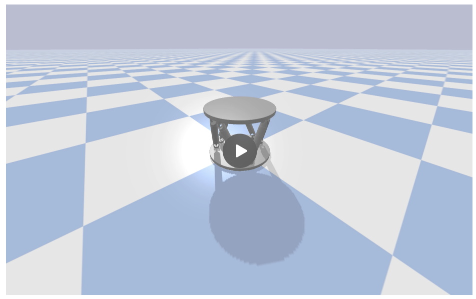

Figure 11. Simulation 3

6 Conclusion

- The simulation was performed using a custom Python script that utilized the PyBullet physics engine for simulating the platform's motion. The script included control algorithms that generated the required forces and torques for achieving the desired motion. We also designed a custom simulation environment that closely resembled the real-world setup of the platform.
- During the simulation, we monitored the position and orientation of the platform at each time step and recorded the resulting motion trajectories. The simulation results showed that the platform achieved the desired RPR 3DOF motion accurately and smoothly.
- Overall, this simulation provides valuable insights into the behaviour of the RPR platform and can be used to optimize its design and control algorithms for real-world applications.

References

1. Dasgupta, B. & Mruthyunjaya, T. The stewart platform manipulator: a review. *Mech. machine theory* **35**, 15–40 (2000).

2. Petrescu, R. V. *et al.* Inverse kinematics of a stewart platform. *J. Mechatronics Robotics* **2**, 45–59 (2018).

3. Craig, J. J. *Introduction to robotics* (Pearson Educacion, 2006).

4. Ghosal, P. A. lec15 direct kinematics of stewart platform manipulators (2023).

5. Lynch, K. & Park, F. Chapter 7: Kinematics of closed chains (2023).

6. Nanua, P. & Waldron, K. Direct kinematic solution of a stewart platform. In *1989 IEEE International Conference on Robotics and Automation*, 431,432,433,434,435,436,437, DOI: 10.1109/ROBOT.1989.100025 (IEEE Computer Society, Los Alamitos, CA, USA, 1989).

7. Narayanan, G. Pulse width modulation for power electronic converters (2023).

[2]**Resources**: Download Code, Combined motion