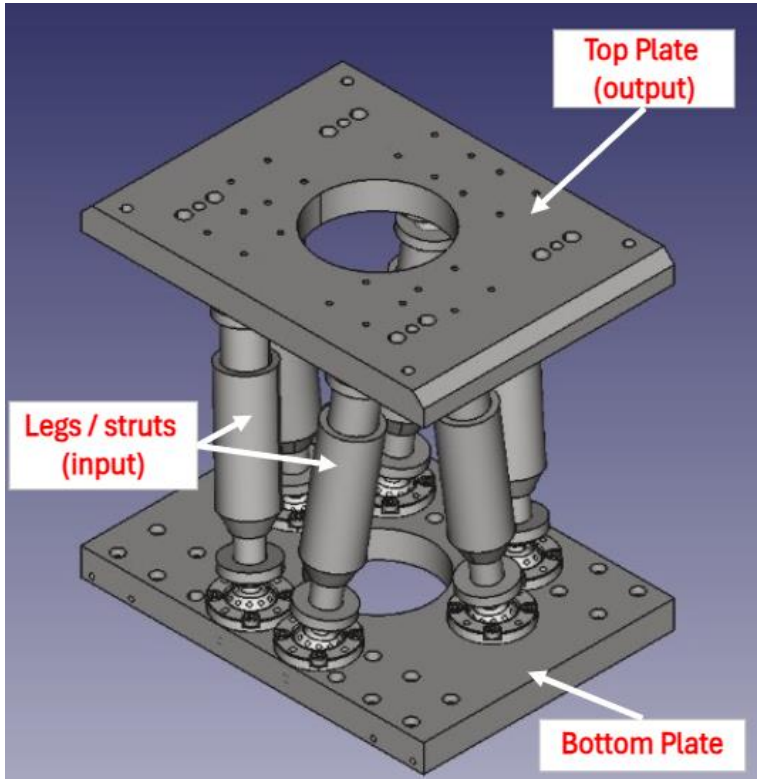
STEWART PLATFORM

by KODANDA CHALLA

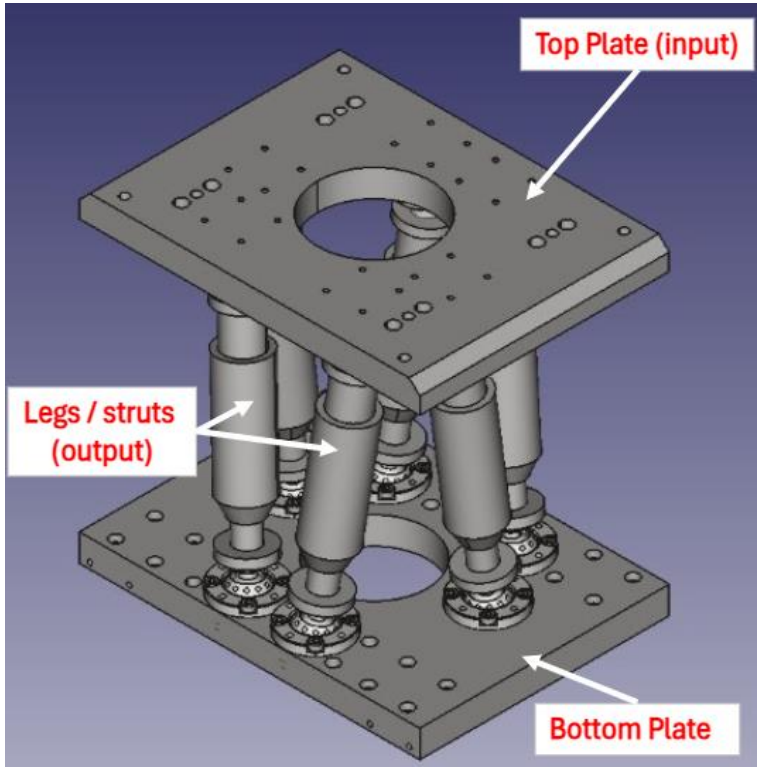
Forward Kinematics

- **Forward kinematics** involves determining the position and orientation of the top platform given the lengths of the six struts. This means calculating where the top platform is and how it is oriented in 3D space based on the known strut lengths.
- Steps:
  - **Input:** Known lengths of the six struts.
  - **Process:** Use geometric and trigonometric relationships to compute the position (x, y, z) and orientation (roll, pitch, yaw) of the top platform.
  - **Output:** The position and orientation of the top platform.
  - Forward kinematics for a Stewart platform is generally complex due to the non-linear relationships between the strut lengths and the platform's pose.



Inverse kinematics

- **Inverse kinematics** involves determining the required lengths of the six struts to achieve a desired position and orientation of the top platform. This means calculating how each strut should extend or retract to move the top platform to a specific pose.
- Steps:
  - **Input:** Desired position (x, y, z) and orientation (roll, pitch, yaw) of the top platform.
  - **Process:** Use the geometric and trigonometric relationships to compute the lengths of the struts.
  - **Output:** The lengths of the six struts.
  - Inverse kinematics is typically easier to solve than forward kinematics for a Stewart platform because it involves direct geometric relationships.



Summary:

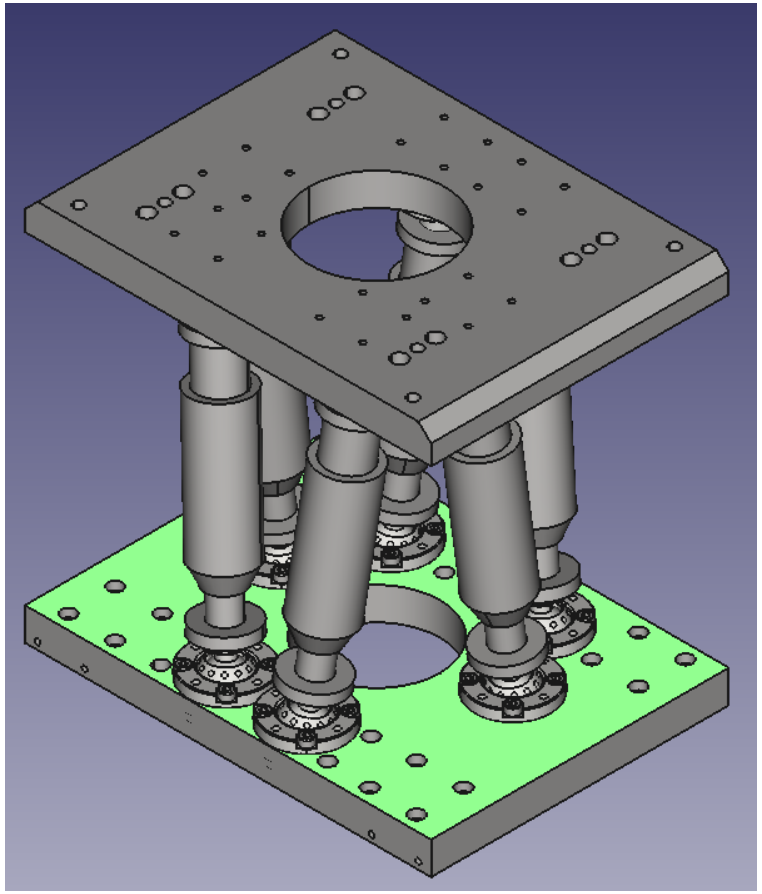
- **Forward Kinematics:** Strut lengths -> Position and Orientation of Top Plate
- **Inverse Kinematics:** Position and Orientation of Top Plate -> Strut lengths

Calibration

- Steps to Determine the Bottom Platform Top Plane and Leg Points
  - Construct Top Plane
  - Construct Side Planes
  - Locate Points
  - Construct Circle Points for Legs
  - Locate Leg Points

- Steps to Determine the Top Platform Top Plane and Leg Points
  - Repeat the same steps outlined above.

Steps to Determine the Bottom Platform Top Plane and Leg Points



1. construct Top Plane: (CMM)*

- Take points along the top edges of the bottom platform, ensuring coverage all around the platform.
- Construct the top plane using these points to accurately represent the top surface of the bottom platform.

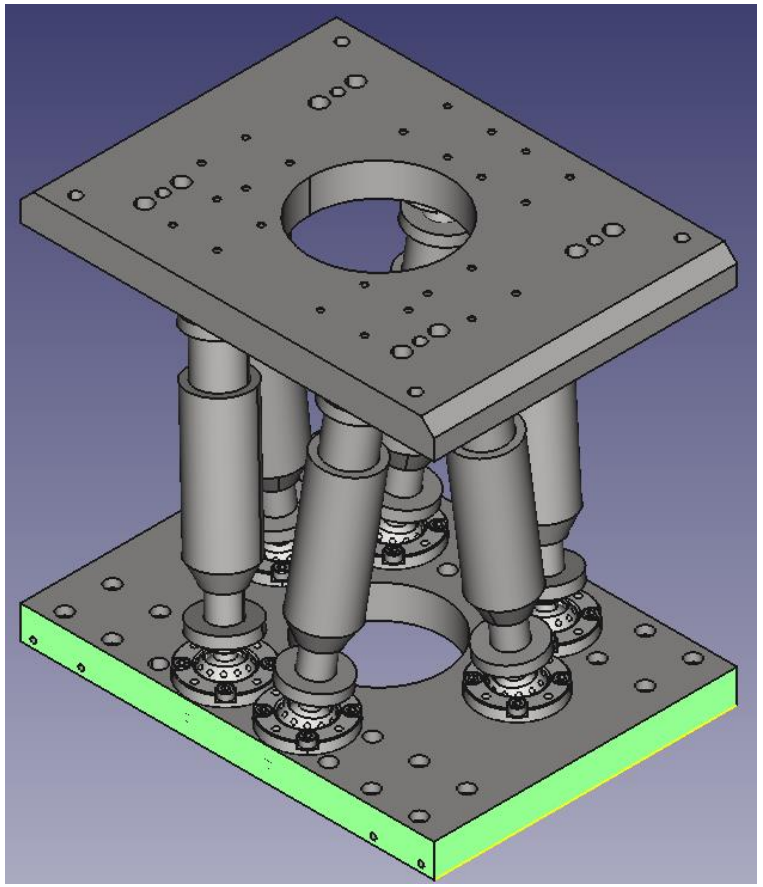
*Coordinate Measuring Machine

2. Construct Side Planes: (CMM)

- Take points along the four sides of the bottom platform.

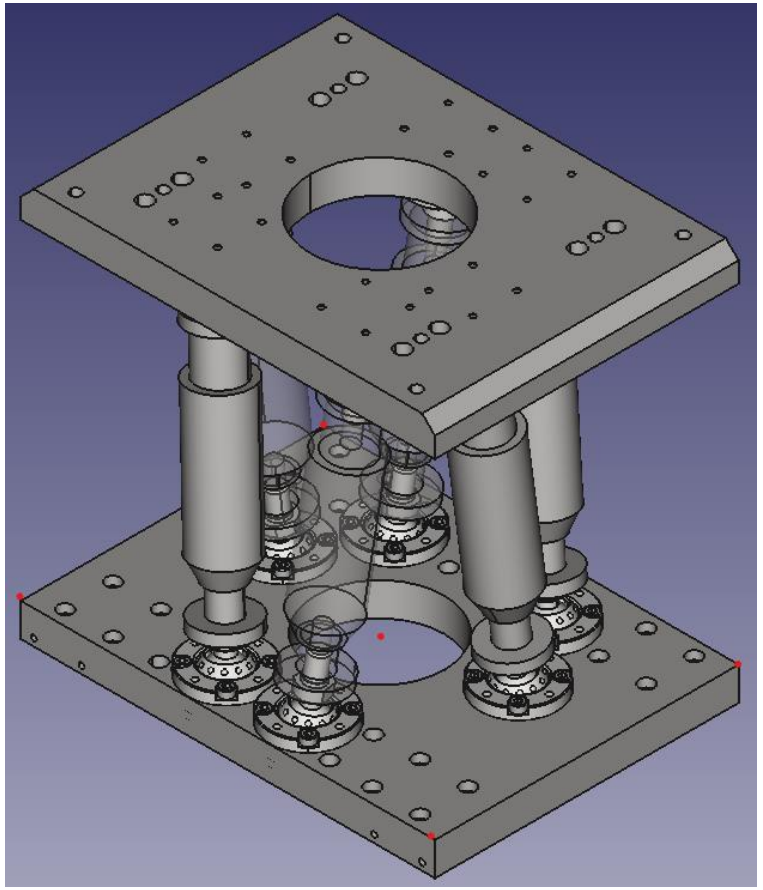
- Construct planes for at least two non-opposite sides of the platform. The other two sides will be parallel and at an offset distance on the bottom platform plane.

- For greater accuracy, construct planes for all four sides.



3. Locate Points: (CMM)

- Using the intersection of the four side planes and the top plane, locate the four corner points (fig) of the top plane of the bottom platform.
- Remember origin at center of rectangle



4. Construct Circle Points for Legs: (MATLAB)

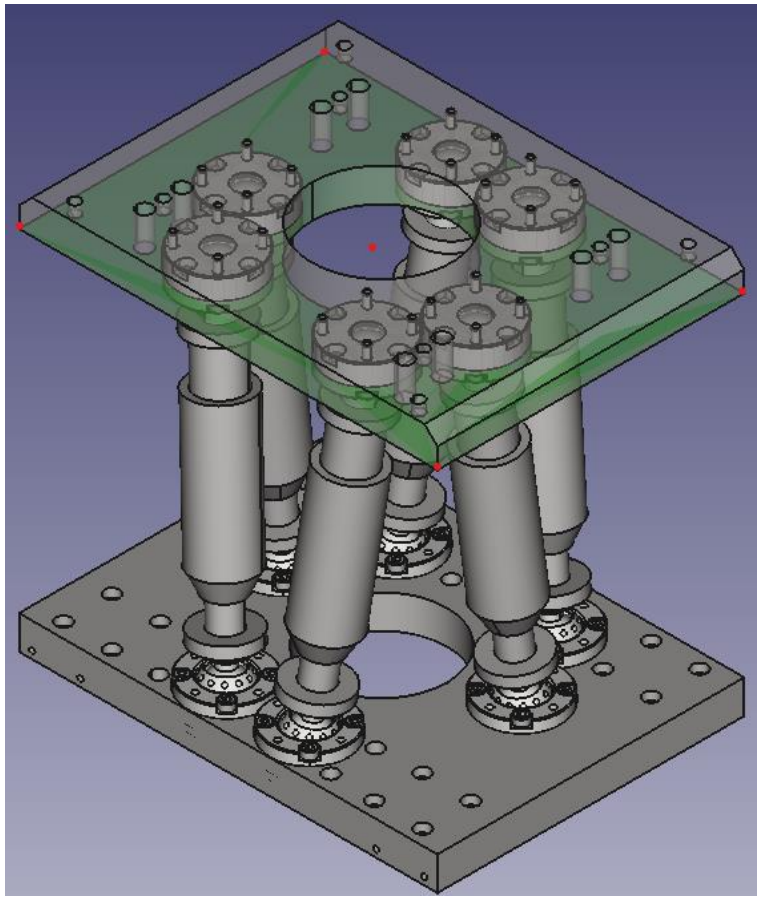
- Replicate planes in the Matlab.
- On the top plane of the bottom platform, construct circles at the desired positions where the legs will be attached.

5. Locate Leg Points: (MATLAB)

- Mark the precise leg points on the top plane of the bottom platform based on the constructed circles.

Steps to Determine the Top Platform Top Plane and Leg Points

- Repeat the same steps outlined for Bottom Platform.
- offset the plane downwards by thickness.
- Else, Take readings on bottom surface of top plane.
- Remember origin at center of bottom rectangle



Checking

- Case 1: After completing the previous two steps:
  - Step1: Determine the top plane and leg points of the bottom platform.
  - Step2: Determine the top plane and leg points of the top platform.
  - Step3: Calculate leg lengths using inverse kinematics (MATLAB) and store it.
- Case 2: Adjust leg lengths manually as needed and note down increments / decrements in length:
  - Step1: Determine the top plane and leg points of the bottom platform.
  - Step2: Determine the top plane and leg points of the top platform.
  - Step3: Calculate leg lengths using inverse kinematics (MATLAB) and store it.
- Verify change: Case 1 Lenths + increments / decrements = Case 2 Lenth, if this satisfies model is correct else non linearity of prototype.

Conclusion

- System is Non- Linear
- Model is reliable for initial adjustments

```
>> out.legLengths

ans =

  179.6890  178.7933  180.0053  181.2739  181.9124  180.1221

>> out.legLengths

ans =

  181.8478  181.1018  181.4217  180.8350  181.3955  181.2932
```

THANK YOU

KODANDA CHALLA