KODANDA CHALLA

Model Based Design (MBD)

Forward Kinematics of the Stewart Platform (HEXAPOD)

In [1]:
```
import numpy as np
from scipy.spatial.transform import Rotation as R
from scipy.optimize import fsolve
import plotly.graph_objects as go
```

Input

- 6 Leg lengths in mm (leg_lengths)

In [2]:
```
leg_lengths = np.array([ 180.769486760529, 184.588371003279, 185.497976872750, 184.773016742414, 185.045189780015, 184.5029496
print(leg_lengths.shape)
```

```
(6,)
```

Calibarated Leg locations on Plates

- Actual Base platform (fixed platform) Leg Points (B)
- Actual Top Platform (Moving Platform) Leg Points (P)

In [3]:
```
B1 = np.array([49.834,41.765,-10.52+11])
B2 = np.array([11.327,63.988,-10.504+11])
B3 = np.array([-61.058, 22.228,-10.654+11])
B4 = np.array([-61.06,-22.233,-10.803+11])
B5 = np.array([11.3,-64.009,-10.898+11])
B6 = np.array([49.825,-41.784,-10.779+11])
B = np.array([B1, B2, B3, B4, B5, B6]).T

P1 = np.array([58.708,21.383,-11.171+11])
P2 = np.array([-10.875,61.556,-10.94+11])
P3 = np.array([-47.912,40.187,-10.888+11])
P4 = np.array([-47.907,-40.162,-11.017+11])
P5 = np.array([-10.873,-61.536,-11.135+11])
P6 = np.array([58.704,-21.364,-11.243+11])
P = np.array([P1, P2, P3, P4, P5, P6]).T

print(B, B.shape,"\n"*2, P, P.shape)
```

```
[[ 49.834  11.327 -61.058 -61.06    11.3     49.825]
 [ 41.765  63.988  22.228 -22.233 -64.009 -41.784]
 [  0.48    0.496   0.346   0.197   0.102   0.221]] (3, 6)

 [[ 5.8708e+01 -1.0875e+01 -4.7912e+01 -4.7907e+01 -1.0873e+01  5.8704e+01]
 [ 2.1383e+01  6.1556e+01  4.0187e+01 -4.0162e+01 -6.1536e+01 -2.1364e+01]
 [-1.7100e-01  6.0000e-02  1.1200e-01 -1.7000e-02 -1.3500e-01 -2.4300e-01]] (3, 6)
```

Non-Linear solver

In [4]:
```
def rotational_Matrix(roll, pitch, yaw):
    # Create a rotation object from Euler angles
    rotation = R.from_euler('XYZ', np.array([roll, pitch, yaw]))
    # Convert to the rotation matrix
    rotation_matrix = rotation.as_matrix()
    return rotation_matrix

def kinematic_equations(x,B,P,leg_lengths):
    px, py, pz, roll, pitch, yaw = x
    rotation_matrix =  rotational_Matrix(roll, pitch, yaw)
    F = np.zeros(6)

    for i in range(6):
        p_global = np.array([px, py, pz]) + np.dot(rotation_matrix, P[:, i])
        d = np.linalg.norm(p_global - B[:, i])
        F[i] = d - leg_lengths[i]
    return F

x0 = np.array([0, 0, np.mean(leg_lengths), 0, 0, 0])
result = fsolve(lambda x: kinematic_equations(x, B, P, leg_lengths), x0)
```

Output

- desired_position (mm)

- desired_orientation (deg)

In [5]:
```
desired_position = result[0:3]
desired_orientation = result [3:]
px, py, pz, roll, pitch, yaw = result

print(f'Position and orientation of the platform:\n'
      f'x: {result[0]:.4f}\n'
      f'y: {result[1]:.4f}\n'
      f'z: {result[2]:.4f}\n'
      f'roll: {np.degrees(result[3]):.4f}\n'
      f'pitch: {np.degrees(result[4]):.4f}\n'
      f'yaw: {np.degrees(result[5]):.4f}')
```

```
Position and orientation of the platform:
x: -2.9130
y: 10.8480
z: 182.8397
roll: -0.5034
pitch: 1.3710
yaw: 3.1790
```

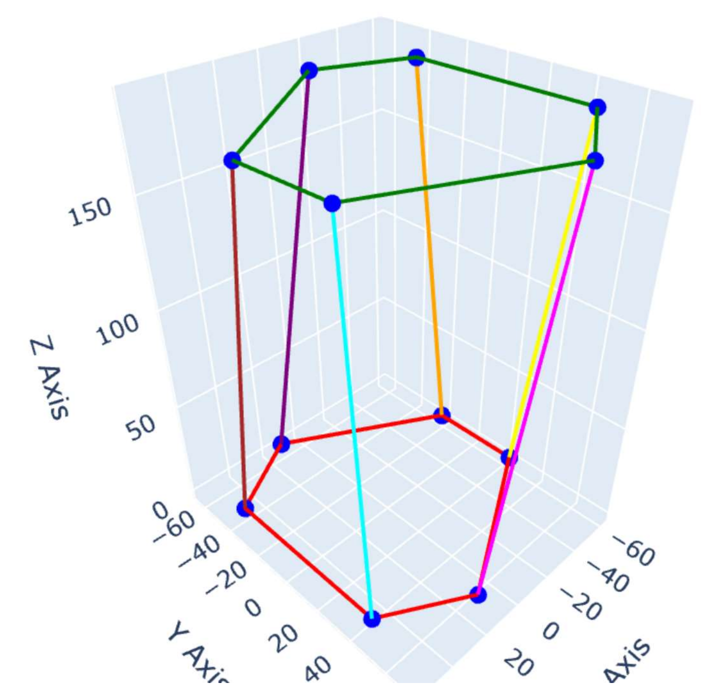
In [6]:
```
P_global = desired_position.reshape(3,1) + np.dot(rotational_Matrix(roll, pitch, yaw) , P)
print (P_global, P_global.shape)
```

```
[[ 54.49831898 -17.17936592 -52.9628508  -48.50642369 -10.3578543
   56.86248352]
 [ 35.43925501  71.70737403  48.32623429 -31.89781827 -51.19350027
   -7.2416684 ]
 [181.07855043 182.70649661 183.82038554 184.28954934 183.4280008
  181.32494693]] (3, 6)
```

Plot

In [7]:
```
def plot_closed_3d_lines(X, Y, Z, Color, Name):
    fig.add_trace(go.Scatter3d(
        x=X,
        y=Y,
        z=Z,
        mode='lines+markers',
        name=Name,          # Name of the line
        line=dict(color=Color, width=4),  # Line color and width
        marker=dict(size=5, color = "blue")   # Marker size and color
    ))

# Plot Base platform leg points
x1 = np.append(B[0], B[0,0])
y1 = np.append(B[1], B[1,0])
z1 = np.append(B[2], B[2,0])

fig = go.Figure()
plot_closed_3d_lines(x1, y1, z1, "red", "Fixed")


# Plot TOP platform leg points
x2 = np.append(P_global[0], P_global[0,0])
y2 = np.append(P_global[1], P_global[1,0])
z2 = np.append(P_global[2], P_global[2,0])
plot_closed_3d_lines(x2, y2, z2, "green", "Moving")


#Plot Legs
colors = ['cyan', 'magenta', 'yellow', 'orange', 'purple', 'brown']
for i in range(P_global.shape[1]):
        x = [B[0, i], P_global[0, i]]
        y = [B[1, i], P_global[1, i]]
        z = [B[2, i], P_global[2, i]]
        plot_closed_3d_lines(x, y, z, colors[i], f'leg {i + 1}')

# Update layout with titles and labels
fig.update_layout(
    title='Hexapod Forward',
    scene=dict(
        xaxis_title='X Axis',
        yaxis_title='Y Axis',
        zaxis_title='Z Axis',
        aspectmode='data',

    )
)

fig.show()
```

Hexapod Forward



```
In [8]: x, y, z = desired_position
        roll, pitch, yaw = desired_orientation

        print("INPUT")
        print("\tLeg Lengths (mm):")
        for i, leg in enumerate (leg_lengths):
            print("\t",i+1,":", leg)

        print("OUTPUT")
        print(f"\tPosition (mm):\n \t x = {x} \n \t y = {y}, \n \t z = {z} \n")
        print(f"\tOrientation (deg): \n \t roll = {np.rad2deg(roll)} \n \t pitch = {np.rad2deg(pitch)}, \n \t yaw = {np.rad2deg(yaw)}
```

```
INPUT
        Leg Lengths (mm):
         1 : 180.769486760529
         2 : 184.588371003279
         3 : 185.49797687275
         4 : 184.773016742414
         5 : 185.045189780015
         6 : 184.502949692505
OUTPUT
        Position (mm):
         x = -2.912999998429377
         y = 10.847999998730808,
         z = 182.8397499928279

        Orientation (deg):
         roll = -0.5033999970290962
         pitch = 1.3710000064312333,
         yaw = 3.1790000007285815
```