摘　要

六自由度运动平台是由六个伺服电机带动电动缸做伸缩变化运动，六个电动缸并联设置共同驱动运动平台。由于运动平台的电机较多，因而其对于平面的控制也更加的复杂。本文将尝试对六自由度运动平台进行建模，运用建立空间直角坐标系的方法，对于平台的运动问题进行研究。

针对问题一，首先建立空间直角坐标系，并运用确定的坐标来表示电动缸的两端坐标，假设上平台在 x 轴和 y 轴上的平动距离为变量，从而推出各个电动缸的坐标变化情况，通过考虑电动缸的限位状况来判断坐标数据是否合法，对于合法的坐标，通过两点之间距离公式来计算电动缸的长度，从而可以得到电动缸长度和上平台平动的函数关系。

针对问题二，首先固定上平台的中心点坐标为定制，接着假定上平台的欧拉角为变量，也就是假定上平台相对于 xyz 轴转动的角度为变量，接着将欧拉角转化成旋转矩阵，从而得出上平台各个点的坐标变化情况，通过坐标来判断电动缸是否满足限位的条件，对于满足条件的变化情况，通过两点之间距离公式得出电动缸的长度随着上表面欧拉角的变化情况。

针对问题三，由于此问题即牵扯到平动问题又牵扯到旋转问题，因而对于问题三，采用的是具例子的方法，通过假定上平台的欧拉角旋转状态以及上平台的平动位移状态，再通过 python 脚本来对这些变量值进行计算，得出电动缸的坐标值，通过坐标值进而再判断其是否符合限位的实际情况，由于牵扯到 6 个变量，不太方便描述，所以我们将举例两组数据，通过这两组数据来计算各个电动缸的长度值。

关键词：空间直角坐标系，欧拉角，旋转矩阵，限位

1 问题重述

1.1 问题背景

1965 年，德国 Stewart 发明了六自由度并联机构，作为飞行模拟器用于训练飞行员。1978 年,澳大利亚著名机构学教授 Hunt 提出将并联机构用于机器人手臂。

此后，日本以及俄罗斯、意大利、德国等欧洲的各大公司相继推出并联机器人作为加工工具的应用机构。随着六自由度并联机构的不断发展，相应的问题也随之出现。由于六自由度并联机器人的一个最大弱点是空间小，应该说这是一个相对的概念。同样的机构尺寸，串联机器人比并联机器人工作空间大；具备同样的工作空间，串联机构比并联机构小。这就导致了六自由度并联机器人工作空间的解析求解是一个非常复杂的问题，它在很大程度上依赖于结构位姿解的研究成果，至今仍没有完善的方法。

1.2 问题要求

题目给出了关于六自由度机械并联结构的示意图以及简化后的模型图，如图 1.1 和 1.2 所示，根据 1.2 的模型图，我们建立空间直角坐标系，将机械结构的运动问题变成坐标点的运算问题。

图 1.1 六自由度机械平台

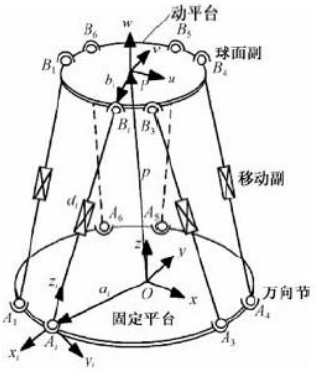

图 1.2 机械平台结构图

问题一：假定上平台的高度不变，即上平台做平行于底座的平动运动。要求建立上平台平动与六根支杆的长度变化的函数关系式。

问题二：假定上平台的中心点固定不变，分析上平台绕各个方向旋转时，建立对应的六支杆的长度关系变化式。

问题三：分析上平台随意运动时，各个支杆的长度变化关系，要求举出一些具体的数值进行计算。

2 模型建立与假设

2.1 六自由度并联结构的模型化处理

在正式研究问题之前，我们需要将六自由度并联机构模型化处理，来为我们的研究做准备。

我们采取的数学模型化处理则是考虑建立空间直角坐标系，运用空间直角坐标的方式，采用 xyz 轴的三维数据来描述该并联结构的运动状态。

当然，由于运动过程中，有许多的点的状态是无关紧要的，它们的状态并不影响最终并联结构的运行，所以可以忽略不计。

所以我们假设底座和上顶板为一个圆柱，假定液压杆为端点固定，长度可变的线段。如图 2.1 所示。

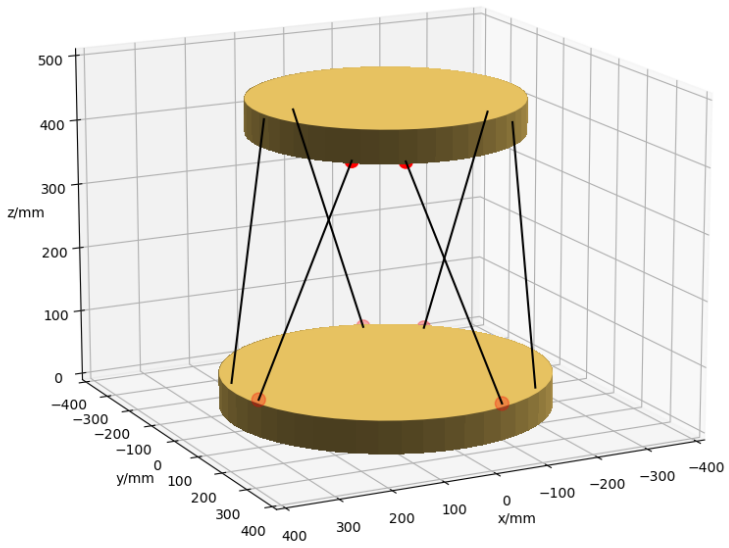

图 2.1 模型示意图

这样做的好处是忽略了许多无关紧要的因素，便于后续的运算。可以把圆柱和直线用空间几何方程描述出来。但是这样的缺点也是十分明显的，根据生活实际情况可知，六液压杆的长短收缩和转动范围是有限制的，圆盘的转动范围也是有限制的，单纯的把液压杆看成厚度忽略不计的线段，会导致线段可以无限伸长，并且无转动死角的问题，这显然是不符合实际的。因而我们需要添加一定的限制条件，限定线段的长度和转动角度。在限制条件的范围内，将其转化成如图 2.1 所示的模型。

2.2 模型的建立与数据规定

我们使用了 Autodesk Fusion 360 建立 3 维模型来进行运动仿真从而得出限位与基本尺寸，由于主要目的是进行运动学仿真，因此忽略了紧固件的设计。我们规定初始状态下，上圆面和底座的圆心的连线和水平面垂直，且上圆面和底面平行，我们以底座的圆心作为空间直角坐标系的原点，以底座的圆心和上圆面的圆心的连线构成的直线作为 y 轴，规定以底座上的两个万向球的球心的连线的中点和底座圆心的连线作为 x 轴，建立空间直角坐标系，规定底座的直径为 600 毫米，规定上半圆面的直径为 500 毫米，规定上圆面的高度为 420 毫米，如图 2.2 所示。

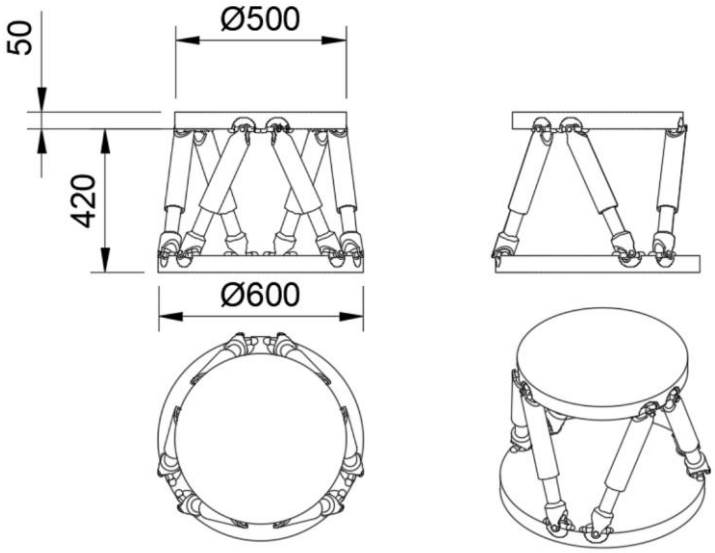

图 2.2

万向节尺寸如图 2.3 到 2.7 所示：

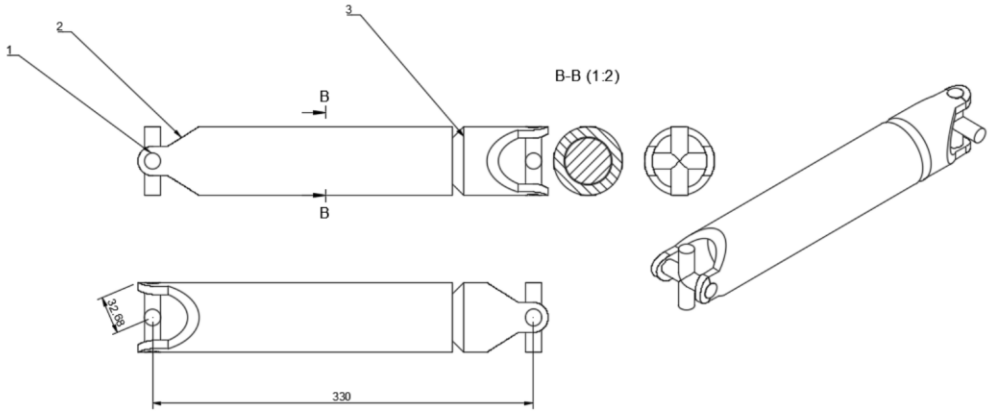

图 2.3 万向节装配图

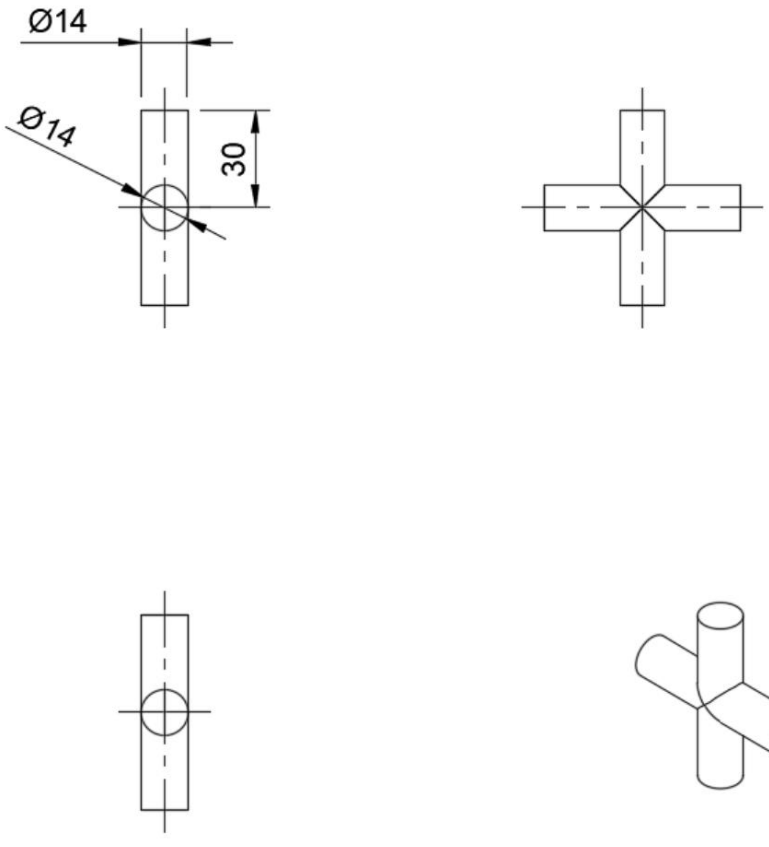

图 2.4 零件 1

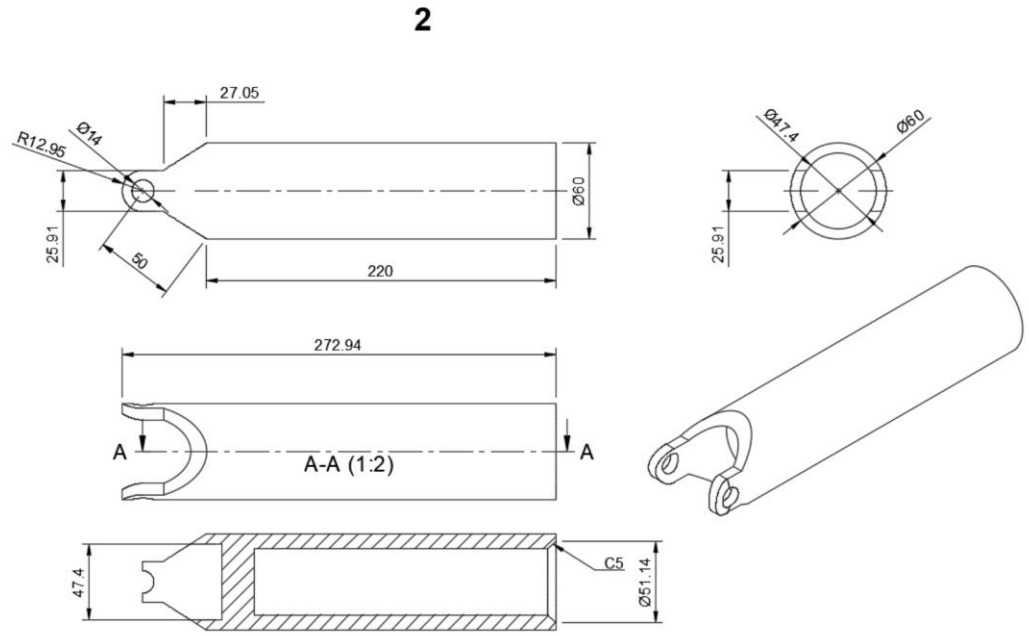

图 2.5 零件 2

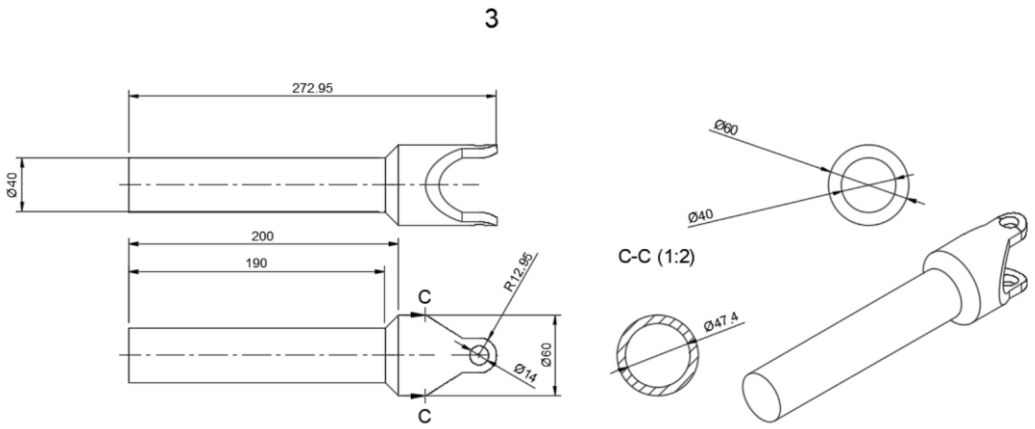

图 2.6 零件 3

上下平面示意图如图 2.7 和 2.8 所示：

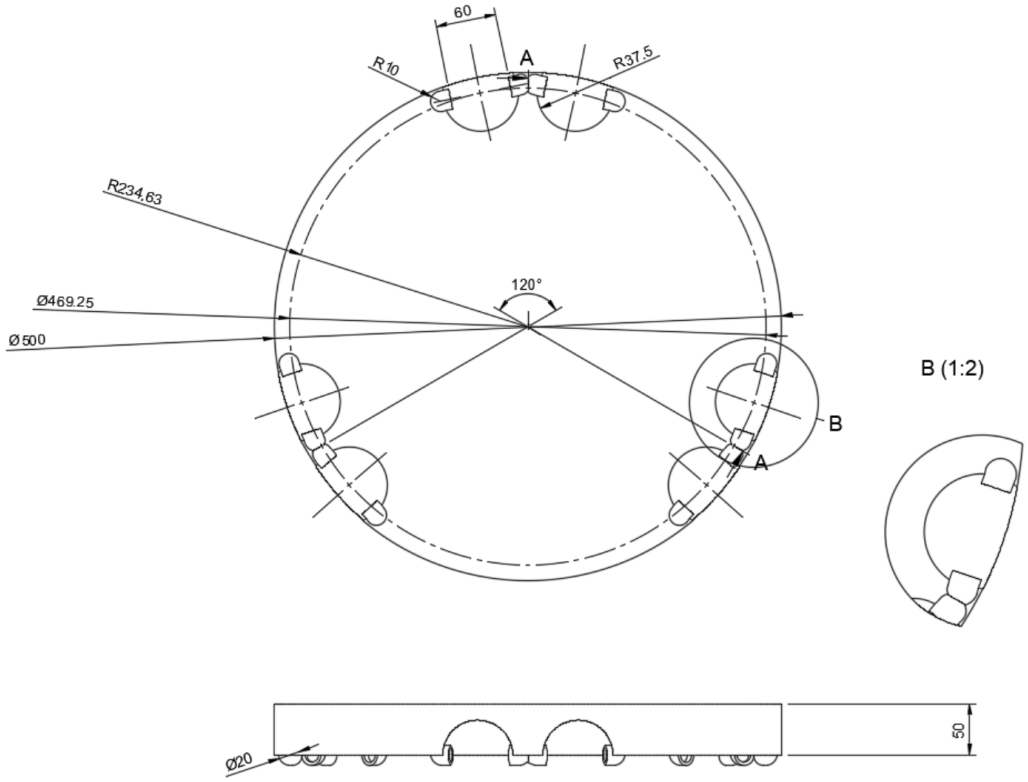

图 2.7 上半圆盘平面示意图

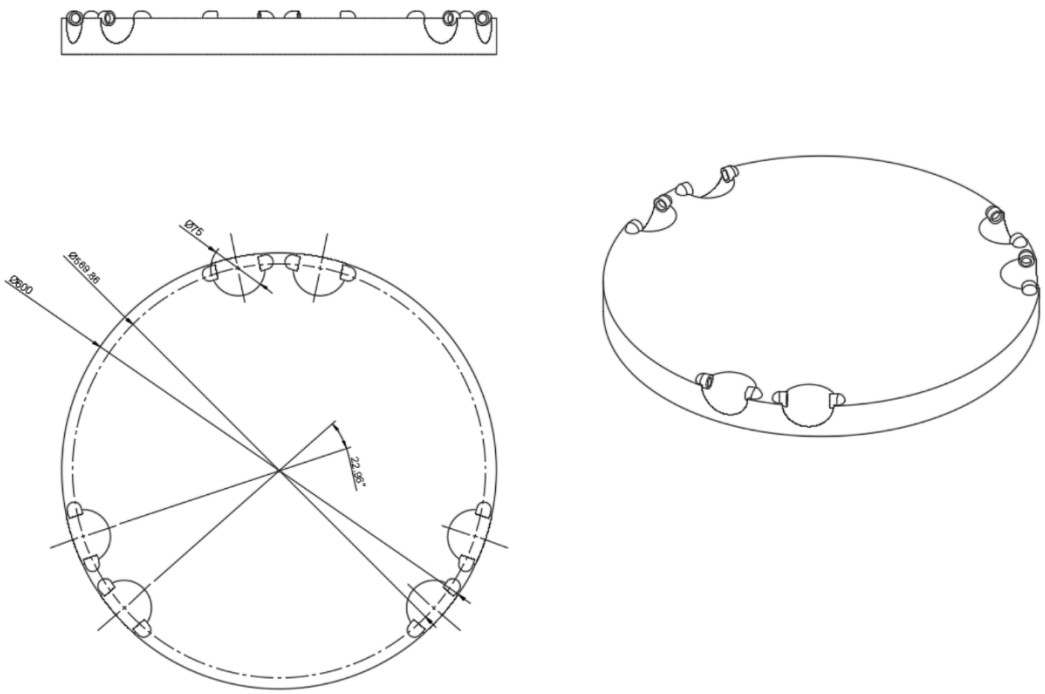

图 2.8 底圆盘平面示意图

对于液压杆部分，我们限定其的长度移动范围是 330 毫米到 510 毫米之间，如图 2.9 所示：

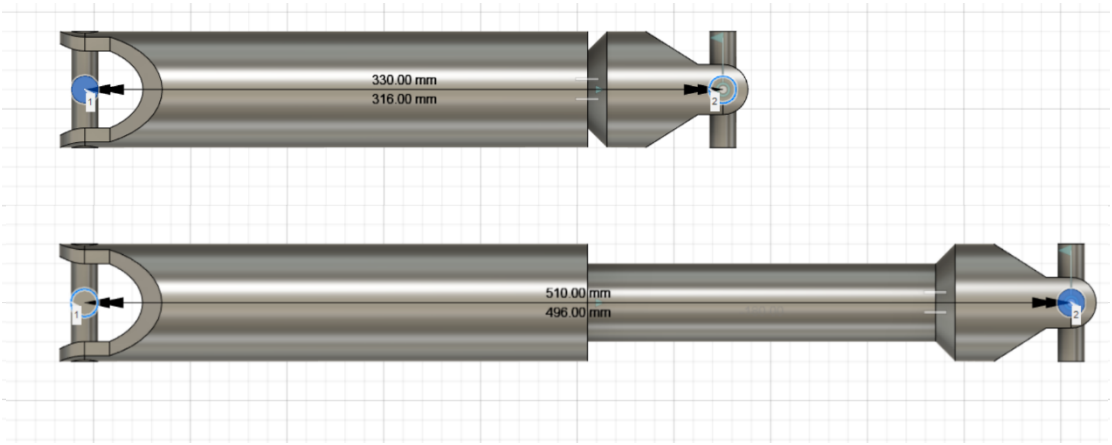

图 2.9

3 符号说明

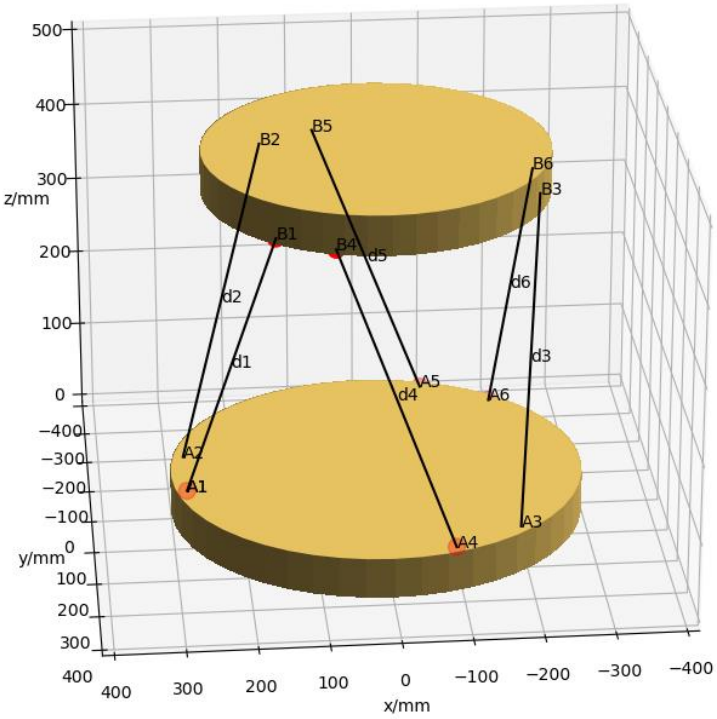

图 3.1 模型示意图

符号名称	意义
A1，A2，A3，A4，A5，A6	底盘面上的六个万向球的球心点
B1，B2，B3，B4，B5，B6	上圆盘面上的六个万向球的球心点
d1，d2，d3，d4，d5，d6	线段 A1B1，A2B2，A3B3，A4B4，A5B5，A6B6 的长度
θ	上平台的章动角
ψ	上平台的旋进角
φ	上平台的自转角
dx	上平台在 x 方向上的位移
dy	上平台在 y 方向上的位移
dz	上平台在 z 方向上的位移

4 问题分析

对于问题一，我们先通过带入 2.2 节中的数据，得到 A1~A6 以及 B1~B6 的参数，如表格 4.1 所示：

点的名称	X 值	Y 值	Z 值
A1	279.2298	56.7084	50.0000
B1	155.1022	175.7817	420.0000
A2	279.2298	-56.7084	50.0000
B2	155.1022	-175.7817	420.0000
A3	-188.7258	213.4658	50.0000
B3	-230.0172	46.6974	420.0000
A4	-90.5039	270.1743	50.0000
B4	74.9150	222.4792	420.0000
A5	-90.5039	-270.1743	50.0000
B5	74.9150	-222.4792	420.0000
A6	-188.7258	-213.4658	50.0000
B6	-230.0172	-46.6974	420.0000

表格 4.1 点的坐标数据表

接着由两点之间距离公式：如果空间中两点的坐标 A(x_1, y_1, z_1)，B(x_2, y_2, z_2)，那么规定两点之间的距离 d 有公式 4.1：

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{4.1}$$

由于第一问只是简单的平动问题，所以上平台每一点的运动坐标都只需要在 x 坐标上加 dx，在 y 坐标上加 dy 即可表示每一点的坐标，通过 4.1 即可算出支杆长度随着位移的变化情况。

对于问题二，在描述上平台运动状态时，由于会存在旋转的情况，因而在这边引入欧拉角的描述方法。

用欧拉角表示旋转的最大好处就是直观，一眼看去就可以清楚知道绕 xyz 轴的旋转角度，但是用欧拉角做位置计算将变得非常的复杂。所以这边还需要引入旋转矩阵的概念。

旋转矩阵描述了三维空间中一个点通过分别绕 xyz 轴进行旋转到另一个点的变化过程，在三维空间中，旋转矩阵是一个 3*3 大小的方阵。

如果我们假设空间中两点的坐标 A(x_1, y_1, z_1)，B(x_2, y_2, z_2)。假设 A 经过旋转矩阵 R 的变化后到达了 B 点，那么有公式 4.2：

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = R \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{4.2}$$

在已知欧拉角的情况下，旋转矩阵 R 可以计算出来，公式如 4.3 所示：

$$R = \begin{bmatrix} cos\varphi cos\psi - sin\varphi sin\psi sin\theta & -sin\varphi cos\theta & cos\varphi sin\psi + sin\varphi sin\theta cos\psi \\ sin\varphi cos\psi + cos\varphi sin\theta sin\psi & cos\varphi cos\theta & sin\varphi sin\psi - cos\varphi sin\theta cos\psi \\ -cos\theta sinh & sin\theta & cos\theta cosh \end{bmatrix} \tag{4.3}$$

假定上平台的欧拉角参数为变量，那么通过 4.2 和 4.3 即可算出上平台每一点旋转后的坐标，再通过 4.1 可算出支杆长度随着旋转的函数关系。

对于问题三，此时平台的运动牵扯到了六个变量，即绕着 xyz 轴的平动位移以及旋转角度，同时还要考虑数据的限位情况，变量过多不便于分析，因而此处通过编写 python 脚本的方式，运用程序的方式给出相应的数值模拟结果。

5 问题求解

5.1 平动问题的研究

在假定上平面做平行于 XOY 平面的平动运动，不发生旋转运动时。此时，其实上圆盘每一点的运动位移都是相同的。由于此时圆盘的厚度并不会影响我们分析的结论，所以此处将圆盘的厚度忽略不计，假定上圆盘是个圆面，而圆心又是上圆盘中最为特殊的一点，所以不妨以上圆盘圆心的位移最为本节分析的对象，通过分析上圆盘圆心的位移来计算六液压杆长度变化情况。

假定上圆盘的圆心为点 O，那么按照 2.2 节中的数据规定，可以知道点 O 的坐标为$(0,0,420)$。假定上圆盘经过平动运动后，在 x 轴方向上的位移为 dx，在 y 轴方向上的位移为 dy。则对于上圆盘底面的点B_i $(i = 1,2,3,4,5,6)$，初始坐标(p_x, p_y, p_z)，它的坐标变为：

$$B_i{}' : (p_x + dx, p_y + dy, p_z)$$

由于其下底面坐标不变，由式 2.1 可知其长度为：

$$d_i = \sqrt{(\overrightarrow{A_i} - \overrightarrow{B_i{}'})^2} \quad (i = 1,2,3,4,5,6)$$

其中$\overrightarrow{A_i}$和$\overrightarrow{B_i{}'}$均表示以原点为起点，自身为终点的向量。

由 2.2 节中的数据，液压杆的长度为 330 到 510 毫米，由此有约束条件：

$$d \in [330, 510]$$

又由于可供液压杆自由转动的范围只有一个半径为 37.5mm 的球体（如图 2.7，2.8 所示）而套筒的半径为 30mm，故其相对于上下平面最小倾角为：

$$\alpha_{min} = \arcsin\left(\frac{30}{37.5}\right) \approx 53.13^{\circ}$$

因此又有约束条件：

$$\alpha \in [53.13^{\circ}, 90^{\circ}]$$

其中，计算倾角θ的公式为公式 5.1：

$$\alpha = \arcsin\left(\frac{(\overrightarrow{B_i{}'} - \overrightarrow{A_i}) \cdot \overrightarrow{n_{平面}}}{d_i}\right) \quad (i = 1,2,3,4,5,6) \tag{5.1}$$

式中$\overline{n_{平面}}$为上/下平面的单位法向量，对于下平面，$\overline{n_{下}}=[0,0,1]$。对于上平面，由于其可以旋转，其初始法向量为$\overline{n_{下}}=[0,0,1]$旋转后的法向量可以由公式 4.2 得出

以 d1 为例，绘制其在约束范围内随着上顶面平动的变化图，其中 x，y 轴为上顶面平动的增量，z 轴为 d1 的长度。图 5.1 为没有约束条件，图 5.2 为有约束条件。

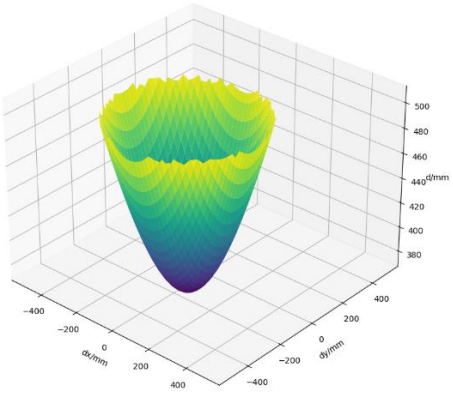

图 5.1 没有约束条件

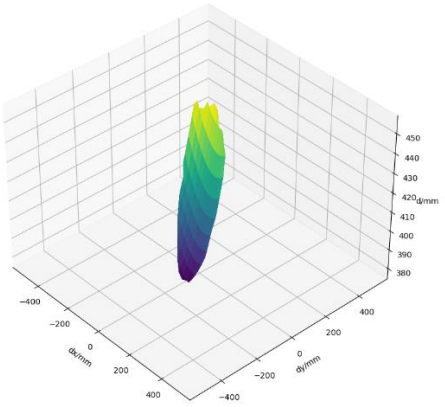

图 5.2 有约束条件

同时，我们还可以得到上平台的运动范围，如图 5.3 所示。

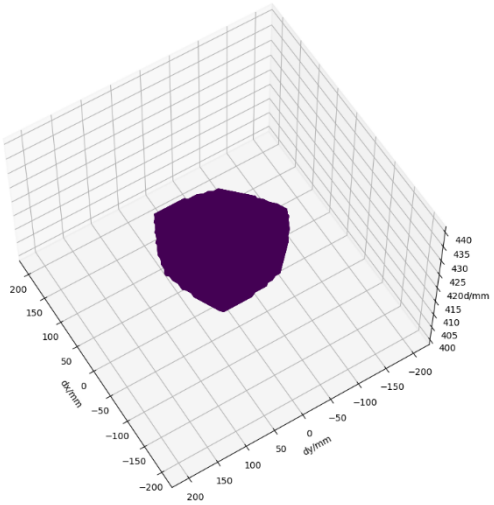

图 5.3

图 5.4 以及图 5.5 为使用 Autodesk Fusion 360 进行运动模拟与 python 脚本数值模拟的对比，其中 dx = 70, dy = -70，脚本源代码见附录 A

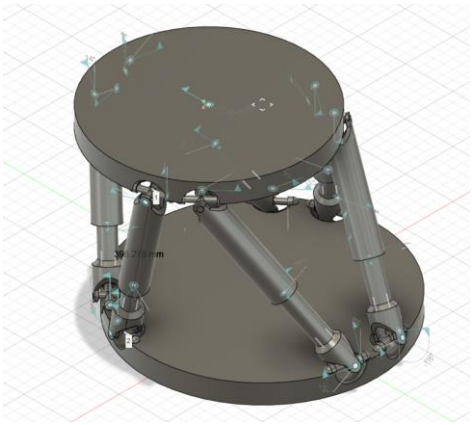

图 5.4 Fusion 360 模拟结果

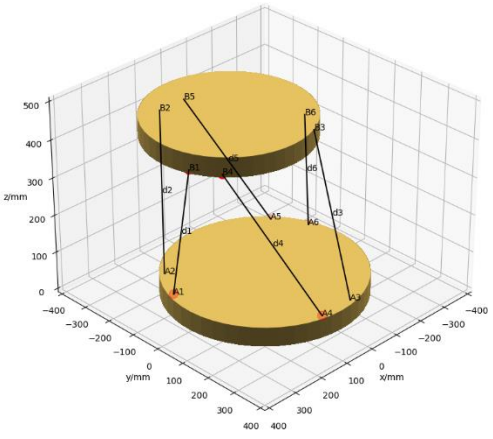

图 5.5 python 脚本模拟结果

注意到在 Fusion 360 中选择的是上下十字轴的顶点进行测量，因此实际长度约为 396.278-14 = 382.278，而使用 python 数值模拟的结果为 382.652，与 Fusion 360 基本一致，足以说明脚本的正确性。

5.2 旋转问题的研究

在假定上平面初始状态下与下平面平行，并且上表面的圆心始终固定的情况下，为了描述上表面的旋转情况，在研究此类问题时，我们引入了欧拉角的概念，它可以清楚直观的描述平台的旋转。

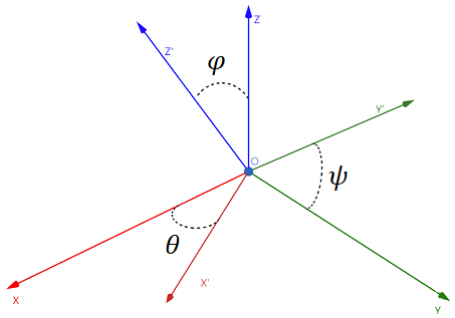

图 5.6 欧拉角描述图

如图 5.6，规定物体绕 z 轴旋转的角度为φ，绕 y 轴旋转的角度为ψ，绕 x 轴旋转的角度为θ。那么根据公式 4.2 和 4.3 就可以快速算出旋转矩阵以及点转移后的坐标。约束条件仍然与 5.1 所述情况一致。

确定了上表面点 B1，B2，B3，B4，B5，B6 的坐标之后，那么液压杆距离就仅需要使用公式 4.1 即可以计算出来。

当ψ，φ均为 0 时：(此时 d3,d4 重合，d2,d5 重合，d1,d6 重合)

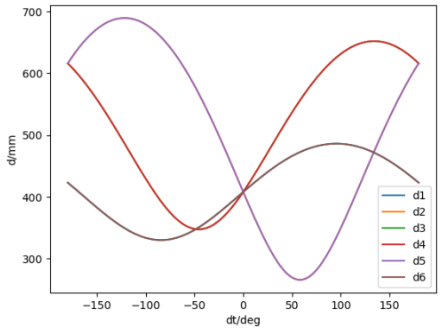

图 5.7 d 随θ变化图(无约束)

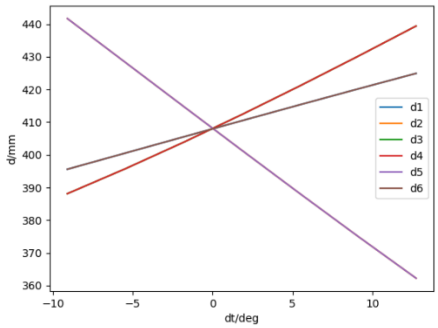

图 5.8 d 随θ变化图(有约束)

当θ，φ均为 0 时：

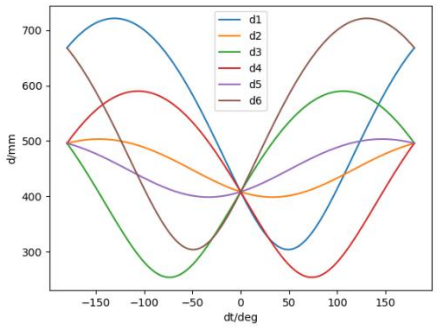

图 5.9 d 随ψ变化图(无约束)

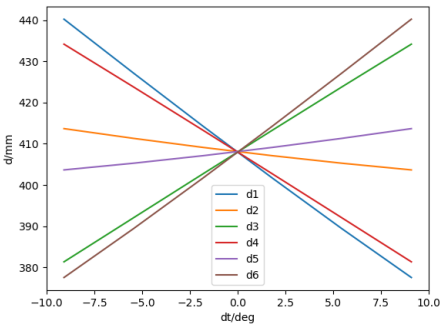

图 5.10 d 随ψ变化图(有约束)

当θ，ψ均为 0 时：(此时 d1,d3,d5 重合，d2, d4, d6 重合)

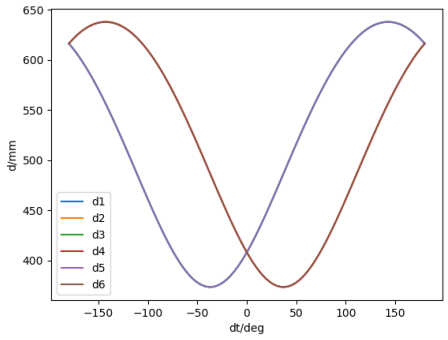

图 5.11 d 随φ变化图(无约束)

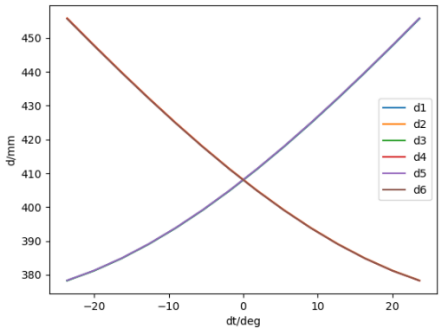

图 5.12 d 随φ变化图(有约束)

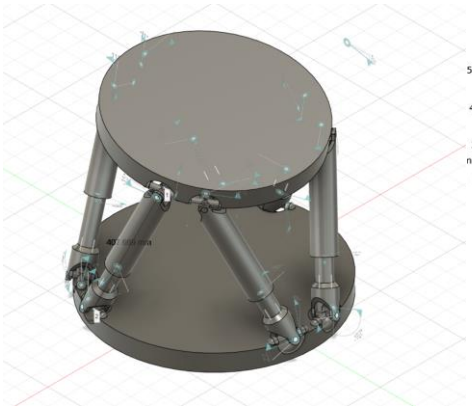

图 5.13 Fusion 360 模拟结果

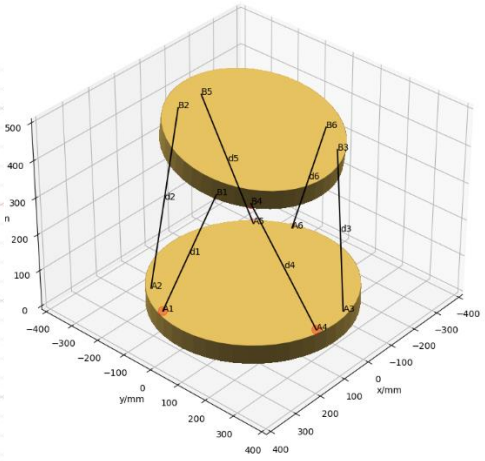

图 5.14 python 脚本模拟结果

图 5.13 和图 5.14 展示了 Fusion 360 与 python 脚本的数值模拟结果，其中ψ，φ为 0，$\theta = -10^{\circ}$。Fusion 360 测量得出 d1 的长为 393.669 毫米 python 脚本得出的数值为 394.340 毫米，基本一致。脚本源码见附录 A。

5.3 复合姿态的研究

由 5.1 与 5.2 可得，对于任意姿态，先转动，后平动即可保持转动的旋转中心为圆柱的中心，所以对于上圆盘面的液压杆端点B_i则有：

$$B_i'^{T} = RB_i^{T} + \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix}$$

其中 R 为旋转矩阵，可由公式 4.3 得到，再通过公式 4.1 出液压杆长度 d，同时应用约束条件$d \in [330, 510]$与$\alpha \in [53.13^{\circ}, 90^{\circ}]$即可知道该姿态是否能够达到以及达到时各液压杆的长度。以下是一些模拟的情况：

$$[dx, dy, dz] = [20, 30, -10], [\theta, \psi, \varphi] = [-10^{\circ}, 10^{\circ}, 10^{\circ}]$$

图像结果如图 5.15 所示，数值模拟结果如表格 5.16 所示

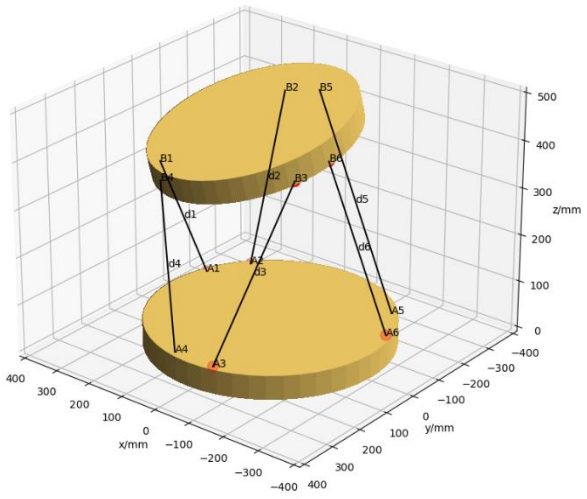

图 5.15

	长度(毫米)	下表面倾角(度)	上表面倾角(度)
d1	386.4396	53.19	61.00
d2	406.0804	74.89	64.18
d3	409.4831	62.97	48.99
d4	327.9708	67.78	63.41
d5	469.3552	60.40	64.77
d6	418.2458	67.58	80.66

表格 5.16

$$[dx, dy, dz] = [60, 40, 10], [\theta, \psi, \varphi] = [-10^{\circ}, 5^{\circ}, 8^{\circ}]$$

图像结果如图 5.17 所示，数值模拟结果如表格 5.18 所示

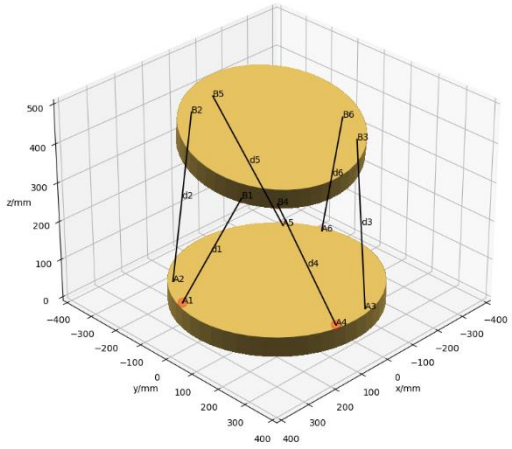

图 5.17

	长度(毫米)	下表面倾角(度)	上表面倾角(度)
d1	416.9138	56.64	66.92
d2	422.8585	79.83	73.32
d3	395.6582	68.82	57.97
d4	376.9724	63.80	60.63
d5	498.2946	58.37	59.88
d6	432.9295	63.36	74.12

表格 5.18

6 模型评价及优化

6.1 模型的优点

1，以圆盘代替上平面和下底座，以线段代替液压杆，忽略液压杆的厚度，便于分析与建模。

2，考虑了约束条件，确保了上平台不会胡乱的随意移动。

3，用点替代万向球，能够较为准确的计算出虚拟软件模拟的结果。

6.2 模型的缺点

1，仅仅假定了一组特定的值进行建模分析，并没有考虑一般情形。

2，由于先前数据的假定，导致约束条件是定值，不具一般性。

3，可能存在一些其他指标没有考虑，仅用了软件虚拟仿真，并没有真实场景应用分析过。

7 参考文献

[1] 胡璠樊.基于欧拉角与微元分析的六自由度机械平台运动模型[J].机电信息,2020,(14)

[2] 刘兴芳. 六自由度并联平台的运动学分析与结构参数优化[D]. 太原: 中北大学, 2018

附 录 A

Python 脚本

```
import numpy as np
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
import matplotlib.pyplot as plt
deg = np.pi/180

def normalize(v):
    return v/(np.sqrt(np.dot(v, v)))
def RotationMatrix(theta, phi, psi):
    '''将观测坐标系中的向量 v 转换成物体坐标系中向量 v'或者将向量(坐标)绕原点旋转.

    Notes
    -----
    此程序旋转顺序(z->y->x),内旋.
    .每种特定顺序的外旋等价于其相反顺序的内旋,反之亦然.
    .坐标系绕原点旋转的旋转矩阵与向量（坐标）绕原点旋转的旋转矩阵互为转置.
    .世界坐标系向目标坐标系旋转的旋转矩阵与目标坐标系向世界坐标系旋转的旋转矩阵互为转置.
    '''

    theta, phi, psi = theta * np.pi / 180, phi * np.pi / 180, psi * np.pi / 180

    Rz = np.mat([[np.cos(psi), np.sin(psi), 0],
                 [-np.sin(psi), np.cos(psi), 0],
                 [0, 0, 1]])

    Ry = np.mat([[np.cos(theta), 0, -np.sin(theta)],
                 [0, 1, 0],
                 [np.sin(theta), 0, np.cos(theta)]])

    Rx = np.mat([[1, 0, 0],
                 [0, np.cos(phi), np.sin(phi)],
                 [0, -np.sin(phi), np.cos(phi)]])

    return Rx * Ry * Rz
class cylinder:
    """Refer in particular to cylinder selfs.

    Attributes
    ----------

    radius: float
```

```
        The radius of the base of a cylinder
    pitch: float
        the pitch angle of the cylinder
    roll: float
        the roll angle of the cylinder
    yaw: float
        the roll angle of the cylinder
    length: float
        the length of the self
    position: list
        the position of the self, [x, y, z]

    """

    def __init__(self, radius, pitch, roll, yaw, length, position_x, position_y,
position_z, **kwargs):
        self.radius = radius
        self.pitch = pitch
        self.roll = roll
        self.yaw = yaw
        self.length = length
        self.position = [position_x, position_y, position_z]
        self.rm = np.array(RotationMatrix(self.pitch, self.roll, self.yaw))
        self.n = np.array([0, 0, 1]) @ self.rm


    def show(self, ax, color,zorder = 1):


        u = np.linspace(0, 2 * np.pi, 50)  # 把圆划分 50 等份
        h = np.linspace(-0.5, 0.5, 2)  # 把高(1m)划分两等份,对应上底和下底

        x = self.radius * np.sin(u)
        y = self.radius * np.cos(u)

        x = np.outer(x, np.ones(len(h)))  # 20*2
        y = np.outer(y, np.ones(len(h)))  # 20*2
        z = np.outer(np.ones(len(u)), h)  # 20*2
        z = z * self.length

        x_rotation = np.ones(x.shape)  # 旋转后的坐标 20*2
        y_rotation = np.ones(y.shape)
        z_rotation = np.ones(z.shape)
```

```
        for i in range(2):
            r = np.c_[x[:, i], y[:, i], z[:, i]]  # 20*3
            rT = r @ self.rm  # 20*3
            x_rotation[:, i] = rT[:, 0]
            y_rotation[:, i] = rT[:, 1]
            z_rotation[:, i] = rT[:, 2]

        ax.view_init(30, 45)
        ax.plot_surface(x_rotation + self.position[0], y_rotation +
self.position[1], z_rotation + self.position[2],
                        color=color, alpha=1, antialiased=False,zorder = zorder)

        verts = [list(zip(x_rotation[:, 0] + self.position[0], y_rotation[:, 0] +
self.position[1],
                        z_rotation[:, 0] + self.position[2]))]

        ax.add_collection3d(Poly3DCollection(verts, facecolors=color))
        verts = [list(zip(x_rotation[:, 1] + self.position[0], y_rotation[:, 1] +
self.position[1],
                        z_rotation[:, 1] + self.position[2]))]

        ax.add_collection3d(Poly3DCollection(verts, facecolors=color))

class line:
    def __init__(self, p1, p2):
        self.p1 = np.array(p1)
        self.p2 = np.array(p2)
        self.v = normalize(self.p2-self.p1)
    def show(self, ax, color = 'black', zorder = 100):
        ax.scatter(*zip(self.p1, self.p2), s = 100, color = 'r',zorder = zorder)
        ax.plot(*zip(self.p1, self.p2), color = color,zorder = zorder)
    def get_length(self):
        d = self.p2-self.p1
        return np.sqrt(np.dot(d, d))
    def get_angle(self, n):
        n1 = normalize(n)
        return np.arcsin(np.dot(n1, self.v))/deg

low = cylinder(300, 0, 0, 0, 50, 0, 0, 25)#下底面 中心坐标为[0, 0, 25] 固定不变
def draw_frame(rotate, position,show_label = False):#上顶面 初始中心坐标为[0, 0, 445]
可以改变
    up = cylinder(250, rotate[1],rotate[0],rotate[2], 50, *position)
    l = []
    a = np.array(RotationMatrix(rotate[1],rotate[0],rotate[2]))
```

```
    for t in np.arange(-np.pi/6, 2*np.pi-np.pi/6, 2*np.pi/3): #定义代表液压杆的直线
        p1 = np.array([284.93*np.cos(t+11.48*deg), 284.93*np.sin(t+11.48*deg), 50])
        p2 = np.array([234.63*np.cos(t+np.pi/3-11.38*deg), 234.63*np.sin(t+np.pi/3-
11.48*deg), -25])
        p2 = p2 @ a
        p2 += up.position


        p3 = np.array([284.93*np.cos(t-11.48*deg), 284.93*np.sin(t-11.48*deg), 50])
        p4 = np.array([234.63*np.cos(t-np.pi/3+11.38*deg), 234.63*np.sin(t-
np.pi/3+11.48*deg), -25])
        p4 = p4 @ a
        p4 += up.position

        l.append(line(p1, p2))
        l.append(line(p3, p4))
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')
    ax.set_xlim(-400, 400)
    ax.set_ylim(-400, 400)
    ax.set_zlim(0, 500)
    ax.set_xlabel("x/mm")
    ax.set_ylabel("y/mm")
    ax.set_zlabel("z/mm")
    low.show(ax,"#E7C261")
    up.show(ax,'#E7C261')
    for i in range(len(l)):
        l[i].show(ax)
        if show_label:
            ax.text(*l[i].p1, f"A{i+1}",zorder = 200)
            ax.text(*l[i].p2, f"B{i+1}", zorder = 200)
            ax.text(*((l[i].p1 + l[i].p2)/2), f"d{i+1}", zorder = 200)
        print("d%d=%.4f"%(i+1, l[i].get_length()), "t1=%.2f
t2=%.2f"%(l[i].get_angle(low.n),l[i].get_angle(up.n)))

    plt.show()

def cal_d_move():#绘制 d1 随 dx,dy 变化的图像
    xx = np.linspace(-200, 200, 100)
    yy = np.linspace(-200, 200, 100)

    def cal_z(x, y):
        up = cylinder(250, 0, 0, 0, 50, x, y, 445)
        l = []
        a = np.array(RotationMatrix(0, 0, 0))
```

```
        for t in np.arange(-np.pi/6, 2*np.pi-np.pi/6, 2*np.pi/3): #定义代表液压杆的直线
            p1 = np.array([284.93*np.cos(t+11.48*deg), 284.93*np.sin(t+11.48*deg), 50])
            p2 = np.array([234.63*np.cos(t+np.pi/3-11.38*deg), 234.63*np.sin(t+np.pi/3-11.48*deg), -25])
            p2 = p2 @ a
            p2 += up.position


            p3 = np.array([284.93*np.cos(t-11.48*deg), 284.93*np.sin(t-11.48*deg), 50])
            p4 = np.array([234.63*np.cos(t-np.pi/3+11.38*deg), 234.63*np.sin(t-np.pi/3+11.48*deg), -25])
            p4 = p4 @ a
            p4 += up.position

            l.append(line(p1, p2))
            l.append(line(p3, p4))
        d_list = np.array([i.get_length() for i in l])
        t_list = np.array([np.array([i.get_angle(low.n), i.get_angle(up.n)]) for i in l])
        #print(d_list, t_list)
        if np.all(d_list>=330) and np.all(d_list<=510) and np.all(t_list<=90) and np.all(t_list>=54):
            #return 420 #绘制顶点范围
            return d_list#绘制液压杆长度
        else:
            return np.nan
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')
    xx, yy = np.meshgrid(xx, yy)
    zz = np.zeros((100, 100, 6))
    for i in range(100):
        for j in range(100):
            x = xx[i][j]
            y = yy[i][j]
            zz[i][j] = cal_z(x, y)
    ax.set_xlabel("dx/mm")
    ax.set_ylabel("dy/mm")
    ax.set_zlabel("d/mm")
    for i in range(6):
        ax.plot_surface(xx, yy, zz[:, :,0],cmap='viridis')
    plt.show()
```

```
def valid(l_list, up:cylinder):#判断情况是否合理
    d_list = np.array([i.get_length() for i in l_list])
    t_list = np.array([np.array([i.get_angle(low.n), i.get_angle(up.n)]) for i in
l_list])
    if np.all(d_list>=330) and np.all(d_list<=510) and np.all(t_list<=90) and
np.all(t_list>=53):
        return True
    else:
        return False

def cal_d_rotate():

    def cal_z(rx,ry,rz):
        up = cylinder(250, ry, rx, rz, 0, 0, 0, 445)
        l = []
        a = np.array(RotationMatrix(ry, rx, rz))
        for t in np.arange(-np.pi/6, 2*np.pi-np.pi/6, 2*np.pi/3): #定义代表液压杆的直
线
            p1 = np.array([284.93*np.cos(t+11.48*deg), 284.93*np.sin(t+11.48*deg),
50])
            p2 = np.array([234.63*np.cos(t+np.pi/3-11.38*deg),
234.63*np.sin(t+np.pi/3-11.48*deg), -25])
            p2 = p2 @ a
            p2 += up.position


            p3 = np.array([284.93*np.cos(t-11.48*deg), 284.93*np.sin(t-11.48*deg),
50])
            p4 = np.array([234.63*np.cos(t-np.pi/3+11.38*deg), 234.63*np.sin(t-
np.pi/3+11.48*deg), -25])
            p4 = p4 @ a
            p4 += up.position

            l.append(line(p1, p2))
            l.append(line(p3, p4))
        d_list = np.array([i.get_length() for i in l])
        t_list = np.array([np.array([i.get_angle(low.n), i.get_angle(up.n)]) for i
in l])
        #print(d_list, t_list)
        if np.all(d_list>=330) and np.all(d_list<=510) and np.all(t_list<=90) and
np.all(t_list>=53):
            #return 420 #绘制顶点范围
            return d_list#绘制液压杆长度
        else:
            return np.array([np.nan for _ in range(6)])
```

```
            #return d_list
    t = np.linspace(-180 , 180, 100)
    y = []
    for i in t:
        y.append(cal_z(i, 0, 0))
    y = np.array(y)
    fig = plt.figure()
    ax = fig.add_subplot()
    ax.set_xlabel("dt/deg")
    ax.set_ylabel("d/mm")
    print(y.shape)
    print(y)
    for i in range(6):
        ax.plot(t, y[:,i],label = f"d{i+1}")
    ax.legend()
    plt.show()



#draw_frame([-10, 5, 8], np.array([0, 0, 445])+np.array([60, 40,
10]),show_label=True)
#cal_d_move()
#cal_d_rotate()
```

其中 cylinder 类用来生成一个圆柱体对象，line 类用来生成一个线对象，在初始化 cylinder 对象时需传入 cylinder 对象的半径 r，旋转参数$[\theta,\psi,\varphi]$高度 h，位置参数$[x_0+dx,y_0+dy,z_0+dz]$，参数位置如下：

$$cylinder(r,\theta,\psi,\varphi,h,x_0+dx,y_0+dy,z_0+dz)$$

在初始化 line 对象时需传入两点的坐标，参数位置如下：

$$line([p1x,p1y,p1z],[p2x,p2y,p2z])$$

draw_frame 函数用来展示某特定状态的数值模拟结果，由于已经定义了 low = cylinder(300, 0, 0, 0, 50, 0, 0, 25)，且有数值设定可知上顶面的半径为 250，高度为 50，初始中心位置为[0, 0, 420+25]以及因此在 draw_frame 函数中，需要传入的参数有上顶面旋转参数$[\theta,\psi,\varphi]$，上顶面位置参$[x_0+dx,y_0+dy,z_0+dz]$，关键字参数 show_label 用以开关标签的显示，可以设置为 True 或者 False，默认为 False。参数位置如下：

$$draw\_frame([\theta,\psi,\varphi],[x_0+dx,y_0+dy,z_0+dz],show\_label=False)$$

cal_d_move 函数用于进行平动的数值模拟，可以在约束范内绘制液压杆长度随着 dx,dy 变化的图像，通过调整返回值，也可以绘制出上顶面圆心可以覆盖的范围。

cal_d_rotate 函数用于进行旋转数值的模拟可以绘制 d 在两个旋转角给定的情况下随着第三个旋转角变化的图像，只需要将代码中的 y.append(cal_z(i, 0, 0))一句的 i 放在对应的位置上即可，其余两个参数可任意填写，对应顺序为θ,ψ,φ。