**实验一 机械臂正逆运动学**

**1. 实验目的**

1. 巩固正逆运动学基础概念。
2. 了解正逆运动学在机械臂控制中的实际用途。

**2．实验内容**

1. 机械臂模型DH参数的计算。
2. 机械臂正运动学的计算。
3. 机械臂逆运动学的计算。

**3. 实验步骤**

1. 根据机械臂图纸及初始位置建立各坐标系，获得参数表。
2. 正运动学实验验证：

编写程序计算以下三组关节角度对应的末端位姿：

第一组：

第二组：

第三组：

在仿真中驱动模型分别到达这三组关节角度，读取末端空间位姿证明正运动学解算代码的正确性。

1. 逆运动学实验验证：

编写程序计算以下两组末端位姿对应的关节角度：

第一组：

第二组：

在仿真中驱动模型分别到达这三组空间位姿，读取模型关节角姿证明逆运动学解算代码的正确性。

1. **实验过程和结果**

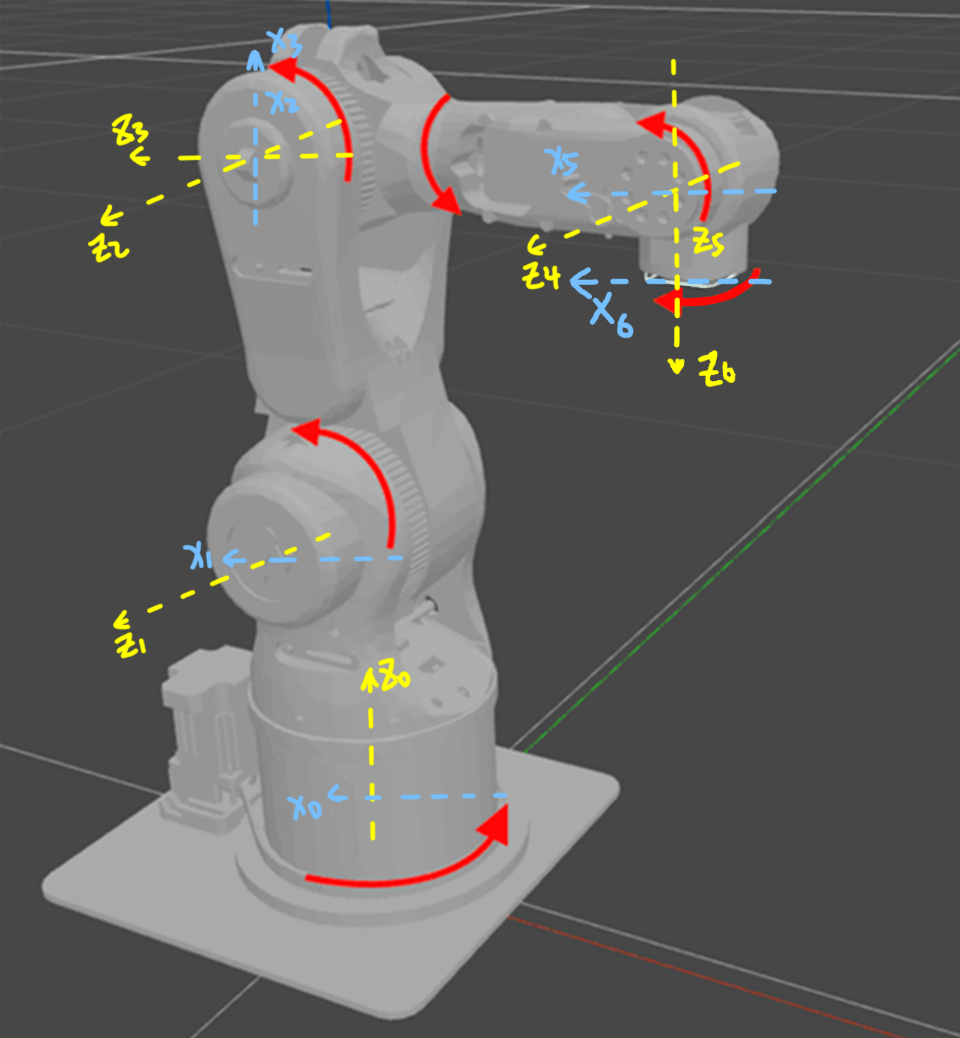
**1、正运动学实验验证：**

1. 机械臂模型系建立及参数计算：

机械臂有6个自由度，且全部为旋转关节，采用如下建系方法进行坐标系建立：

* 以各连杆后端的旋转轴为轴，采用右手准则确定轴正方向
* 轴垂直于轴和轴构成平面
* 轴、轴、轴构成右手系

最终建立如下坐标系：



**图4-1：机械臂坐标系**

之后根据坐标系获得参数

* ：轴和轴在轴方向上的距离
* ：轴绕轴旋转到轴经过的角度（以轴的**右手准则**，有正负）
* ：轴和轴在轴方向上的距离
* ：轴绕轴旋转到轴经过的角度（以轴的右手准则，有正负，此为控制该机械臂的**输入量**，存在初始角度）

根据上述原则最终获得参数表如下：

**表4-1：机械臂参数表(单位：)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| **1** | **0** |  | **284** |  |
| **2** | **225** |  | **0** |  |
| **3** | **0** |  | **0** |  |
| **4** | **0** |  | **228.9** |  |
| **5** | **0** |  | **0** |  |
| **6** | **0** |  | **55** |  |

1. 旋转矩阵和的计算：

旋转矩阵是坐标系到坐标系的旋转矩阵，即：

从坐标变换角度分析，从坐标系到坐标系需要经过如下四个步骤：

1. 平移变换：坐标系沿着轴平移，平移矩阵为：
2. 旋转变换：坐标系绕着轴旋转，旋转矩阵为：
3. 平移变换：坐标系沿着轴（或轴）平移，平移矩阵为：
4. 旋转变换：坐标系沿着轴（或轴）旋转，旋转矩阵为：

所以旋转矩阵按如下方式计算，包括两次平移和两次旋转：

从世界坐标系到坐标系的旋转矩阵按如下方式计算：

1. 机械臂末端在世界坐标系下坐标计算：

机械臂的坐标系的坐标原点恰好在机械臂末端，所以在坐标系下的坐标为：

所以在世界坐标系下的坐标按如下式计算：

通过这样的计算获得：

1. 机械臂末端的角计算：

记旋转矩阵为如下表达：

展开找到对应项计算后得：

1. 机械臂末端点位姿计算及仿真代码编写，进行实验验证坐标系、参数及旋转矩阵等的正确性：

将上述参数、理论旋转矩阵、点位姿计算公式编写成程序进行计算，输入关节角数据（），输出点位姿（）；同时将关节角数据发送给，得到机械臂点的仿真位姿，将理论位姿和仿真位姿进行对比，进行验证。详细代码见附录，仿真及理论计算结果如下：

* 设置角度为：

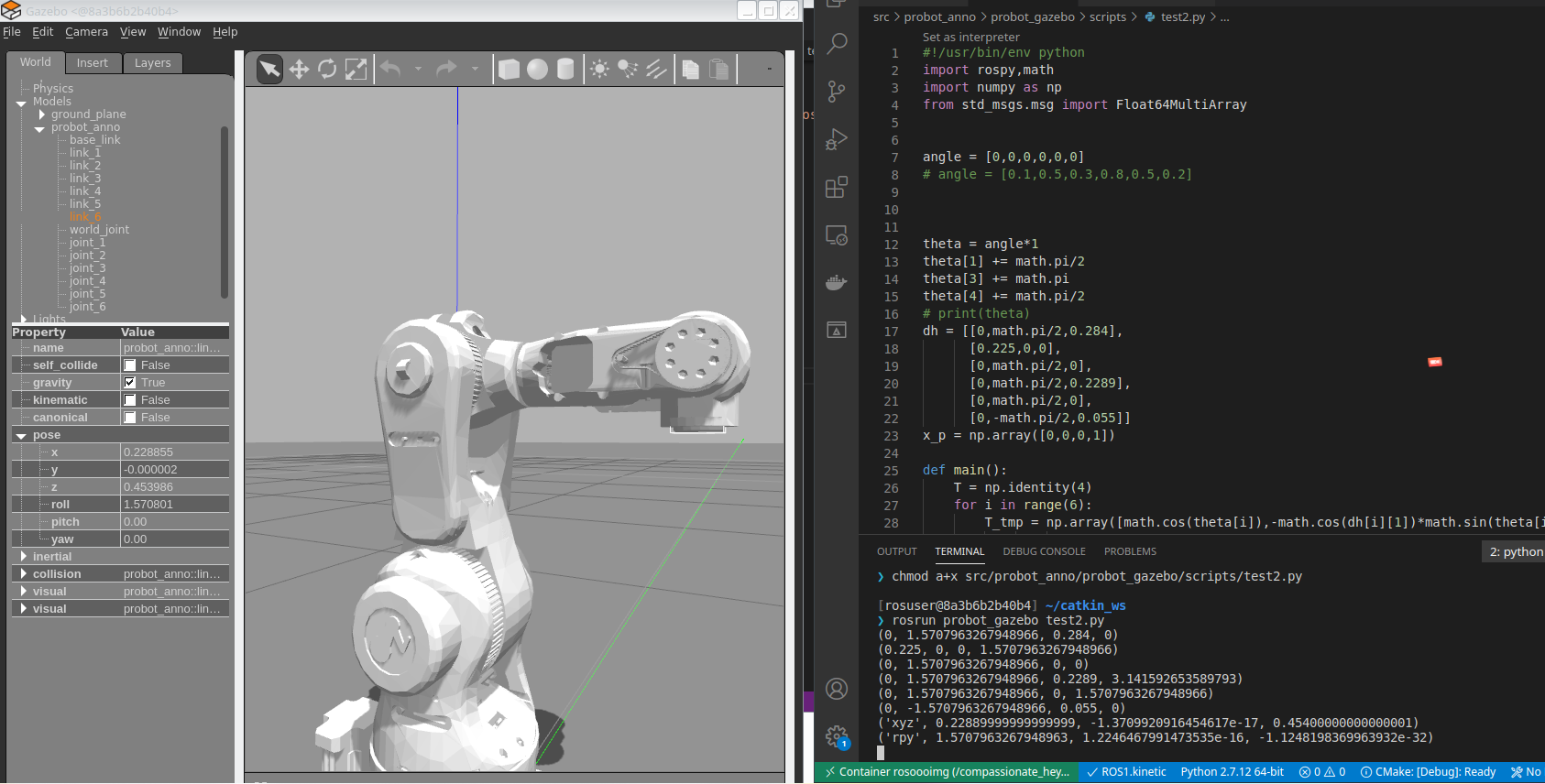
程序计算结果：

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Value | Property | Value |
|  | 0.2289 |  | 1.5708 |
|  | 0 |  | 0 |
|  | 0.4540 |  | 0 |

仿真结果：

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Value | Property | Value |
|  | 0.2289 |  | 1.5708 |
|  | 0 |  | 0 |
|  | 0.4540 |  | 0 |

运行结果展示：



* 设置角度为：

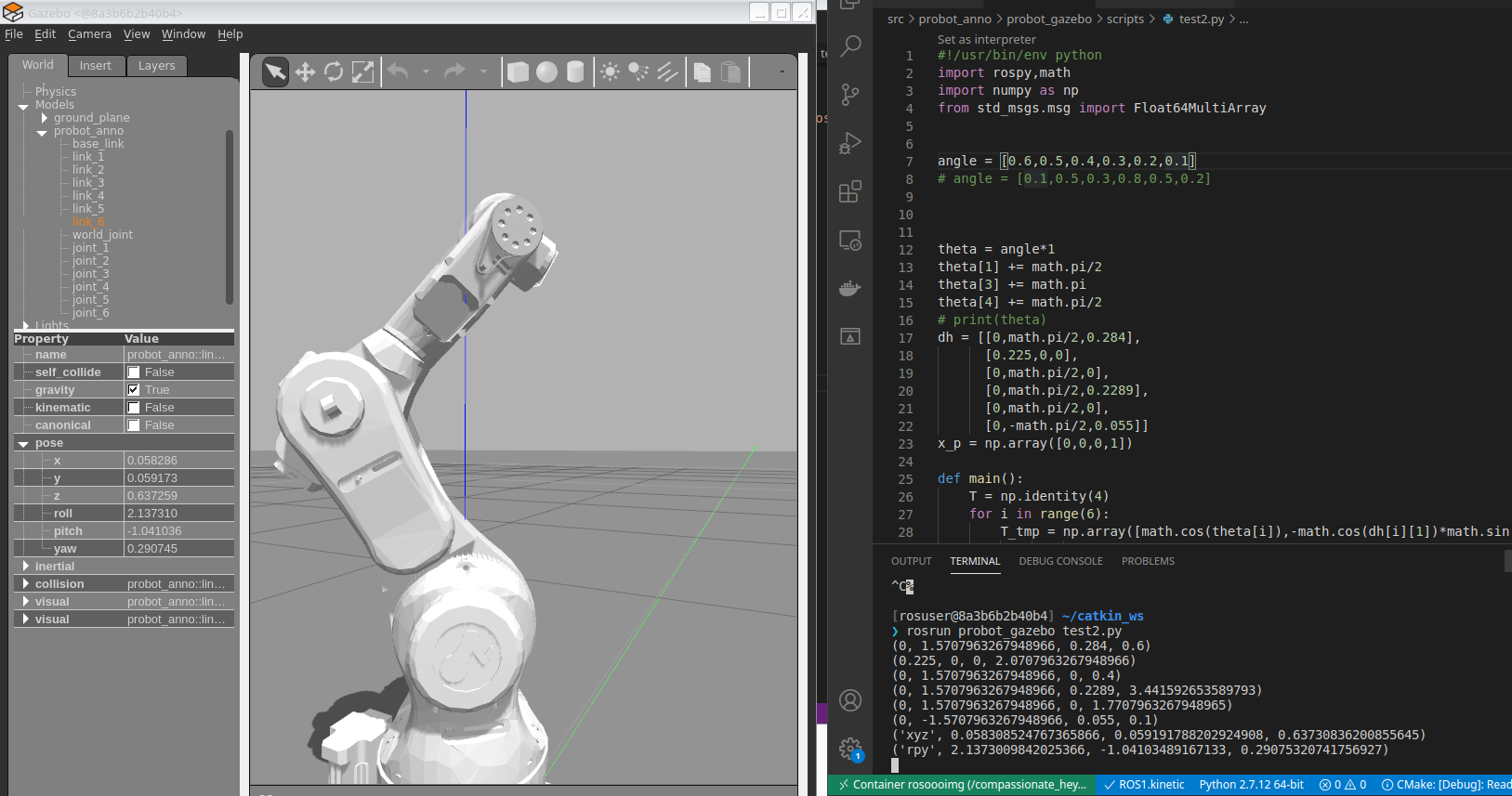
程序计算结果：

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Value | Property | Value |
|  | 0.0583 |  | 2.1373 |
|  | 0.0592 |  | -1.0410 |
|  | 0.6373 |  | 0.2908 |

仿真结果：

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Value | Property | Value |
|  | 0.0583 |  | 2.1373 |
|  | 0.0592 |  | -1.0410 |
|  | 0.6373 |  | 0.2908 |

运行结果展示：



* 设置角度为：

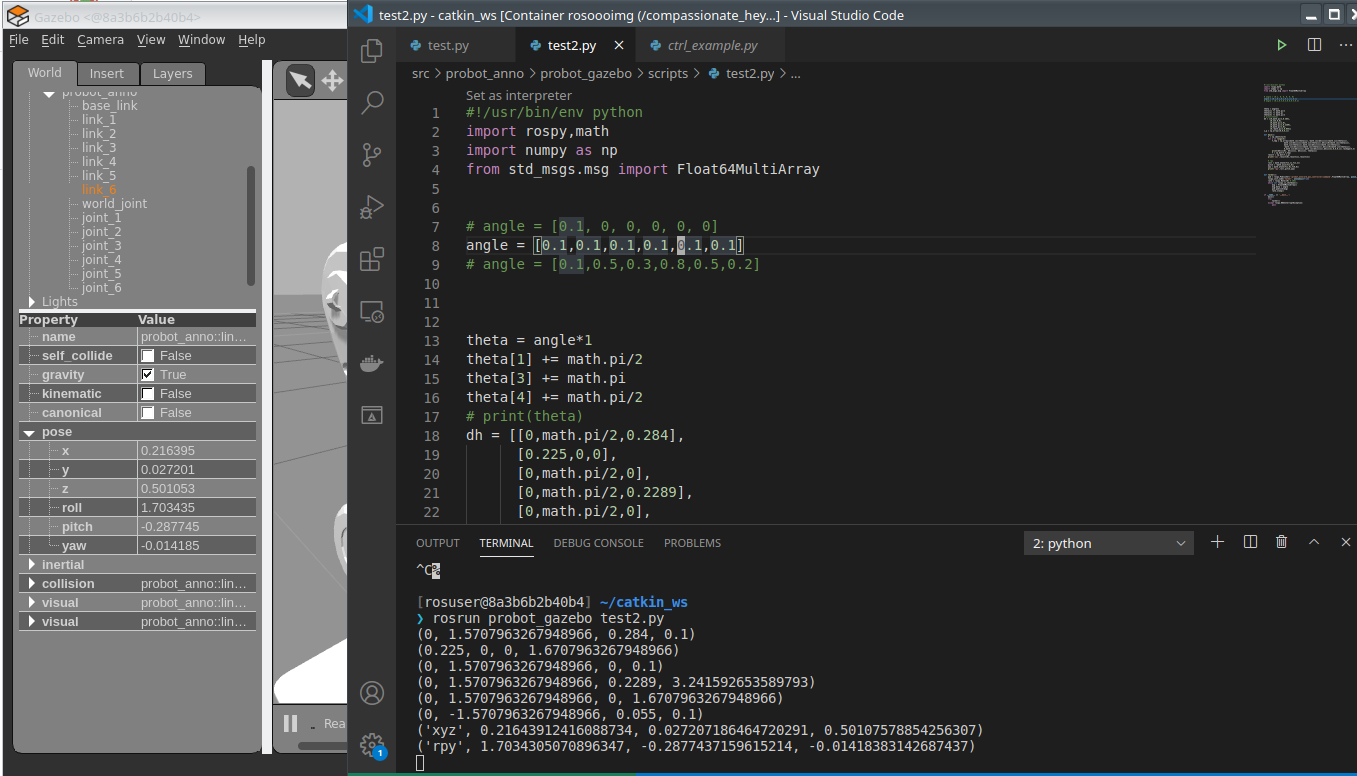
程序计算结果：

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Value | Property | Value |
|  | 0.2164 |  | 1.7034 |
|  | 0.0272 |  | -0.2877 |
|  | 0.5010 |  | -0.0141 |

仿真结果：

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Value | Property | Value |
|  | 0.2164 |  | 1.7034 |
|  | 0.0272 |  | -0.2877 |
|  | 0.5010 |  | -0.0141 |

运行结果展示：



**2、逆运动学实验验证：**

1. 各坐标系之间的位姿变换矩阵：

已知anno机械臂的DH参数表，可以首先写出各坐标系之间的位姿变换矩阵：















其中：，，，。

1. 总的位姿变换矩阵：

我们已知终点的位置(fx,fy,fz)与rpy姿态角，可以直接写出总的位姿变换矩阵：



所以其实我们所有要做的事情就是求解如下的矩阵方程：



1. 求解矩阵方程：

首先求得：





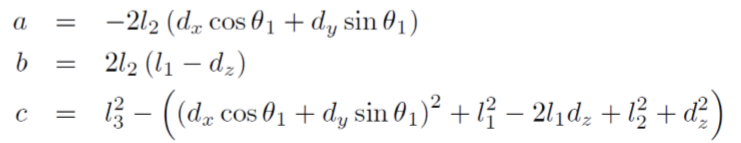


化简上式可得如下表达式：



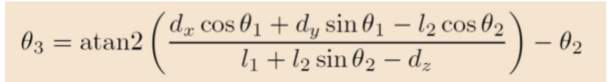
其中dx、dy、dz均已知，直接求解得到如下结论：











1. 矩阵方程的解：

已经得到的值后，我们便可求出，进而通过求出。

令，再与上面的表达式对比，简单的我们得出如下结论：



1. 注意事项：

值得注意的是，我们要考虑到反正切函数的多解型，所以我们在使用反正切函数得到一个角度时，还要将其加上180的角度考虑是否有其他的可能解。另外在求得4、5、6三个角时，我们并没有用到全部的条件，所以得到的角度可以采用另外的等式去验证。

1. 核心代码注解：

正向运动学函数，输入角度可得到末端执行器最终的位姿：

1. # forward kinematics
2. **def** forward(angle):
3. ……………………..
4. **return**(pose)

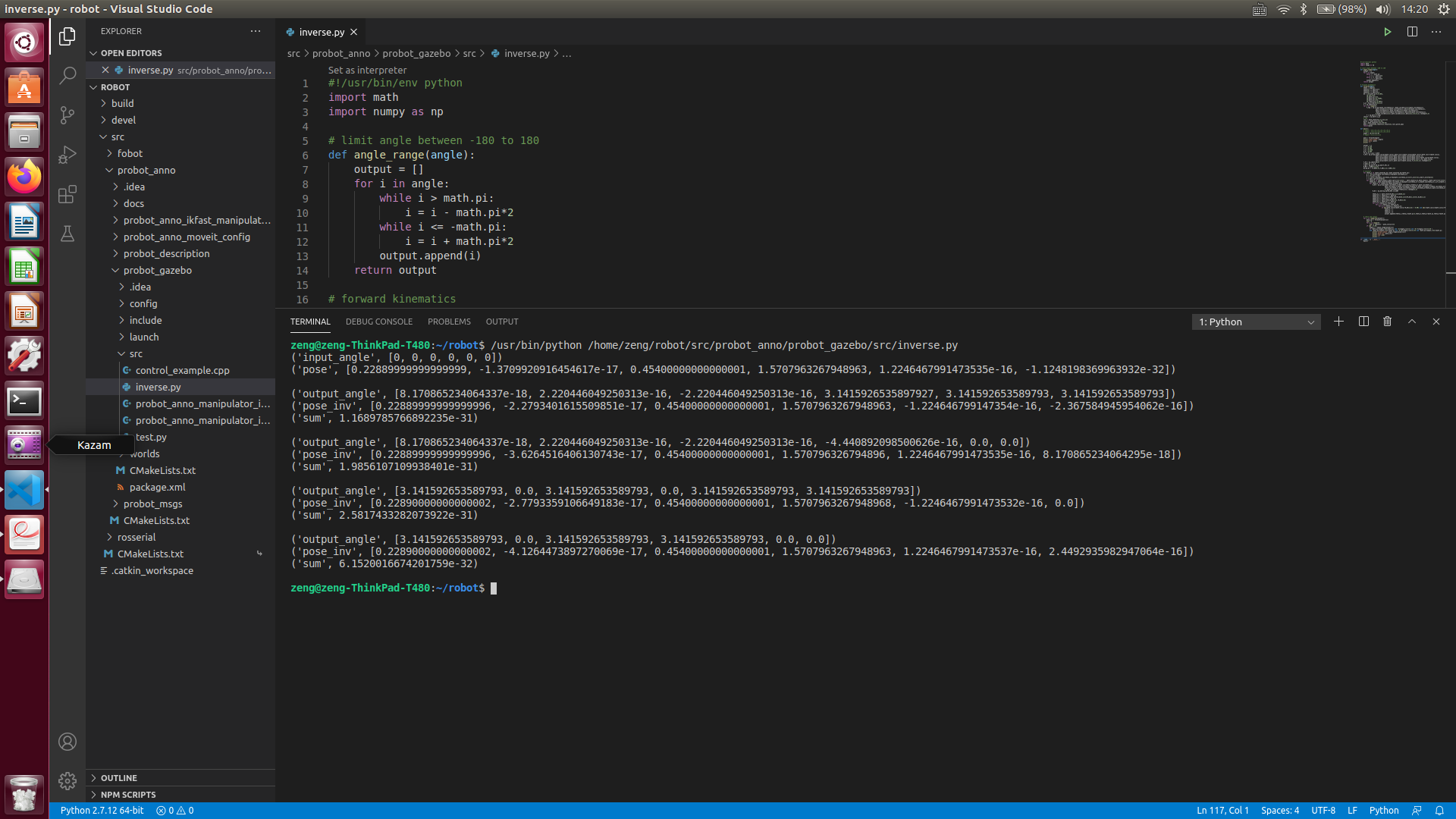
判断得到的所有角度组合是否满足条件：

1. # check and output
2. **for** i **in** range(len(answer)):
3. pose\_inv = forward(answer[i])
4. sum = 0
5. **for** j **in** range(6):
6. sum += (pose[j] - pose\_inv[j])\*\*2
7. **if** sum < 0.01:
8. angle\_r = angle\_range(answer[i])
9. **if** -math.pi<=angle\_r[0]<=math.pi **and** -2<=angle\_r[1]<=2 **and** -0.7<=angle\_r[2]<=3.84 \
10. **and** -math.pi<=angle\_r[3]<=math.pi **and** -0.79<=angle\_r[4]<=3.93 **and** -math.pi<=angle\_r[5]<=math.pi:
11. **print**('output\_angle',angle\_range(answer[i]))
12. **print**('pose\_inv',pose\_inv)
13. **print**('sum',sum)
14. **print**('')

将解出的角度再次代入正运动学模型，得到的末端位姿，将其与实际的末端位姿每一项的差求平方和，由sum表示，用来估计结果的偏差大小。另外各关节的旋转角度受到机械结构的限制，所以还要进行一下筛选，这里我们用如下函数将所有角度限制在-180到180度之间，便于比较：

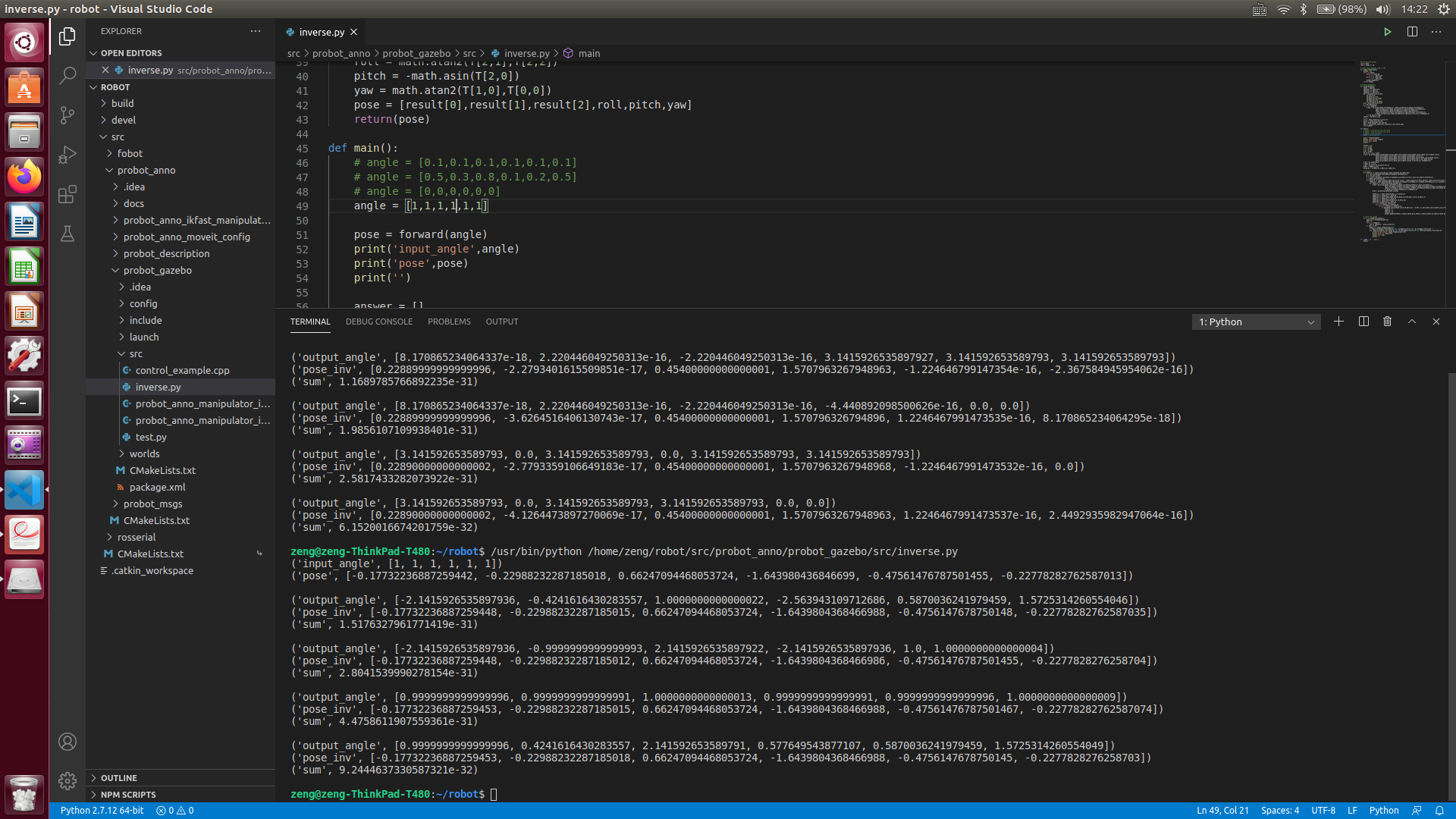
1. **def** angle\_range(angle):
2. output = []
3. **for** i **in** angle:
4. **while** i > math.pi:
5. i = i - math.pi\*2
6. **while** i <= -math.pi:
7. i = i + math.pi\*2
8. output.append(i)
9. **return** output
10. 逆运动学实验结果：

1.初始位置（各输入角度为0）：



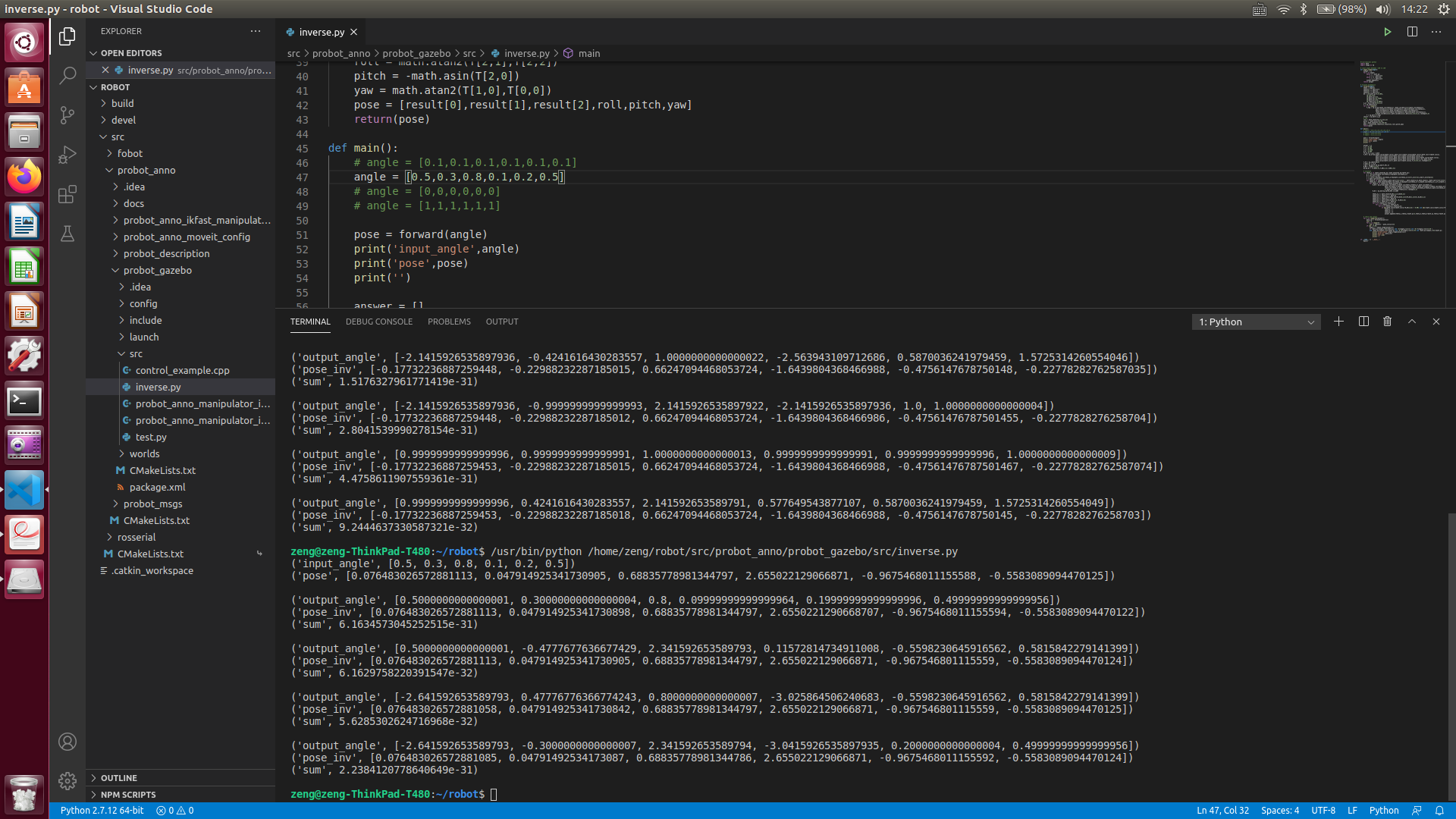
可以看到有四种角度组合能达到同样的位置。

2.输入角度[1,1,1,1,1,1]：

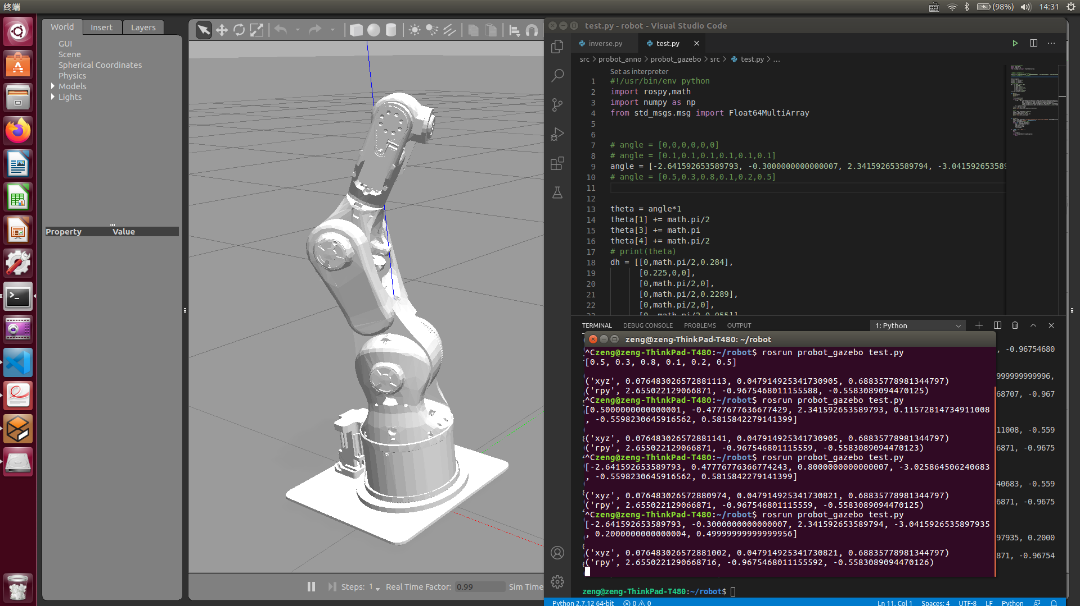
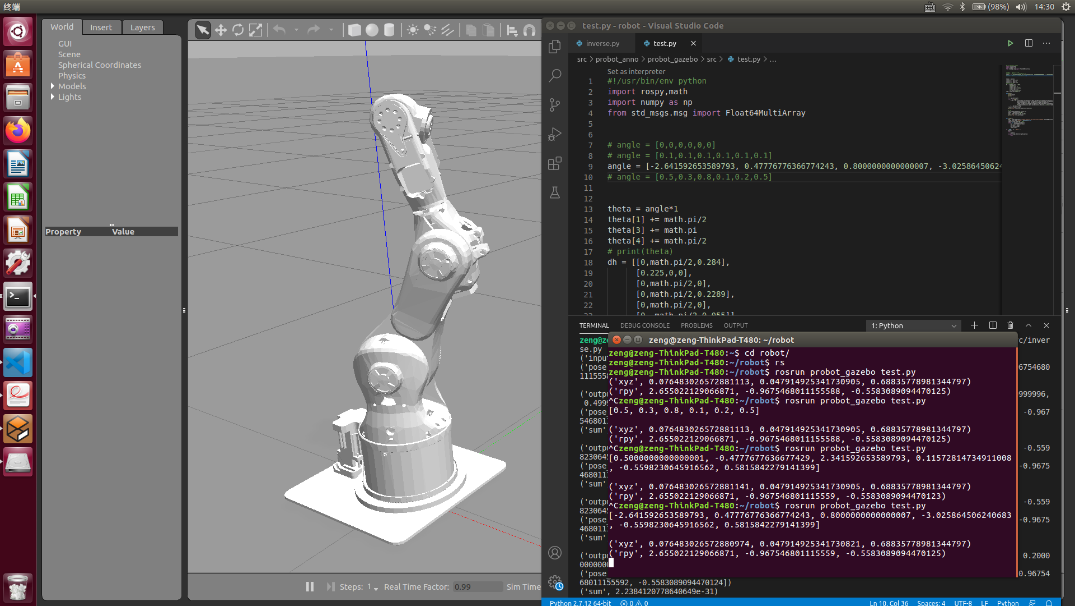
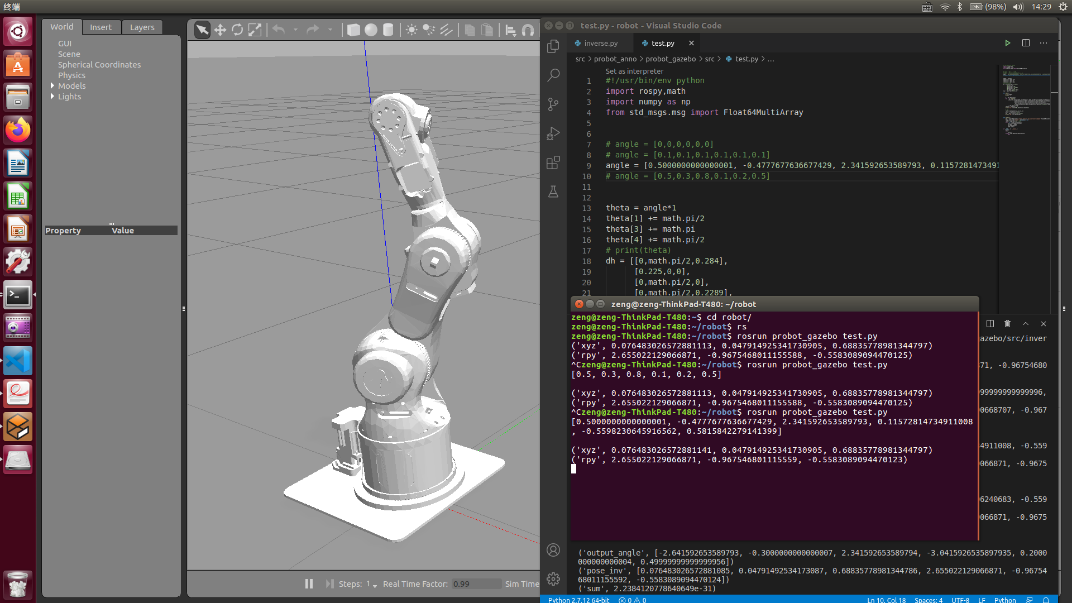
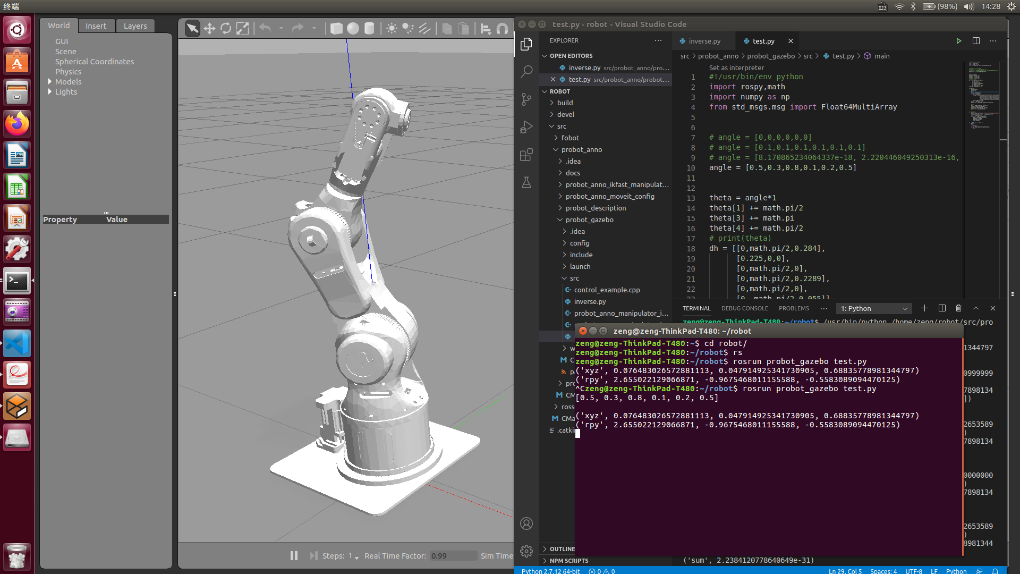


同样能得到四种角度组合达到同样的位置。

3、输入角度[0.5,0.3,0.8,0.1,0.2,0.5]：



这里我们可以在gazebo里看看是哪4种状态达到了同样的末端位置：



**5． 实验要点、心得**

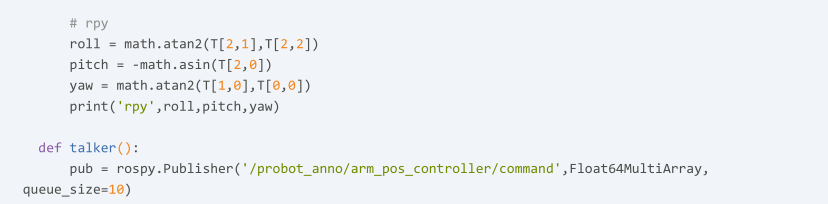
对于正运动学实验，最重要的还是根据机械臂图纸建立坐标系，然后确定参数，之后的旋转矩阵等均是根据参数进行计算，所以一旦参数发生错误，则之后的计算同样也不正确。我们组一开始遇到的问题是没有考虑到的初值，因为根据坐标系的建立，轴和轴存在初始夹角，如之类的夹角。之后同样由于此夹角的存在，之后角的计算公式也会发生一些变化，综合考虑这些因素即可。

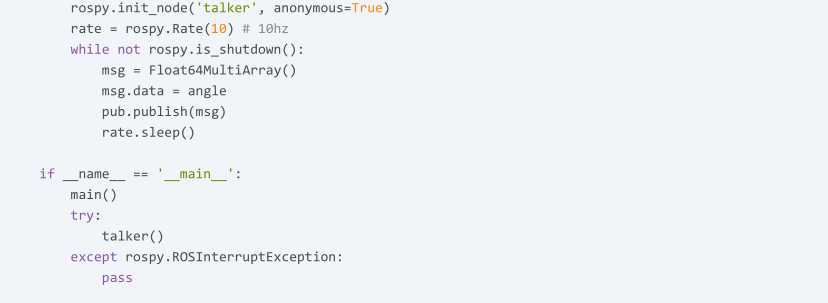
对于逆运动学实验，主要还是考察了大家的数学功底，尤其是对矩阵的处理能力。另外逆运动学与正运动学息息相关，所以我们还是要熟练掌握坐标系之间的位姿变换以及描述连杆机械臂的DH参数使用方法。

**附录**









1. #!/usr/bin/env python
2. **import** math
3. **import** numpy as np
5. # limit angle between -180 to 180
6. **def** angle\_range(angle):
7. output = []
8. **for** i **in** angle:
9. **while** i > math.pi:
10. i = i - math.pi\*2
11. **while** i <= -math.pi:
12. i = i + math.pi\*2
13. output.append(i)
14. **return** output
16. # forward kinematics
17. **def** forward(angle):
18. theta = angle\*1
19. theta[1] += math.pi/2
20. theta[3] += math.pi
21. theta[4] += math.pi/2
22. dh = [[0,math.pi/2,0.284],
23. [0.225,0,0],
24. [0,math.pi/2,0],
25. [0,math.pi/2,0.2289],
26. [0,math.pi/2,0],
27. [0,-math.pi/2,0.055]]
28. x\_p = np.array([0,0,0,1])
29. T = np.identity(4)
30. **for** i **in** range(6):
31. T\_tmp = np.array([math.cos(theta[i]),-math.cos(dh[i][1])\*math.sin(theta[i]),
32. math.sin(dh[i][1])\*math.sin(theta[i]),dh[i][0]\*math.cos(theta[i]),
33. math.sin(theta[i]),math.cos(dh[i][1])\*math.cos(theta[i]),
34. -math.sin(dh[i][1])\*math.cos(theta[i]),dh[i][0]\*math.sin(theta[i]),
35. 0,math.sin(dh[i][1]),math.cos(dh[i][1]),dh[i][2],0,0,0,1]).reshape(4,4)
36. T = np.dot(T,T\_tmp)
37. result = np.dot(T,x\_p)
38. # rpy
39. roll = math.atan2(T[2,1],T[2,2])
40. pitch = -math.asin(T[2,0])
41. yaw = math.atan2(T[1,0],T[0,0])
42. pose = [result[0],result[1],result[2],roll,pitch,yaw]
43. **return**(pose)
45. **def** main():
46. # angle = [0.1,0.1,0.1,0.1,0.1,0.1]
47. angle = [0.5,0.3,0.8,0.1,0.2,0.5]
48. # angle = [0,0,0,0,0,0]
49. # angle = [1,1,1,1,1,1]
51. pose = forward(angle)
52. **print**('input\_angle',angle)
53. **print**('pose',pose)
54. **print**('')
56. answer = []
57. l\_2 = 0.225
58. l\_1 = 0.284
59. l\_3 = 0.2289
60. d\_6 = 0.055
61. fx,fy,fz,r,p,y = pose
62. T\_70 = np.array([math.cos(p)\*math.cos(y),math.sin(r)\*math.sin(p)\*math.cos(y)-math.cos(r)\*math.sin(y),
63. math.sin(r)\*math.sin(y)+math.cos(r)\*math.sin(p)\*math.cos(y),fx,
64. math.cos(p)\*math.sin(y),math.cos(r)\*math.cos(y)+math.sin(r)\*math.sin(p)\*math.sin(y),
65. math.cos(r)\*math.sin(p)\*math.sin(y)-math.cos(y)\*math.sin(r),fy,-math.sin(p),
66. math.cos(p)\*math.sin(r),math.cos(r)\*math.cos(p),fz,0,0,0,1]).reshape(4,4)
67. T\_76 = np.identity(4)
68. T\_76[1,3] -= d\_6
69. T\_60 = np.dot(T\_70,np.mat(T\_76).I)
70. R\_60 = T\_60[0:3,0:3]
71. dx,dy,dz = [T\_60[0,3],T\_60[1,3],T\_60[2,3]]
73. # solution
74. **for** theta\_1 **in** [math.atan2(dy,dx),math.atan2(dy,dx)+math.pi]:
75. a = -2\*l\_2\*(dx\*math.cos(theta\_1)+dy\*math.sin(theta\_1))
76. b = 2\*l\_2\*(l\_1-dz)
77. c = l\_3\*\*2-((dx\*math.cos(theta\_1)+dy\*math.sin(theta\_1))\*\*2+l\_1\*\*2-2\*l\_1\*dz+l\_2\*\*2+dz\*\*2)
78. r = math.sqrt(a\*\*2+b\*\*2)
79. **for** theta\_2 **in** [math.atan2(c,math.sqrt(r\*\*2-c\*\*2)) - math.atan2(a,b),math.atan2(c,-math.sqrt(r\*\*2-c\*\*2)) - math.atan2(a,b)]:
80. theta\_3a = math.atan2(dx\*math.cos(theta\_1)+dy\*math.sin(theta\_1)-l\_2\*math.cos(theta\_2),l\_1+l\_2\*math.sin(theta\_2)-dz)-theta\_2
81. **for** theta\_3 **in** [theta\_3a,theta\_3a+math.pi]:
82. R\_30 = np.array([math.cos(theta\_1)\*math.cos(theta\_2+theta\_3),math.sin(theta\_1),
83. math.cos(theta\_1)\*math.sin(theta\_2+theta\_3),math.sin(theta\_1)\*math.cos(theta\_2+theta\_3),
84. -math.cos(theta\_1),math.sin(theta\_1)\*math.sin(theta\_2+theta\_3),math.sin(theta\_2+theta\_3),
85. 0,-math.cos(theta\_2+theta\_3)]).reshape(3,3)
86. R\_63 = np.dot(np.mat(R\_30).I,R\_60)
88. theta\_4\_1 = math.atan2(R\_63[1,1],R\_63[0,1])
89. theta\_4\_2 = theta\_4\_1 + math.pi
90. theta\_5\_1 = math.atan2(math.sqrt(R\_63[0,1]\*\*2+R\_63[1,1]\*\*2),R\_63[2,1])
91. theta\_5\_2 = theta\_5\_1 + math.pi
92. theta\_6\_1 = math.atan2(R\_63[2,2],-R\_63[2,0])
93. theta\_6\_2 = theta\_6\_1 + math.pi
94. **for** i **in** [theta\_4\_1,theta\_4\_2]:
95. **for** j **in** [theta\_5\_1,theta\_5\_2]:
96. **for** k **in** [theta\_6\_1,theta\_6\_2]:
97. **if** abs(math.cos(k)\*math.sin(j)-R\_63[2,0]) < 0.001 **and** abs(-math.cos(i)\*math.sin(j)-R\_63[0,1]) < 0.001:
98. theta\_4 = i
99. theta\_5 = j
100. theta\_6 = k
101. answer.append([theta\_1,theta\_2-math.pi/2,theta\_3,theta\_4-math.pi,theta\_5-math.pi/2,theta\_6])
103. # check and output
104. **for** i **in** range(len(answer)):
105. pose\_inv = forward(answer[i])
106. sum = 0
107. **for** j **in** range(6):
108. sum += (pose[j] - pose\_inv[j])\*\*2
109. **if** sum < 0.01:
110. angle\_r = angle\_range(answer[i])
111. **if** -math.pi<=angle\_r[0]<=math.pi **and** -2<=angle\_r[1]<=2 **and** -0.7<=angle\_r[2]<=3.84 \
112. **and** -math.pi<=angle\_r[3]<=math.pi **and** -0.79<=angle\_r[4]<=3.93 **and** -math.pi<=angle\_r[5]<=math.pi:
113. **print**('output\_angle',angle\_range(answer[i]))
114. **print**('pose\_inv',pose\_inv)
115. **print**('sum',sum)
116. **print**('')
118. **if** \_\_name\_\_ == '\_\_main\_\_':
119. main()