

## ▼ forward kinematics

input: joint variables, output: Cartesian point (n, o, a, p) and (x, y, z,  $\varphi$ ,  $\theta$ ,  $\psi$ ).

```
1 theta_input=[50,50,50,50,50,50]
```

output: Cartesian point (n, o, a, p) and (x, y, z,  $\varphi$ ,  $\theta$ ,  $\psi$ ).

```
1 from math import sin,cos,pi,radians,atan2,atan,sqrt,asin,acos
2 import numpy as np
3
4 #setting
5 d=[0,0,0.149,0.433,0,0]
6 a=[0,0.432,-0.02,0,0,0]
7 alpha=[-0.5*pi, 0, 0.5*pi, -0.5*pi, 0.5*pi, 0]
8 theta=np.radians(theta_input)
9
10 #transform matrix(frame i+1 relate to i)
11 def A(i):
12     matrix=np.array([[cos(theta[i]) , -sin(theta[i])*cos(alpha[i]) , sin(theta[i])*sin(alpha[i]) , a[i]*cos(theta[i])
13                      [sin(theta[i]) , cos(theta[i])*cos(alpha[i]) , -cos(theta[i])*sin(alpha[i]) , a[i]*sin(theta[i])
14                      [0 , sin(alpha[i]) , cos(alpha[i]) , d[i]
15                      [0 , 0 , 0 , 1
16     return matrix
17
18 #iterate to find the transform matrix(frame 6 relate to 1)
19 Cartesian_point=np.identity(4)
20 for i in range(0,6):
21     Cartesian_point=np.dot(Cartesian_point,A(i))
22
23 #Computation of the Orientation Angles and Position
24 x=Cartesian_point[0][3]
25 y=Cartesian_point[1][3]
```

```

26 z=Cartesian_point[2][3]
27 phi = atan(Cartesian_point[1][2]/Cartesian_point[0][2])+pi
28 thetaa = atan((cos(phi)*Cartesian_point[0][2] + sin(phi)*Cartesian_point[1][2])/Cartesian_point[2][2])+pi
29 psi = atan((-sin(phi)*Cartesian_point[0][0] + cos(phi)*Cartesian_point[1][0])/(-sin(phi)*Cartesian_point[0][1] + cos(phi)*Cartesian_point[1][1]))
30 output=np.array([x,y,z, phi*180/pi, thetaa*180/pi, psi*180/pi])
31
32 #print Ans
33 np.set_printoptions(precision=4)
34 print("Cartesian_point(n,o,a,p)=")
35 print(Cartesian_point)
36 print("\n (x, y, z, φ, θ, ψ)=",output)

```

```

↳ Cartesian_point(n,o,a,p)=
[[ -0.8955  0.4342 -0.0976  0.3407]
 [  0.1912  0.5734  0.7966  0.6378]
 [  0.4019  0.6947 -0.5965 -0.3864]
 [  0.      0.      0.      1.    ]]

(x, y, z, φ, θ, ψ)= [  0.3407  0.6378 -0.3864  96.9846 126.6226 120.0473]

```

## ▼ inverse kinematics

input: Cartesian point (n, o, a, p) *italicized text*

```

1 Cartesian_point=np.array([[-0.895511      ,0.43420623      , -0.09759607      ,0.34068237],
2                             [ 0.19121987      ,0.57343036      ,0.79662575      ,0.63781229],
3                             [ 0.40186441      ,0.69472482      , -0.59654205      , -0.38642471],
4                             [ 0.      ,0.      ,0.      ,1.      ]])

```

output: the corresponding joint variables.

```

1 from math import sin,cos,pi,radians,atan2,sqrt,asin,acos
2 import numpy as np
3

```

```

4 #restrict of the corresponding_joint_variables
5 def check_output(var_array):
6     var_range=[160,125,135,140,100,260]
7     for i in range(var_array.size):
8         if var_array[i]>var_range[i] or var_array[i]<var_range[i]:
9             print('!!! 0',i+1,' is out of range !!!')
10
11
12
13
14
15
16
17
18 f11=Cartesian_point[0][0]
19 f12=Cartesian_point[0][1]
20 f13=Cartesian_point[0][2]
21 px=Cartesian_point[0][3]
22
23 f21=Cartesian_point[1][0]
24 f22=Cartesian_point[1][1]
25 f23=Cartesian_point[1][2]
26 py=Cartesian_point[1][3]
27
28 f31=Cartesian_point[2][0]
29 f32=Cartesian_point[2][1]
30 f33=Cartesian_point[2][2]
31 pz=Cartesian_point[2][3]
32
33 # 8 solutions totally
34 for c1 in range(2):#2 solutions of theta1
35     for c3 in range(2):#2 solutions of theta3
36         for c5 in range(2):#2 solutions of theta5
37             d3=d[3-1]
38             if c1==0:
39                 ctal=atan2(py,px)-atan2(d3, sqrt(px**2+py**2-d3**2))
40             else:
41                 ctal=atan2(py,px)-atan2(d3, -sqrt(px**2+py**2-d3**2))

```

```

41     cta1=atan2(py,px)-atan2(d3,-sqrt(px**2+py**2-d3**2))
42
43
44     a2=a[2-1]
45     a3=a[3-1]
46     d3=d[3-1]
47     d4=d[4-1]
48     M=(px**2+py**2+pz**2-a2**2-a3**2-d3**2-d4**2)/(2*a2)
49     if c3==0:
50         cta3=atan2(M,sqrt(a3**2+d4**2-M**2))-atan2(a3,d4)
51     else:
52         cta3=atan2(M,-sqrt(a3**2+d4**2-M**2))-atan2(a3,d4)
53
54
55
56     TMP=np.array([[cos(cta1)*px+sin(cta1)*py,-pz],[pz,cos(cta1)*px+sin(cta1)*py]])
57     TMP_inv=np.linalg.inv(TMP)
58     matrix23=np.dot(TMP_inv, np.array([[a3+a2*cos(cta3)],[d4+a2*sin(cta3)]]))
59     cta23=atan2(matrix23[1],matrix23[0])
60     cta2=cta23-cta3
61
62
63
64     if c5==0:
65         cta5=acos( cos(cta1)*sin(cta23)*f13+sin(cta1)*sin(cta23)*f23+cos(cta23)*f33 )
66     else:
67         cta5=-acos(cos(cta1)*sin(cta23)*f13+sin(cta1)*sin(cta23)*f23+cos(cta23)*f33)
68
69
70
71     c4s5 = cos(cta1)*cos(cta23)*f13 + sin(cta1)*cos(cta23)*f23 - sin(cta23)*f33;
72     s4s5 = -sin(cta1)*f13 + cos(cta1)*f23;
73     s5c6 = -1*(cos(cta1)*sin(cta23)*f11+sin(cta1)*sin(cta23)*f21+cos(cta23)*f31);
74     s5s6 = (cos(cta1)*sin(cta23)*f12+sin(cta1)*sin(cta23)*f22+cos(cta23)*f32);
75     cta4 = atan2(s4s5, c4s5);
76     cta6 = atan2(s5s6, s5c6);
77     if cta5<0:
78         if cta4>=0:cta4=cta4-pi

```

```

79     else :cta4=cta4+pi
80     if cta6>=0:cta6=cta6-pi
81     else :cta6=cta6+pi
82
83
84
85
86
87     corresponding_joint_variables=np.array([cta1,cta2,cta3,cta4,cta5,cta6])*180/pi
88     np.set_printoptions(precision=4)
89     print("corresponding_joint_variables=",corresponding_joint_variables)
90     check_output(corresponding_joint_variables)
91     print('-----')

```

```

corresponding_joint_variables= [50. 50. 50. 50. 50. 50.]

```

```

-----
corresponding_joint_variables= [ 50.   50.   50. -130.  -50. -130.]

```

```

-----
corresponding_joint_variables= [ 50.          7.2799 135.2892  97.0136  36.2456 -11.2196]
!!!  θ 3   is out od range  !!!

```

```

-----
corresponding_joint_variables= [ 50.          7.2799 135.2892 -82.9864 -36.2456 168.7804]
!!!  θ 3   is out od range  !!!

```

```

-----
corresponding_joint_variables= [-106.2171 -187.2799  50.          -66.5364  20.1631  -9.1114]
!!!  θ 2   is out od range  !!!

```

```

-----
corresponding_joint_variables= [-106.2171 -187.2799  50.          113.4636 -20.1631  170.8886]
!!!  θ 2   is out od range  !!!

```

```

-----
corresponding_joint_variables= [-106.2171 -230.          135.2892 -149.3667  38.3558  80.7952]
!!!  θ 2   is out od range  !!!
!!!  θ 3   is out od range  !!!
!!!  θ 4   is out od range  !!!

```

```

-----
corresponding_joint_variables= [-106.2171 -230.          135.2892  30.6333 -38.3558 -99.2048]
!!!  θ 2   is out od range  !!!
!!!  θ 3   is out od range  !!!
-----

```

