

---

---

---

---

---

---



# Homework - 04

## Trajectory Generation

Task to draw a circle of radius 10cm within 5 seconds

Using the inverse velocity kinematics approach

(Inverse Jacobians)

Consider Joint 3 to be locked  $\theta_3 = 0$

$$\boldsymbol{\dot{q}}(\text{at } t=0) = [v_1, v_2, v_4, v_5, v_6, v_7]^T$$

$$= [0.0, 0.0, \pi/2, 0.0, \pi/2, 0.0]^T$$

Step 1 setting up the frames

# HOMEWORK - 4

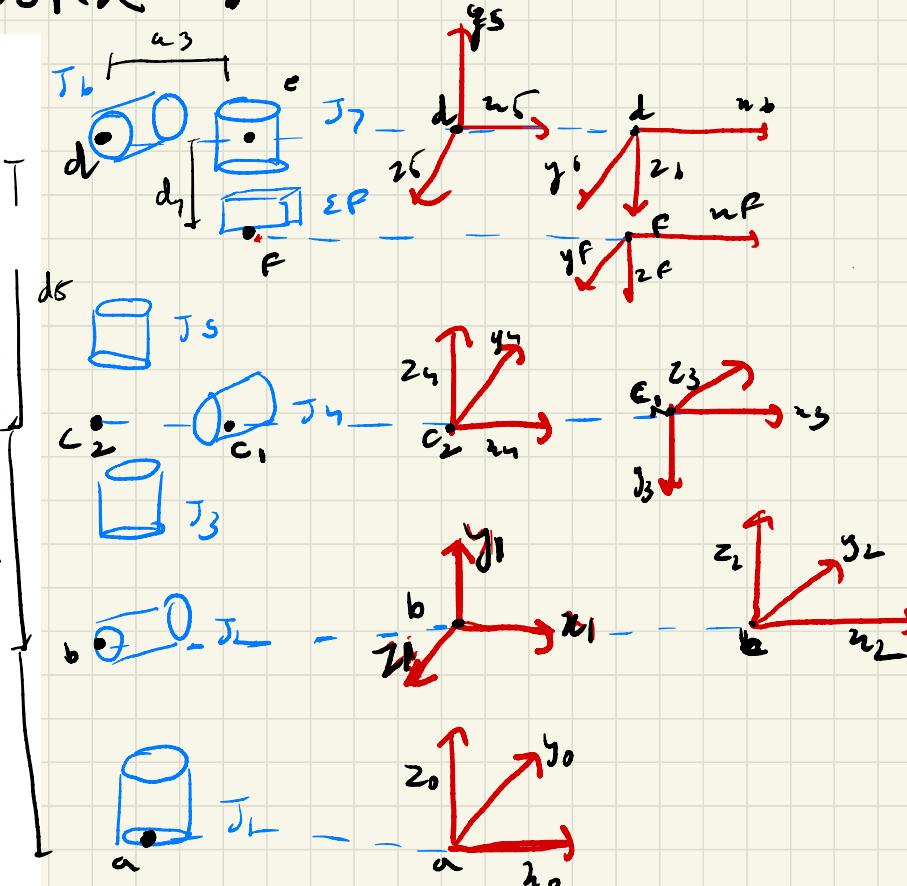
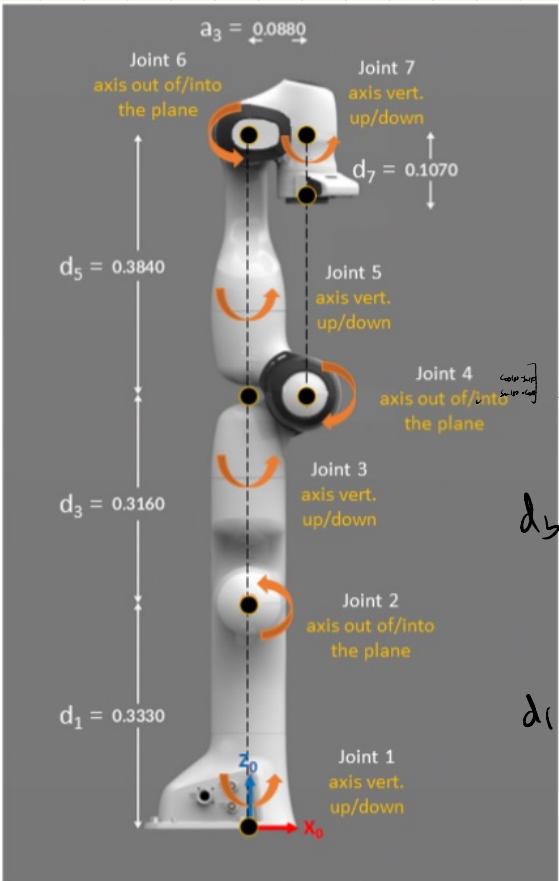


Fig. 2: Panda Robot - Home configuration

(Orange arrows indicate the positive sense of joint rotation)

DH Table from Selected from  
Spong

.	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\Theta_1$	$d_1$	0	$\pi/2$
2	$\Theta_2$	0	0	$-\pi/2$
3	0	$d_3$	$-a_3$	$\pi/2$
4	$\Theta_4$	0	$-a_3$	$\pi/2$
5	$\Theta_5$	$d_5$	0	$\pi/2$
6	$\Theta_6$	0	$a_3$	$\pi/2$
7	$\Theta_7$	$d_7$	0	0

\*  $\Theta_3 = 0$  rest all is same

$$z_i \frac{1}{n} z_{i-1}$$

D-H parameters

$d_i$ :  $n_i$  distance from  $n_{i-1}$  along  $z_{i-1}$

$\Theta_i$ :  $n_i$  angle from  $n_{i-1}$  around  $z_{i-1}$

$a_i$ :  $z_i$  distance from  $z_{i-1}$  along  $n_i$

$\alpha_i$ :  $z_i$  angle from  $z_{i-1}$  around  $n_i$

=> This is computed in Jupiter notebook

=> The 5 geometric configurations are also made in the code

=> Even  $T_i$  is also been printed in it.

To add the end effector pen length odd 10cm to  $d_7$

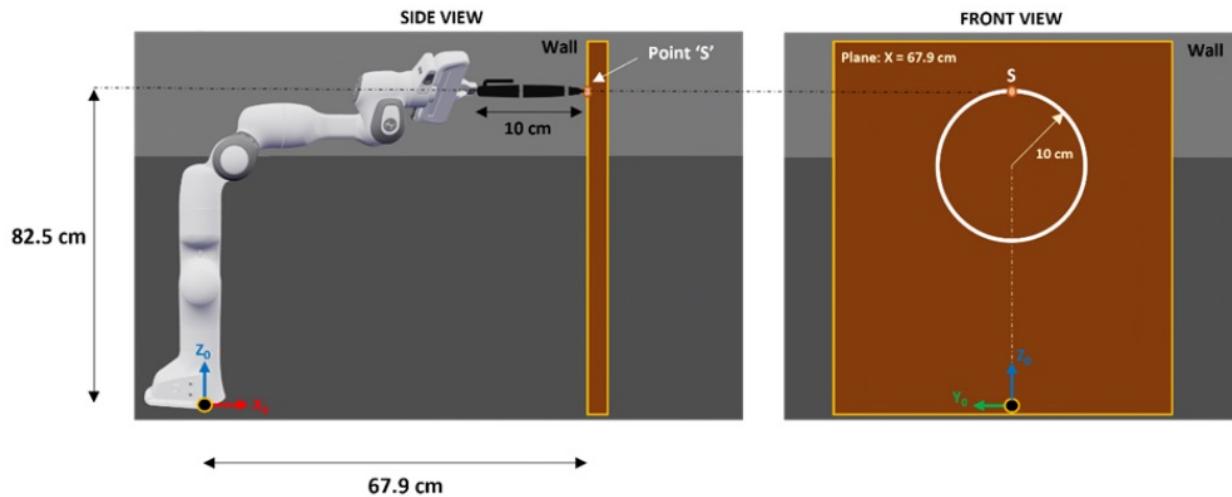
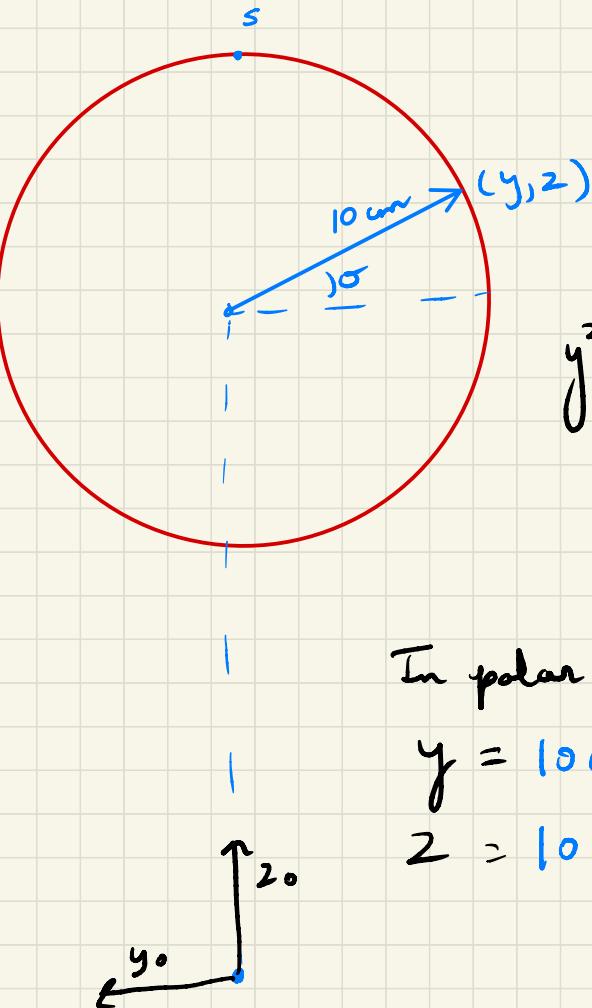


Fig. 1: Panda Robot with the pen (Side and Front Views)



plane  $n = 67.9\text{cm}$   
wall

eqn of circle

$$y^2 + (z - 72.5)^2 = 10^2$$

in cartesian form  
with respect to origin  
frame

In polar coordinates

$$y = 10 \cos(\theta)$$

$$z = 10 \sin(\theta)$$

$$y = 10 \cos(\theta)$$

$$z = 10 \sin(\theta) + 72.5$$

$$v_y = \dot{y} = -10 \sin(\theta) \dot{\theta}$$

$$v_z = \dot{z} = 10 \cos(\theta) \dot{\theta}$$

$$\dot{\theta} = \frac{2\pi}{5} \text{ as time } 5 \text{ secs}$$

$$\theta = \theta(t) = \frac{2\pi}{5} t$$

As our starting point is at 8

$$\theta = \frac{\pi}{2} + \theta$$

$$\dot{x} = \begin{bmatrix} 0 \\ -10 \sin\left(\frac{2\pi}{5}t\right) \frac{2\pi}{5} \\ 10 \cos\left(\frac{2\pi}{5}t\right) \frac{2\pi}{5} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\dot{x} = \begin{bmatrix} 0 \\ -10 \cos\left(\frac{2\pi}{5}t\right) \frac{2\pi}{5} \\ -10 \sin\left(\frac{2\pi}{5}t\right) \frac{2\pi}{5} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

To find the Jacobian :

Step 1 Calculate  $J_i^T$

write J

Step 2 Calculate  $Z_i$

Step 3 Calculate  $h(q_1, q_2, \dots, q_n)$

Step 4 Calculate  $\partial h / \partial q_i$

$$T_0' = A_1$$

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & \sin\omega_i \cos\theta_i & \sin\omega_i \sin\theta_i \\ \sin\theta_i & \cos\theta_i & -\sin\omega_i \cos\theta_i & \sin\omega_i \sin\theta_i \\ 0 & \sin\omega_i & \cos\omega_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_0'' = A_1 A_2 A_3 A_4$$

$$T_0''' = A_1 A_2 A_3 A_4 A_5$$

$$T_0'''' = A_1 A_2 A_3 A_4 A_5 A_6$$

$$T_0''''' = A_1 A_2 A_3 A_4 A_5 A_6 A_7$$

$$h(q_1, q_2, \dots, q_7) = X_p(q_1, q_2, \dots, q_7) = \begin{bmatrix} u_p \\ y_p \\ z_p \end{bmatrix}$$

$${}^0_T = \begin{bmatrix} {}^0 R_n & {}^0 p \\ - & - \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

p: origin of the end effector coordinates

$u_p, y_p, z_p$  are end effector coordinates

${}^oZ_i$  is the 3rd column of the  $T_i^o$  matrix for  
 $i = 0, 1, 2, 4, 5, 6, 7$

$${}^oJ = \begin{bmatrix} \frac{d{}^oX_p}{dq_1} & \frac{d{}^oX_p}{dq_2} & \frac{d{}^oX_p}{dq_4} & \frac{d{}^oX_p}{dq_5} & \frac{d{}^oX_p}{dq_6} & \frac{d{}^oX_p}{dq_7} \\ {}^oZ_1 & {}^oZ_2 & {}^oZ_4 & {}^oZ_5 & {}^oZ_6 & {}^oZ_7 \end{bmatrix}$$

At  $t=0$

$$q\text{-current} = \begin{bmatrix} 0 \\ 0 \\ \rho_0/L \\ 0 \\ \rho L \\ 0 \end{bmatrix}$$

$$q\text{-dot} = J^{-1} \times \dot{X}$$

for integration

$\Delta t = \frac{T}{N} \rightarrow$  total time  
 $N \rightarrow$  points to be plotted

$$q\text{-current} = q\text{-current} + q\text{-dot} \Delta t$$

Use this q-current to get the end effector position

$x_p$ .

This process will be continue  $\textcircled{N}$  number of times

& then  $x_p$  can be used to plot the trajectory.

~~X~~

Please refer the codes & tool with the explanation  
for better understanding of the process

# JACOBIAN MATRIX

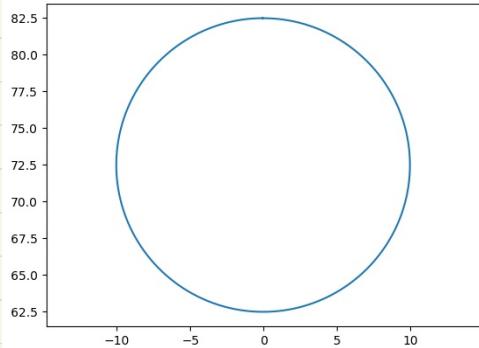
```
In [67]: q_current=Matrix(([0],  
[0],  
[pi/2],  
[0],  
[pi],  
[0]))  
J_initial=J.subs({theta_1:q_current[0],  
theta_2:q_current[1],  
theta_4:q_current[2],  
theta_5:q_current[3],  
theta_6:q_current[4],  
theta_7:q_current[5]})  
J_initial
```

```
Out[67]: 
$$\begin{bmatrix} 0 & -49.2 & 17.6 & 0 & -8.8 & 0 \\ 67.9 & 0 & 0 & -8.8 & 0 & 0 \\ 0 & 67.9 & -59.1 & 0 & 20.7 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

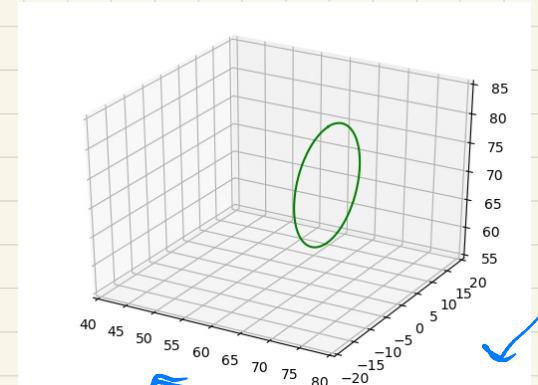
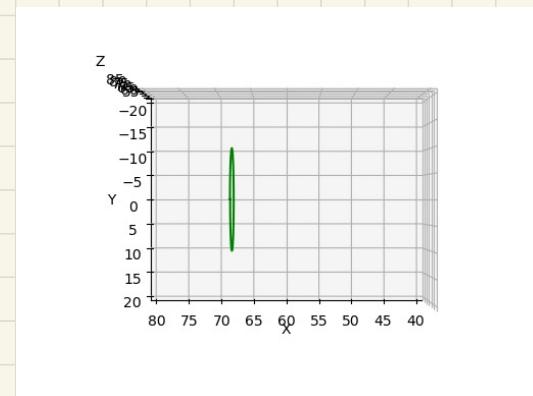
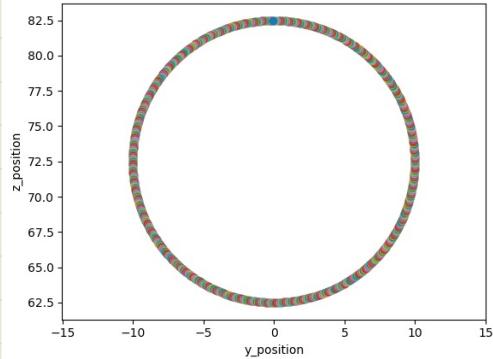
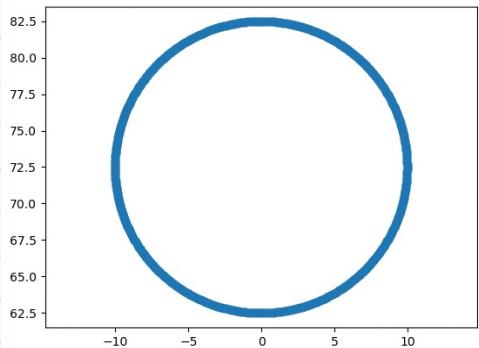
```

# PLOTS

Z  
↑



Z  
↑



x

# HOMEWORK-4

```
In [1]: #sympy library used to express matrices in terms of symbols  
# matplotlib lib used to plot the graph
```

```
from sympy import *  
from IPython.display import Image,display,HTML  
import matplotlib.pyplot as plt  
from matplotlib import cm  
import numpy as np  
  
from mpl_toolkits.mplot3d.axes3d import get_test_data  
init_printing()  
  
thetadot,t=symbols('thetadot,t')  
theta,d,a,alpha=symbols('theta,d,a,alpha')  
d_1,d_3,d_5,d_7,a_3=symbols('d_1,d_3,d_5,d_7,a_3')  
theta_1,theta_2,theta_3,theta_4,theta_5,theta_6,theta_7=symbols('theta_1,theta_2,theta_3  
Rot_z_theta=Matrix([[cos(theta),-sin(theta),0,0],  
[sin(theta),cos(theta),0,0],  
[0,0,1,0],  
[0,0,0,1]])  
Trans_z_d=Matrix([[1,0,0,0],  
[0,1,0,0],  
[0,0,1,d],  
[0,0,0,1]])  
Trans_x_a=Matrix([[1,0,0,a],  
[0,1,0,0],  
[0,0,1,0],  
[0,0,0,1]])  
Rot_x_alpha=Matrix([[1,0,0,0],  
[0,cos(alpha),-sin(alpha),0],  
[0,sin(alpha),cos(alpha),0],  
[0,0,0,1]])
```

```
In [2]: # Transformation matrix for each frame  
A=Rot_z_theta*Trans_z_d*Trans_x_a*Rot_x_alpha
```

```
In [3]: # Substituting values form DH table  
A_1=A.subs({theta:theta_1,d:d_1,a:0,alpha:pi/2})  
A_2=A.subs({theta:theta_2,d:0,a:0,alpha:-pi/2})  
A_3=A.subs({theta:0,d:d_3,a:a_3,alpha:-pi/2})  
A_4=A.subs({theta:theta_4,d:0,a:-a_3,alpha:pi/2})  
A_5=A.subs({theta:theta_5,d:d_5,a:0,alpha:pi/2})  
A_6=A.subs({theta:theta_6,d:0,a:a_3,alpha:pi/2})  
A_7=A.subs({theta:theta_7,d:d_7,a:0,alpha:0})
```

```
In [4]: # Transformation matrix for each frame with respect to origin
```

```
Transformation=A_1*A_2*A_3*A_4*A_5*A_6*A_7  
Transformation  
T_1=A_1  
T_2=A_1*A_2  
T_4=A_1*A_2*A_3*A_4  
T_5=A_1*A_2*A_3*A_4*A_5  
T_6=A_1*A_2*A_3*A_4*A_5*A_6  
T_7=A_1*A_2*A_3*A_4*A_5*A_6*A_7
```

```
In [5]: # 1st frame wrt origin
```

```
T_1
```

$$\text{Out[5]: } \begin{bmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & -\cos(\theta_1) & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
In [6]: # 2nd frame wrt origin
```

```
T_2
```

$$\text{Out[6]: } \begin{bmatrix} \cos(\theta_1)\cos(\theta_2) & -\sin(\theta_1) & -\sin(\theta_2)\cos(\theta_1) & 0 \\ \sin(\theta_1)\cos(\theta_2) & \cos(\theta_1) & -\sin(\theta_1)\sin(\theta_2) & 0 \\ \sin(\theta_2) & 0 & \cos(\theta_2) & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
In [7]: # 4th framw frame wrt origin
```

```
T_4
```

$$\text{Out[7]: } \begin{bmatrix} \sin(\theta_2)\sin(\theta_4)\cos(\theta_1) + \cos(\theta_1)\cos(\theta_2)\cos(\theta_4) & -\sin(\theta_1) & -\sin(\theta_2)\cos(\theta_1)\cos(\theta_4) + \sin(\theta_1)\sin(\theta_2)\sin(\theta_4) & 0 \\ \sin(\theta_1)\sin(\theta_2)\sin(\theta_4) + \sin(\theta_1)\cos(\theta_2)\cos(\theta_4) & \cos(\theta_1) & -\sin(\theta_1)\sin(\theta_2)\cos(\theta_4) + \sin(\theta_1)\cos(\theta_2)\sin(\theta_4) & 0 \\ \sin(\theta_2)\cos(\theta_4) - \sin(\theta_4)\cos(\theta_2) & 0 & \sin(\theta_2)\sin(\theta_4) + \cos(\theta_2)\cos(\theta_4) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
In [8]: # 5th frame wrt origin
```

```
T_5
```

$$\text{Out[8]: } \begin{bmatrix} (\sin(\theta_2)\sin(\theta_4)\cos(\theta_1) + \cos(\theta_1)\cos(\theta_2)\cos(\theta_4))\cos(\theta_5) - \sin(\theta_1)\sin(\theta_5) & -\sin(\theta_2)\cos(\theta_1)\cos(\theta_4)\cos(\theta_5) - \sin(\theta_1)\sin(\theta_2)\sin(\theta_4)\cos(\theta_5) & 0 & 0 \\ (\sin(\theta_1)\sin(\theta_2)\sin(\theta_4) + \sin(\theta_1)\cos(\theta_2)\cos(\theta_4))\cos(\theta_5) + \sin(\theta_5)\cos(\theta_1) & -\sin(\theta_1)\sin(\theta_2)\cos(\theta_4)\cos(\theta_5) - \sin(\theta_1)\cos(\theta_2)\sin(\theta_4)\cos(\theta_5) & 0 & 0 \\ (\sin(\theta_2)\cos(\theta_4) - \sin(\theta_4)\cos(\theta_2))\cos(\theta_5) & \sin(\theta_2)\sin(\theta_4) + \cos(\theta_2)\cos(\theta_4) & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
In [9]: # 6th frame wrt origin
```

```
T_6
```

```
Out[9]: ((sin (θ₂) sin (θ₄) cos (θ₁) + cos (θ₁) cos (θ₂) cos (θ₄)) cos (θ₅) - sin (θ₁) sin (θ₅)) cos (θ₆)
         + (- sin (θ₂) cos (θ₁) cos (θ₄) + sin (θ₄) cos (θ₁) cos (θ₂)) sin (θ₆)

         ((sin (θ₁) sin (θ₂) sin (θ₄) + sin (θ₁) cos (θ₂) cos (θ₄)) cos (θ₅) + sin (θ₅) cos (θ₁)) cos (θ₆)
         + (- sin (θ₁) sin (θ₂) cos (θ₄) + sin (θ₁) sin (θ₄) cos (θ₂)) sin (θ₆)

         (sin (θ₂) sin (θ₄) + cos (θ₂) cos (θ₄)) sin (θ₆) + (sin (θ₂) cos (θ₄) - sin (θ₄) cos (θ₂)) cos (θ₅) cos
         (θ₆)

         0
```

```
In [10]: # 7th frame wrt origin
T_7
```

```
Out[10]: (((sin (θ₂) sin (θ₄) cos (θ₁) + cos (θ₁) cos (θ₂) cos (θ₄)) cos (θ₅) - sin (θ₁) sin (θ₅)) cos (θ₆)
         + (- sin (θ₂) cos (θ₁) cos (θ₄) + sin (θ₄) cos (θ₁) cos (θ₂)) sin (θ₆)) cos (θ₇)
         + ((sin (θ₂) sin (θ₄) cos (θ₁) + cos (θ₁) cos (θ₂) cos (θ₄)) sin (θ₅) + sin (θ₁) cos (θ₅)) sin (θ₇)

         (((sin (θ₁) sin (θ₂) sin (θ₄) + sin (θ₁) cos (θ₂) cos (θ₄)) cos (θ₅) + sin (θ₅) cos (θ₁)) cos (θ₆)
         + (- sin (θ₁) sin (θ₂) cos (θ₄) + sin (θ₁) sin (θ₄) cos (θ₂)) sin (θ₆)) cos (θ₇)
         + ((sin (θ₁) sin (θ₂) sin (θ₄) + sin (θ₁) cos (θ₂) cos (θ₄)) sin (θ₅) - cos (θ₁) cos (θ₅)) sin (θ₇)

         ((sin (θ₂) sin (θ₄) + cos (θ₂) cos (θ₄)) sin (θ₆) + (sin (θ₂) cos (θ₄) - sin (θ₄) cos (θ₂)) cos (θ₅) cos
         (θ₆)) cos (θ₇) + (sin (θ₂) cos (θ₄) - sin (θ₄) cos (θ₂)) sin (θ₅) sin (θ₇)

         0
```

```
In [11]: # End effector position
Xp=Transformation.col(3)
Xp.row_del(3)
```

```
In [12]: # substituting values to end effector positions
Xp_copy=Xp.subs({d_1:33.30,
                  d_3:31.60,
                  d_5:38.40,
                  a_3:8.80,
                  d_7:20.7})
```

```
In [13]: # Taking individual x,y,z for the end effector position
```

```
x=Xp[0]
y=Xp[1]
z=Xp[2]
```

In [14]: # Setting up the Jacobian Matrix

```
J=Matrix(([diff(x,theta_1),diff(x,theta_2),diff(x,theta_4),diff(x,theta_5),diff(x,theta_6),
          [diff(y,theta_1),diff(y,theta_2),diff(y,theta_4),diff(y,theta_5),diff(y,theta_6),
          [diff(z,theta_1),diff(z,theta_2),diff(z,theta_4),diff(z,theta_5),diff(z,theta_6),
          [T_1[0,2],T_2[0,2],T_4[0,2],T_5[0,2],T_6[0,2],T_7[0,2]],
          [T_1[1,2],T_2[1,2],T_4[1,2],T_5[1,2],T_6[1,2],T_7[1,2]],
          [T_1[2,2],T_2[2,2],T_4[2,2],T_5[2,2],T_6[2,2],T_7[2,2]]]))
```

In [15]: # Substituting values to Jacobian Matrix

```
J=J.subs({d_1:33.30,
          d_3:31.60,
          d_5:38.40,
          a_3:8.80,
          d_7:20.7})
```

In [16]: J

Out[16]:

$$\begin{aligned} & 20.7 ((-\sin(\theta_1) \sin(\theta_2) \sin(\theta_4) - \sin(\theta_1) \cos(\theta_2) \cos(\theta_4)) \cos(\theta_5) - \sin(\theta_5) \cos(\theta_1)) \sin(\theta_6) \\ & + 8.8 ((-\sin(\theta_1) \sin(\theta_2) \sin(\theta_4) - \sin(\theta_1) \cos(\theta_2) \cos(\theta_4)) \cos(\theta_5) - \sin(\theta_5) \cos(\theta_1)) \cos(\theta_6) \\ & \quad + 8.8 (\sin(\theta_1) \sin(\theta_2) \cos(\theta_4) - \sin(\theta_1) \sin(\theta_4) \cos(\theta_2)) \sin(\theta_6) \\ & - 20.7 (\sin(\theta_1) \sin(\theta_2) \cos(\theta_4) - \sin(\theta_1) \sin(\theta_4) \cos(\theta_2)) \cos(\theta_6) + 8.8 \sin(\theta_1) \sin(\theta_2) \sin(\theta_4) \\ & + 38.4 \sin(\theta_1) \sin(\theta_2) \cos(\theta_4) + 31.6 \sin(\theta_1) \sin(\theta_2) - 38.4 \sin(\theta_1) \sin(\theta_4) \cos(\theta_2) + 8.8 \sin \\ & \quad (\theta_1) \cos(\theta_2) \cos(\theta_4) - 8.8 \sin(\theta_1) \cos(\theta_2) \\ & 20.7 ((\sin(\theta_2) \sin(\theta_4) \cos(\theta_1) + \cos(\theta_1) \cos(\theta_2) \cos(\theta_4)) \cos(\theta_5) - \sin(\theta_1) \sin(\theta_5)) \sin(\theta_6) \\ & + 8.8 ((\sin(\theta_2) \sin(\theta_4) \cos(\theta_1) + \cos(\theta_1) \cos(\theta_2) \cos(\theta_4)) \cos(\theta_5) - \sin(\theta_1) \sin(\theta_5)) \cos(\theta_6) \\ & \quad + 8.8 (-\sin(\theta_2) \cos(\theta_1) \cos(\theta_4) + \sin(\theta_4) \cos(\theta_1) \cos(\theta_2)) \sin(\theta_6) \\ & - 20.7 (-\sin(\theta_2) \cos(\theta_1) \cos(\theta_4) + \sin(\theta_4) \cos(\theta_1) \cos(\theta_2)) \cos(\theta_6) - 8.8 \sin(\theta_2) \sin(\theta_4) \cos \\ & \quad (\theta_1) - 38.4 \sin(\theta_2) \cos(\theta_1) \cos(\theta_4) - 31.6 \sin(\theta_2) \cos(\theta_1) + 38.4 \sin(\theta_4) \cos(\theta_1) \cos(\theta_2) \\ & \quad - 8.8 \cos(\theta_1) \cos(\theta_2) \cos(\theta_4) + 8.8 \cos(\theta_1) \cos(\theta_2) \\ & 0 \end{aligned}$$

$$\sin(\theta_1)$$

$$-\cos(\theta_1)$$

$$0$$

In [17]: # The velocity matrix for end effector velocity

```
X_dot=Matrix(([0],
              [-10*sin(pi/2+ thetadot*t)*thetadot],
              [10*cos(pi/2 + thetadot*t)*thetadot],
```

```

    [0],
    [0],
    [0)))
X_dot=X_dot.subs({thetadot:(2*pi)/5})
X_dot

```

Out[17]:

$$\begin{bmatrix} 0 \\ -4\pi \cos\left(\frac{2\pi t}{5}\right) \\ -4\pi \sin\left(\frac{2\pi t}{5}\right) \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In [45]:

```

# List1 to store the q_current values
# y_plt to store the y values of the end effector
# z_plt to store the z values of the end effector

x=67.9
list1=[]
x_plt=[]
y_plt=[]
z_plt=[]
i=0
j=0
x=[]

# The initial q_current values are given to us
q_current=Matrix(([0],
                  [0],
                  [pi/2],
                  [0],
                  [pi],
                  [0]))

# Using while loop for
while(i<=5):
    Orientation=J

    Orientation=Orientation.subs({theta_1:q_current[0],
                                   theta_2:q_current[1],
                                   theta_4:q_current[2],
                                   theta_5:q_current[3],
                                   theta_6:q_current[4],
                                   theta_7:q_current[5]
                                  })

    Orientation_inverse=Orientation.evalf().inv()

    q_dot=Orientation_inverse*X_dot
    Z=q_dot
    Z=Z.subs({t:i}).evalf()
    q_current=q_current+Z*0.005

    list1.append(q_current)

    x_plt.append(Xp_copy[0].subs({theta_1:q_current[0],
                                   theta_2:q_current[1],
                                   theta_4:q_current[2],
                                   theta_5:q_current[3],
                                   theta_6:q_current[4],
                                   theta_7:q_current[5]
                                  }))

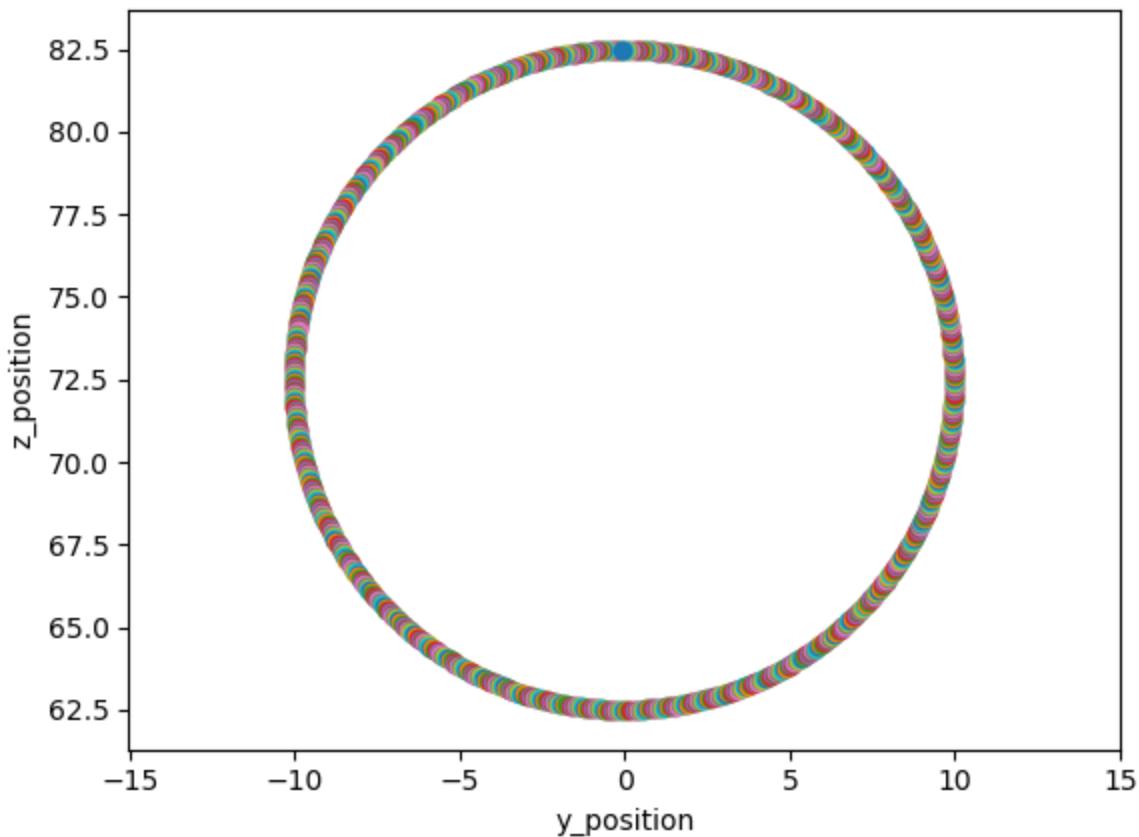
    y_plt.append(Xp_copy[1].subs({theta_1:q_current[0],
                                   theta_2:q_current[1],
                                   theta_4:q_current[2],
                                   theta_5:q_current[3],
                                   theta_6:q_current[4],
                                   theta_7:q_current[5]
                                  }))

    z_plt.append(Xp_copy[2].subs({theta_1:q_current[0],
                                   theta_2:q_current[1],
                                   theta_4:q_current[2],
                                   theta_5:q_current[3],
                                   theta_6:q_current[4],
                                   theta_7:q_current[5]
                                  }))

```

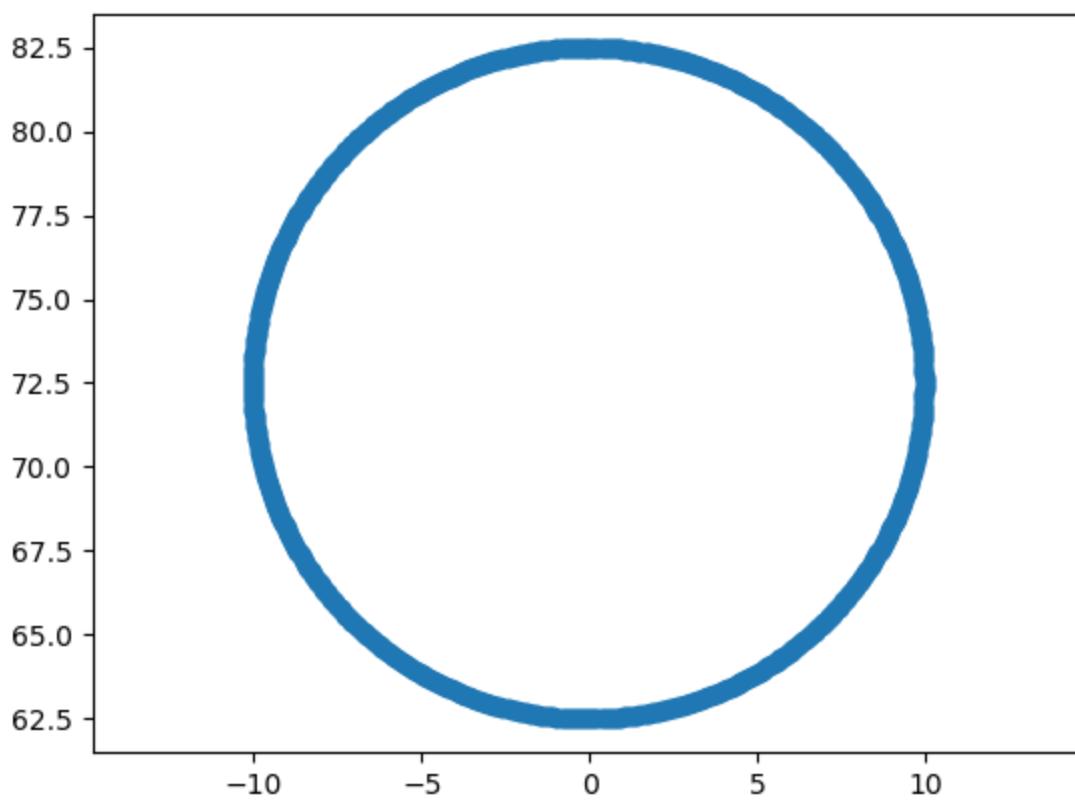
```
theta_5:q_current[3],  
theta_6:q_current[4],  
theta_7:q_current[5})).evalf()  
  
y_plt.append(Xp_copy[1].subs({theta_1:q_current[0],  
theta_2:q_current[1],  
theta_4:q_current[2],  
theta_5:q_current[3],  
theta_6:q_current[4],  
theta_7:q_current[5})).evalf())  
  
z_plt.append(Xp_copy[2].subs({theta_1:q_current[0],  
theta_2:q_current[1],  
theta_4:q_current[2],  
theta_5:q_current[3],  
theta_6:q_current[4],  
theta_7:q_current[5})).evalf())  
print(1)  
i=i+0.005  
plt.axis("equal")  
plt.scatter(y_plt[j],z_plt[j])  
j+=1  
plt.xlabel("y_position")  
plt.ylabel("z_position")  
x.append(67.9)
```

```
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1
```



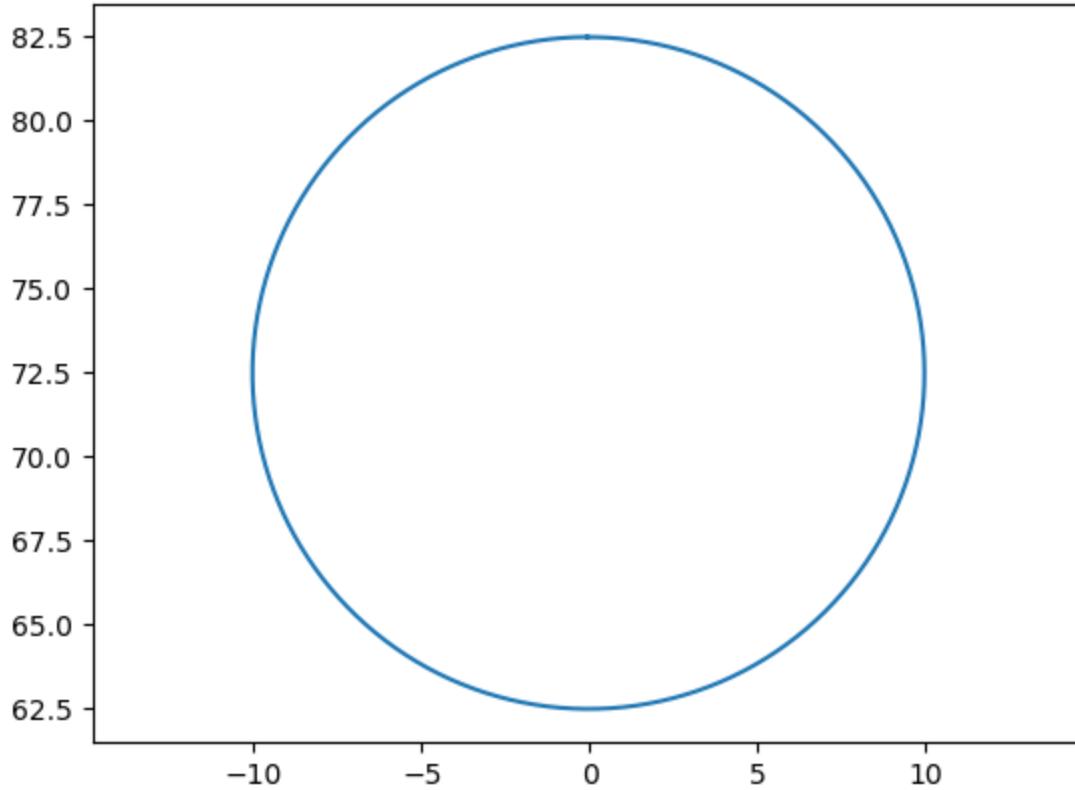
```
In [46]: # Plot of the y and z scatter  
plt.axis("equal")  
plt.scatter(y_plt, z_plt)
```

```
Out[46]: <matplotlib.collections.PathCollection at 0x7f18fcf03940>
```



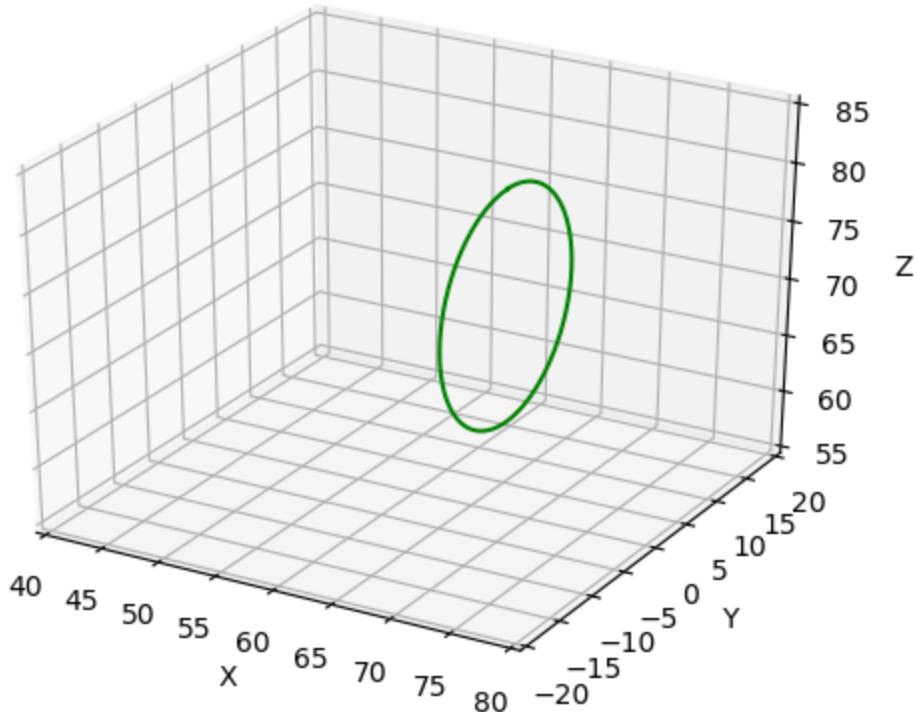
```
In [47]: # normal plot for y and z  
plt.axis("equal")  
plt.plot(y=plt,z=plt)
```

```
Out[47]: [<matplotlib.lines.Line2D at 0x7f1908956d60>]
```



```
In [62]: from mpl_toolkits.mplot3d import Axes3D  
from matplotlib import pyplot as plt  
  
fig = plt.figure()  
ax = fig.add_subplot(111,projection='3d')
```

```
ax.axes.set_xlim3d(left=40, right=80)
ax.axes.set_ylim3d(bottom=-20, top=20)
ax.axes.set_zlim3d(bottom=55, top=85)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.plot3D(x=plt,y=plt,z=plt, color="green")
plt.show()
```



In [60]:

In [ ]: