**Derive the forward linear velocity kinematics for the Lynx robot for when you are tracking the end effector. Explain your steps and your result. Include a copy of any code you write for this step.**

We know that the manipulator Jacobian relates the vector of joint velocities to the body velocity of the end effector. For reference (from Chapter 4 of the textbook)

The $i^{th}$ column of the Jacobian matrix corresponds to the $i^{th}$ joint of the robot manipulator, and takes one of two forms depending on whether the $i^{th}$ joint is prismatic or revolute

$$J_i = \begin{cases} \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{if joint i is revolute} \\ \\ \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & \text{if joint i is prismatic} \end{cases} \quad (4.131)$$

The above formulas make the determination of the Jacobian of any manipulator simple since all of the quantities needed are available once the forward kinematics are worked out. Indeed, the only quantities needed to compute the Jacobian are the unit vectors $z_i$ and the coordinates of the origins $o_1, \ldots, o_n$. A moment's reflection shows that the coordinates for $z_i$ with respect to the base frame are given by the first three elements in the third column of $T_i^0$ while $o_i$ is given by the first three elements of the fourth column of $T_i^0$. Thus, only the third and fourth columns of the $T$ matrices are needed in order to evaluate the Jacobian according to the above formulas.

Since our manipulator has all revolute joints, we will use the latter form for each column in the Jacobian matrix. Thus, we must compute

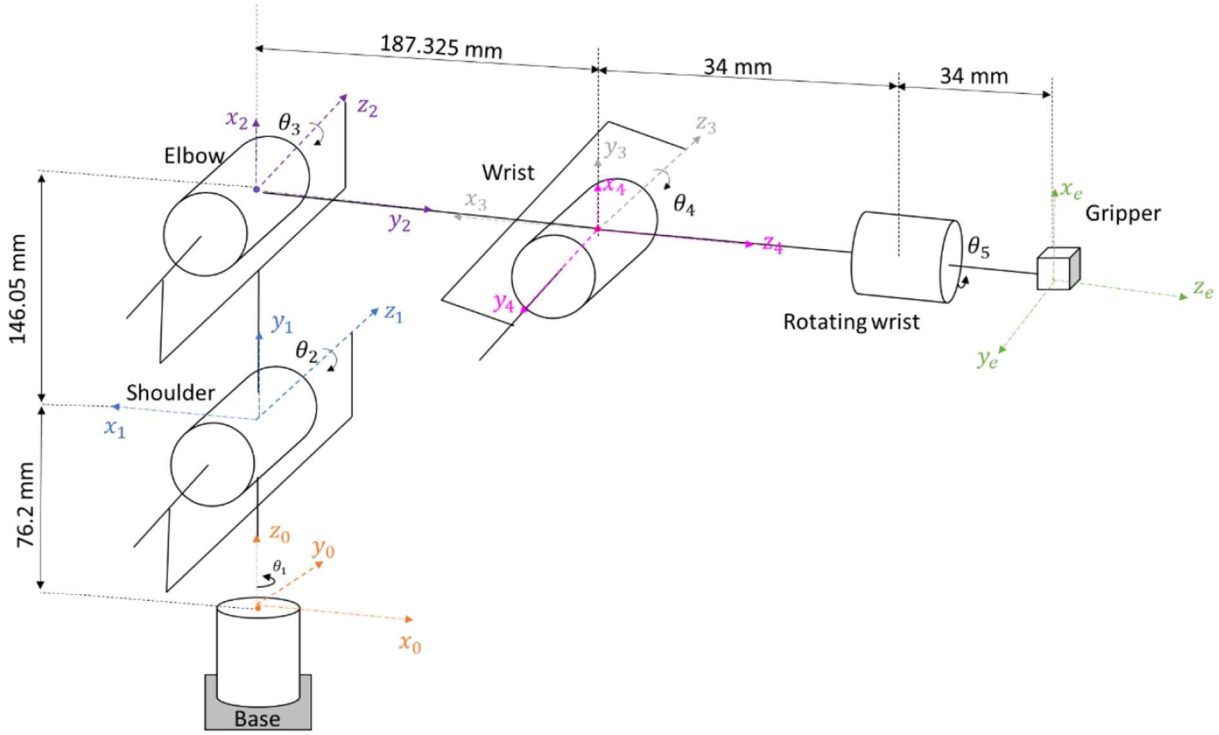
- **Z** i
- **Z** i-1
- **O** n – **O** i-1

We can compute the first two quantities using the first 3 elements in the 3rd column from the transformation matrix

We can compute the last quantity from the first three elements in the fourth column of the transformation matrix.

We can reuse the Python form of all the transformation matrices from Lab 1, and algebraically manipulate them to compute the relevant quantities for each column of the Jacobian.

From Lab 1, our coordinate frames and transformation matrices assigned according to DH convention:




$$A_1^0 = \begin{bmatrix} \cos(\pi + \theta_1) & -\sin(\pi + \theta_1)\,cos(-pi/2) & \sin(\pi + \theta_1)\,sin(-\pi/2) & 0 \\ \sin(\pi + \theta_1) & \cos(\pi + \theta_1)\,cos(-\pi/2) & -\cos(\pi + \theta_1)\,sin(-\pi/2) & 0 \\ 0 & sin(-\pi/2) & cos(-\pi/2) & 76.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^1 = \begin{bmatrix} \cos(\pi/2 + \theta_2) & -\sin(\pi/2 + \theta_2)\cos(0) & \sin(\pi/2 + \theta_2)\sin(0) & -146.05 * \cos(\pi/2 + \theta_2) \\ \sin(\pi/2 + \theta_2) & \cos(\pi/2 + \theta_2)\cos(0) & -\cos(\pi/2 + \theta_2)\sin(0) & -146.05 * \sin(\pi/2 + \theta_2) \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^2 = \begin{bmatrix} \cos(\theta_3 - \pi/2) & -\sin(\theta_3 - \pi/2)\cos(0) & \sin(\theta_3 - \pi/2)\sin(0) & -187.325 * \cos(\theta_3 - \pi/2) \\ \sin(\theta_3 - \pi/2) & \cos(\theta_3 - \pi/2)\cos(0) & -\cos(\theta_3 - \pi/2)\sin(0) & -187.325 * \sin(\theta_3 - \pi/2) \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^4 = \begin{bmatrix} \cos(\theta_4 + \pi/2) & -\sin(\theta_4 + \pi/2)\cos(-\pi/2) & \sin(\theta_4 + \pi/2)\sin(-\pi/2) & 0 \\ \sin(\theta_4 + \pi/2) & \cos(\theta_4 + \pi/2)\cos(-\pi/2) & -\cos(\theta_4 + \pi/2)\sin(-\pi/2) & 0 \\ 0 & \sin(-\pi/2) & \cos(-\pi/2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_e^4 = \begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5)\cos(0) & \sin(\theta_5)\sin(0) & 0 \\ \sin(\theta_5) & \cos(\theta_5)\cos(0) & -\cos(\theta_5)\sin(0) & 0 \\ 0 & \sin(0) & \cos(0) & 68 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Do a sanity check: Let us say the Lynx starts in the zero position and only one of the joints moves. What do you expect the corresponding velocity of the end effector to be?**
**Does your FK reflect this velocity?**

Initial conditions:
q=[0,np.pi/4,0,0,0,0]
qdot=[0,2,0,0,0,0]
z0= [0,0,1]

Corresponding velocity of the end effector:
[-154.53818703  0.  -567.6299686 ]

This makes sense since both X and Z are moving when Joint 2 moves.

**Code**

```
import numpy as np
from calculateFK import calculateFK


# inputs
q=[0,np.pi/4,0,0,0,0]
qdot=[0,2,0,0,0,0]
z0=np.array([0,0,1])

# z matrix calculated for joints 1,2 and 3 at once
z13=np.array([-np.sin(q[0]),np.cos(q[0]),0])

# taking an instance of the calculate FK function
fk = calculateFK()
Jp,T0e = fk.forward(q)

# using the 3rd column of the T0e matrix to find z matrix for joints 4 and e
z4e = np.array([T0e[0,2],T0e[1,2],T0e[2,2]])

# for loop to fill complete jacobian matrix
J = np.zeros((3,6))
for i in range(1,6):
    # Finding the difference between joint origin and end effector
    Jo=Jp[5]-Jp[i-1]
    #determining the cross product based on the joint considered
    if i == 1:
        Jr=np.cross(z0,Jo)
    elif i <= 3:
        Jr=np.cross(z13,Jo)
    else:
        Jr=np.cross(z4e,Jo)

    # jacobian matrix being filled
```

```
    J[0, i-1]=Jr[0]
    J[1, i-1]=Jr[1]
    J[2, i-1]=Jr[2]


print("Jacobian")
print(J)


# multiply result with qdot to get final linear velocity matrix
linv=np.matmul(J,qdot).T
print(linv)
```