ENPM 662 – Final Exam

Name: Suhas Nagaraj

UID: 119505373

Question 1: Workspace, velocity kinematics and dynamics

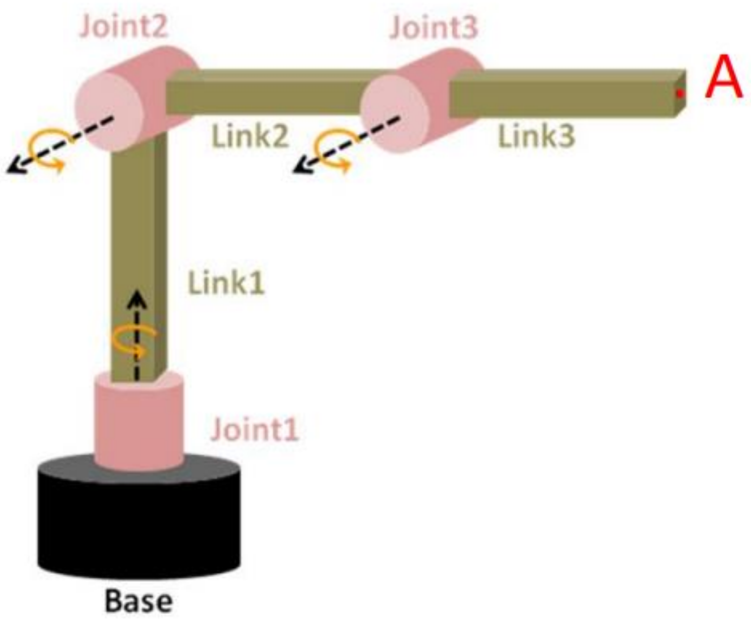


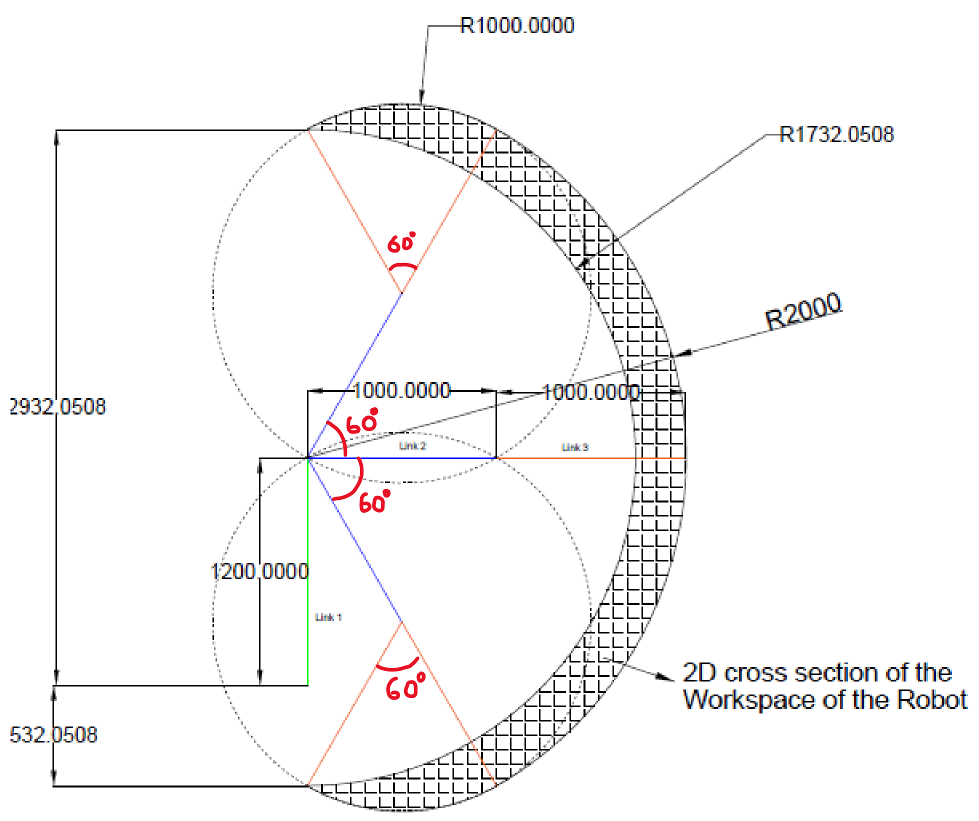
Figure 1: 3DOF Robot

Workspace of the Robot

The workspace is the region in space that the end effector of the robot can reach. The workspace of the given robot, is found by considering following limits of joint rotation:

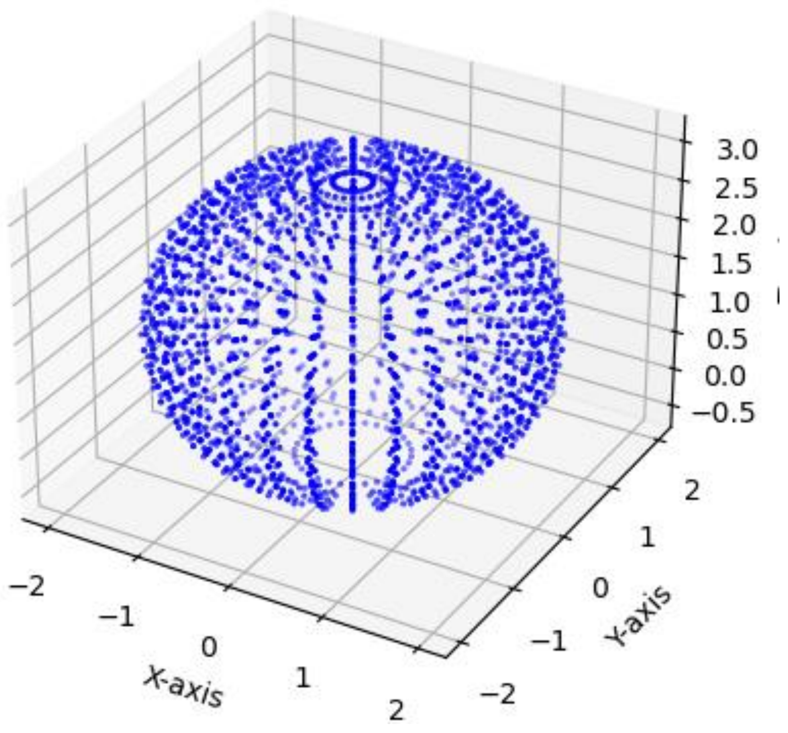
- Joint 1 has no limits of rotation
- Joint 2 has limits of ± π/3 radians from the home position
- Joint 3 has limits of ± π/3 radians from the home position

The workspace was studied and was drawn using AutoCAD and the 3d workspace was plotted using python script.



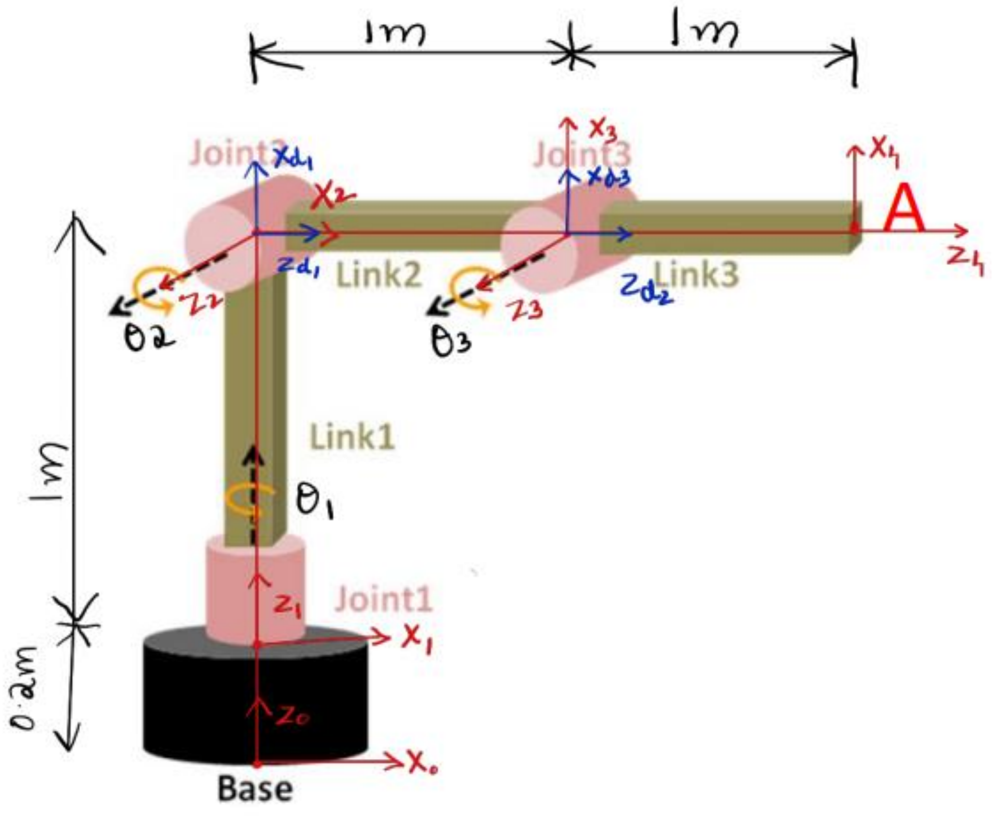
2D Cross-Section of the workspace as drawn on AutoCad

If we rotate the cross-section by 360 degrees about z axis, we will get the 3d workspace of the robot.



3D workspace generated from python script

Jacobian of the Robot



DH Frame Assignment

DH Table:

Frames	a (in m)	α (in degree)	θ (in degree)	d (in m)
Frame 0 – Frame 1	0	0^0	0^0	0.2
Frame 1 – Frame 2	0	90^0	θ_1	1.0
Frame 2 – Frame d1	0	90^0	$\theta_2 + 90^0$	0
Frame d1 – Frame 3	0	-90^0	0^0	1.0
Frame 3 – Frame d2	0	90^0	θ_3	0
Frame d2 – Frame 4	0	0^0	0^0	1.0

Link Transformation Matrices:

The general form of Link Transformation Matrix is given by:

$$T_i = \begin{bmatrix} \cos\theta_i & -(\sin\theta_i * \cos\alpha_i) & (\sin\theta_i * \sin\alpha_i) & (a_i * \cos\theta_i) \\ \sin\theta_i & (\cos\theta_i * \cos\alpha_i) & -(\cos\theta_i * \sin\alpha_i) & (a_i * \sin\theta_i) \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where “i” is the link number.

Therefore, referring to the DH table, the link transformation matrices for the links are given by:

i. For Frame 0 – Frame 1

$${}^0T_1 = \begin{bmatrix} \cos 0 & -(\sin 0 * \cos 0) & (\sin 0 * \sin 0) & (0 * \cos 0) \\ \sin 0 & (\cos 0 * \cos 0) & -(\cos 0 * \sin 0) & (0 * \sin 0) \\ 0 & \sin 0 & \cos 0 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii. For Frame 1 – Frame 2

$${}^1T_2 = \begin{bmatrix} \cos\theta_1 & -(\sin\theta_1 * \cos 90) & (\sin\theta_1 * \sin 90) & (0 * \cos\theta_1) \\ \sin\theta_1 & (\cos\theta_1 * \cos 90) & -(\cos\theta_1 * \sin 90) & (0 * \sin\theta_1) \\ 0 & \sin 90 & \cos 90 & 1.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^1T_2 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & 1.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. For Frame 2 – Frame d1

$^{2}T_{d1}$=

$$\begin{bmatrix} \cos(\theta_2+90) & -(sin(\theta_2+90)*\cos 90) & (sin(\theta_2+90)*\ \sin 90) & (0*\cos(\theta_2+90)) \\ sin(\theta_2+90) & (\cos(\theta_2+90)*\cos 90) & -(\cos(\theta_2+90)*\sin 90) & (0*sin(\theta_2+90)) \\ 0 & \sin 90 & \cos 90 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}T_{d1}=\begin{bmatrix} -\sin\theta_2 & 0 & \cos\theta_2 & 0 \\ \cos\theta_2 & 0 & \sin\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iv. For Frame d1 – Frame 3

$$^{d1}T_{3}=\begin{bmatrix} \cos 0 & -(\sin 0*\cos -90) & (\sin 0*\ \sin -90) & (0*\cos 0) \\ \sin 0 & (\cos 0*\ \cos -90) & -(\cos 0*\sin -90) & (0*\cos 0) \\ 0 & \sin -90 & \cos -90 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{d1}T_{3}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

v. For Frame 3 – Frame d2

$$^{3}Td_{2}=\begin{bmatrix} \cos\theta_3 & -(\sin\theta_3*\cos 90) & (\sin\theta_3*\ \sin 90) & (0*\cos\theta_3) \\ \sin\theta_3 & (\cos\theta_3*\cos 90) & -(\cos\theta_3*\sin 90) & (0*\sin\theta_3) \\ 0 & \sin 90 & \cos 90 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{3}T_{d2}=\begin{bmatrix} \cos\theta_3 & 0 & \sin\theta_3 & 0 \\ \sin\theta_3 & 0 & -\cos\theta_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

vi. For Frame d2 – Frame 4

$$^{d2}T_4=\begin{bmatrix}\cos 0 & -(\sin 0 * \cos 0) & (\sin\theta_1 * \ \sin 0) & (0 * \cos 0)\\ \sin 0 & (\cos 0 * \ \cos 0) & -(\cos 0 * \sin 0) & (0 * \cos 0)\\ 0 & \sin 0 & \cos 0 & 1\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$^{d2}T_4 = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1\end{bmatrix}$$

The Final Transformation Matrix is given by post multiplying the matrices like:

$^{0}T_4 = {}^{0}T_1 * {}^{1}T_2 * {}^{2}T_{d1} * {}^{d1}T_3 * {}^{3}T_{d2} * {}^{d2}T_4$

The Final Transformation Matrix:

$$\left[\begin{matrix} -\sin(\theta_2(t))\cos(\theta_1(t))\cos(\theta_3(t)) - \sin(\theta_3(t))\cos(\theta_1(t))\cos(\theta_2(t))\\ -\sin(\theta_1(t))\sin(\theta_2(t))\cos(\theta_3(t)) - \sin(\theta_1(t))\sin(\theta_3(t))\cos(\theta_2(t))\\ -\sin(\theta_2(t))\sin(\theta_3(t)) + \cos(\theta_2(t))\cos(\theta_3(t))\\ 0\end{matrix}\right.$$

$$\begin{matrix}\sin(\theta_1(t))\\ -\cos(\theta_1(t))\\ 0\\ 0\end{matrix}$$

$$\begin{matrix} -\sin(\theta_2(t))\sin(\theta_3(t))\cos(\theta_1(t)) + \cos(\theta_1(t))\cos(\theta_2(t))\cos(\theta_3(t))\\ -\sin(\theta_1(t))\sin(\theta_2(t))\sin(\theta_3(t)) + \sin(\theta_1(t))\cos(\theta_2(t))\cos(\theta_3(t))\\ \sin(\theta_2(t))\cos(\theta_3(t)) + \sin(\theta_3(t))\cos(\theta_2(t))\\ 0\end{matrix}$$

$$\left.\begin{matrix} -\sin(\theta_2(t))\sin(\theta_3(t))\cos(\theta_1(t)) + \cos(\theta_1(t))\cos(\theta_2(t))\cos(\theta_3(t)) + \cos(\theta_1(t))\cos(\theta_2(t))\\ -\sin(\theta_1(t))\sin(\theta_2(t))\sin(\theta_3(t)) + \sin(\theta_1(t))\cos(\theta_2(t))\cos(\theta_3(t)) + \sin(\theta_1(t))\cos(\theta_2(t))\\ \sin(\theta_2(t))\cos(\theta_3(t)) + \sin(\theta_2(t)) + \sin(\theta_3(t))\cos(\theta_2(t)) + 1.2\\ 1\end{matrix}\right]$$

Validation: When $(\theta_1, \theta_2, \theta_3) = (0,0,0)$ – Home position

$$\begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 1.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Which is exactly what we get when we verify geometrically (at home position)

Constructing the Jacobian Matrix

Method followed – Second Method

Steps:

1. Calculating the matrix $\boldsymbol{T_i}$

$T_1 = T_1^0 = T_1^0 * T_2^1 * T_{d1}^2$

$T_2 = T_2^0 = T_1^0 * T_2^1 * T_{d1}^2 * T_3^{d1}$

$T_3 = T_4^0 = T_1^0 * T_2^1 * T_{d1}^2 * T_3^{d1} * T_{d2}^3 * T_4^{d2}$

2. Calculating $\boldsymbol{Z_i}$ (First 3 row values of the 3rd column of the T matrix)

$Z_1 = T_1[0{:}3,2]$

$Z_2 = T_2[0{:}3,2]$

$Z_3 = T_3[0{:}3,2]$

3. Calculating h(q_1, q_2, ….. q_n)

h(q_1, q_2, ….. q_n) = $P_8^0 = T_6\ [0{:}3,3]$

4. Calculating $\partial\mathbf{h}/\partial\mathbf{q}_i$

i. $\partial h/\partial q_1 = \partial h/\partial \theta_1$
ii. $\partial h/\partial q_2 = \partial h/\partial \theta_2$
iii. $\partial h/\partial q_3 = \partial h/\partial \theta_3$

5. Writing the Jacobian Matrix

$$J_8^0 = J = \begin{bmatrix} \partial h/\partial \theta_1 & \partial h/\partial \theta_2 & \partial h/\partial \theta_3 \\ Z_1 & Z_2 & Z_3 \end{bmatrix}$$

Analytical Jacobian matrix obtained:

$$\left[\begin{matrix} \sin(\theta_1(t))\sin(\theta_2(t))\sin(\theta_3(t)) - \sin(\theta_1(t))\cos(\theta_2(t))\cos(\theta_3(t)) - \sin(\theta_1(t))\cos(\theta_2(t)) \\ -\sin(\theta_2(t))\sin(\theta_3(t))\cos(\theta_1(t)) + \cos(\theta_1(t))\cos(\theta_2(t))\cos(\theta_3(t)) + \cos(\theta_1(t))\cos(\theta_2(t)) \\ 0 \\ \cos(\theta_1(t))\cos(\theta_2(t)) \\ \sin(\theta_1(t))\cos(\theta_2(t)) \\ \sin(\theta_2(t)) \end{matrix}\right.$$

Column 1

$$\begin{matrix} -\sin(\theta_2(t))\cos(\theta_1(t))\cos(\theta_3(t)) - \sin(\theta_2(t))\cos(\theta_1(t)) - \sin(\theta_3(t))\cos(\theta_1(t))\cos(\theta_2(t)) \\ -\sin(\theta_1(t))\sin(\theta_2(t))\cos(\theta_3(t)) - \sin(\theta_1(t))\sin(\theta_2(t)) - \sin(\theta_1(t))\sin(\theta_3(t))\cos(\theta_2(t)) \\ -\sin(\theta_2(t))\sin(\theta_3(t)) + \cos(\theta_2(t))\cos(\theta_3(t)) + \cos(\theta_2(t)) \\ \sin(\theta_1(t)) \\ -\cos(\theta_1(t)) \\ 0 \end{matrix}$$

Column 2

$$\left.\begin{matrix} -\sin(\theta_2(t))\cos(\theta_1(t))\cos(\theta_3(t)) - \sin(\theta_3(t))\cos(\theta_1(t))\cos(\theta_2(t)) \\ -\sin(\theta_1(t))\sin(\theta_2(t))\cos(\theta_3(t)) - \sin(\theta_1(t))\sin(\theta_3(t))\cos(\theta_2(t)) \\ -\sin(\theta_2(t))\sin(\theta_3(t)) + \cos(\theta_2(t))\cos(\theta_3(t)) \\ -\sin(\theta_2(t))\sin(\theta_3(t))\cos(\theta_1(t)) + \cos(\theta_1(t))\cos(\theta_2(t))\cos(\theta_3(t)) \\ -\sin(\theta_1(t))\sin(\theta_2(t))\sin(\theta_3(t)) + \sin(\theta_1(t))\cos(\theta_2(t))\cos(\theta_3(t)) \\ \sin(\theta_2(t))\cos(\theta_3(t)) + \sin(\theta_3(t))\cos(\theta_2(t)) \end{matrix}\right]$$

Column 3

Plot of the Joint Torques and Joint Angles

Given:

- The robot should move in a straight line in the z axis (vertical) by 0.707m and 1.707m away from the base.
- It is assumed that the robot starts at an initial position where the end effector is at 1.707 m in the X axis direction and at a height of 1.907m; the robot has the initial joint values of:

  [q1 , q2 , q3] = [0 , 90 , -90] at (t = 0)

$$\begin{bmatrix} 0 & 0 & 1.0 & 1.70710678118655 \\ 0 & -1 & 0 & 0 \\ 1.0 & 0 & 0 & 1.90710678118655 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Time considered for plotting: T = 10 seconds

Velocity Matrix considered, $\dot{X}$= $[0 \quad 0 \quad -0.0707 \quad 0 \quad 0 \quad 0]$

- This is because the robot must move in the negative Z direction by 0.707m in 15 seconds. Therefore, the linear velocity is -0.707/10 = -0.0707 m/s
- The angular velocity component is considered to make the robot move in an exact straight line vertically.

Now from t=0 to t=10, for a timestep of dt (dt is assumed as 0.05 seconds for this problem), perform the following calculations:

- Substitute current values of (q1 , q2 , q3) to the Jacobian equation J
- Substitute the current value of t to velocity trajectory equation V
- Use the Inverse Velocity Kinematics equation to calculate the Joint angular velocities

  $\dot{q}$ = J^{-1} * V

$$\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = J^{-1} * \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Use the joint angular velocities obtained from above equation to perform numeric integration and get the new joint angle values.

  i.e. q_{new} = $q_{current}$ + $\dot{q}$ * dt

- Plug in the new q values obtained after numeric integration into the forward position kinematics equation and extract the position of the end effector with respect to the base. This is done by extracting the last column of the final transformation matrix after substitution.
- Store the position values and plot.

Robot Dynamic equation

The robot dynamic equation is given by:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + J^T(q)F$$

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) - J^T(q)F$$

Computing gravity matrix $\boldsymbol{g(q)}$:

$$g_i(q) = \frac{\partial P(q)}{\partial q_i}$$

$$g_i(q) = \begin{bmatrix} g_1(q) \\ g_2(q) \\ g_3(q) \end{bmatrix}$$

Where P(q) is the total potential energy of the system

Where,

$g(q)$ is the gravity matrix

J^T is the transpose of the Jacobian

F is the force matrix

Computing Total Potential Energy P(q):

We know that Potential Energy,

PE = m*g*h

Where m is the mass of the body

g is the acceleration due to gravity

h is the height of the centre of mass of the body

In a serial robot, total potential energy is equal to the sum of the potential energy of its links.

Therefore, P = P_1 + P_2 + P_3

Note - the centre of mass of the link is assumed to be at l/2, where l is the length of the link.

$$P_i = m_i * g * h_i$$

Link mass m_i

Density of the link - 7600 Kg/m3

Area of cross section of the link, A = (0.09*0.09) - (0.08*0.08) = 0.0017 m^2

Length of the link, L= 1m

Volume = A*L = 0.0017 m^3

Mass of the link = Density * Volume = 7600 * 0.0017 = 12.92 kg

Computing link height h_i (q)

The height of the centre of mass of individual links is a function of joint angles as it changes when the joint angle changes. To obtain this height as a function of joint angles, we can follow the steps:

i. Assume that the centre of mass of the link is at the half of the height of the link. (i.e. if the link length is ‘d’, then the centre of mass is assumed to be at ‘d/2’)
ii. Compute the transformation matrices, which gives us the transformation from the centre of the link to the base frame. This is done by considering the link length as half its true length, while substituting in the transformation matrix.
iii. Extract the height of the link from the computed transformation matrices (4th column and 3rd row element)

This gives us the height of the links as a function of joint angles ‘q’.

Substituting the values to get gravity matrix g(t)

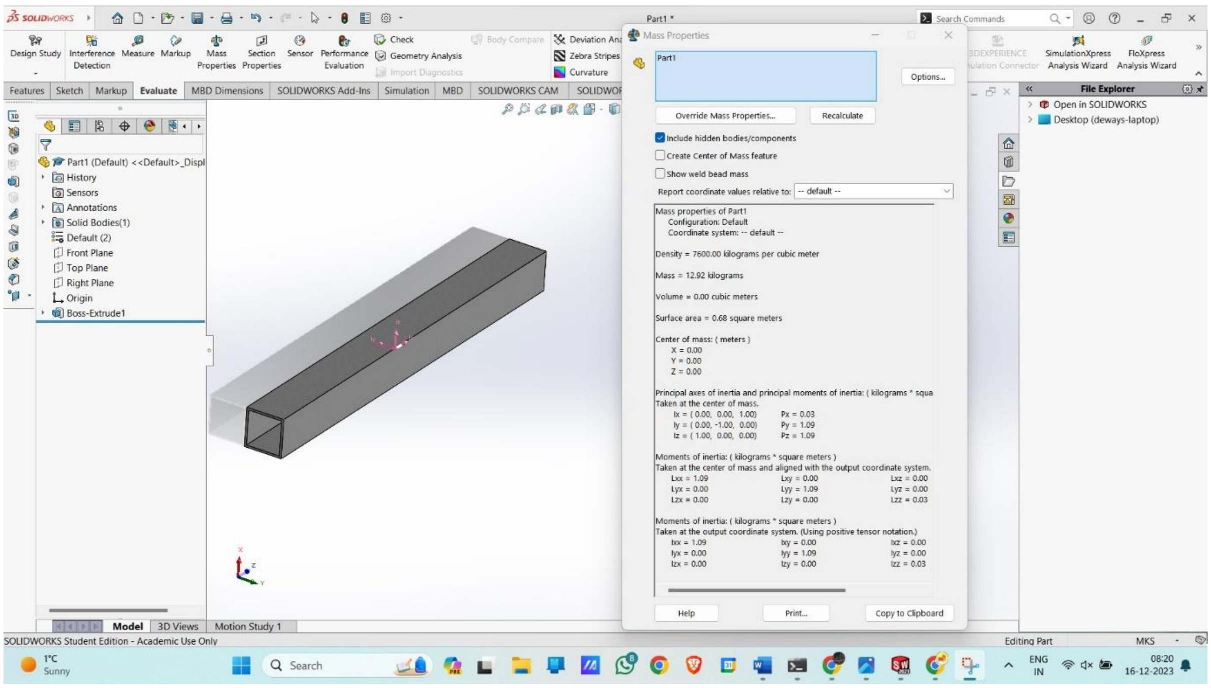
- $P_i = m_i * g * h_i$
- P = P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8

- $$g(q) = \begin{bmatrix} g_1(q) \\ g_2(q) \\ g_3(q) \end{bmatrix} = \begin{bmatrix} \frac{\partial P(q)}{\partial q_1} \\ \frac{\partial P(q)}{\partial q_2} \\ \frac{\partial P(q)}{\partial q_3} \end{bmatrix}$$

Computing M matrix

Steps followed:

1. Obtain the moment of inertia tensor of the links:
   This is done using SolidWorks where on inspecting the property of the link, we get the inertia tensor.



```
Mass properties of Part1
     Configuration: Default
     Coordinate system: -- default --

Density = 7600.00 kilograms per cubic meter

Mass = 12.92 kilograms

Volume = 0.00 cubic meters

Surface area = 0.68 square meters

Center of mass: ( meters )
      X = 0.00
      Y = 0.00
      Z = 0.00

Principal axes of inertia and principal moments of inertia: ( kilograms * squa
Taken at the center of mass.
       Ix = ( 0.00,  0.00,  1.00)        Px = 0.03
       Iy = ( 0.00, -1.00,  0.00)        Py = 1.09
       Iz = ( 1.00,  0.00,  0.00)        Pz = 1.09

Moments of inertia: ( kilograms * square meters )
Taken at the center of mass and aligned with the output coordinate system.
      Lxx = 1.09                  Lxy = 0.00                 Lxz = 0.00
      Lyx = 0.00                  Lyy = 1.09                 Lyz = 0.00
      Lzx = 0.00                  Lzy = 0.00                 Lzz = 0.03

Moments of inertia: ( kilograms * square meters )
Taken at the output coordinate system. (Using positive tensor notation.)
      Ixx = 1.09                  Ixy = 0.00                 Ixz = 0.00
      Iyx = 0.00                  Iyy = 1.09                 Iyz = 0.00
      Izx = 0.00                  Izy = 0.00                 Izz = 0.03
```

Include hidden bodies/components: ✓
Create Center of Mass feature: ☐
Show weld bead mass: ☐
Report coordinate values relative to: -- default --

2. Obtain the inertia in terms of the base frame

   This is done by multiplying the inertia tensor with transformation matrix to the centre of mass/centroid of the link.

$$I_{cm} = R^T * I * R$$

3. Compute the linear velocity Jacobian and angular velocity Jacobian from position vector to the centre of mass of the links. My approach in code is as follows:

```
Jv1=Matrix([[diff(R1[0:3,3],theta_1),diff(R1[0:3,3],theta_2),diff(R1[0:3,3],theta_3)]])
Jv2=Matrix([[diff(R2[0:3,3],theta_1),diff(R2[0:3,3],theta_2),diff(R2[0:3,3],theta_3)]])
Jv3=Matrix([[diff(R3[0:3,3],theta_1),diff(R3[0:3,3],theta_2),diff(R3[0:3,3],theta_3)]])

Z=Matrix([[0],[0],[0]])
Jw1=Matrix([[R1[0:3,2],Z,Z]])
Jw2=Matrix([[R1[0:3,2],R2[0:3,2],Z]])
Jw3=Matrix([[R1[0:3,2],R2[0:3,2],R3[0:3,2]]])
```

4. Substitute the values in the equation:

$$M(q) = \sum_{i=1}^{n} \left[m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \right]$$

From Spong Textbook

Computing C matrix

C matrix is computed using the M matrix obtained above.

The C matrix elements are given by:

$$\begin{aligned} c_{kj} &= \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i \\ &= \sum_{i=1}^{n} \frac{1}{2} \left\{ \frac{\partial d_{kj}}{\partial q_j} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \dot{q}_i \end{aligned}$$

From Spong Textbook

This is computed in the code as follows:

```
C111=0.5*(diff(M[0,0],theta_1)+diff(M[0,0],theta_1)-diff(M[0,0],theta_1))*qd1
C112=0.5*(diff(M[1,0],theta_1)+diff(M[1,0],theta_1)-diff(M[0,0],theta_2))*qd1
C113=0.5*(diff(M[2,0],theta_1)+diff(M[2,0],theta_1)-diff(M[0,0],theta_3))*qd1
C121=0.5*(diff(M[0,1],theta_1)+diff(M[0,0],theta_2)-diff(M[0,1],theta_1))*qd1
C122=0.5*(diff(M[1,1],theta_1)+diff(M[1,0],theta_2)-diff(M[0,1],theta_2))*qd1
C123=0.5*(diff(M[2,1],theta_1)+diff(M[2,0],theta_2)-diff(M[0,1],theta_3))*qd1
C131=0.5*(diff(M[0,2],theta_1)+diff(M[0,0],theta_3)-diff(M[0,2],theta_1))*qd1
C132=0.5*(diff(M[1,2],theta_1)+diff(M[1,0],theta_3)-diff(M[0,2],theta_2))*qd1
C133=0.5*(diff(M[2,2],theta_1)+diff(M[2,0],theta_3)-diff(M[0,2],theta_3))*qd1

C211=0.5*(diff(M[0,0],theta_2)+diff(M[0,1],theta_1)-diff(M[1,0],theta_1))*qd2
C212=0.5*(diff(M[1,0],theta_2)+diff(M[1,1],theta_1)-diff(M[1,0],theta_2))*qd2
C213=0.5*(diff(M[2,0],theta_2)+diff(M[2,1],theta_1)-diff(M[1,0],theta_3))*qd2
C221=0.5*(diff(M[0,1],theta_2)+diff(M[0,1],theta_2)-diff(M[1,1],theta_1))*qd2
C222=0.5*(diff(M[1,1],theta_2)+diff(M[1,1],theta_2)-diff(M[1,1],theta_2))*qd2
C223=0.5*(diff(M[2,1],theta_2)+diff(M[2,1],theta_2)-diff(M[1,1],theta_3))*qd2
C231=0.5*(diff(M[0,2],theta_2)+diff(M[0,1],theta_2)-diff(M[1,2],theta_1))*qd2
C232=0.5*(diff(M[1,2],theta_2)+diff(M[1,1],theta_2)-diff(M[1,2],theta_2))*qd2
C233=0.5*(diff(M[2,2],theta_2)+diff(M[2,1],theta_2)-diff(M[1,2],theta_3))*qd2

C311=0.5*(diff(M[0,0],theta_3)+diff(M[0,2],theta_1)-diff(M[2,0],theta_1))*qd3
C312=0.5*(diff(M[1,0],theta_3)+diff(M[1,2],theta_1)-diff(M[2,0],theta_2))*qd3
C313=0.5*(diff(M[2,0],theta_3)+diff(M[2,2],theta_1)-diff(M[2,0],theta_3))*qd3
C321=0.5*(diff(M[0,1],theta_3)+diff(M[0,2],theta_2)-diff(M[2,1],theta_1))*qd3
C322=0.5*(diff(M[1,1],theta_3)+diff(M[1,2],theta_2)-diff(M[2,1],theta_2))*qd3
C323=0.5*(diff(M[2,1],theta_3)+diff(M[2,2],theta_2)-diff(M[2,1],theta_3))*qd3
C331=0.5*(diff(M[0,2],theta_3)+diff(M[0,2],theta_2)-diff(M[2,2],theta_1))*qd3
C332=0.5*(diff(M[1,2],theta_3)+diff(M[1,2],theta_2)-diff(M[2,2],theta_2))*qd3
C333=0.5*(diff(M[2,2],theta_3)+diff(M[2,2],theta_2)-diff(M[2,2],theta_3))*qd3
```

```
C11=C111+C211+C311
C12=C121+C221+C321
C13=C131+C231+C331
C21=C112+C212+C312
C22=C122+C222+C322
C23=C132+C232+C332
C31=C113+C213+C313
C32=C123+C223+C323
C33=C133+C233+C333

C=Matrix([[C11,C12,C13],[C21,C22,C23],[C31,C32,C33]])
```

Computing the Joint torques

It is given that the robot should exert a force of 10N in the positive X direction with respect to the base. Hence, the force matrix is defined as

F = $[10 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$

Note: To obtain the $\ddot{q}$ values, we do numerical differentiation on the $\dot{q}$ values in every iteration

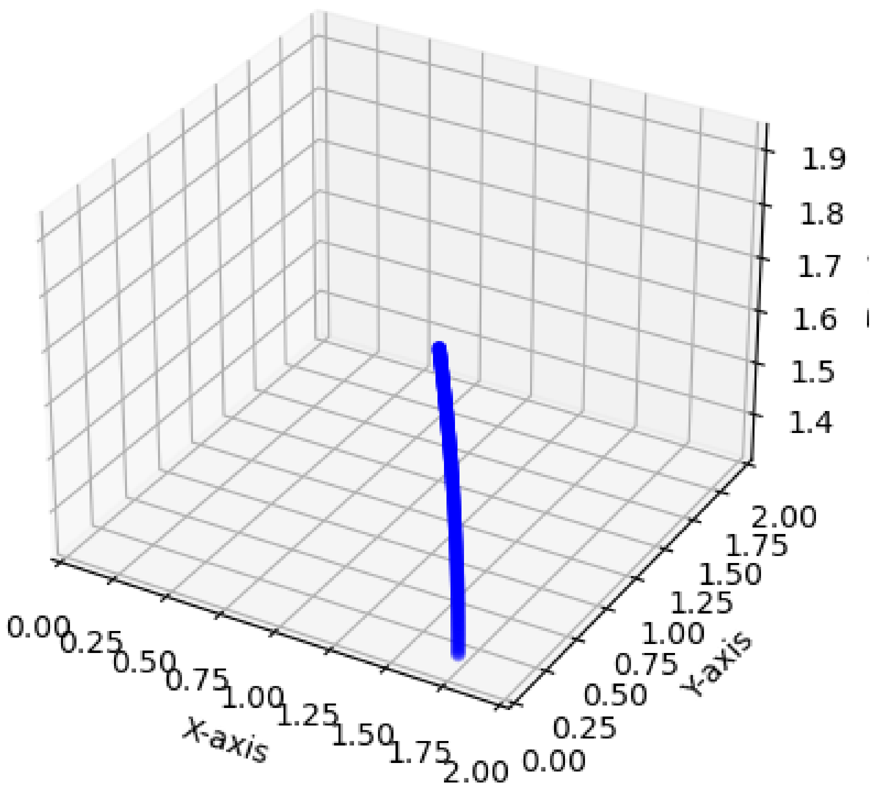
Substitute the matrix M, matrix C, gravity matrix g, force matrix F along with $\ddot{q}$ *and* $\dot{q}$ in the equation:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) - J^T(q)F$$

```
tou_s=(MS*q_ddot)+(CS*q_dot)+GS-(JT*F)
```

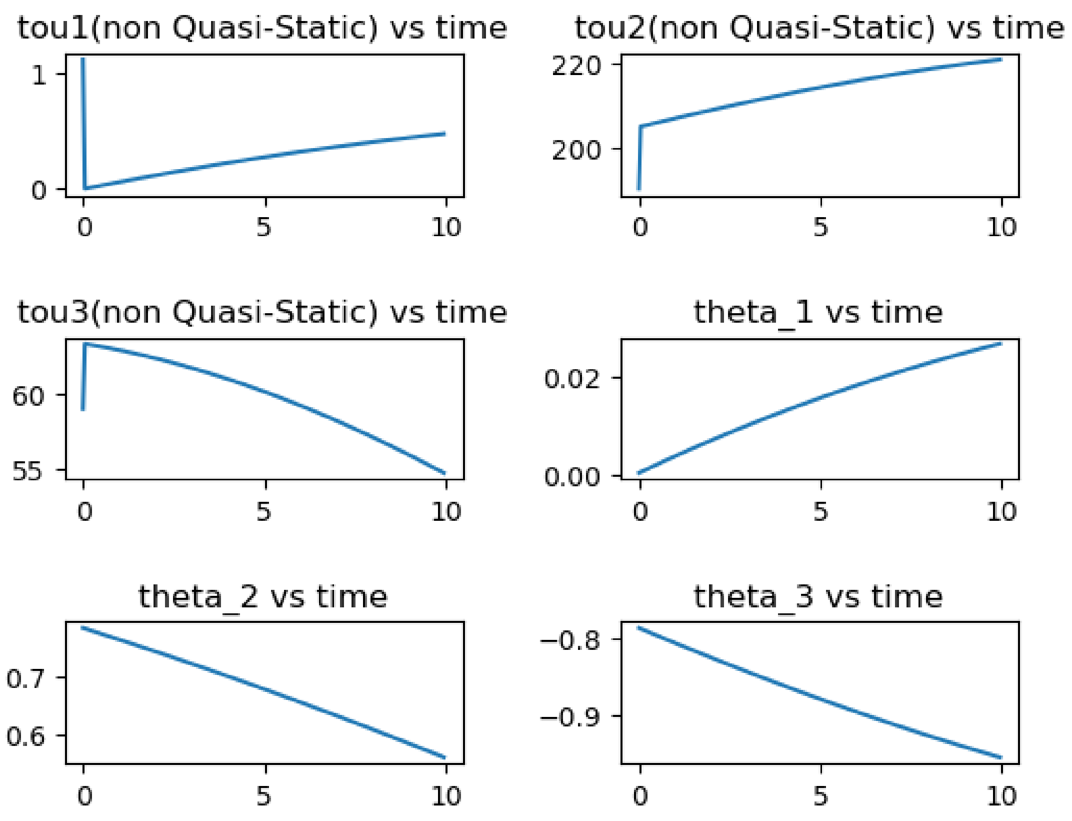
Results Obtained and Plots

3D trajectory of the End Effector



Torque plots:

The joint torques are plotted for the robot along with the joint angles:



Joint Torques vs Time and Joint Angles vs Time plots

(The sudden jump in torque values is due to the initial acceleration being zero. It creates an impulse. But the torques stabilize and give the values after first iteration)

Question 2: Rotational Matrices

A. <u>Any rotation matrix is orthogonal, with determinant equal to 1.</u>

The general form of a rotational matrix in 2D is given by:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

A matrix is orthogonal if the product of the matrix with its transpose is equal to 1. Hence,

$$R^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$R * R^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} * \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} (\cos\theta * \cos\theta) + \ (-\sin\theta * \ -\sin\theta) & (\cos\theta * \sin\theta) + (-\sin\theta * \cos\theta) \\ (\cos\theta * \sin\theta) + (-\sin\theta * \cos\theta) & (\cos\theta * \cos\theta) + \ (-\sin\theta * \ -\sin\theta) \end{bmatrix}$$

$$= \begin{bmatrix} \cos^2\theta + \sin^2\theta & 0 \\ 0 & \cos^2\theta + \sin^2\theta \end{bmatrix}$$

$R * \ R^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$

Hence, the rotational matrix is orthogonal.

Now, taking the determinant of the rotational matrix:

Det R = $\begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$ = $(\cos\theta * \cos\theta) - \ (\sin\theta * (-\sin\theta)) = \cos^2\theta + \sin^2\theta$ = 1

Hence it is proved that a general 2D rotational matrix is orthogonal and its determinant is equal to 1.

Extending this to 3D, any 3D rotation can be split into 3 Euler angle rotations about the 3 axes X, Y and Z. That is,

3D rotational matrix, $R \ = \ R_\psi * R_\theta * \ R_\phi$

Here, each individual rotational matrices about X Y and Z axes are orthogonal and their determinant is equal to 1.

Hence, $R * \ R^T$ = $(R_\psi * R_\theta * \ R_\phi)$ * $(R_\psi * R_\theta * \ R_\phi)^{\,T}$

$R * \ R^T$ = $R_\psi * R_\theta * \ R_\phi * {R_\phi}^T * {R_\theta}^T * {R_\psi}^T$

$R * \ R^T$ = $R_\psi * R_\theta * \ I * {R_\theta}^T * {R_\psi}^T$ = $R * \ R^T$ = $R_\psi * I * {R_\psi}^T = I$

Hence R is orthogonal.

Also, $|R|$ = $|R_\psi * R_\theta * \ R_\phi|$ = $|R_\psi| * |R_\theta| * |R_\phi|$

$|R|$ = 1 * 1 * 1 = 1

Hence the determinant of R is 1.

This proves that any rotational matrix is orthogonal, and its determinant is equal to 1.

B. <u>The length of a free vector is not changed by rotation, that is, $||v||$ = $||Rv||$</u>

The length of a vector is given by len(v) = $||v||$ = $\sqrt{v_x^2 + v_y^2 + v_z^2}$

This can also be represented as the square root of the dot product of the transpose of vector v and vector v.

That is, let's consider vector v as $v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$

Now, $v^T = [v_x \quad v_y \quad v_z]$

$v^T * v = [v_x \quad v_y \quad v_z] * \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = v_x^2 + v_y^2 + v_z^2$

$\sqrt{v^T * v} = \sqrt{v_x^2 + v_y^2 + v_z^2} = \|v\|$

Similarly, we can represent the length of the vector $\|Rv\|$ as:

$\|Rv\| = \sqrt{(Rv)^T * (Rv)}$

But $(Rv)^T = v^T * R^T$

Therefore, $\|Rv\| = \sqrt{v^T * R^T * R * v}$

We already showed that $R^T * R$ = 1

Hence, $\|Rv\| = \sqrt{v^T * v}$

But we already showed that $\sqrt{v^T * v}$ = $\|v\|$

Therefore, $\|Rv\|$ = $\|v\|$ ----> Proved

C. <u>The distance between points is not changed by rotation, $\|p1 - p2\| = \|Rp1 - Rp2\|$</u>

Let p1 be represented by (x1,y1,z1) and p2 be represented by (x2,y2,z2)

The distance between 2 points is given by the formula,

$\|\mathrm{p1} - \mathrm{p2}\| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$

This can also be represented as,

$\|\mathrm{p1} - \mathrm{p2}\|^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2$

This can be represented as follows:

$\|\mathrm{p1} - \mathrm{p2}\|^2 = [x_1 - x_2 \quad y_1 - y_2 \quad z_1 - z_2] * \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{bmatrix}$ = $(\mathrm{p1} - \mathrm{p2})^T * (\mathrm{p1} - \mathrm{p2})$

This can be compared to the property: $A \,.\, B = A^T * B$

Hence, $\|\mathrm{p1} - \mathrm{p2}\|^2 = (\mathrm{p1} - \mathrm{p2}).(\mathrm{p1} - \mathrm{p2})$

$\|\mathrm{p1}-\mathrm{p2}\|^2 = \ (\mathrm{p1.p1}) - 2(\mathrm{p1.p2}) + (\mathrm{p2.p2})$

Similarly, we can express the $\|Rp1 - \ Rp2\|$ as:

$\|Rp1 - \ Rp2\|$ = $(R\mathrm{p1} - \mathrm{Rp2}).(R\mathrm{p1} - \mathrm{Rp2})$

$\|Rp1 - \ Rp2\|$ = $(R\mathrm{p1.Rp1}) - (R\mathrm{p1.Rp2}) - (R\mathrm{p1.Rp2}) + (R\mathrm{p2.Rp2})$

$\|Rp1 - \ Rp2\|$ = $(R\mathrm{p1.Rp1}) - (R\mathrm{p1.Rp2}) - (R\mathrm{p1.Rp2}) + (R\mathrm{p2.Rp2})$

Using the same property $A\,.\,B \ = \ A^T * B$

$\|Rp1 - \ Rp2\|^2$ = $((R\mathrm{p1})^T * (\mathrm{Rp1})) - 2 * ((R\mathrm{p1})^T * (\mathrm{Rp2})) + ((R\mathrm{p2})^T * (\mathrm{Rp2}))$

$\|Rp1 - \ Rp2\|^2$ = $((\mathrm{p1}^T * \mathrm{R}^T * \mathrm{R} * \mathrm{p1})) - 2 * (\mathrm{p1}^T * \mathrm{R}^T * \mathrm{R} * \mathrm{p2}) + (\mathrm{p2}^T * \mathrm{R}^T * \mathrm{R} * \mathrm{p2})$

But $\mathrm{R}^T * \mathrm{R} = 1$

$\|Rp1 - \ Rp2\|^2$ = $((\mathrm{p1}^T * \mathrm{R}^T * \mathrm{R} * \mathrm{p1})) - 2 * (\mathrm{p1}^T * \mathrm{R}^T * \mathrm{R} * \mathrm{p2}) + (\mathrm{p2}^T * \mathrm{R}^T * \mathrm{R} * \mathrm{p2})$

$\|Rp1 - \ Rp2\|^2$ = $((\mathrm{p1}^T * \mathrm{p1})) - 2 * (\mathrm{p1}^T * \mathrm{p2}) + (\mathrm{p2}^T * \mathrm{p2})$

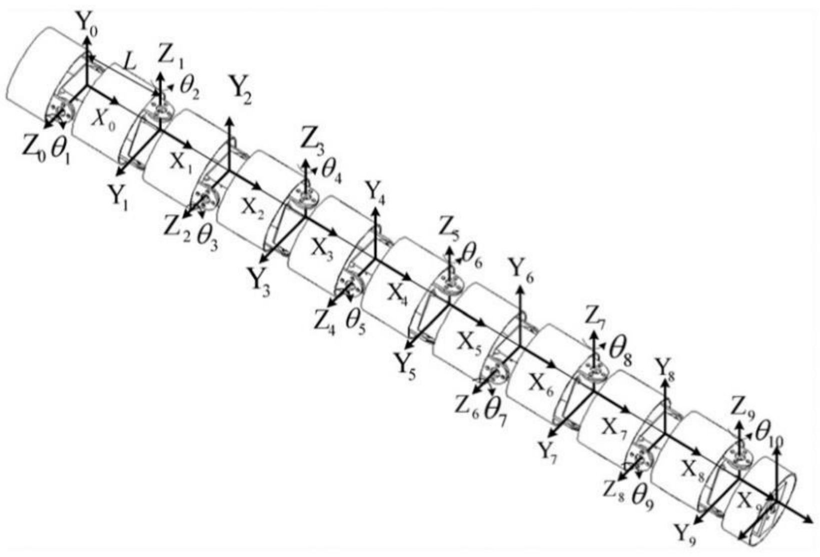
$\|Rp1 - \ Rp2\|^2$ = $((\mathrm{p1.p1})) - 2 * (p1.\mathrm{p2}) + (\mathrm{p2.p2})$

Therefore, comparing both results,

$\|Rp1 - \ Rp2\|^2$ = $\|\mathrm{p1}-\mathrm{p2}\|^2$

Hence $\|Rp1 - \ Rp2\| = \ \|\mathrm{p1}-\mathrm{p2}\|$ --------> Proved

Question 3: Kinematics and motion



- In this robot, alternate joints axes are perpendicular to each other.
- So, for it to achieve a slithering motion, the joints whose axes are perpendicular to the base/ground should contribute to the slithering action, whereas the joints with axes parallel to the base/ground should contribute to the forward movement.
- The movement of the bot resembles a sine wave, and hence the angles inputs to the joints can be determines using this concept.

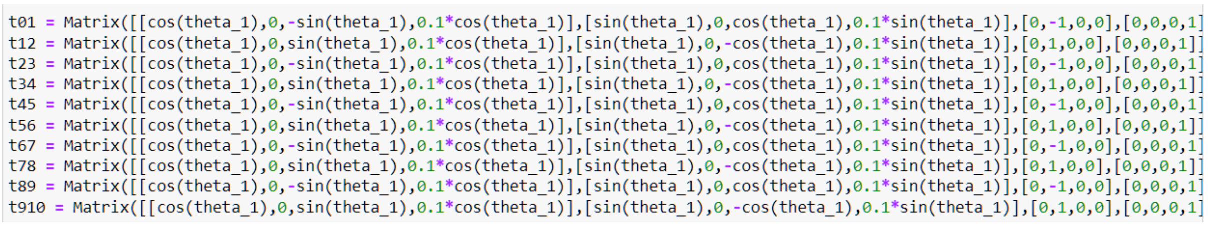
DH table for this robot

The X and Z axes are considered as it is given in the figure:

Frames	a (in m)	α (in degree)	θ (in degree)	d (in m)
Frame 0 – Frame 1	0.1	-90^0	θ_1	0
Frame 1 – Frame 2	0.1	90^0	θ_2	0
Frame 2 – Frame 3	0.1	-90^0	θ_3	0
Frame 3 – Frame 4	0.1	90^0	θ_4	0
Frame 4 – Frame 5	0.1	-90^0	θ_5	0
Frame 5 – Frame 6	0.1	90^0	θ_6	0
Frame 6 – Frame 7	0.1	-90^0	θ_7	0
Frame 7 – Frame 8	0.1	90^0	θ_8	0
Frame 8 – Frame 9	0.1	-90^0	θ_9	0
Frame 9 – Frame 10	0.1	90^0	θ_{10}	0

Note: The distance between the joints is assumed as 0.1m and the total length of the robot as 1m.

Transformation matrices:



```
t01 = Matrix([[cos(theta_1),0,-sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,cos(theta_1),0.1*sin(theta_1)],[0,-1,0,0],[0,0,0,1]
t12 = Matrix([[cos(theta_1),0,sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,-cos(theta_1),0.1*sin(theta_1)],[0,1,0,0],[0,0,0,1]]
t23 = Matrix([[cos(theta_1),0,-sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,cos(theta_1),0.1*sin(theta_1)],[0,-1,0,0],[0,0,0,1]
t34 = Matrix([[cos(theta_1),0,sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,-cos(theta_1),0.1*sin(theta_1)],[0,1,0,0],[0,0,0,1]]
t45 = Matrix([[cos(theta_1),0,-sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,cos(theta_1),0.1*sin(theta_1)],[0,-1,0,0],[0,0,0,1]
t56 = Matrix([[cos(theta_1),0,sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,-cos(theta_1),0.1*sin(theta_1)],[0,1,0,0],[0,0,0,1]]
t67 = Matrix([[cos(theta_1),0,-sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,cos(theta_1),0.1*sin(theta_1)],[0,-1,0,0],[0,0,0,1]
t78 = Matrix([[cos(theta_1),0,sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,-cos(theta_1),0.1*sin(theta_1)],[0,1,0,0],[0,0,0,1]]
t89 = Matrix([[cos(theta_1),0,-sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,cos(theta_1),0.1*sin(theta_1)],[0,-1,0,0],[0,0,0,1]
t910 = Matrix([[cos(theta_1),0,sin(theta_1),0.1*cos(theta_1)],[sin(theta_1),0,-cos(theta_1),0.1*sin(theta_1)],[0,1,0,0],[0,0,0,1]
```

Final Transformation matrix:

$${}^{0}T_{10} = {}^{0}T_{1} * {}^{1}T_{2} * {}^{2}T_{3} * {}^{3}T_{4} * {}^{4}T_{5} * {}^{5}T_{6}\, {}^{6}T_{7} * {}^{7}T_{8} * {}^{8}T_{9} * {}^{9}T_{10}$$

The final transformation matrix when all angles are zero:

$$\begin{bmatrix} 1 & 0 & 0 & 1.0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Forward Kinematics

- The forward kinematics for this unique robot involves giving necessary joint angle inputs so that the robot executes a desired motion (snake gait)
- The movement of the snake (gait) is oscillatory in nature, and it resembles a sine wave. Hence, I took this as a base to build upon my concept.

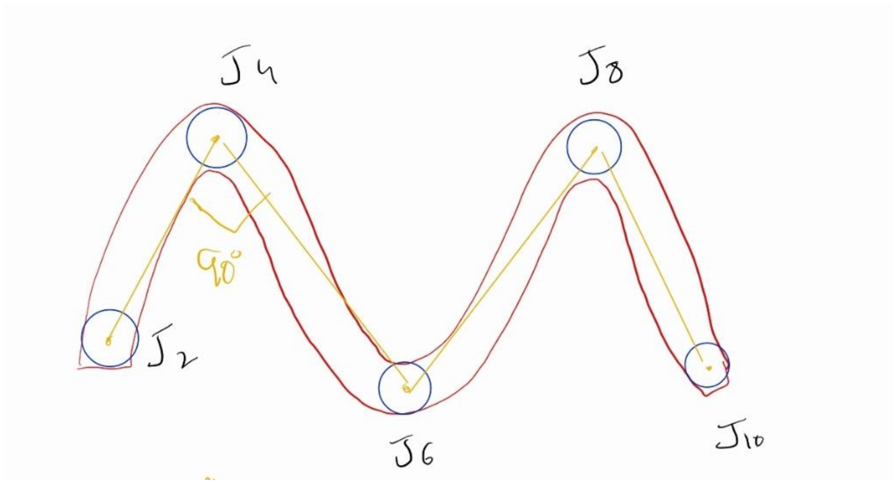


Figure A

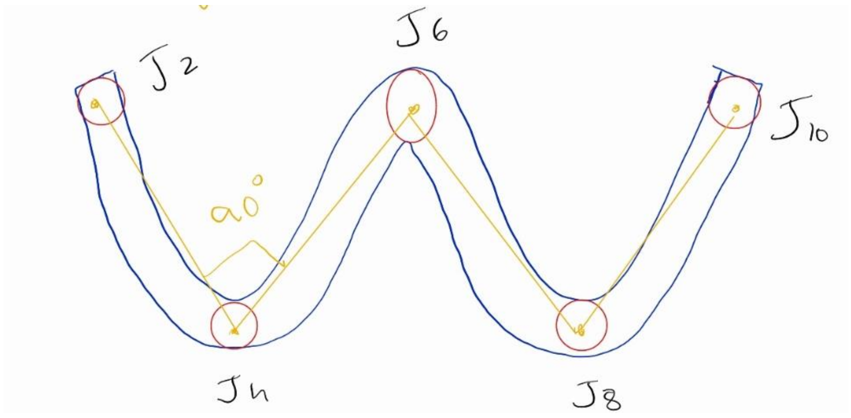


Figure B

- I observed that the movement in a gait requires consecutive joints (joints with axis perpendicular to the ground) to have a phase shift of some degrees, so that we get a continuous wave like form.
- In the figures attached above, I have considered a phase difference of 90 degrees between consecutive joints (joints with axis perpendicular to the ground)
- It is seen that, after time T/2 (where T is the time-period of the sine wave), the robot assumes the form as shown in 2nd figure.

- Hence, by giving a required **phase difference** between consecutive joints and by considering the oscillating behaviour of the end effector, we can achieve the required gait movement.
- In my case, I have considered that the joint actuator has the limits of $+\frac{\pi}{4}\ to -\frac{\pi}{4}$

Joint angle inputs considered for gait motion:

$$\theta_2 = 0.7853 * sin(0.25 * \pi * t)$$

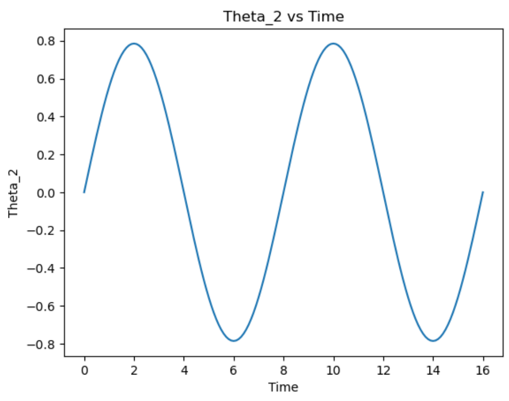
$$\theta_4 = 0.7853 * sin((0.25 * \pi * t)\ + (3.14 * 1))$$

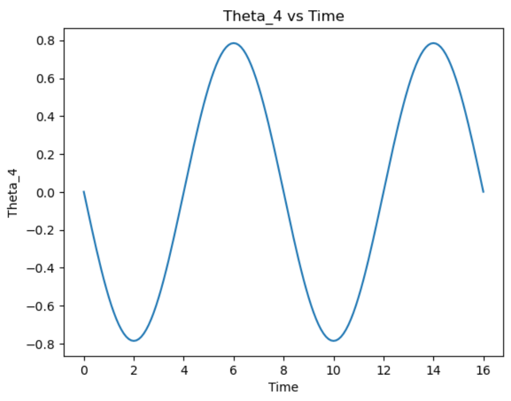
$$\theta_6 = 0.7853 * \sin((0.25 * \pi * t) + (3.14 * 2))$$

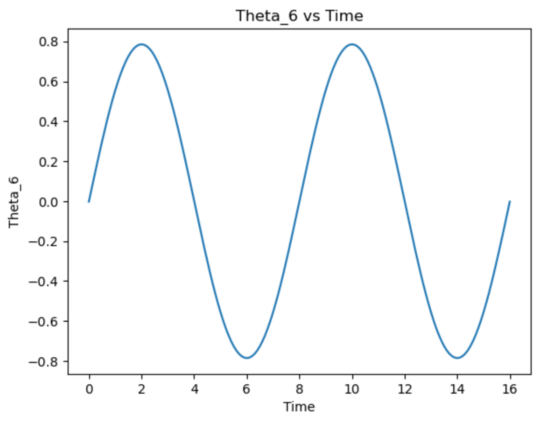
$$\theta_8 = 0.7853 * \sin((0.25 * \pi * t) + (3.14 * 3))$$

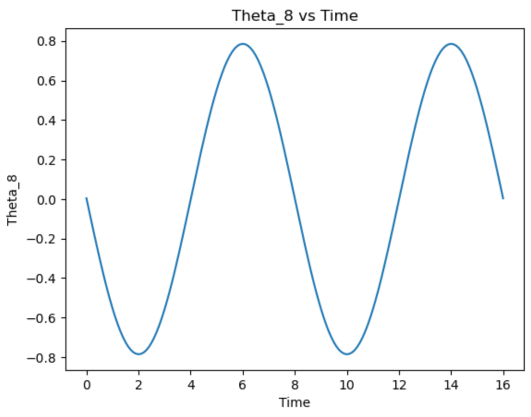
$$\theta_{10} = 0.7853 * \sin((0.25 * \pi * t) + (3.14 * 4))$$

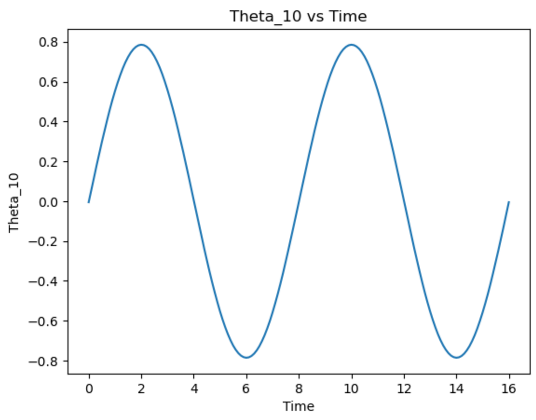
Angle plots:



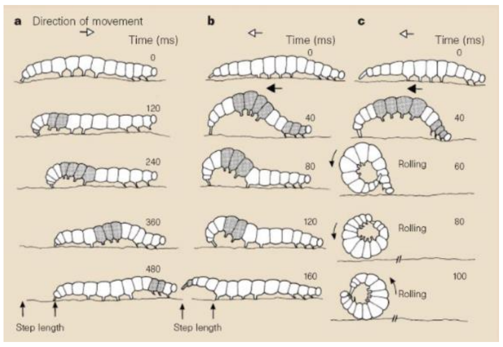




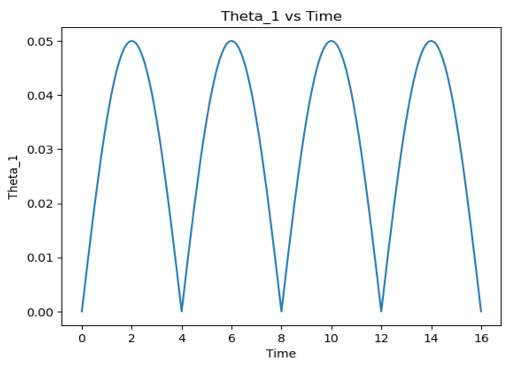




- To move the robot forward, we can make use of the remaining joints.
- The joints 1,3,5,7,9 have axes parallel to the ground plane. Hence, they cannot contribute to the gait movement.
- Instead, they can be used to propel the robot forward. I have considered 0 to $+\frac{\pi}{4}$ actuator motion for this purpose.
- This motion resembles like how worms move. They lift a part of their body up and propel themselves forward.
- But this method depends on the friction between the ground plane and the robot surface in contact, if the friction is less, there will be slip and the robot will not be able to move forward.



Hence the joint movement resembling this motion is given by:



Here the joint lifts the body of the robot by some small height and uses the friction to drive the robot forward.

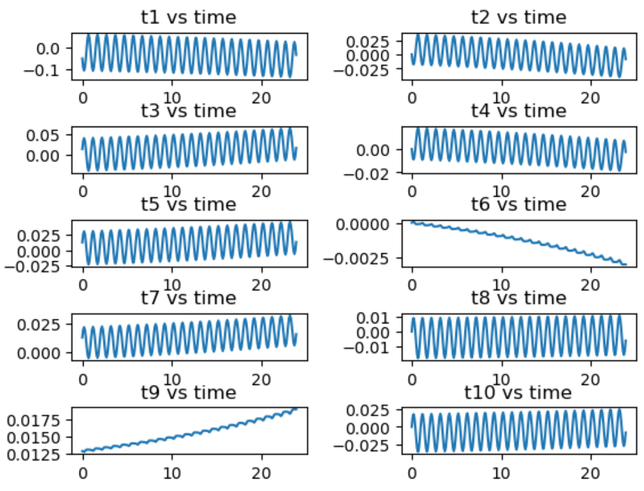
Attempt of Inverse velocity kinematics:

Steps:

- From the parameters in DH table, the Jacobian was calculated.
- Linear velocity matrix of the following was considered:

```
x_dot=0.5/8
y_dot=0
z_dot=(1*math.pi)*cos(2*math.pi*t)
phi_dot=0
theta_dot=0
psi_dot=0
```

- Here, as I wanted the robot to move linearly in the x direction and in a sinusoidal way in the z direction (where x-z forms the ground plane), I gave the inputs of scalar for x_dot and a sin wave for z_dot.
- From the obtained joint angular velocities, joint angles were calculated by numerical integration.
- The joint angle plots are as follows:



Alternative approach – Using Serpenoid curve formula to calculate the joint angles

The serpenoid curve formula proposed by Hirose [3], is one of the methods to calculate the joint angles of a snake like robot. Here, the joint angles are calculated using the formula,

$$\theta_i = -2\alpha_0 \sin(\frac{K_n * \pi}{L} s + \frac{2K_n * \pi}{n} i) + K_1 l$$

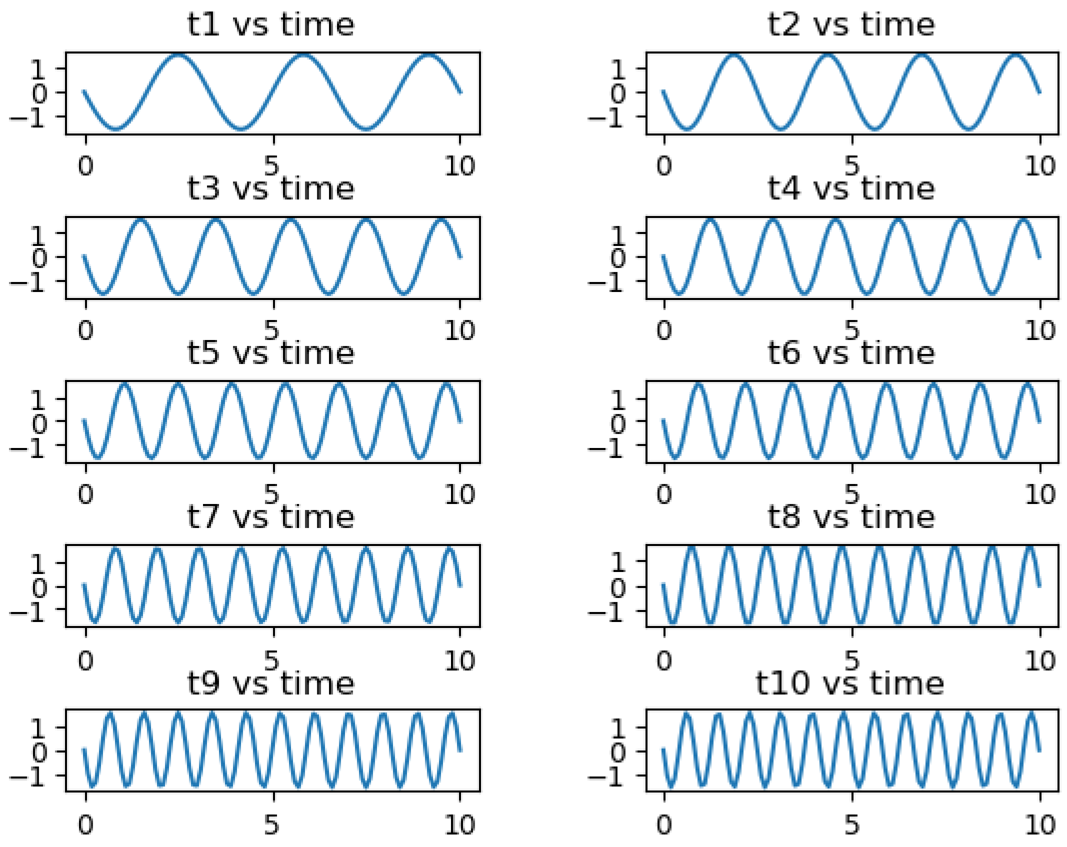
Where,

α_0 *is the initial angle,* K_n *is the number of period,* L *is the length of the robot*

i *is the joint number and* l *is the link length,* s *is the axial distance of the rear*

n *is the number of joints,* $K1$ *is the deflective curvature which is used for steering the bot*

Joint angle plot obtained from python code for the above formula:



I have considered the following:

α_0=45 degrees, L = 1m, $l = 0.1m$, s = 0.4 m, K1=0, n=10, i=1,2…n

References:

1) Spong, M. W., Hutchinson, S., & Vidyasagar, M. *Robot Modelling, and control*. (1st ed.). John Wiley & Sons, inc.
2) Brackenbury, J. Caterpillar kinematics. Nature 390, 453 (1997). https://doi.org/10.1038/37253
3) Mu, Zonggao & Wang, Haomiao & Xu, Wenfu & Liu, Tianliang & Wang, Hongtao. (2017). Two types of snake-like robots for complex environment exploration: Design, development, and experiment. Advances in Mechanical Engineering. 9. 168781401772185. 10.1177/1687814017721854.
4) https://www.instructables.com/Bioinspired-Robotic-Snake/
5) https://github.com/YogeshPhalak/Simulation-of-the-Snake-locomotion-mechanisms