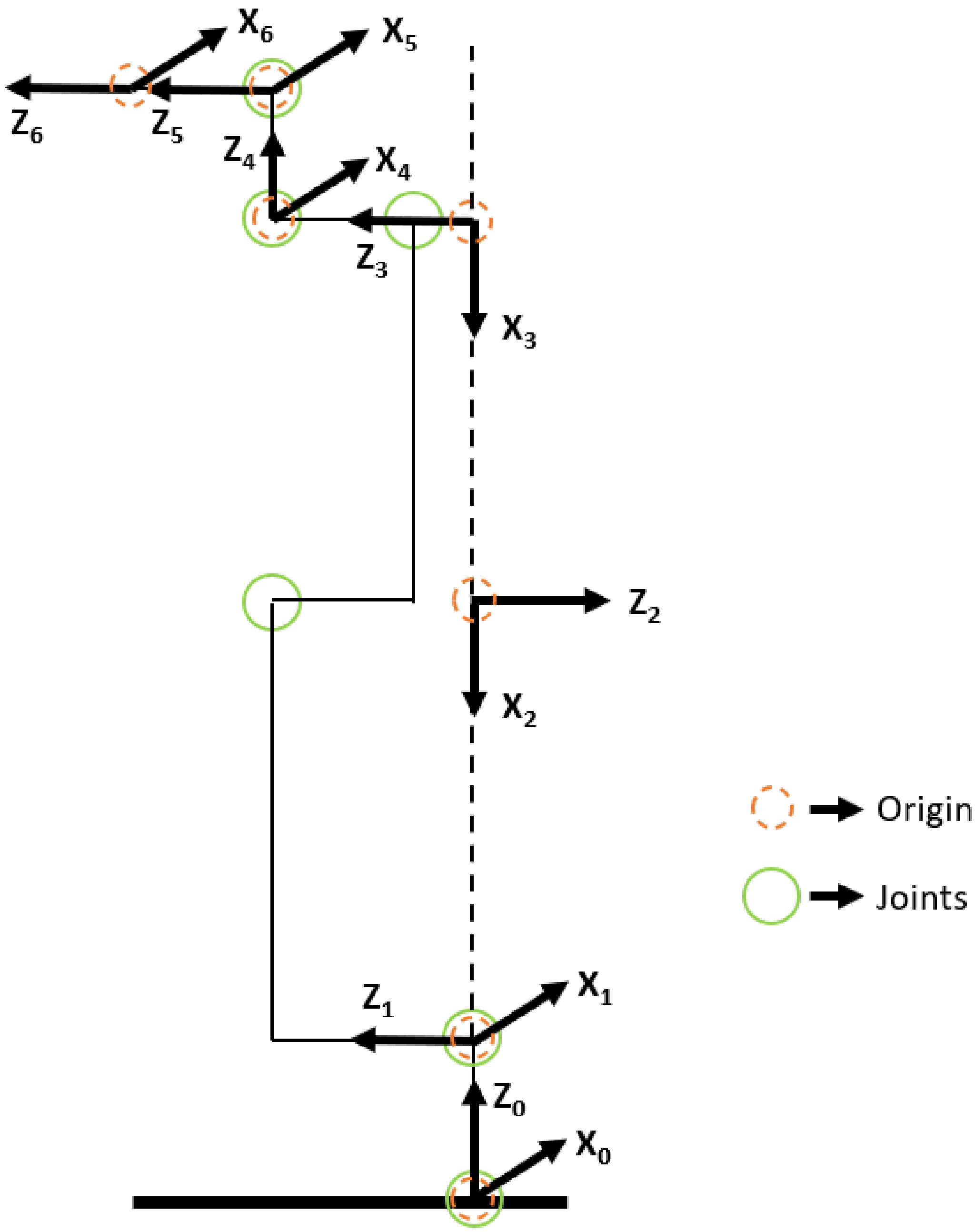
HOMEWORK 5

December 2023

1 Manipulator Dynamics - UR10

The DH coordinate frame for the robot is assigned as follows:

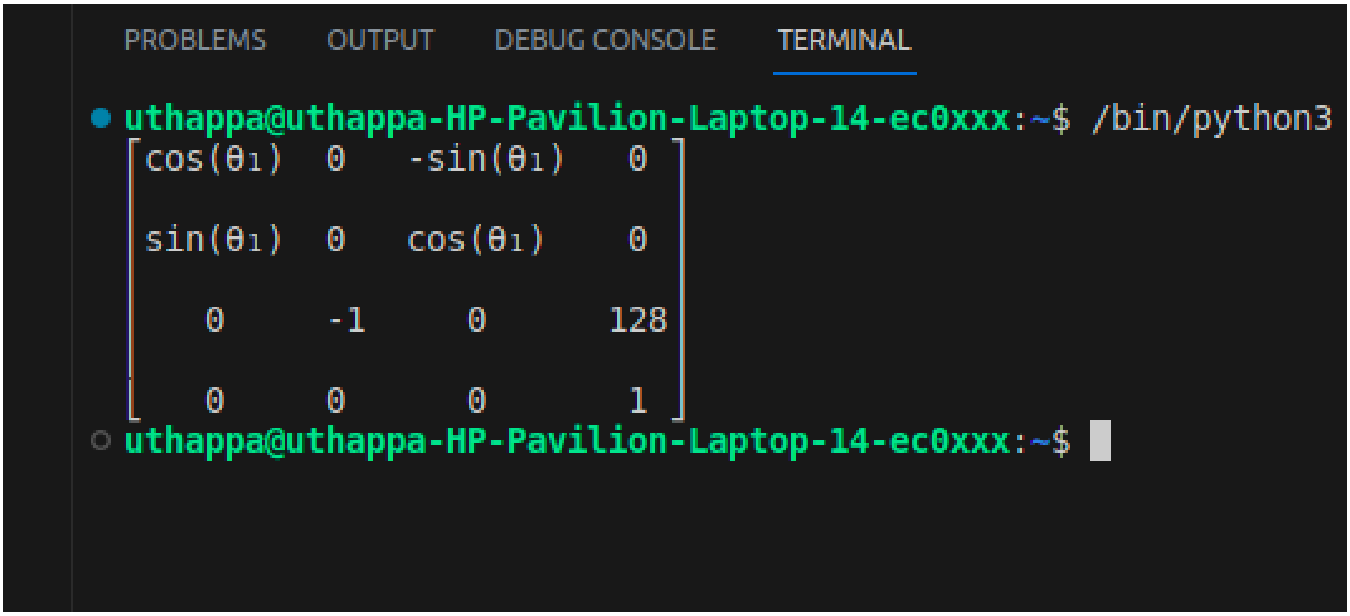


After assigning the DH coordinate frames for the robot, we construct the DH table as follows:

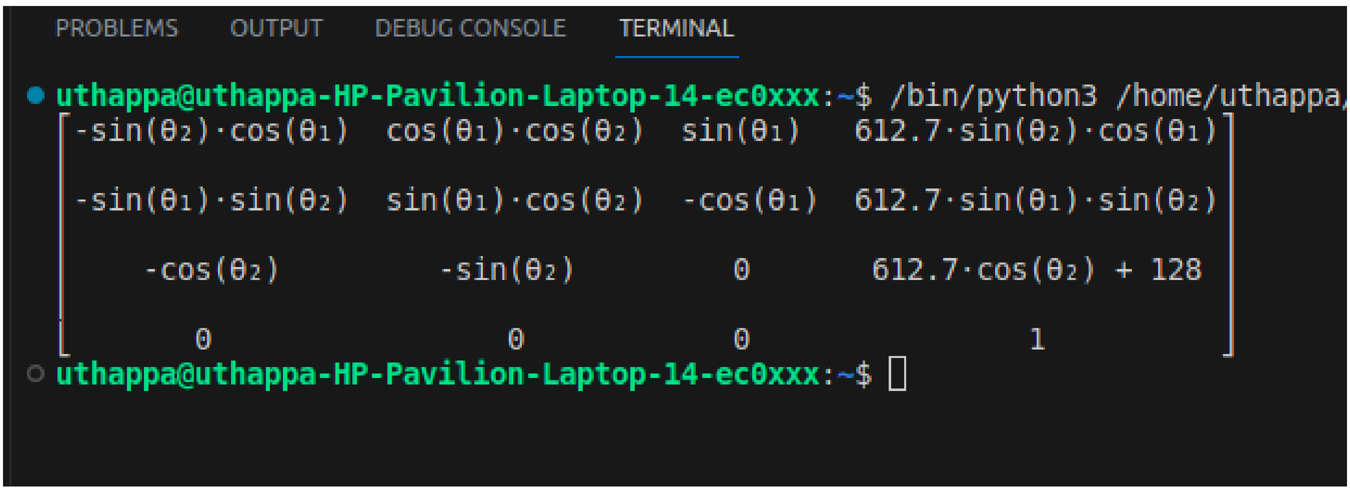
frame	a	α	d	θ
0 - 1	0	-90	128	θ_1
1 - 2	-612.7	180	0	90 + θ_2
2 - 3	-571.6	-180	0	θ_3
3 - 4	0	90	163.9	-90 + θ_4
4 - 5	0	-90	115.7	θ_5
5 - 6	0	0	192.2	θ_6

Once the DH parameters are set, the transformation matrices for each frame with respect to the base frame is noted down as follows:

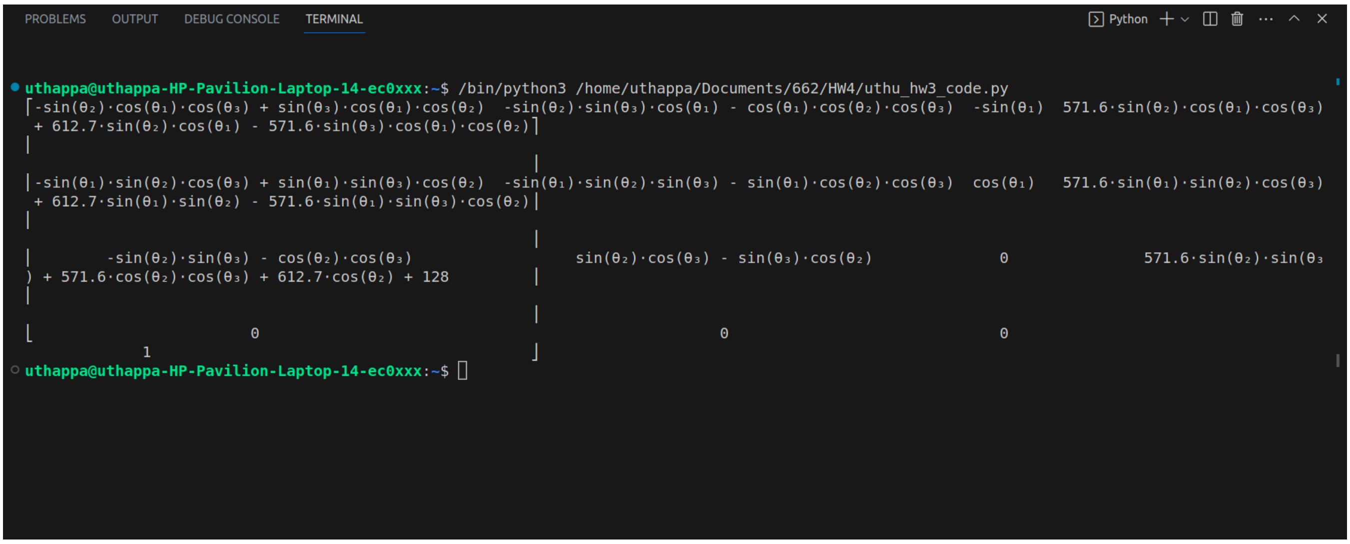
T_1^0 = Transformation of frame 1 with respect to the base frame.



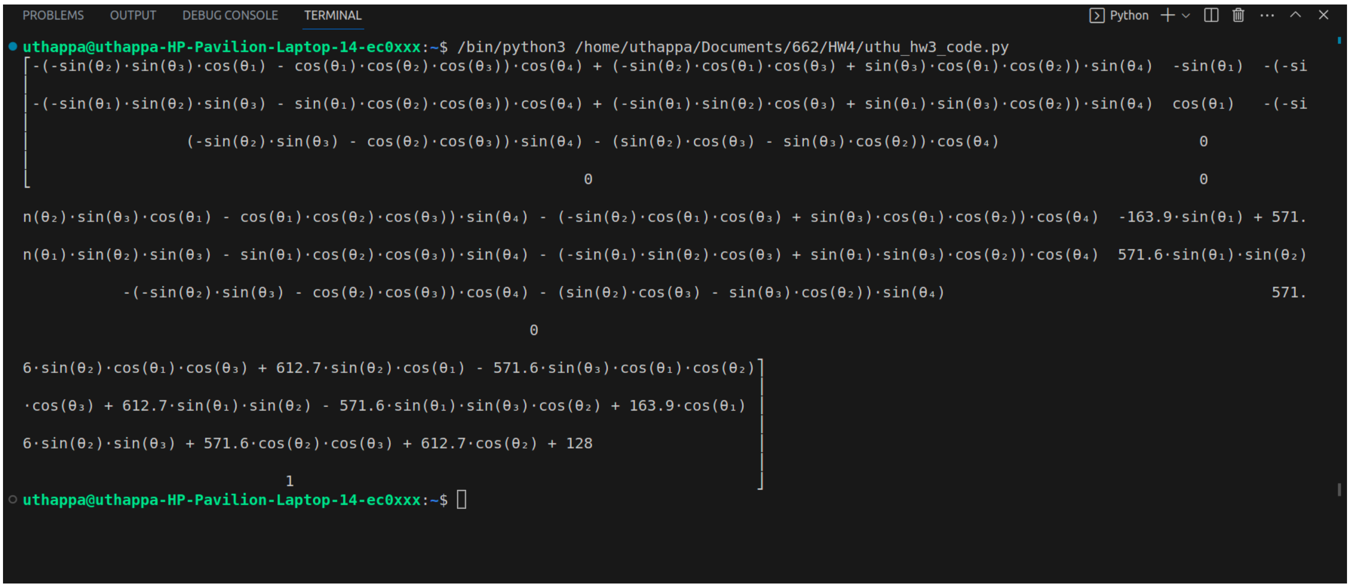
T_2^0 = Transformation of frame 2 with respect to the base frame.



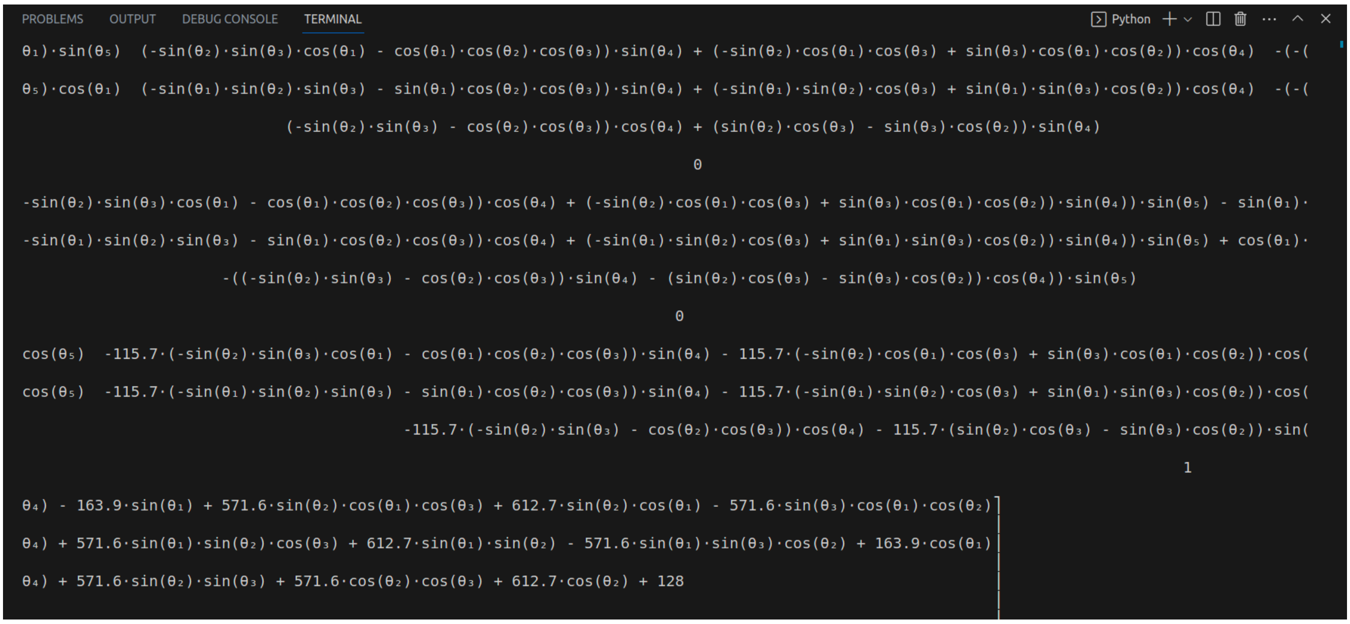
T_3^0 = Transformation of frame 3 with respect to the base frame.



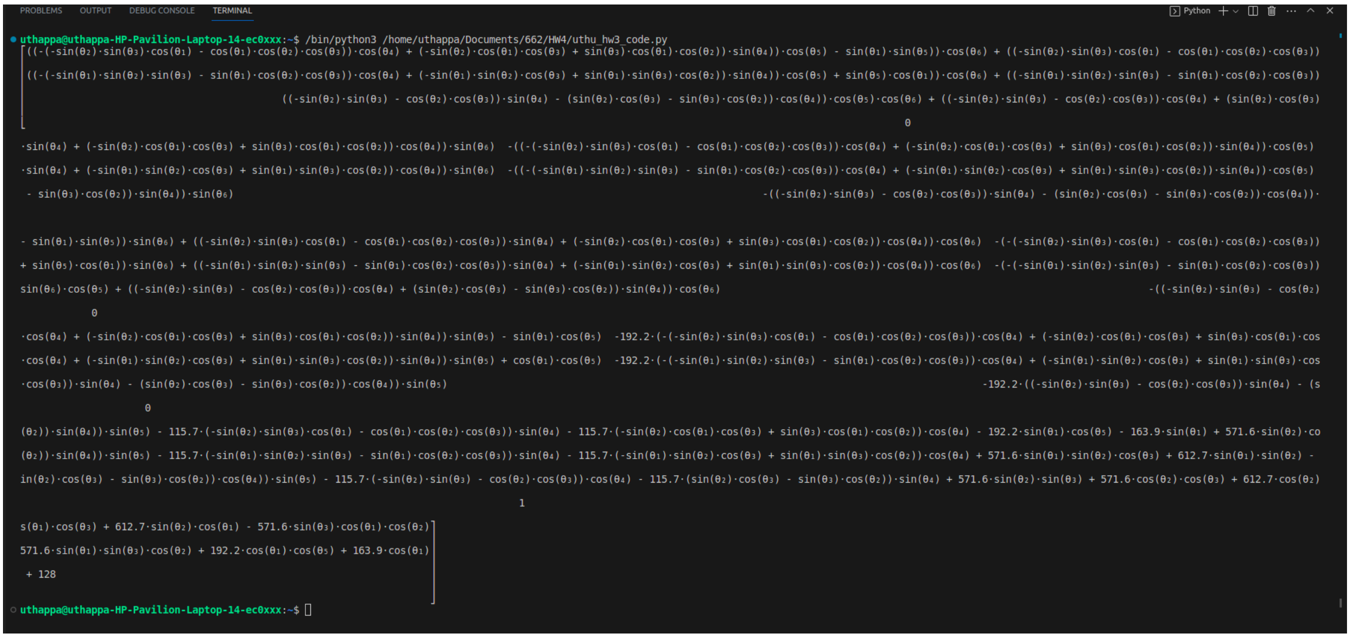
T_4^0 = Transformation of frame 4 with respect to the base frame.



T_5^0 = Transformation of frame 5 with respect to the base frame.



T_6^0 = Transformation of frame 6 with respect to the base frame.



Calculation of the jacobian:

Once the transformation matrices are established we proceed with the calculation of the jacobian.
Using method 2 for the calculation of the jacobian.

$$J = \begin{bmatrix} \frac{\partial P}{\partial q_1} & \frac{\partial P}{\partial q_2} & \frac{\partial P}{\partial q_3} & \frac{\partial P}{\partial q_4} & \frac{\partial P}{\partial q_5} & \frac{\partial P}{\partial q_6} \\ Z_0 & Z_1 & Z_2 & Z_3 & Z_4 & Z_5 \end{bmatrix}$$

We will first write down the Z_i part.

$$J_\omega = \begin{bmatrix} Z_0 & Z_1 & Z_2 & Z_3 & Z_4 & Z_5 \end{bmatrix}$$

Which is given by:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                                    Python

uthappa@uthappa-HP-Pavilion-Laptop-14-ec0xxx:~$ /bin/python3 /home/uthappa/Documents/662/HW4/uthu_hw3_code.py
⎡0  -sin(θ₁)  sin(θ₁)   -sin(θ₁)  -(-sin(θ₂)⋅sin(θ₃)⋅cos(θ₁) - cos(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) - (-sin(θ₂)⋅cos(θ₁)⋅cos(θ₃) + sin(θ₃)⋅cos(θ₁)⋅co
s(θ₂))⋅cos(θ₄)  -(-(-sin(θ₂)⋅sin(θ₃)⋅cos(θ₁) - cos(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅cos
⎢

⎢0  cos(θ₁)   -cos(θ₁)  cos(θ₁)   -(-sin(θ₁)⋅sin(θ₂)⋅sin(θ₃) - sin(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) - (-sin(θ₁)⋅sin(θ₂)⋅cos(θ₃) + sin(θ₁)⋅sin(θ₃)⋅co
s(θ₂))⋅cos(θ₄)  -(-(-sin(θ₁)⋅sin(θ₂)⋅sin(θ₃) - sin(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅cos
⎢

⎣1     0         0         0                   -(-sin(θ₂)⋅sin(θ₃) - cos(θ₂)⋅cos(θ₃))⋅cos(θ₄) - (sin(θ₂)⋅cos(θ₃) - sin(θ₃)⋅cos(θ₂))⋅sin(θ₄)
                                         -((-sin(θ₂)⋅sin(θ₃) - cos(θ₂)⋅cos

(θ₄) + (-sin(θ₂)⋅cos(θ₁)⋅cos(θ₃) + sin(θ₃)⋅cos(θ₁)⋅cos(θ₂))⋅sin(θ₄))⋅sin(θ₅) - sin(θ₁)⋅cos(θ₅)⎤
                                                                                              ⎥
(θ₄) + (-sin(θ₁)⋅sin(θ₂)⋅cos(θ₃) + sin(θ₁)⋅sin(θ₃)⋅cos(θ₂))⋅sin(θ₄))⋅sin(θ₅) + cos(θ₁)⋅cos(θ₅)⎥
                                                                                              ⎥
(θ₃))⋅sin(θ₄) - (sin(θ₂)⋅cos(θ₃) - sin(θ₃)⋅cos(θ₂))⋅cos(θ₄))⋅sin(θ₅)                          ⎦
```

We proceed with the calculation of J_v

$$J_v = \begin{bmatrix} \frac{\partial P}{\partial q_1} & \frac{\partial P}{\partial q_2} & \frac{\partial P}{\partial q_3} & \frac{\partial P}{\partial q_4} & \frac{\partial P}{\partial q_5} & \frac{\partial P}{\partial q_6} \end{bmatrix}$$

The function P is given by the end effector coordinates of the final transformation matrix. $P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$

P_x, P_y and P_z is as follows:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                                    Python

uthappa@uthappa-HP-Pavilion-Laptop-14-ec0xxx:~$ /bin/python3 /home/uthappa/Documents/662/HW4/uthu_hw3_code.py

P_x is as follows

-192.2⋅(-(-sin(θ₂)⋅sin(θ₃)⋅cos(θ₁) - cos(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅cos(θ₄) + (-sin(θ₂)⋅cos(θ₁)⋅cos(θ₃) + sin(θ₃)⋅cos(θ₁)⋅cos(θ₂))⋅sin(θ₄))⋅sin(θ₅) -
115.7⋅(-sin(θ₂)⋅sin(θ₃)⋅cos(θ₁) - cos(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) - 115.7⋅
(-sin(θ₂)⋅cos(θ₁)⋅cos(θ₃) + sin(θ₃)⋅cos(θ₁)⋅cos(θ₂))⋅cos(θ₄) - 192.2⋅sin(θ₁)⋅cos(θ₅) - 163.9⋅sin(θ₁) + 571.6⋅sin(θ₂)⋅cos(θ₁)⋅cos(θ₃) + 612.7⋅sin
(θ₂)⋅cos(θ₁) - 571.6⋅sin(θ₃)⋅cos(θ₁)⋅cos(θ₂)

P_y is as follows

-192.2⋅(-(-sin(θ₁)⋅sin(θ₂)⋅sin(θ₃) - sin(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅cos(θ₄) + (-sin(θ₁)⋅sin(θ₂)⋅cos(θ₃) + sin(θ₁)⋅sin(θ₃)⋅cos(θ₂))⋅sin(θ₄))⋅sin(θ₅) -
115.7⋅(-sin(θ₁)⋅sin(θ₂)⋅sin(θ₃) - sin(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) - 115.7⋅
(-sin(θ₁)⋅sin(θ₂)⋅cos(θ₃) + sin(θ₁)⋅sin(θ₃)⋅cos(θ₂))⋅cos(θ₄) + 571.6⋅sin(θ₁)⋅sin(θ₂)⋅cos(θ₃) + 612.7⋅sin(θ₁)⋅sin(θ₂) - 571.6⋅sin(θ₁)⋅sin(θ₃)⋅cos
(θ₂) + 192.2⋅cos(θ₁)⋅cos(θ₅) + 163.9⋅cos(θ₁)

P_z is as follows

-192.2⋅((-sin(θ₂)⋅sin(θ₃) - cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) - (sin(θ₂)⋅cos(θ₃) - sin(θ₃)⋅cos(θ₂))⋅cos(θ₄))⋅sin(θ₅) - 115.7⋅(-sin(θ₂)⋅sin(θ₃) - cos(θ₂)⋅
cos(θ₃))⋅cos(θ₄) - 115.7⋅(sin(θ₂)⋅cos(θ₃) - sin(θ₃)⋅cos(θ₂))⋅sin(θ₄) + 571.
6⋅sin(θ₂)⋅sin(θ₃) + 571.6⋅cos(θ₂)⋅cos(θ₃) + 612.7⋅cos(θ₂) + 128
uthappa@uthappa-HP-Pavilion-Laptop-14-ec0xxx:~$
```

These P_x, P_y and P_z are differentiated with respect to $theta1$, $theta2$, $theta3$, $theta4$, $theta5$ and $theta6$ respectively and then setup to form J_v.
Finally the jacobian matrix is setup as:

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$$

Since the Jacobian matrix is huge it is not included in the report and is present in the code. The jacobian matrix at home position of robot is obtained as follows:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

uthappa@uthappa-HP-Pavilion-Laptop-14-ec0xxx:~$ /bin/python3 /home/uthappa/Documents/662/HW4/uthu_hw3_code.py
⎡-356.1  1300.0  -687.3  115.7  -192.2  0⎤
⎢  0       0       0       0      0     0⎥
⎢  0       0       0       0      0     0⎥
⎢  0       0       0       0      0     0⎥
⎢  0       1      -1       1      0     1⎥
⎣  1       0       0       0      1     0⎦
```

Calculation of the g matrix:

We will first calculate the potential energy for individual link as follows:
Mass of each link is obtained from the UR10 data sheet present in the Universal Robotics webpage.
The mass of each link are as follows:
Mass of link1 = m1 = 7.1 kg
Mass of link2 = m2 = 12.7 kg
Mass of link3 = m3 = 4.27 kg
Mass of link4 = m4 = 2 kg
Mass of link5 = m5 = 2 kg
Mass of link6 = m5 = 0.365 kg

In order to calculate the potential energy we make a assumption that the mass is concentrated at the centre of mass of the link. With this assumption we proceed to calculate the distance of the centre of the mass from the base frame.
Height of link1 = h1 = $T_1[2,3]/2$
Height of link2 = h2 = $T_2[2,3]/2 + T_{01}[2,3]$
Height of link3 = h3 = $T_3[2,3]/2 + T_{02}[2,3]$
Height of link4 = h4 = $T_4[2,3]/2 + T_{03}[2,3]$
Height of link5 = h5 = $T_5[2,3]/2 + T_{04}[2,3]$
Height of link6 = h6 = $T_6[2,3]/2 + T_{05}[2,3]$

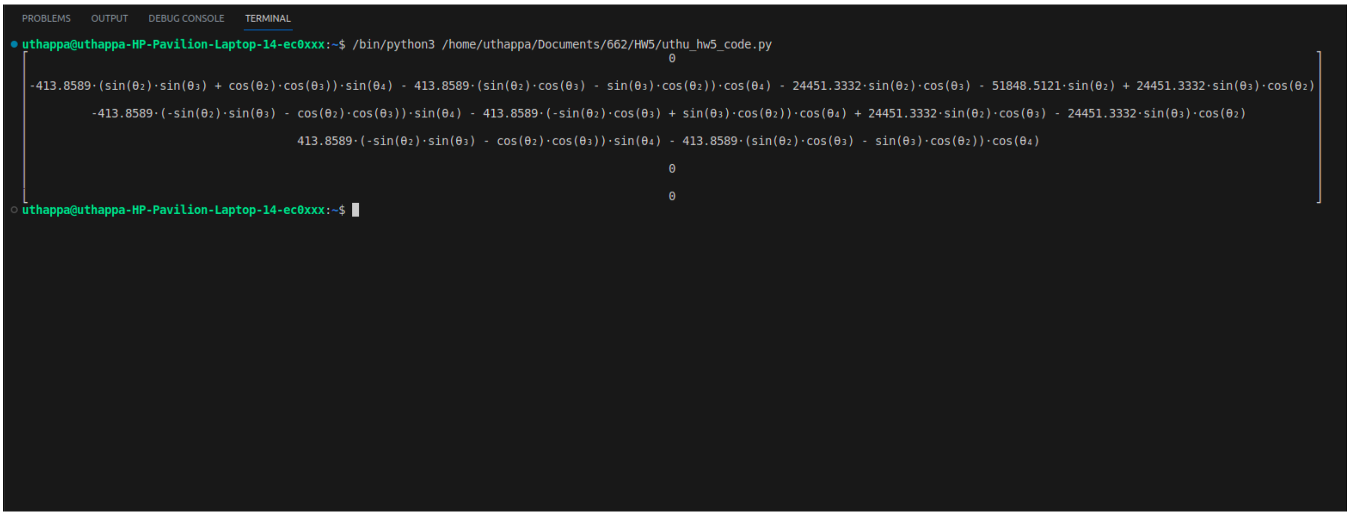
Potential energy for each link is then calculated as follows:
$PE_i = m_i * g * h_i$

The total potential energy is calculated as:
PE $= PE_1 + PE_2 + PE_3 + PE_4 + PE_5 + PE_6$

The individual components of the g matrix is obtained as follows:

$g_i = \frac{\partial PE}{\partial q_i}$

The g matrix is as follows:



```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
uthappa@uthappa-HP-Pavilion-Laptop-14-ec0xxx:~$ /bin/python3 /home/uthappa/Documents/662/HW5/uthu_hw5_code.py
⎡                                                                    0                                                                    ⎤
⎢-413.8589⋅(sin(θ₂)⋅sin(θ₃) + cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) - 413.8589⋅(sin(θ₂)⋅cos(θ₃) - sin(θ₃)⋅cos(θ₂))⋅cos(θ₄) - 24451.3332⋅sin(θ₂)⋅cos(θ₃) - 51848.5121⋅sin(θ₂) + 24451.3332⋅sin(θ₃)⋅cos(θ₂)⎥
⎢       -413.8589⋅(-sin(θ₂)⋅sin(θ₃) - cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) - 413.8589⋅(-sin(θ₂)⋅cos(θ₃) + sin(θ₃)⋅cos(θ₂))⋅cos(θ₄) + 24451.3332⋅sin(θ₂)⋅cos(θ₃) - 24451.3332⋅sin(θ₃)⋅cos(θ₂)⎥
⎢                        413.8589⋅(-sin(θ₂)⋅sin(θ₃) - cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) - 413.8589⋅(sin(θ₂)⋅cos(θ₃) - sin(θ₃)⋅cos(θ₂))⋅cos(θ₄)⎥
⎢                                                                    0                                                                    ⎥
⎣                                                                    0                                                                    ⎦
uthappa@uthappa-HP-Pavilion-Laptop-14-ec0xxx:~$
```

Computation of joint torques.

We setup the parametric equation of a circle as follows:
$x = r * sin(\theta)$
$z = r * cos(\theta)$
where θ is the angle subtended by the circle and r is the radius of the circle. Here r = 100mm.

Also the circle needs to be drawn in 200s, hence angular velocity $\omega = 2\pi/200$.

From the parametric equation we obtain the velocity $\dot{x}$ and $\dot{y}$.
$\dot{x} = r * cos(\theta) * \dot{\theta}$
$\dot{z} = -r * sin(\theta) * \dot{\theta}$

This forms the end effector velocity matrix as follows:

$$\dot{X} = \begin{bmatrix} \dot{x} \\ 0 \\ \dot{z} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In order to find out the joint velocities we make use of the equation:
$\dot{q} = J^{-1} * \dot{X}$

The Jacobian matrix is not invertible and hence we find the sudo inverse of it.

Once we obtain $\dot{q}$ we proceed with the numerical integration as follows:
$q = q + \dot{q} * dt$

From the Euler-lagrangian equation, we have:
$M\ddot{q} + C\dot{q} + g = \tau + J^T F$

Since the robot is quasi-static, $\ddot{q}$ and $\dot{q}$ is zero. The dynamic equation reduces to,
$g = \tau + J^T F$

The force F can be considered as follows:

$$F = \begin{bmatrix} 0 \\ -5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

We calculate the joint torques as follows:
$\tau = g - J^T F$

This joint torque values are calculated for different joint angles required to plot a circle.
All the calculations have been presented in the code.

The obtained plot for the joint torques is as follows: