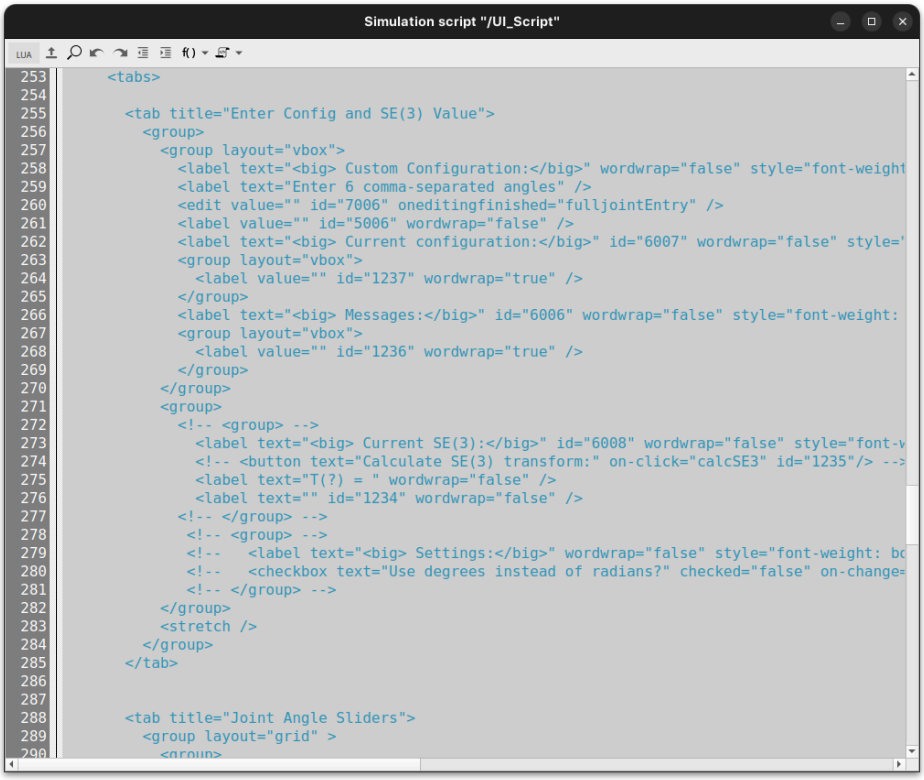
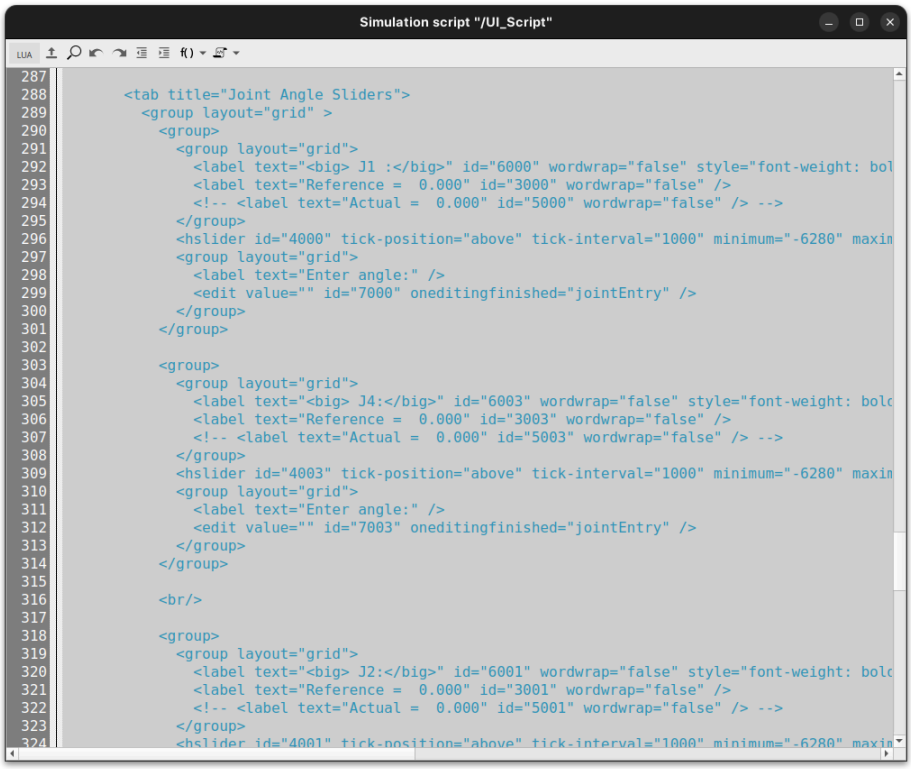
ME449 Assignment 1 - Pushkar Dave

Screenshot of UI Script.Title of configuration box has been changed to **‘Custom Configuration’**

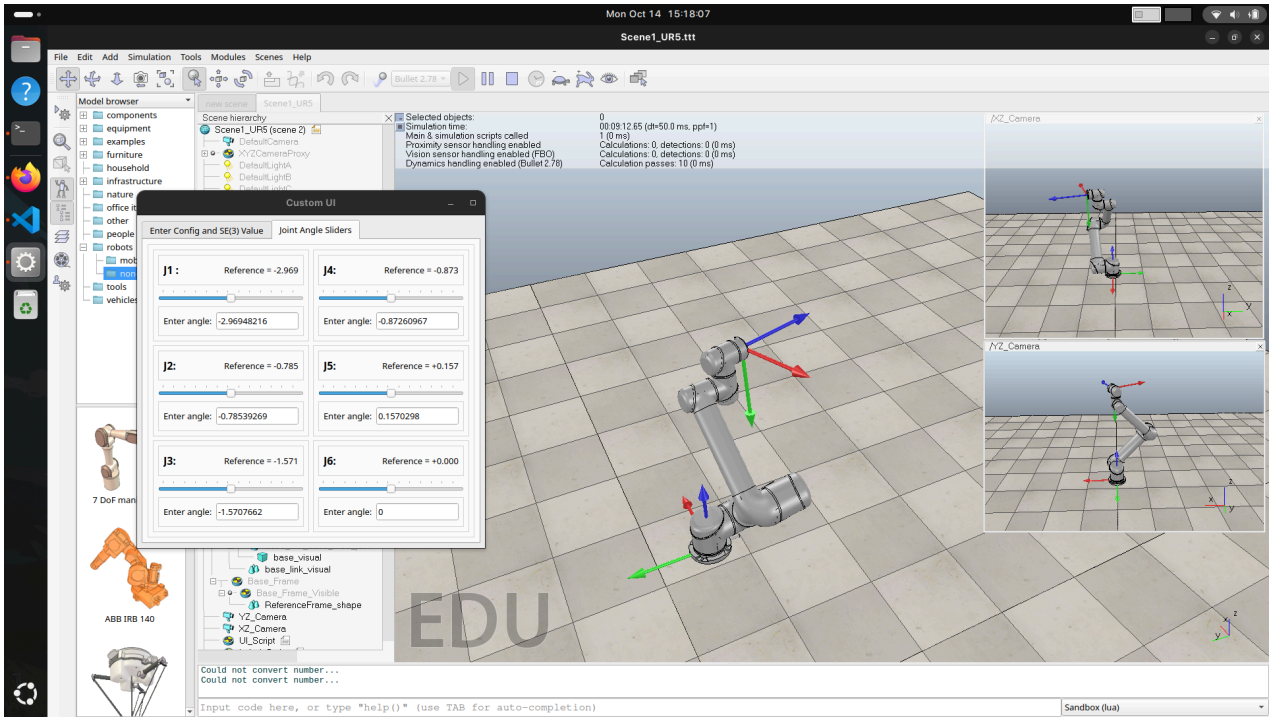


Screenshot of UI Script. Joint names have been changed from **Joint$_i$** to **J$_i$**

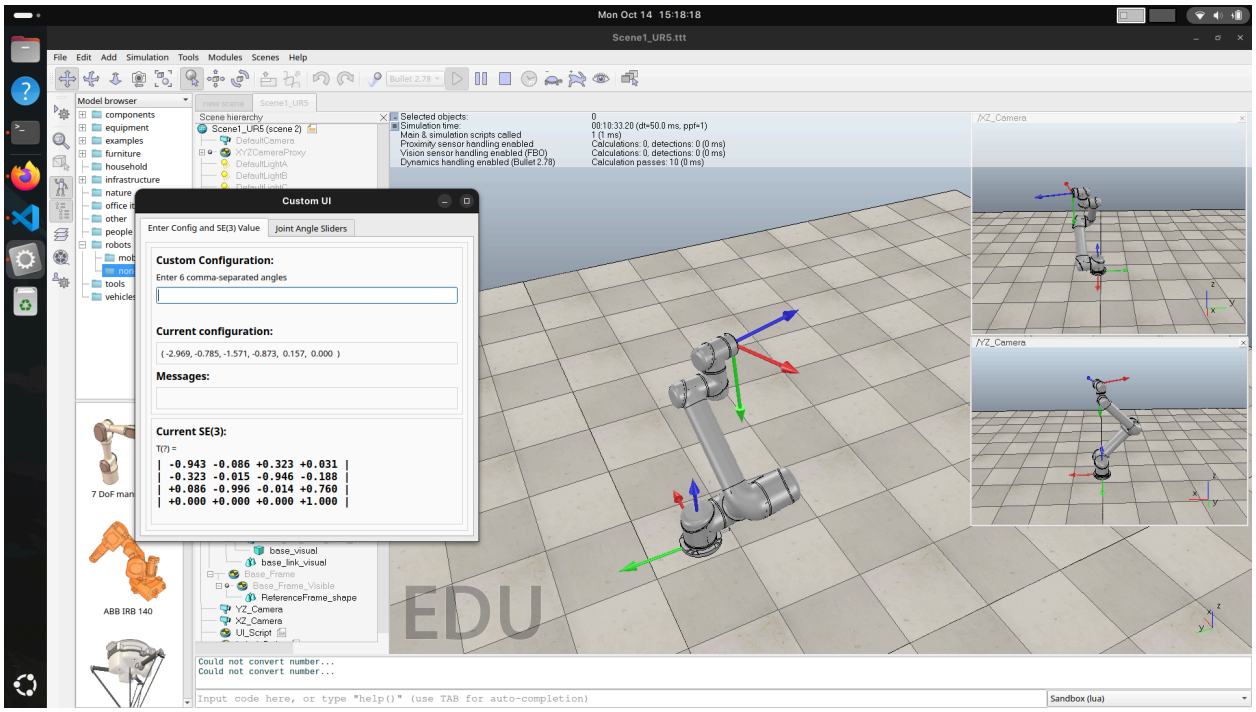


ME449 Assignment 1 - Pushkar Dave

Screenshot of the scene with robot configuration, modified UI and calculated joint angles as input.



Screenshot of the scene with robot configuration and SE(3) calculation



ME449 Assignment 1 - Pushkar Dave

Calculated joint angles: (θ_1, θ_2, θ_3, θ_4, θ_5 , θ_6):
[-2.969, -0.785, -1.570, -0.872, 0.157, 9.047e-07]
(Note: the last value is small enough to be approximated as zero)

Calculated SE(3) Matrix:
[[-0.9418 -0.0859 0.3249]
[-0.3249 -0.0151 -0.9456]
[0.0861 -0.9962 -0.0136]]

Code:

```
import modern_robotics as mr
import numpy as np

# provided rotation matrices
R13 = np.asanyarray([[-0.7071, 0, -0.7071], [0, 1, 0], [0.7071, 0,
-0.7071]])
Rs2 = np.asanyarray([[-0.6964, 0.1736, 0.6964], [-0.1228, -0.9848,
0.1228], [0.7071, 0, 0.7071]])
R25 = np.asanyarray([[-0.7566, -0.1198, -0.6428], [-0.1564, 0.9877, 0],
[0.6348, 0.1005, -0.7661]])
R12 = np.asanyarray([[0.7071, 0, -0.7071], [0, 1, 0], [0.7071, 0,
0.7071]])
R34 = np.asanyarray([[0.6428, 0, -0.7660], [0, 1, 0], [0.7660, 0,
0.6428]])
Rs6 = np.asanyarray([[0.9418, 0.3249, -0.0859], [0.3249, -0.9456,
-0.0151], [-0.0861, -0.0136, -0.9962]])
R6b = np.asanyarray([[-1, 0, 0], [0, 0, 1], [0, 1, 0]])

# rotation axes for the joints
w1 = np.asanyarray([0, 0, 1])
w2 = np.asanyarray([0, 1, 0]) # w3, w4, w6 have the same axis as w2
w5 = np.asanyarray([0, 0, -1])



# calculate the necessary transformation matrices
Rs1 = np.matmul(Rs2, mr.RotInv(R12))

# Next sequential R
R23 = np.matmul(mr.RotInv(R12),R13)

Rs3 = np.matmul(Rs2, R23)
```

```
# Intermediate steps to find R45
R41 = np.matmul(mr.RotInv(R34),mr.RotInv(R13))
R15 = np.matmul(R12, R25)

R45 = np.matmul(R41, R15)

Rs4 = np.matmul(Rs3,R34)

Rs5 = np.matmul(Rs4, R45)
# Intermediate steps to find R56
R26 = np.matmul(mr.RotInv(Rs2), Rs6)
R56 = np.matmul(mr.RotInv(R25),R26)


# calculating the vectors after taking matrix log of the rotation matrices
between the links
omega_1 = mr.so3ToVec(mr.MatrixLog3(Rs1))
omega_2 = mr.so3ToVec(mr.MatrixLog3(R12))
omega_3 = mr.so3ToVec(mr.MatrixLog3(R23))
omega_4 = mr.so3ToVec(mr.MatrixLog3(R34))
omega_5 = mr.so3ToVec(mr.MatrixLog3(R45))
omega_6 = mr.so3ToVec(mr.MatrixLog3(R56))

# calculating the joint angles
theta_1 = omega_1 * w1
theta_2 = omega_2 * w2
theta_3 = omega_3 * w2
theta_4 = omega_4 * w2
theta_5 = omega_5 * w5
theta_6 = omega_6 * w2

print(theta_1[2], theta_2[1], theta_3[1], theta_4[1], theta_5[2],
theta_6[1])

# calculating Rsb
Rsb = np.matmul(Rs6, R6b)
print(Rsb)
```

Code Explanation:

ME449 Assignment 1 - Pushkar Dave

Based on the given rotation matrices, the code first calculates the rotation matrices Rs1, R12, R34 (already given), R45 and R56 using the subscript cancellation rule (and matrix multiplication).
Then, we find the Matrix Log of these matrices and convert them into the vector form. When we multiply this vector with the vector representing the axis of rotation of the joint, we get the Joint Angle with appropriate sign.
For calculating Rsb, we multiply the given matrices Rs6 and R6b, and which gives us the matrix Rsb (based on subscript cancellation rule).