Introduction to Robotics: Homework II

Ονοματεπώνυμο: Ζωγράφου Μαρία-Νίκη

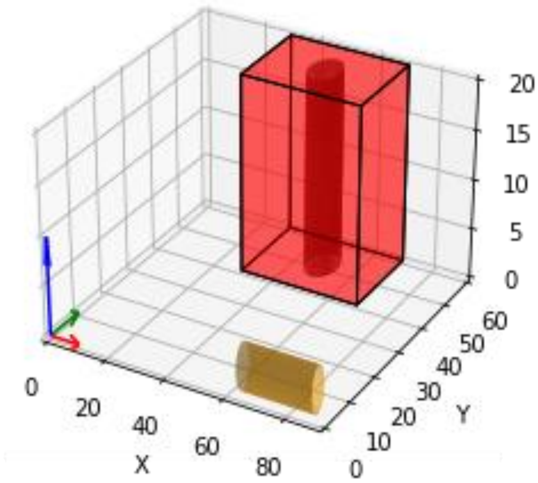
Αριθμός Μητρώου: 1096060

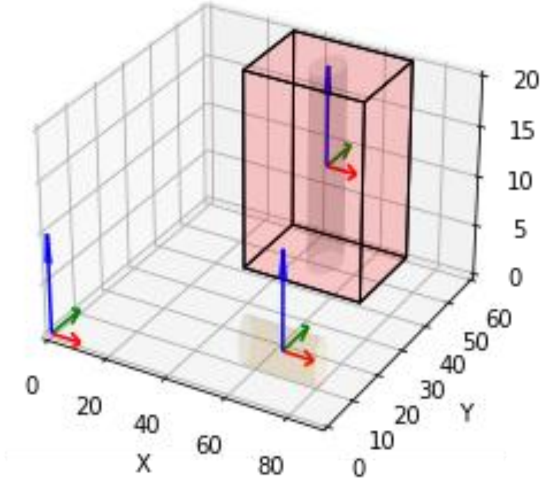
Περιεχόμενα

Ερωτήματα 1-4:

1. Choose the location of the world frame and assign a fixed body frame to each object;

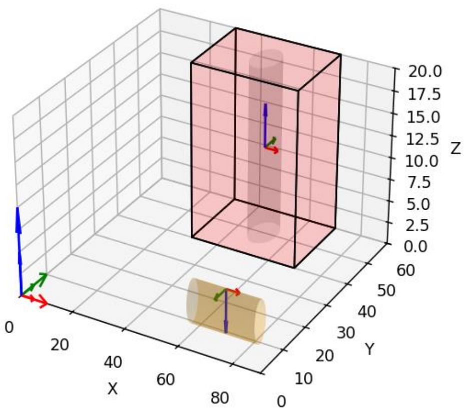
2. Select an appropriate transformation matrix





for the base frame of the manipulator with respect to the world frame, ensuring it can perform the manipulation task;

Τοποθετούμε το world frame στο (0,0,0) και το robot στο (0,0,0) για ευκολία Base Frame = World Frame. Red:x, Green:y, Blue:z.




```
box transformation matrix:
[[ 1.  0.  0. 50.]
 [ 0.  1.  0. 50.]
 [ 0.  0.  1. 10.]
 [ 0.  0.  0.  1.]]
horizontal cylinder transformation
matrix
[[ 1.   0.   0.   69.5]
 [ 0.   1.   0.    9.5]
 [ 0.   0.   1.    2.5]
 [ 0.   0.   0.    1. ]]

Franka Panda base frame transformation matrix:
[[ 1.  0.  0. 0.]
 [ 0.  1.  0. 0.]
 [ 0.  0.  1. 0.]
 [ 0.  0.  0. 1.]]
```

3. Determine the relative transformation matrix between the manipulator's end-effector and the cylinder, allowing the manipulator to grasp the cylinder (aka, what should be the orientation and position of the cylinder with respect to the end-effector such that it can be grasped);

```
relative transformation matrix between the manipulator's end-effe
ctor and the cylinder
[[ 1.0 0.0  0.0 0.0]
 [ 0.0 -1.0 0.0  0.0]
 [ 0.0 0.0 -1.0  2.5]
 [ 0.0 0.0  0.0  1.0]]
```

4. Calculate the world frame transformation matrix of the cylinder such that, when released, it will be positioned to drop precisely into the hole in the box;

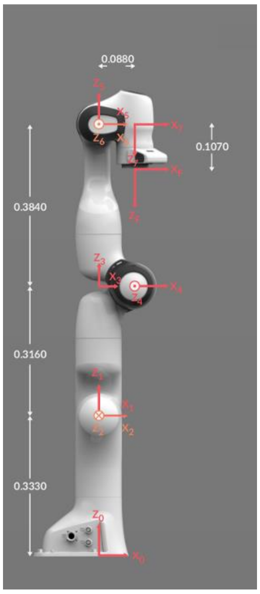
$$T = \begin{matrix} 0.0 & 0.0 & 1 & 0.475 \\ 0.0 & -1.0 & 0.0 & 0.5 \\ 1.0 & 0.0 & 0.0 & 0.325 \end{matrix}$$

Ερώτημα 5:

Using inverse kinematics calculate the joint positions of the arm such that the end-effector is able to grasp the cylinder

Υπολογισμός Screw Axis:

Γνωρίζουμε ότι $screw\ \boldsymbol{S} = (h, \boldsymbol{r}, \boldsymbol{q})$, όπου $\boldsymbol{r}$ μοναδιαίο διάνυσμα στην κατεύθυνση του άξονα περιστροφής και $\boldsymbol{q}$ σημείο πάνω στον άξονα. Γνωρίζουμε επίσης twist $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{r}\dot{\theta} \\ -\boldsymbol{r}\dot{\theta} \times \boldsymbol{q} + h\boldsymbol{r}\dot{\theta} \end{bmatrix}$, όπου το $h\boldsymbol{r}\dot{\theta}$ αντιπροσωπεύει το translation πάνω στον screw axis -μηδενίζεται στην περίπτωση μας- και το εξωτερικό γινόμενο $-\boldsymbol{r}\dot{\theta} \times \boldsymbol{q}$ που αντιπροσωπεύει την κίνηση λόγω της περιστροφής γύρω από τον άξονα. Προτιμούμε να χρησιμοποιούμε αντί για το προηγούμενο $\boldsymbol{S}$, το $screw\ \boldsymbol{S} = \frac{\boldsymbol{V}}{\|\boldsymbol{\omega}\|}$ (κανονικοποιημένο ως προς το μέτρο της γωνιακής ταχύτητας). Έχουμε λοιπόν:



- **Joint 1:** z translation=0 και περιστροφή κατά z άρα q=[0 0 0]^T και r=[0 0 1]^T, οπότε $-\boldsymbol{r}\dot{\theta} \times \boldsymbol{q} = [\boldsymbol{0\ 0\ 0}]^{\boldsymbol{T}}$, επομένως **S= [0, 0, 1, 0, 0, 0]^T.**
- **Joint 2:** z translation=0.333 και r= [0 1 0]^T επομένως **S= [0, 1, 0, -0.333, 0, 0]^T.**
- **Joint 3:** z translation=0.649 άρα q=[0 0 0.649]^T και άξονας r=[0 0 1]^T, επομένως **q**×**r**=[0 0 0]^T. **S=[0 0 1 0 0 0]^T**.
- **Joint 4: S=[0, -1, 0, 0.649, 0,-0.088]^T**
- **Joint 5: S= [0, 0, 1, 0, 0, 0]^T**
- **Joint 6: S= [0, -1, 0, 1.033, 0, 0]^T**
- **Joint 7: r=[0 0 -1]^T** και **q=[0.088 0 0.926]^T** άρα **S= [0, 0, -1, 0, 0.088, 0]^T**

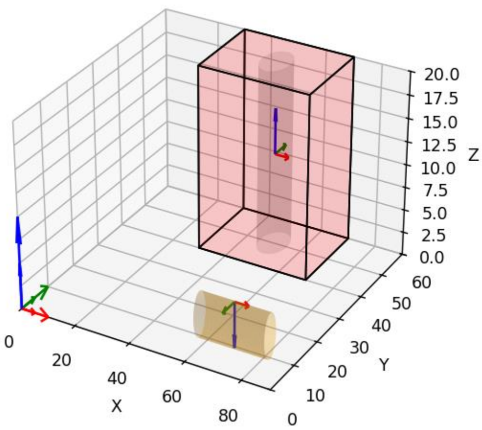
Υπολογισμός Home Configuration M:

Ορίζουμε M να είναι η θέση και προσανατολισμός του end effector όταν όλες οι γωνίες των joints ρυθμιστούν στο μηδέν (μηδενική θέση του ρομπότ). M=T01*T02*...T07, όπου η τελευταία στήλη είναι το translation και στις πρώτες 2 στήλες και πρώτες 2 σειρές φαίνεται το rotation.

Περιστροφή: στο ρομποτικό βραχίονα ο end effector είναι ανάποδα από τον αρχικό προσανατολισμό του world frame κατά y και κατά z. Το x παραμένει με ίδιο προσανατολισμό. Δηλαδή $y_{new} = -1y_{wf}$ και $z_{new} = -1z_{wf}$ και $x_{new} = x_{wf}$.

Υπάρχει επίσης μετακίνηση κατά x=*0.088* και κατά z=*0.926*.

$$\boldsymbol{M}= \begin{matrix} 1 & 0 & 0 & 0.088 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.926 \\ 0 & 0 & 0 & 1 \end{matrix}$$



Υπολογισμός Επιθυμητής Τελικής Θέσης:

Θέλουμε ο end effector στη μέση του μήκους του κυλίνδρου και από πάνω του (0.695,0.095,0.05).

Λαμβάνουμε υπόψιν μας και τον προσανατολισμό που πρέπει να έχει ο end effector για να υπολογίσουμε το Target pose. Επομένως:

```
    target_pose = np.array([
        [1.0, 0.0,  0.0, 0.695],
        [0.0, -1.0, 0.0,  0.0950],
        [0.0, 0.0, -1.0,  0.05],
        [0.0, 0.0,  0.0,  1.0]])
```

Προσανατολισμός του end effector: $x{=}x_{wf}$, $y{=}{-}y_{wf}$, $z{=}{-}z_{wf}$ (όπως στο home configuration).

Εφαρμογή των τύπων του forward kinematics με power exponentials με Python:

```
def poe_forward_kinematics(screw_axes, joint_angles,M):
# Initialize the transformation matrix to identity
    T = np.eye(4)
    # List to store joint positions, starting from the base frame
    positions = [[0, 0, 0]]#The base frame origin assumed to be [0,0,0]
# Iterate over each joint to compute the transformation step by step
    for i in range(len(joint_angles)):
        # Construct the twist matrix (se(3)) from the screw axis
        w_hat = np.zeros((4, 4))
        w_hat[:3, :3] = hat(screw_axes[:3, i])  # Angular velocity
(skew-symmetric)
        w_hat[:3, 3] = screw_axes[3:, i]       # Linear velocity
      #e^sθ
        exp_twist = expm(w_hat * joint_angles[i])
        # Update the overall transformation matrix
        T = T @ exp_twist
        # Compute the current position
        T_current = T @ M
        positions.append(T_current[:3, 3])  # Extract the position
    # Compute the final end-effector transformation relative to the
base frame
    end_effector_position = T @ M
    return end_effector_position, positions
```

Με την χρήση Newton-Raphson εφαρμόζουμε inverse kinematics στην python. Στόχος των inverse kinematics είναι να υπολογιστούν οι γωνίες των αρθρώσεων θ=[θ1,θ2,…,θn] του ρομποτικού βραχίονα ώστε το τελικό άκρο (end-effector) να φτάσει στην επιθυμητή θέση και προσανατολισμό. Αρχική υπόθεση για Newton Raphson (7x7 πίνακας με στοιχεία τιμής 0.6):

Το robot όντως φτάνει στην επιθυμητή θέση μετά από 11 επαναλήψεις. Το τελικό σφάλμα θέσης είναι 4.787669461820572e-05

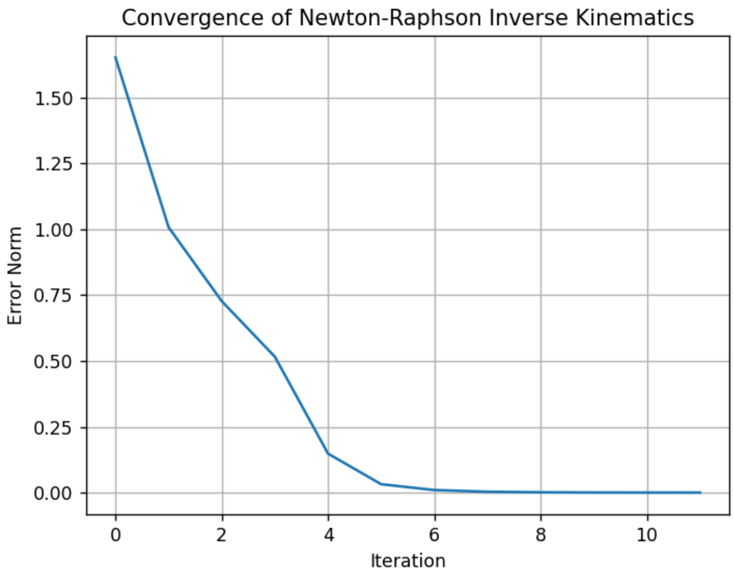
Verify your result with forward kinematics

```
Final Pose from Forward Kinematics:
 [[ 1.00000000e+00 -3.16997605e-07  1.67444021e-06  6.94954237e-01]
 [-3.16997532e-07 -1.00000000e+00 -4.35735782e-08  9.49931656e-02]
 [ 1.67444022e-06  4.35730474e-08 -1.00000000e+00  5.00120603e-02]
 [ 1.09426672e-16 -1.87551326e-17  2.22044605e-16  1.00000000e+00]]
```

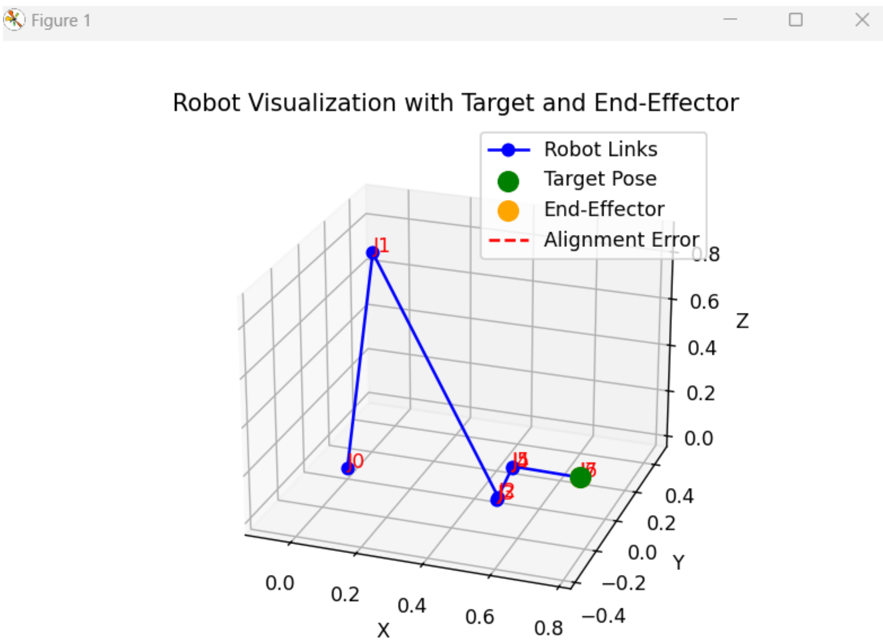
Βλέπουμε ότι με στρογγυλοποίηση είμαστε στο επιθυμητό target pose.

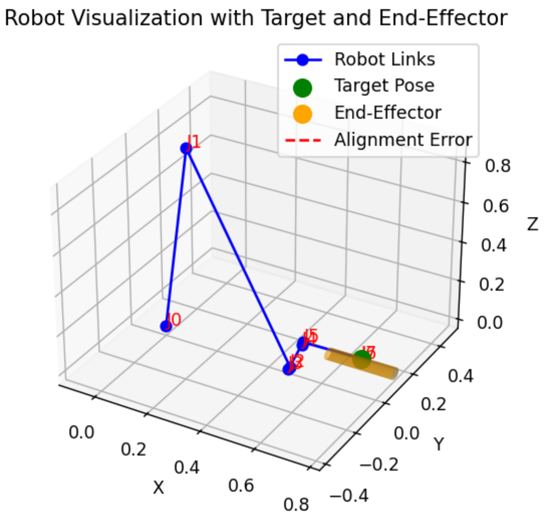
Επιτυχής Σύγκλιση Newton-Raphson:

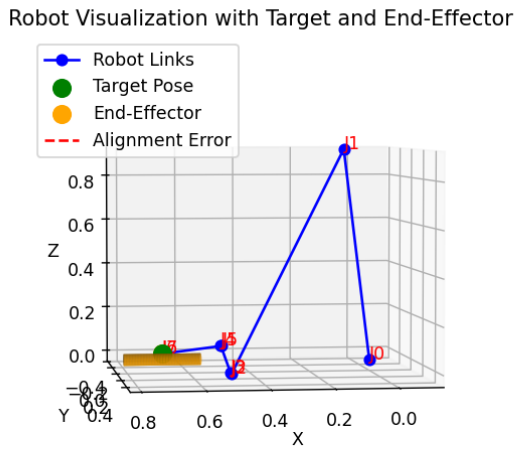
```
iteration 0 error norm = 1.6534
iteration 1 error norm = 1.0081
iteration 2 error norm = 0.7273
iteration 3 error norm = 0.5161
iteration 4 error norm = 0.1480
iteration 5 error norm = 0.0319
iteration 6 error norm = 0.0097
iteration 7 error norm = 0.0033
iteration 8 error norm = 0.0011
iteration 9 error norm = 0.0004
iteration 10 error norm = 0.0001
iteration 11 error norm = 0.0000
Converged after 11 iterations.
```

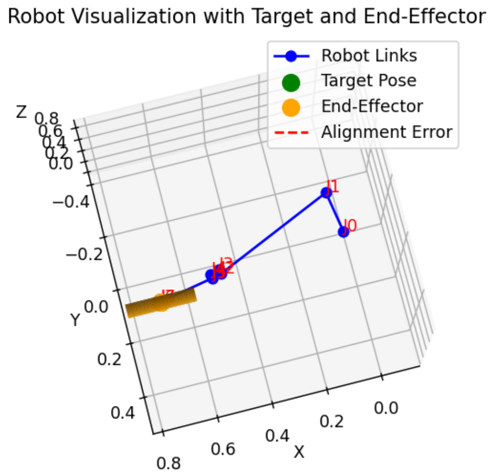


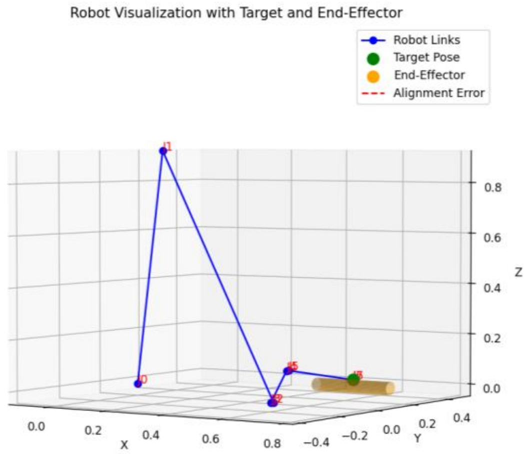
Αναπαράσταση του ρομπότ:











Τελικές Θέσεις των Αρθρώσεων:

```
Joint Position:
Joint0: [0, 0, 0]
Joint1: [0.08774949 0.00663525 0.926      ]
Joint2: [ 0.46201873  0.03493593 -0.04740922]
Joint3: [ 0.46430801  0.01717852 -0.04583501]
Joint4: [0.49980421 0.0459517  0.07782033]
Joint5: [0.49711737 0.06108508 0.07647082]
Joint6: [0.69495424 0.09499317 0.05001206]
Joint7: [0.69495424 0.09499317 0.05001206]
```

Τελικές Γωνίες των Αρθρώσεων:

```
Joint Angles:
Joint theta0: 0.07547220840406976
Joint theta1: 2.110900381730033
Joint theta2: -0.2046003154601239
Joint theta3: 0.48052166414480674
Joint theta4: 0.175556963772218
Joint theta5: 1.638666126760329
Joint theta6: 0.16974519941867655
```

Ερώτημα 6:

Using inverse kinematics calculate the joint positions of the manipulator such that the cylinder (grasped by the arm) is positioned in the "placing" position above the hole.

Υπολογισμός Επιθυμητής Τελικής Θέσης:

Ο κύλινδρος πρέπει να ευθυγραμμιστεί με την κυλινδρική τρύπα του κουτιού. Ο end effector πρέπει επομένως να βρίσκεται πάνω στην περίμετρο της κυλινδρικής τρύπας.

$Translation: x = 45\ cm, y = 50\ cm, z = 32.5\ cm.$

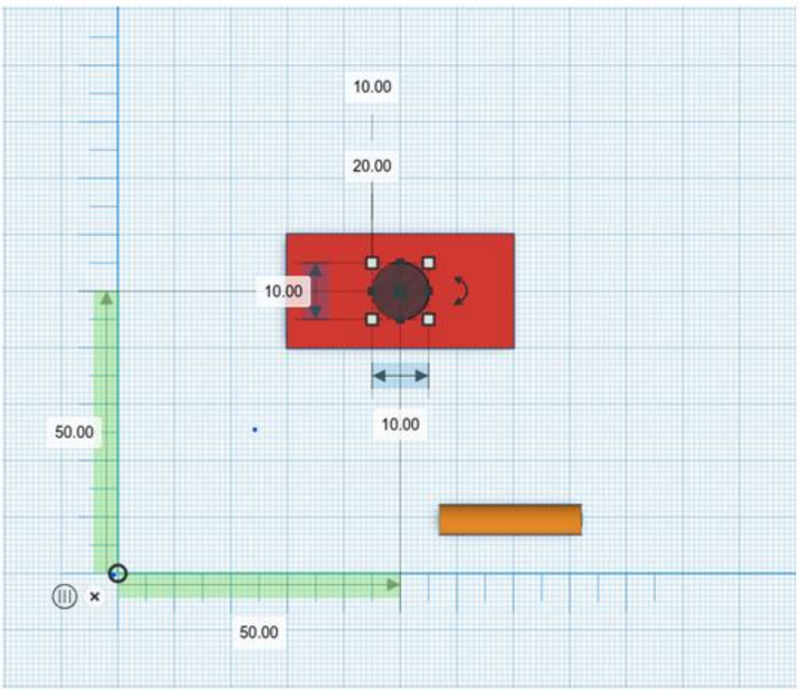
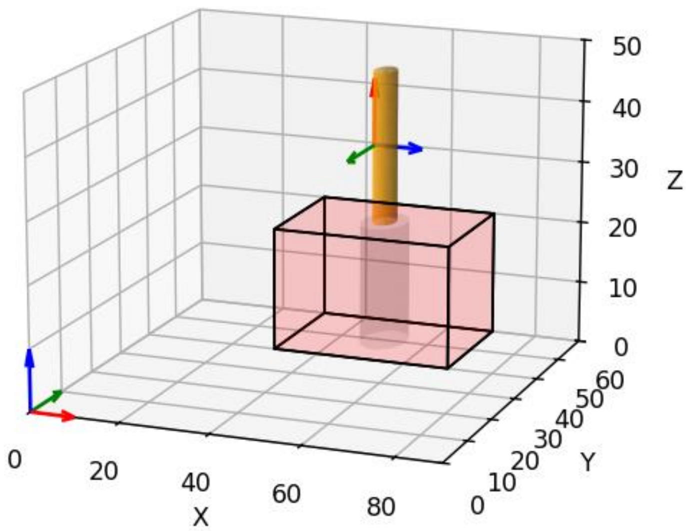


Figure 5: Scenario measurements. **Measurements are in cm.**

Όσον αφορά τον προσανατολισμό ο end-effector χρειάζεται να περιστρέψει τον κύλινδρο κατά 90 μοίρες γύρω από τον y. $x_{new} = z_{wf},\ y_{new} = -y_{wf},\ \ z_{new} = x_{wf}$. Μπορούμε επίσης να το υπολογίσουμε και ως R6= RotX(0)@RotY(np.pi)@RotZ(np.pi)@RotY(np.pi/2).

```
target_pose = np.array([
    [0.0, 0.0,  1.0, 0.475],
    [0.0, -1.0, 0.0,  0.50],
    [1.0, 0.0, 0.0,  0.325],
    [0.0, 0.0,  0.0,  1.0]])
```

Σημείωση: Παρατηρούμε ότι ο κύλινδρος έχει μισή ακτίνα από την κυλινδρική τρύπα, οπότε έχουμε περιθώριο για το πού θέλουμε να τοποθετηθεί. Έστω ότι θέλουμε να είναι στη μέση της τρύπας, άρα ο end-effector στο *x=0.475m.*

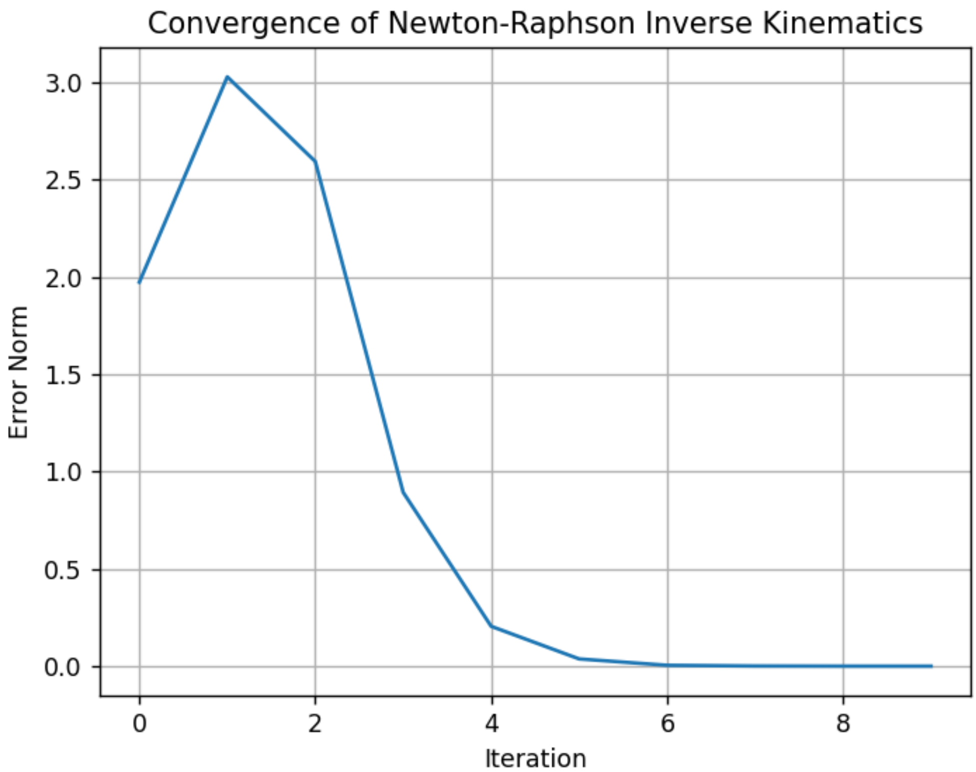


Όπως και προηγουμένως με την χρήση Newton-Raphson εφαρμόζουμε inverse kinematics στην python. Κρατάμε το ίδιο αρχικό guess initial_guess = np.ones(7)*0.6.

Το robot όντως φτάνει στην επιθυμητή θέση (Pose error= 4.8349700742601593e-05).

Επιτυχής Σύγκλιση Newton-Raphson:

```
iteration 0 error norm = 1.9736
iteration 1 error norm = 3.0289
iteration 2 error norm = 2.5942
iteration 3 error norm = 0.8931
iteration 4 error norm = 0.2049
iteration 5 error norm = 0.0372
iteration 6 error norm = 0.0045
iteration 7 error norm = 0.0010
iteration 8 error norm = 0.0002
iteration 9 error norm = 0.0000
Converged after 9 iterations.
```



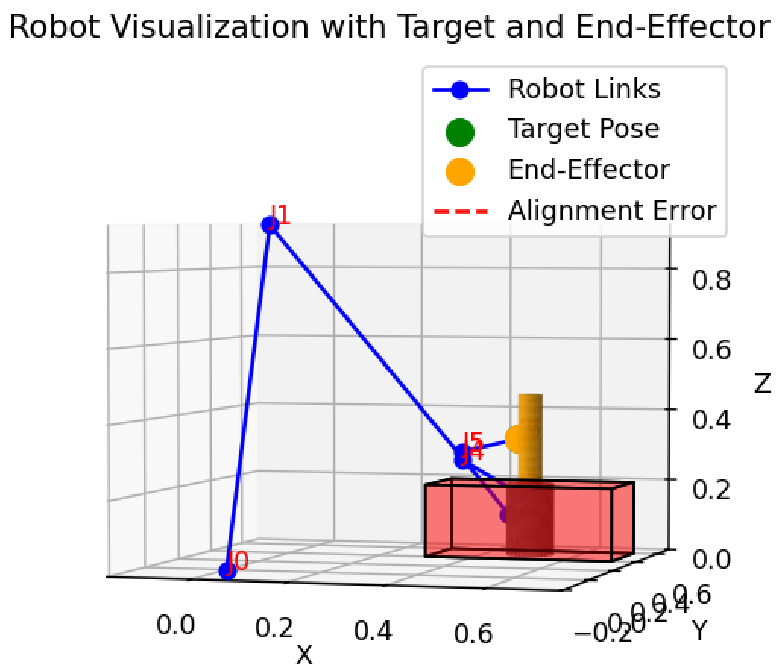
Τελικές Θέσεις των Αρθρώσεων:

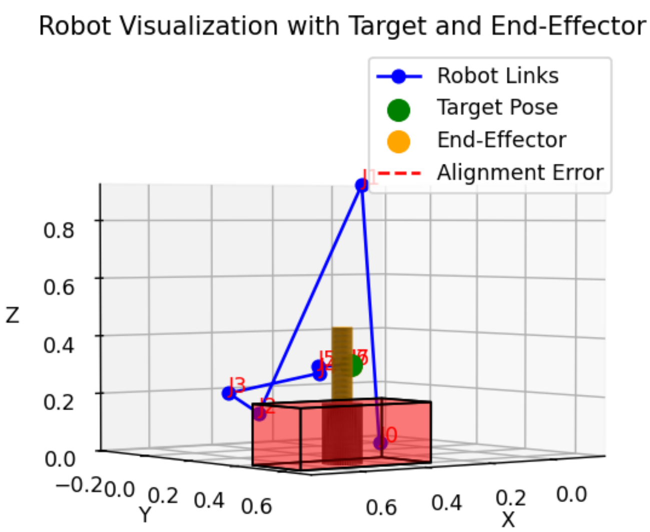
```
Joint Position:
Joint theta0: [0, 0, 0]
Joint theta1: [0.08263038 0.03026914 0.926      ]
Joint theta2: [0.535194   0.19605214 0.14719382]
Joint theta3: [0.57498465 0.11973905 0.21279928]
Joint theta4: [0.41204171 0.2854361  0.28303676]
Joint theta5: [0.41468679 0.28246331 0.30491658]
Joint theta6: [0.47496979 0.49996233 0.324999  ]
Joint theta7: [0.47496979 0.49996233 0.324999  ]
```

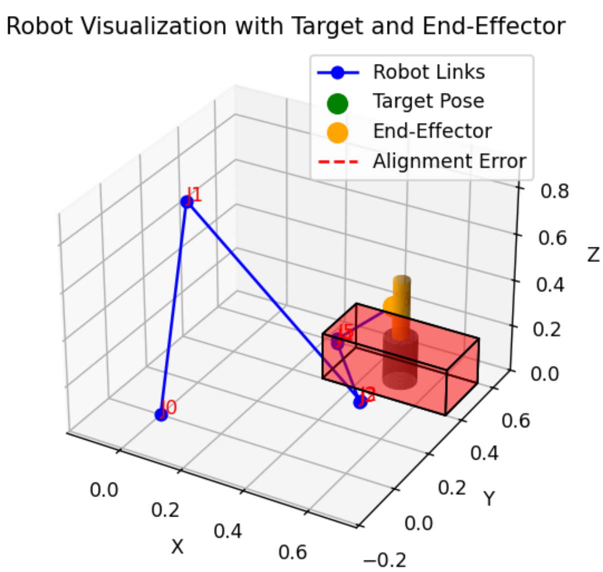
Τελικές Γωνίες των Αρθρώσεων:

```
Joint Angles:
Joint theta0: 0.35113896233647857
Joint theta1: 1.7386017313033406
Joint theta2: -1.3244497806143547
Joint theta3: 0.907235376404434
Joint theta4: -0.2533900130603458
Joint theta5: 1.9151085057764075
Joint theta6: 1.478723269549855
```

Αναπαράσταση του ρομπότ:







Verify your result with forward kinematics

```
Final Pose from Forward Kinematics:
 [[ 3.14178234e-07 -1.42895764e-06  1.00000000e+00  4.74969794e-01]
 [ 7.01249633e-07 -1.00000000e+00 -1.42895786e-06  4.99962330e-01]
 [ 1.00000000e+00  7.01250082e-07 -3.14177232e-07  3.24998997e-01]
 [-2.04154519e-17  2.21104084e-16  2.22044605e-16  1.00000000e+00]]
```

Βλέπουμε ότι με στρογγυλοποίηση είμαστε στο επιθυμητό target pose.

Ερώτημα 7:

Αρχεία κώδικα

hw1script.py: χρήσιμες συναρτήσεις από την προηγούμενη εργασία

visualizebox.py: βοηθητικά σχήματα για την αναφορά

hw2pt1.py: ερώτημα 5

hw2pt2.py: ερώτημα 6