# Introduction to Robotics: Homework III

Ονοματεπώνυμο: Ζωγράφου Μαρία-Νίκη

Αριθμός Μητρώου: 1096060

# Περιεχόμενα

Ερώτημα 1-4 ..... 3
Ερώτημα 5 ..... 4
Προσομοίωση και Αποτελέσματα: ..... 6
Ερώτημα 6-7 ..... 7
Αποτελέσματα και προσομοίωση: ..... 8

## Ερώτημα 1-4

Για αποστάσεις σε μέτρα:

Τοποθετούμε το world frame στο $(0,0,0)$ $T_w = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Transformation matrix για το κουτί: $T_{box} = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Transformation matrix για τον κύλινδρο: $T_{cylinder} = \begin{bmatrix} 1 & 0 & 0 & 0.695 \\ 0 & 1 & 0 & 0.095 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Franka Panda base frame transformation matrix:$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Home Configuration του end effector $T_s = \begin{bmatrix} 1 & 0 & 0 & 0.088 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.926 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Ορίζουμε για σημείο σύλληψης του κυλίνδρου:

The End-Effector–Cylinder Relative Transformation $T_{grasp} = \begin{bmatrix} 1 & 0 & 1 & 0.65 \\ 0 & -1 & 0 & 0.095 \\ 1 & 0 & -1 & 0.05 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Ορίζουμε για σημείο απελευθέρωσης του κυλίνδρου:

World–Cylinder (Release) Transformation $T_{release} = \begin{bmatrix} 0 & 0 & 1 & 0.5 \\ 0 & -1 & 0 & 0.5 \\ 1 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

# Ερώτημα 5

**Σχεδιασμός τροχιάς στο SE(3) για την εκτέλεση του έργου χειρισμού**

Η τροχιά του ρομποτικού βραχίονα πρέπει να σχεδιαστεί ώστε να εκτελέσει δύο κύρια καθήκοντα:

1. **Προσέγγιση του αντικειμένου με σωστό προσανατολισμό και θέση**
2. **Μεταφορά του αντικειμένου και απελευθέρωσή του στην κατάλληλη θέση**

Ο σχεδιασμός της τροχιάς γίνεται στο χώρο των ομογενών μετασχηματισμών SE(3). Θα χρησιμοποιήσουμε πολυωνυμικές συναρτήσεις για την παραγωγή ομαλών τροχιών.

**Κυβικές συναρτήσεις παρεμβολής (Cubic Splines)**: Χρησιμοποιούμε κυβικές πολυωνυμικές συναρτήσεις στον χώρο SE3 επομένως μιλάμε για Rotation R και γωνιακή ταχύτητα ω:

$$\boldsymbol{R}(t) = \exp(\hat{\boldsymbol{c}}_3 t^3 + \hat{\boldsymbol{c}}_2 t^2 + \hat{\boldsymbol{c}}_1 t + \hat{\boldsymbol{c}}_0)$$
$$\dot{\boldsymbol{R}}(t) = (3\hat{\boldsymbol{c}}_3 t^2 + 2\hat{\boldsymbol{c}}_2 t + \hat{\boldsymbol{c}}_1)\boldsymbol{R}(t)$$

Έστω ξεκινάμε την στιγμή $t = 0$ από το $(\boldsymbol{R}_s, \boldsymbol{\omega}_s)$ και θέλουμε να φτάσουμε την στιγμή $t = T$ στο $(\boldsymbol{R}_g, \boldsymbol{\omega}_g)$, έχουμε:

$$\boldsymbol{R}(0) = \exp(\hat{\boldsymbol{c}}_0) = \boldsymbol{R}_s \Rightarrow \boldsymbol{c}_0 = \overline{\log(\boldsymbol{R}_s)} = \boldsymbol{\phi}_s$$

$$\dot{\boldsymbol{R}}(0) = \hat{\boldsymbol{c}}_1 \boldsymbol{R}(0) \Rightarrow \hat{\boldsymbol{\omega}}_s \boldsymbol{R}_s = \hat{\boldsymbol{c}}_1 \boldsymbol{R}_s \Rightarrow \boldsymbol{c}_1 = \boldsymbol{\omega}_s$$

$$\boldsymbol{c}_2 = \frac{3\boldsymbol{\phi}_g}{T^2} - \frac{3\boldsymbol{\phi}_s}{T^2} - \frac{2\boldsymbol{\omega}_s}{T} - \frac{\boldsymbol{\omega}_g}{T}$$

$$\boldsymbol{c}_3 = -\frac{2\boldsymbol{\phi}_g}{T^3} + \frac{2\boldsymbol{\phi}_s}{T^3} + \frac{\boldsymbol{\omega}_s}{T^2} + \frac{\boldsymbol{\omega}_g}{T^2}$$

Όπου $\boldsymbol{R}_s = \exp(\hat{\boldsymbol{\phi}}_s)$, $\boldsymbol{R}_g = \exp(\hat{\boldsymbol{\phi}}_g)$.

Μετατρέπουμε τις παραπάνω εξισώσεις έτσι ώστε να χρησιμοποιούμε το T(t) transformation matrix που περιγράφει τόσο τη θέση όσο και τον προσανατολισμό του σώματος. Χρησιμοποιώντας την ίδια λογική που χρησιμοποιούμε για τις γραμμικές τροχιές θέτουμε ότι ξεκινάμε από την κατάσταση $(\boldsymbol{T}_s, \mathcal{V}_s)$ όπου $\mathcal{V}_s$ αρχικό twist, Ts αρχικό transformation matrix και θέλουμε να φτάσουμε στην $(\boldsymbol{T}_g, \mathcal{V}_g)$.

$$\boldsymbol{T}(t) = \exp(\hat{\boldsymbol{c}}_3 t^3 + \hat{\boldsymbol{c}}_2 t^2 + \hat{\boldsymbol{c}}_1 t + \hat{\boldsymbol{c}}_0)$$
$$\dot{\boldsymbol{T}}(t) = (3\hat{\boldsymbol{c}}_3 t^2 + 2\hat{\boldsymbol{c}}_2 t + \hat{\boldsymbol{c}}_1)\boldsymbol{T}(t)$$

Οι συντελεστές υπολογίζονται ως εξής:

Έστω αρχικό Transformation Matrix T(0) και twist $\mathcal{V}_s$:

$\boldsymbol{T}(0) = \exp(\hat{\boldsymbol{c}}_0) = \boldsymbol{T}_s \Rightarrow \boldsymbol{c}_0 = \overline{\log \boldsymbol{T}_s} = \boldsymbol{\tau}_s$, εύρεση $c_0$.

$\dot{\boldsymbol{T}}(0) = \hat{\boldsymbol{c}}_1 \boldsymbol{T}(0) \Rightarrow [\mathcal{V}_s]\boldsymbol{T}_s = \hat{\boldsymbol{c}}_1 \boldsymbol{T}_s \Rightarrow \boldsymbol{c}_1 = \mathcal{V}_s$, εύρεση $c_1$.

Στην συνέχεια υπολογισμός:

$$\boldsymbol{c}_2 = \frac{3\tau_g}{T^2} - \frac{3\tau_s}{T^2} - \frac{2\mathcal{V}_s}{T} - \frac{\mathcal{V}_g}{T}$$

$$\boldsymbol{c}_3 = -\frac{2\boldsymbol{\tau}_g}{T^3} + \frac{2\boldsymbol{\tau}_s}{T^3} + \frac{\mathcal{V}_s}{T^2} + \frac{\mathcal{V}_g}{T^2}$$

Σημείωση: $\boldsymbol{T}_s = \exp(\hat{\boldsymbol{\tau}}_s), \boldsymbol{T}_g = \exp(\hat{\boldsymbol{\tau}}_g)$

*Σημείωση:*

$\hat{V} = \begin{bmatrix} \hat{\omega} & u \\ 0 & 0 \end{bmatrix}$, όπου ω skew symmetric του διανύσματος περιστροφής ω και το υ το διάνυσμα **γραμμικής ταχύτητας**

**Στην σχεδίαση της τροχιάς θα υπάρχουν δύο στάδια:**

1. **Προσέγγιση του κυλίνδρου:** Από αρχική θέση έως target pose 1 για να γίνει grasp ο κύλινδρος
2. **Μεταφορά του κυλίνδρου:** Από την θέση που έγινε το grasp έως τη θέση πάνω από το κουτί

Θα γίνει προσαρμογή της τελικής θέσης του κυλίνδρου ώστε να τοποθετηθεί ακριβώς πάνω από την τρύπα και ρύθμιση της τελικής ταχύτητας σε **μηδενική** για ομαλή απελευθέρωση.

Για την εξασφάλιση της ακρίβειας της τροχιάς:

1. Σπάμε το χρονικό διάστημα σε στιγμές t ενός πλήθους ορισμένο από τον χρήστη.
    - Υπολογίζουμε το $T(\mathrm{t})$ και το $\dot{\boldsymbol{T}}(\mathrm{t})$ για κάθε χρονική στιγμή t. Αποκτάμε έτσι όλες τις θέσεις, περιστροφές και ταχύτητες του robot.

```
positions.append(T_t[:3, 3]) # column 3 first 3 rows
rotations.append(T_t[:3, :3]) # first 3 rows first 3 columns
transformations.append(T_t) # T(t)
velocities.append(V_t) # V(t)
```

Εξήγηση: το T(t) έχει την μορφή

$$T_t = \begin{bmatrix} R_t & p_t \\ 0 & 1 \end{bmatrix}$$

Η παραπάνω προσέγγιση επιτρέπει στο ρομπότ να ακολουθήσει μια συνεχή και ομαλή τροχιά, διασφαλίζοντας την ακριβή τοποθέτηση του κυλίνδρου μέσα στην τρύπα.

Στο αρχείο exercise3-5.ipynb (για jupyter notebook) ή στο script exercise5script.py, σχεδιάζουμε στο τέλος τον κύλινδρο, το κουτί και τις δύο τροχιές που θα πρέπει να ακολουθήσει το ρομπότ.

## Προσομοίωση και Αποτελέσματα:

Αρχική θέση end effector:

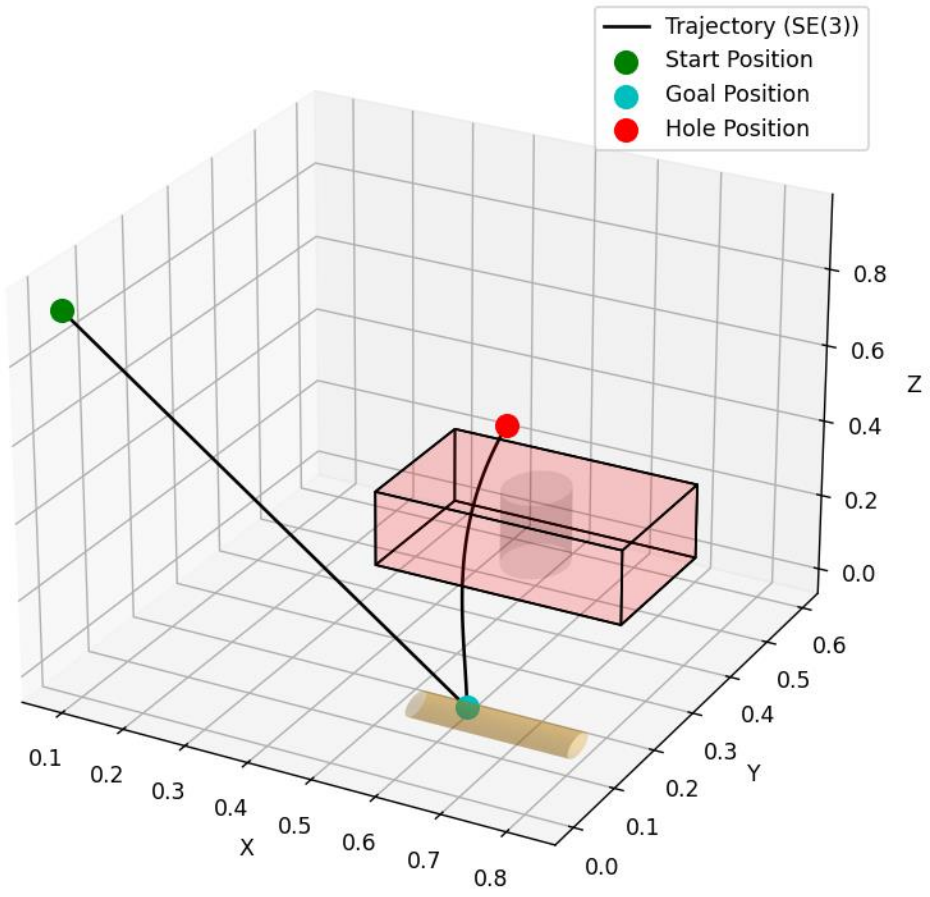
```
T_s = np.array([
    [1, 0, 0, 0.088],
    [0, -1, 0, 0],
    [0, 0, -1, 0.926],
    [0, 0, 0, 1]
])
```

Θέση για να συλληφθεί (grasp) ο κύλινδρος:

```
T_g = np.array([
    [1, 0, 0, 0.65],
    [0, -1, 0, 0.095],
    [0, 0, -1, 0.05],
    [0, 0, 0, 1]
])
```

Transformation matrix για το που θα αφεθεί ο κύλινδρος:

```
T_h = np.array([
    [0, 0, 1, 0.45],
    [0, -1, 0, 0.5],
    [1, 0, 0, 0.35],
    [0, 0, 0, 1]])
```



## Ερώτημα 6-7

Χρησιμοποιούμε Product of Exponentials Forward Kinematics για να υπολογίζουμε το end effector pose με βάση τις γωνίες των joints. (screw axis ξ, γωνίες θ, M home configuration)

$$T(\theta) = e^{\dot{\xi}_1\theta_1} e^{\xi_2\theta_2} \dots e^{\dot{\xi}_n\theta_n} M$$

Για την ακολούθηση της τροχιάς θα χρησιμοποιήσουμε **Inverse Kinematics με** Jacobian pseudoinverse-based task-space velocity controller.

Ορίζουμε error ως $e = \begin{bmatrix} e_p \\ e_o \end{bmatrix}$, όπου $e_p$ = επιθυμητή θέση – τωρινή θέση και $e_o$ είναι η απόκλιση από τον σωστό προσανατολισμό και προκύπτει από την διαφορά των λογαρίθμων των rotation matrices. Για να μεταφράσουμε αυτό το σφάλμα σε αλλαγές των γωνιών των αρθρώσεων, χρησιμοποιούμε τη σχέση $e = \boldsymbol{J}\dot{\theta}$, από την οποία προκύπτει ότι $\dot{\theta} = \boldsymbol{J}^{\dagger}\, e$ (joint velocities).

Το **Pseudoinverse Jacobian** είναι:

$\boldsymbol{J}^{\dagger} = (\boldsymbol{J}^{\top}\boldsymbol{J} + \lambda^2\boldsymbol{I})^{-1}\boldsymbol{J}^{\top}$, για $n < m, (\boldsymbol{J}^{\dagger} = \boldsymbol{I})$ (n rows, m columns)

$\boldsymbol{J}^{\dagger} = \boldsymbol{J}^{\boldsymbol{T}}(\boldsymbol{J}\boldsymbol{J}^{\boldsymbol{T}} + \lambda^2\boldsymbol{I})^{-1}$, για $n > m, (\boldsymbol{J}^{\dagger}\boldsymbol{J} = \boldsymbol{I})$ και $\lambda \in \mathbb{R}^{+}$

Το λ είναι ένας όρος απόσβεσης για τη σταθεροποίηση των υπολογισμών σε καταστάσεις κοντά σε κινηματικές μοναδικότητες.

Για να ανανεώσουμε τις γωνίες των αρθρώσεων εκτελούμε:

$$\theta_{\text{new}} = \theta_{\text{current}} + \dot{\theta}\Delta t$$

**Βήματα Ελέγχου Συνοπτικά:**

1. Υπολογισμός τρέχουσας θέσης με Forward Kinematics.

2. Υπολογισμός σφάλματος e: Διαφορά μεταξύ επιθυμητού και τρέχοντος pose.

3. Υπολογισμός της Ιακωβιανής: Αριθμητικός υπολογισμός μέσω μικρών διαταραχών των γωνιών.

4. Υπολογισμός ταχυτήτων αρθρώσεων: Χρήση του $\dot{\theta} = \boldsymbol{J}^{\dagger}\, e$ για την εύρεση των απαραίτητων ταχυτήτων.

5. Ανανέωση των γωνιών: Εφαρμογή Euler integration ως $\theta_{\text{new}} = \theta_{\text{current}} + \dot{\theta}\Delta t$

Επανάληψη μέχρι το σφάλμα να πέσει κάτω από ένα όριο by default 1e-6.

**Αριθμητικός Υπολογισμός Ιακωβιανής:**

Βρίσκουμε το αρχικό transformation matrix για τις τωρινές γωνίες. Εφαρμόζουμε στις γωνίες των αρθρώσεων μια μικρή διαταραχή delta και υπολογίζουμε το νέο Tranformation matrix.

Κρατάμε την τέταρτη στήλη του πίνακα Tperturbed, αλλά μόνο τις τρεις πρώτες γραμμές για το p δηλαδή το διάνυσμα θέσης. Βρίσκουμε την διαφορά διανύσματος θέσης με dp = (p_perturbed - p_current) / delta. Υστέρα βρίσκουμε το σφάλμα προσανατολισμού.

R_error = R_perturbed @ R_current.T # Σχετικός μετασχηματισμός περιστροφής log_R = logm(R_error) # Υπολογισμός του λογαρίθμου του σφάλματος περιστροφής log_R = np.real(log_R) # Απορρίπτουμε τυχόν φανταστικά μέρη

Υπολογίζουμε την σχετική περιστροφή μεταξύ δυο καταστάσεων: $R_{error} = R_{pertrubed}R_{current}^{T}$ και ύστερα τον λογάριθμο του. Ο πίνακας **logR είναι αντισυμμετρικός** και η σχέση του με τη γωνιακή ταχύτητα δίνεται ως:

$$\omega = \begin{bmatrix} logR_{3,2} \\ logR_{1,3} \\ logR_{2,1} \end{bmatrix}$$

Διαιρούμε με το βήμα delta για κανονικοποίηση.

Τέλος προσθέτουμε σε μια στήλη της ιακωβιανής τις γραμμικές ταχύτητες και τις γωνιακές ταχύτητες. Έτσι έχουμε 6 γραμμές (3 γραμμικές 3 γωνιακές) και n στήλες για κάθε Joint.

## Αποτελέσματα και προσομοίωση:

**Screw axis του ρομπότ:**

```
    screw_axes = np.array([
        [0, 0, 1, 0, 0, 0],
        [0, 1, 0, -0.333, 0, 0],
        [0, 0, 1, 0, 0, 0],
        [0, -1, 0, 0.649, 0, -0.088],
        [0, 0, 1, 0, 0, 0],
        [0, -1, 0, 1.033, 0, 0],
        [0, 0, 1, 0, -0.088, 0]
    ]).T  # Transpose to make it 6x7
```

Αρχική θέση, target poses:

```
T_s = np.array([
        [1, 0, 0, 0.088],
        [0, -1, 0, 0],
        [0, 0, -1, 0.926],
        [0, 0, 0, 1]
```

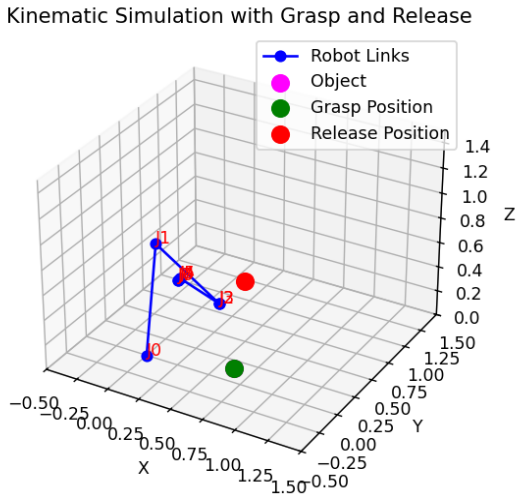
```
    ])

    target_pose1 = np.array([
        [1, 0, 0, 0.65],
        [0, -1, 0, 0.095],
        [0, 0, -1, 0.05],
        [0, 0, 0, 1]
    ])

    target_pose2 = np.array([
        [0, 0, 1, 0.5],
        [0, -1, 0, 0.5],
        [1, 0, 0, 0.5],
        [0, 0, 0, 1]
    ])
```

Η προσομοίωση γίνεται με animation στο αρχείο ex6and7.py

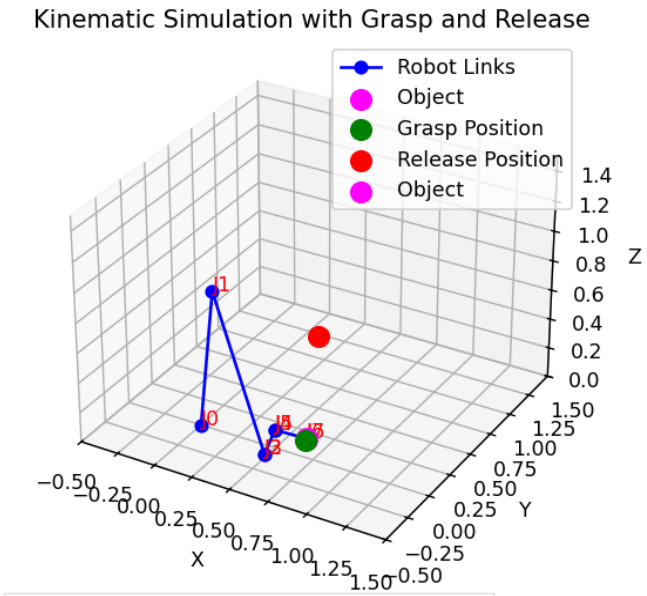
Inverse Kinematics: Converged in 25 steps.




Τυχαίο στιγμιότυπο

Όταν ο end effector βρεθεί στο σημείο που μπορεί να γραπώσει τον κύλινδρο (grasp), όταν έχει δηλαδή κατάλληλη απόσταση και προσανατολισμό, τυπώνει το τωρινό transformation matrix καθώς και τον αριθμό του χρονικού βήματος στο οποίο «έπιασε τον κύλινδρο».

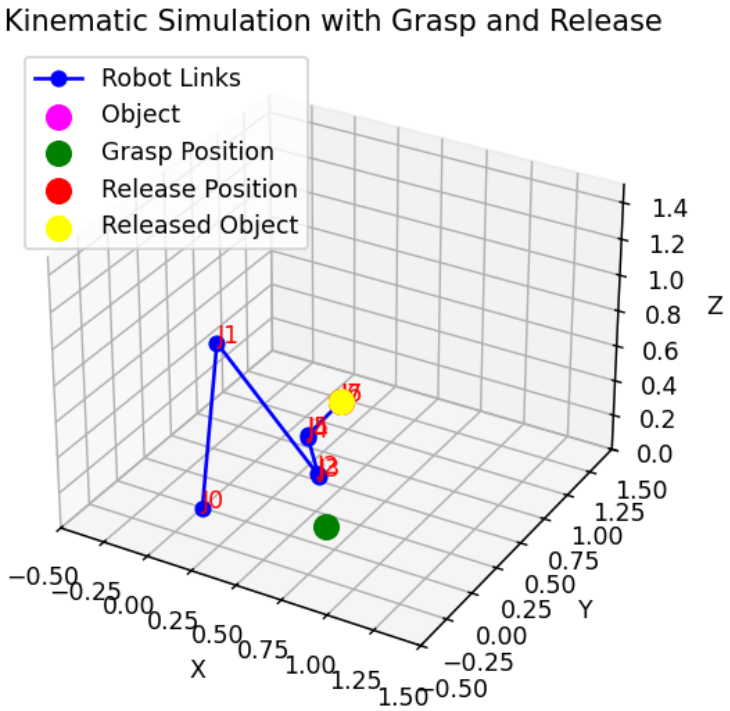
```
Grasped object at step 96.
[[ 1.00000000e+00  9.08008193e-07 -8.20707101e-06  6.45320271e-01]
 [ 9.08010667e-07 -1.00000000e+00  3.01482902e-07  9.42145378e-02]
 [-8.20707074e-06 -3.01490354e-07 -1.00000000e+00  5.70160929e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```




Cylinder grasped

Όταν ο end effector βρεθεί στο σημείο που μπορεί να ελευθερώσει τον κύλινδρο (release), όταν έχει δηλαδή κατάλληλη απόσταση και προσανατολισμό, τυπώνει το τωρινό transformation matrix καθώς και τον αριθμό του χρονικού βήματος στο οποίο «άφησε τον κύλινδρο».

```
Released object at step 197.
[[ 5.37250728e-03  1.19530523e-06  9.99985568e-01  5.00214484e-01]
 [-3.60958124e-06 -1.00000000e+00  1.21471526e-06  4.97769558e-01]
 [ 9.99985568e-01 -3.61605521e-06 -5.37250728e-03  4.97844618e-01]
 [-6.01807753e-17  9.32964412e-17  0.00000000e+00  1.00000000e+00]]
```




Cylinder released