Introduction to Robotics: Homework IV

Ονοματεπώνυμο: Ζωγράφου Μαρία-Νίκη

Αριθμός Μητρώου: 1096060

Περιεχόμενα

Ερώτημα 1

Κάνουμε extract τα δεδομένα από το αρχείο urdf. Μας δίνεται η δυνατότητα από την βιβλιοθήκη urdfpy να εξάγουμε προγραμματιστικά τα δεδομένα και να τα τυπώσουμε(αρχείο erotima1.py).

	name	center_of_mass	mass
0	arm_link_0	[0, 0, 0.02725]	0 # δεν δίνεται μάζα default σε 0
1	arm_link_1	[0.0, 0.0, 0.0615]	0.190421352
2	arm_link_2	[0.0, 0.0, 0.1585]	0.29302326
3	arm_link_3	[0.0, 0.0, 0.101]	0.21931466
4	arm_link_4	[0.0, 0.0, 0.08025]	0.15813986
5	arm_link_5	[0.0, 0.0, 0.0]	0 # δεν έχει CoM είναι απλά ο end effector

	name	type	joint_frame	rotation_axis
0	arm_joint_1	revolute	[0.0, 0.0, 0.0545]	[0.0, 0.0, 1.0]
1	arm_joint_2	revolute	[0.0, 0.0, 0.123]	[0.0, 1.0, 0.0]
2	arm_joint_3	revolute	[0.0, 0.0, 0.317]	[0.0, 1.0, 0.0]
3	arm_joint_4	revolute	[0.0, 0.0, 0.202]	[0.0, 1.0, 0.0]
4	arm_joint_5	fixed	[0.0, 0.0, 0.1605]	[0, 0, 0]

Inertia Matrices=

$$\boldsymbol{I_b} = \begin{bmatrix} \boldsymbol{I_{XX}} & \boldsymbol{I_{XY}} & \boldsymbol{I_{XZ}} \\ \boldsymbol{I_{XY}} & \boldsymbol{I_{YY}} & \boldsymbol{I_{YZ}} \\ \boldsymbol{I_{XZ}} & \boldsymbol{I_{YZ}} & \boldsymbol{I_{ZZ}} \end{bmatrix}$$

arm_link_0: δεν έχει Inertia Matrix

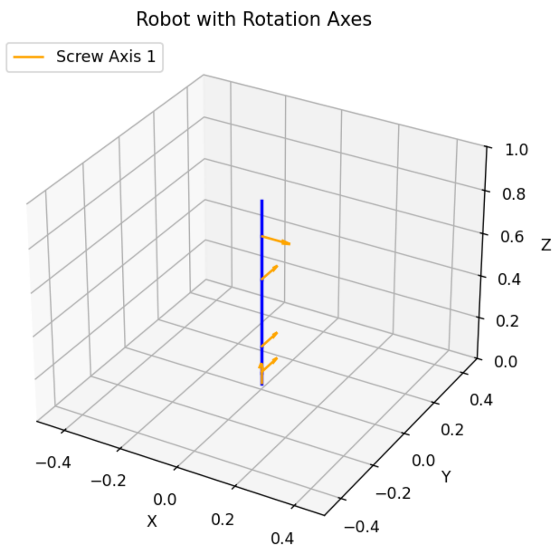
arm_link_1: **ixx**= 0.000279744834534, **ixy**= 0.0, **ixz**= 0.0, **iyy**= 0.000265717763008, **iyz**= 0.0, **izz**= 6.53151584738e-05

arm_link_2: **ixx**= 0.00251484771035, **ixy**= 0.0, **ixz**= 0.0, **iyy**= 0.00248474836108, **iyz**= 0.0, **izz**=9.19936757328e-05

arm_link_3: **ixx**= 0.000791433503053, **ixy**= 0.0, **ixz**= 0.0, **iyy**= 0.000768905501178, **iyz**= 0.0, **izz**= 6.88531064581e-05

arm_link_4: **ixx**= 0.00037242266488, **ixy**= 0.0, ixz= 0.0, **iyy**= 0.000356178538461, **iyz**= 0.0**,** **izz**=4.96474819141e-05

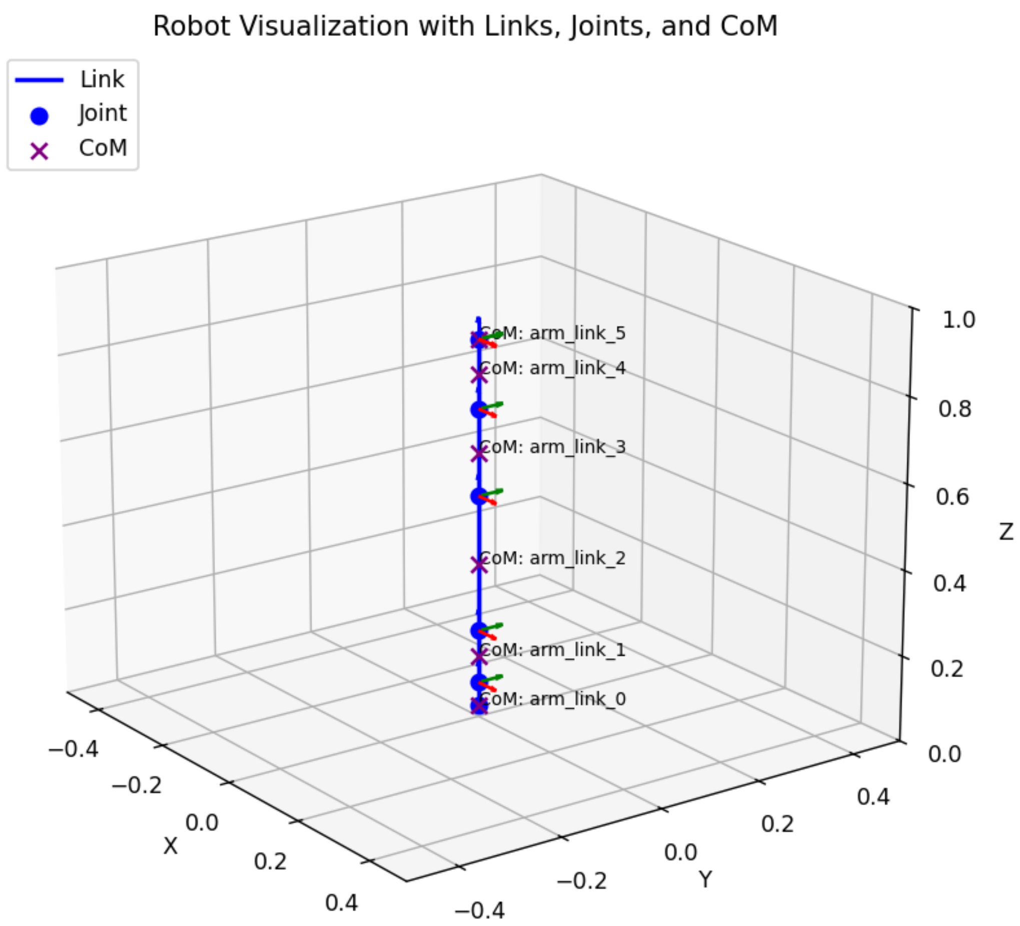
arm_link_5: δεν έχει Inertia Matrix



Υπολογισμός Screw Axes:

screw axes
[0. 0. 1. 0. 0. 0.]
[0.   1.   0.   -0.1775 0.   0.   ]
[0.   1.   0.   -0.4945 0.   0.   ]
[0.   1.   0.   -0.6965 0.   0.   ]
[1.   0.   0.   0.   0.857 0.   ]

Οπτικοποίηση του Ρομπότ:



Ερώτημα 2

Δημιουργία Forward Dynamics Simulator με Closed Form Derivation

$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + \boldsymbol{F}_{ext} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}))$, με Fext = 0 στην περίπτωση μας.

Θα χρησιμοποιήσουμε Newton-Euler Inverse Dynamics:

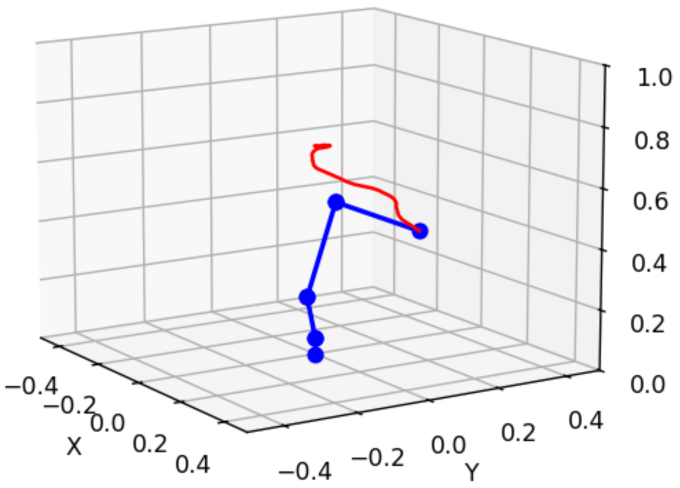
Από το βιβλίο Modern Robotics έχουμε:

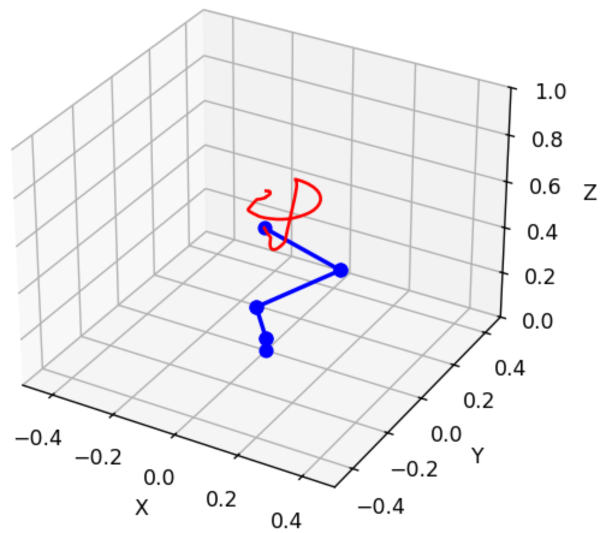
> When the manipulator is at the home position, with all joint variables zero, we denote the configuration of frame $\{j\}$ in $\{i\}$ as $M_{i,j} \in SE(3)$, and the configuration of $\{i\}$ in the base frame $\{0\}$ using the shorthand $M_i = M_{0,i}$. With these definitions, $M_{i-1,i}$ and $M_{i,i-1}$ can be calculated as

Αρχικά $M_{i,j} \in SE(3)$ και έχουμε $M_i = M_{i,0},\ M_{i-1,i} = M_{i-1}^{-1} M_i\ \ και\ M_{i,i-1} = M_i^{-1} M_{i-1}$

Θα βασιστούμε σε exponential coordinates για τους 3d υπολογισμούς μας, με την παρακάτω συνάρτηση: exp_se3. Κάνουμε $R = I + \sin(\theta)\, W + (1 - \cos(\theta)) W^2$ (Rodriguez Formula).

Θα δημιουργήσουμε μία συνάρτηση που δέχεται σαν εισόδους τα q, $\dot{q}, \tau$ και θα έχει σαν output τα $\ddot{q}$. Ύστερα θα υλοποιήσουμε ολοκλήρωση Euler, όπου απλά θα υπολογίζουμε τα $q_{next} = q + dt * \dot{q}$ (με το προηγούμενο $\dot{q}$) και $\dot{q_{next}} = \dot{q} + dt * \ddot{q}$ (με το νέο $\ddot{q}$). Ελέγχουμε ότι τα q δεν ξεπερνάνε τα joint limits που δίνονται στο urdf αρχείο [-1.57,+1.57]. Με χρήση forward kinematics μπορούμε να βρούμε τα Joint angles σε ποια θέση οδηγούν τον end effector και να αναπαραστήσουμε το ρομπότ. Δίνοντας τυχαία τ (ροπές) στο ρομπότ δημιουργούμε μια προσομοίωση τεστ (είναι animated στο αρχείο erotima2.py και διαφορετικά seeds δίνουν διαφορετικά αποτελέσματα):





Closed Form Derivation από το Modern Robotics

Θέλουμε κλειστή μορφή για τα $M(\boldsymbol{q} \mid \bar{q}, \mathbf{c}(\boldsymbol{q}.\boldsymbol{q})$ και $\boldsymbol{g}(\boldsymbol{q})$, οπότε:

$$
\begin{aligned}
&v_{\text{all}} = [v_1 \quad \cdots \quad v_n]^T \in \mathbb{R}^{6n} \\
&F_{all} = [F_1 \quad \cdots \quad F_n]^T \in \mathbb{R}^{6n} \\
&\mathcal{A}_{al} = \begin{bmatrix} \mathcal{A}_1 & 0 & \cdots & 0 \\ 0 & \mathcal{A}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \mathcal{A}_n \end{bmatrix} \in \mathbb{R}^{6n\times 6s} \\
&\mathcal{G}_{all} = \begin{bmatrix} \mathcal{G}_1 & 0 & \cdots & 0 \\ 0 & \mathcal{G}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \mathcal{G}_n \end{bmatrix} \in \mathbb{R}^{60\times 6n} \\
&[adv_{2l}] = \begin{bmatrix} [adv_1] & 0 & \cdots & 0 \\ 0 & [adv_2] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & [adv_n] \end{bmatrix} \in \mathbb{R}^{60\times 6n} \\
&\left[ad_{A_{2,}}\right] = \begin{bmatrix} \left[ad_{A_{1q_1}}\right] & 0 & \cdots & 0 \\ 0 & \left[ad_{A_{2q_2}}\right] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \left[ad_{A_{n,q_n}}\right] \end{bmatrix} \in \mathbb{R}^{6n\times 6n} \\
&W(\boldsymbol{q}) = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ \left[Ad_{\boldsymbol{r}_{2,1}}\right] & 0 & \cdots & 0 & 0 \\ 0 & \left[Ad\boldsymbol{T}_{2,2}\right] & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & \cdots & \cdots & \left[AdT_{n,\varepsilon-1}\right] & 0 \end{bmatrix} \in \mathbb{R}^{6n\times 6n} \\
&\mathcal{L}(\boldsymbol{q}) = (\boldsymbol{I} - \mathcal{W}(\boldsymbol{q}))^{-1} = \begin{bmatrix} \boldsymbol{I} & 0 & 0 & \cdots & 0 \\ \left[Ad_{\boldsymbol{T}_{2,1}}\right] & \boldsymbol{I} & 0 & \cdots & 0 \\ \left[Ad_{T_{3,1}}\right] & \left[Ad_{T_{3,2}}\right] & \boldsymbol{I} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \left[\mathrm{Ad}_{T_{n,1}}\right] & \left[Ad_{T_{n,2}}\right] & \left[Ad_{T_{n,3}}\right] & \cdots & \boldsymbol{I} \end{bmatrix} \in \mathbb{R}^{6n\times 6n}
\end{aligned}
$$

$$
V_{\text{base}} = \begin{bmatrix} \left[Ad\boldsymbol{r}_{2,0}\right]v_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{6n}
$$

$$
V_{\text{hase}} = \begin{bmatrix} \left[Ad\boldsymbol{T}_{1,0}\right)\dot{V}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{6n}
$$

$$
\mathcal{F}_{\text{tip}} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \left[Ad\boldsymbol{T}_{e+1,}\right]^T \mathcal{F}_{s+1} \end{bmatrix} \in \mathbb{R}^{6n}
$$

Καταλήγουμε στις εξής σχέσεις:

$$\begin{aligned}
\mathcal{V}_{\text{all}} &= \mathcal{L}(\boldsymbol{q})(\mathcal{A}_{\text{all}}\,\dot{\boldsymbol{q}} + \mathcal{V}_{\text{base}}) \\
\dot{\mathcal{V}}_{\text{all}} &= \mathcal{L}(\boldsymbol{q})\big(\mathcal{A}_{\text{all}}\,\ddot{\boldsymbol{q}} - [\,\text{ad}\,\mathcal{A}_{\text{ald}}\,]\mathcal{W}(\boldsymbol{q})\mathcal{V} - \big[ad_{\mathcal{A}_{\text{ald}}}\big]\mathcal{V}_{\text{base}} + \dot{\mathcal{V}}_{\text{base}}\big) \\
\mathcal{F}_{\text{all}} &= \mathcal{L}^{T}(\boldsymbol{q})\left(\mathcal{G}_{\text{all}}\,\dot{\mathcal{V}}_{\text{all}} - \big[ad_{V_{\text{all}}}\big]^{T}\mathcal{G}_{\text{all}}\,\mathcal{V}_{\text{all}} + \mathcal{F}_{\text{tip}}\right) \\
\tau &= \mathcal{A}_{\text{all}}^{T}\,\mathcal{F}_{\text{all}}
\end{aligned}$$

Και:

$$\begin{aligned}
\boldsymbol{M}(\boldsymbol{q}) &= \mathcal{A}_{211}^{T}\mathcal{L}^{T}(\boldsymbol{q})\mathcal{G}_{\text{all}}\,\mathcal{L}(\boldsymbol{q})\mathcal{A}_{\text{all}} \\
\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= -\mathcal{A}_{\text{all}}^{T}\,\mathcal{L}^{T}(\boldsymbol{q})\left(\mathcal{G}_{\text{all}}\,\mathcal{L}(\boldsymbol{q})\left[ad_{\mathcal{A}_{\text{al|}}\,]}\right]\mathcal{W}(\boldsymbol{q}) + \left[ad_{v_{a1}}\right]^{T}\mathcal{G}_{\text{all}}\right)\mathcal{L}(\boldsymbol{q})\mathcal{A}_{\text{al|}}\,\dot{\boldsymbol{q}} \\
\boldsymbol{g}(\boldsymbol{q}) &= \mathcal{A}_{\text{all}}^{T}\,\mathcal{L}^{T}(\boldsymbol{q})\mathcal{G}_{\text{an}}\,\mathcal{L}(\boldsymbol{q})\dot{\mathcal{V}}_{\text{base}}
\end{aligned}$$

Ερώτημα 3:

Για να ακολουθήσει την ημιτονοειδή τροχιά θα χρησιμοποιήσουμε Inverse Dynamics. Δεδομένης μιας επιθυμητής επιτάχυνσης $\ddot{\boldsymbol{q}}_d$, μπορούμε να υπολογίσουμε τη δύναμη (wrench) που απαιτείται για την επίτευξή της με τη σχέση:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_d + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{F}_{\text{ext}}$$

Αυτή η σχέση θα χρησιμοποιηθεί και για έλεγχο και για σύνθεση τροχιάς.

Δύο Υλοποιήσεις για έλεγχο και τροχιά παρακάτω:

Υλοποίηση 1 - Open Loop Controller

Ο ελεγκτής βασίζεται μόνο στην αντίστροφη κινηματική (inverse kinematics) και την ολοκλήρωση Euler. Δεν αποτελεί ρεαλιστικό σύστημα ελέγχου, καθώς λειτουργεί μόνο υπό ιδανικές συνθήκες της προσομοίωσης. Η επιτάχυνση ορίζεται σταθερά ως:

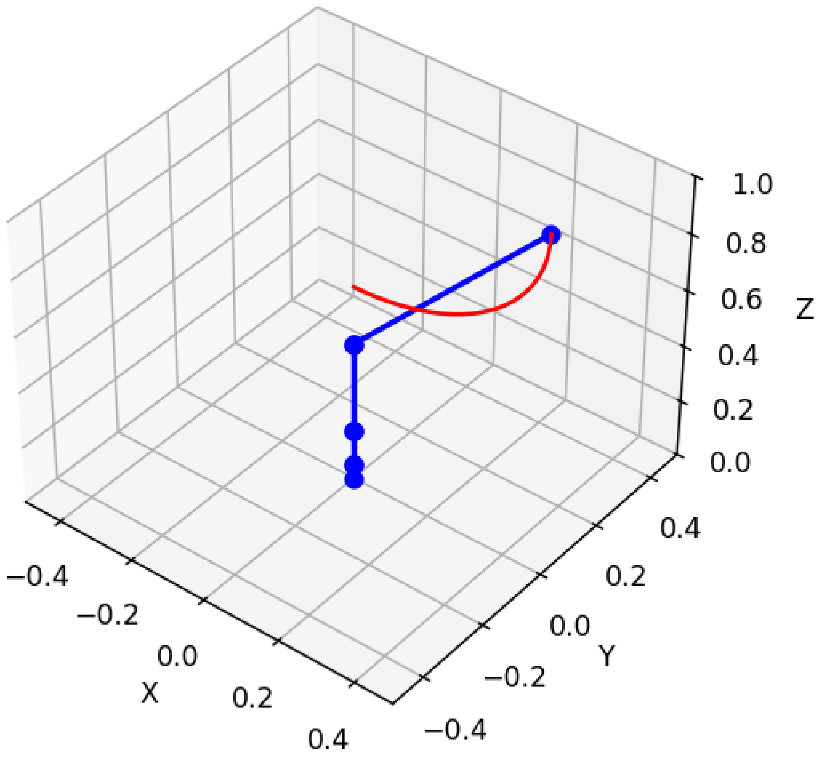
$$\alpha = sin\left(\frac{2\pi}{T_{sim}} * dt * k\right) \text{μόνο για το τελευταίο joint} \Rightarrow u = \cos(t) \text{ και θέση } p = sint(t).$$

1. Εισάγουμε τις τρέχουσες τιμές των q (θέσεις), $\dot{q}$(ταχύτητες), και $\ddot{q}$(επιταχύνσεις) στην συνάρτηση για τα inverse dynamics. Αυτή μας επιστρέφει τις απαιτούμενες ροπές τ για τις αρθρώσεις.

2. Περνάμε τα τ στην Step function η οποία εφαρμόζει τα forward dynamics, και την ολοκλήρωση Euler επιστρέφοντας τις νέες καταστάσεις q και $\dot{q}$ (next time step).

3. Χρησιμοποιούμε τις νέες τιμές των θέσεων q για να υπολογίσουμε τις αντίστοιχες θέσεις των joints του ρομπότ (forward kinematics).

Συνάρτηση animate_trajectory_open_loop() στο erotima3.py. Παίρνουμε το ¼ ενός ημιτόνου:



Υλοποίηση 2 - feedforward plus feedback linearizing

Υλοποιούμε ελεγκτή PD της εξής μορφής

$$\tau(t) = M(q(t))\ddot{q}_d(t) + C(q(t), \dot{q}(t)) + K_p[q_d(t) - q(t)] + K_d[\dot{q}_d(t) - \dot{q}(t)]$$

Κρατάμε την προηγούμενη λογική, αλλά για την εύρεση του τ δεν στηριζόμαστε μόνο στα Inverse Dynamics αλλά και σε δύο διορθωτικούς όρους με proportional και differential gain. Δηλαδή Kp * error θέσης και Kd * error ταχύτητας για feedback.

- $q_d(t)$ η επιθυμητή τροχιά των γωνιών των αρθρώσεων joint
- $\dot{q}_d(t), \ddot{q}_d(t)$ η επιθυμητή ταχύτητα και επιτάχυνση
- K_p, K_d τα κέρδη επιλέγουμε τιμές χαμηλές (Kp= 1.0 και Kd= 2.0)
- $M(\cdot)$ και $C(\cdot)$ από τα closed form dynamics που ορίσαμε (mass matrix και Coriolis/centrifugal).
- Προσθήκη $g(q)$

Στον κώδικα ορίζουμε Επιθυμητή τροχιά q= [0,0,0, A*sin(ω*t)] και επιθυμητή ταχύτητα q'=[0,0,0, A*ω*cos(ω*t)] και επιθυμητή επιτάχυνση q''=[0,0,0, -A*($ω^2$)*sin(ω*t)].

Ο συγκεκριμένος έλεγχος είναι ευαίσθητος στο time step και την ρύθμιση των κερδών. (συνάρτηση animate_trajectory_control() στο erotima3.py)

Για Tsim=1.5 και dt= 0.0015 παίρνουμε το ¼ του ημίτονου και ένα κομμάτι της επιστροφής στην αρχική θέση: