Ρομποτική ΙΙ

Εξαμηνιαία Εργασία 1

Κινηματικός έλεγχος ρομποτικού χειριστή με πλεονάζοντες βαθμούς ελευθερίας: Παρακολούθηση τροχιάς και αποφυγή εμποδίου

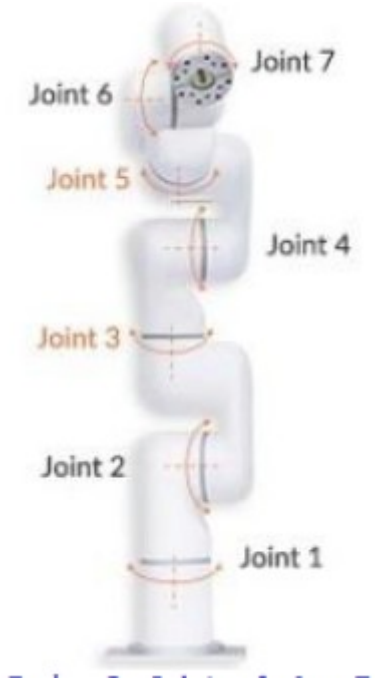
Αντωνίου Κωνσταντίνος
Α.Μ.: 03116169
kostas.an2016@gmail.com

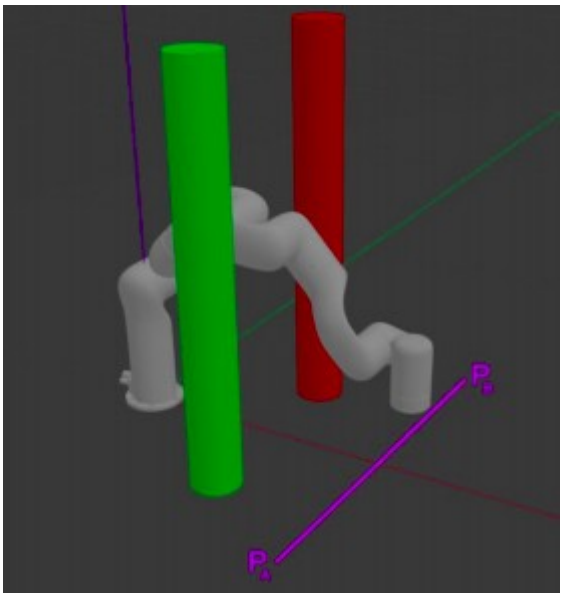
Μετζάκης Ιωάννης
Α.Μ.: 03116202
johnmetzakis@gmail.com

ΣΗΜΜΥ 8ο

Διάταξη:

Η διάταξη μας αποτελείται απο ένα ρομπότ τύπου cobot με 7 βαθμούς ελευθερίας αποτελούμενο απο 7 στροφικές αρθρώσεις και 7 συνδέσμους. Στην διάταξη επίσης υπάρχουν και 2 κυλινδρικά εμπόδια τα οποία απέχουν μεταξύ τους απόσταση 40cm και βρίσκονται αντιδιαμετρικά του άξονα x. Το κάθε κυλινδρικό εμπόδιο έχει διατομή 10cm. Η διάταξη αρχικοποίησης του ρομπότ καθώς και το αρχικό configuration που μας δίνεται φαίνονται αντίστοιχα παρακάτω:





Θεωρητική Ανάλυση:

Οι ζητούμενες ενέργειες που θέλουμε να επιτευχθούν με το συγκεκριμένο ρομπότ είναι οι εξής:

1. ***Παρακολούθηση τροχιάς***:
Θέλουμε το ρομπότ να πηγαίνει από το σημείο Pa (y=-0.2) μέχρι το σημείο Pb (y=0.2) και απο εκεί πάλι μέχρι το σημείο Pa (1 ολόκληρη κίνηση) διατηρώντας σταθερά τα x, z στις τιμές x=0.6043 και z=0.1508. Η διαδικασία αυτή επιθυμούμε να επαναλαμβάνεται μέχρι να σταματήσουμε την εκτέλεση του προγράμματος.

2. ***Αποφυγή εμποδίων***:
Παράλληλα με την πρώτη διεργάσια θέλουμε το ρομπότ να προσαρμόζεται και να αποφεύγει τα κυλινδρικά εμπόδια καθώς αυτά κουνιούνται ή παραμένουν σε μία σταθερή θέση της επιλογής μας πάνω στην ευθεία x=0.3.

Προαπαιτούμενες εργασίες:

Αρχικά θα υπολογίσουμε τον πίνακα A^0_7 ο οποίος θα μας δώσει την θέση του τελικού στοιχείου δράσης σε σχέση με το πλαίσιο της βάσης του ρομπότ. Ακολουθούμε τις παρακάτω σχέσεις:

$$\boldsymbol{A_1^0(q_1)} = Rot(z, q_1) \cdot Tra(z, l_1), \quad \boldsymbol{l_1} = 26.7cm$$

$$\boldsymbol{A_2^1(q_2)} = Rot\left(x, -\frac{\pi}{2}\right) \cdot Rot(z, q_2)$$

$$\boldsymbol{A_3^2(q_3)} = Rot\left(x, +\frac{\pi}{2}\right) \cdot Rot(z, q_3) \cdot Tra(z, l_2), \quad \boldsymbol{l_2} = 29.3cm$$

$$\boldsymbol{A_4^3(q_4)} = Rot\left(x, +\frac{\pi}{2}\right) \cdot Tra(x, l_3) \cdot Rot(z, q_4), \quad \boldsymbol{l_3} = 5.25cm$$

$$\boldsymbol{A_5^4(q_5)} = Rot\left(x, +\frac{\pi}{2}\right) \cdot Tra(x, l_4 \sin\theta_1) \cdot Rot(z, q_5) \cdot Tra(z, l_4 \cos\theta_1), \quad \begin{matrix} \boldsymbol{l_4} = 35.12\ cm \\ \boldsymbol{\theta_1} = 0.2225\ rad \end{matrix}$$

$$\boldsymbol{A_6^5(q_5)} = Rot\left(x, +\frac{\pi}{2}\right) \cdot Rot(z, q_6)$$

$$\boldsymbol{A_7^6(q_6)} = Rot\left(x, -\frac{\pi}{2}\right) \cdot Tra(x, l_5 \sin\theta_2) \cdot Rot(z, q_7) \cdot Tra(z, l_5 \cos\theta_2), \quad \begin{matrix} \boldsymbol{l_5} = 12.32\ cm \\ \boldsymbol{\theta_2} = 0.6646\ rad \end{matrix}$$

Γνωρίζουμε ότι ο πίνακας $A^0{}_7$ προκύπτει από τον πολλαπλασιασμό των παραπάνων πινάκων δηλαδή:

$A^0{}_7 = A^0{}_1 * A^1{}_2 * A^2{}_3 * A^3{}_4 * A^4{}_5 * A^5{}_6 * A^6{}_7$

Ο πίνακας που προκύπτει είναι πολυ ογκώδης. Ενδεικτικά οι θέσεις του end effector που προκύπτουν φαίνονται παρακάτω:

$p_x =$

$$\begin{aligned}
&l_2 \sin(q_2)\cos(q_1) + l_3(-\sin(q_1)\sin(q_3) + \cos(q_1)\cos(q_2)\cos(q_3)) \\
&- l_4(-(-\sin(q_1)\sin(q_3) + \cos(q_1)\cos(q_2)\cos(q_3))\sin(q_4) + \sin(q_2)\cos(q_1)\cos(q_4))\cos(\theta_1) \\
&+ l_4((-\sin(q_1)\sin(q_3) + \cos(q_1)\cos(q_2)\cos(q_3))\cos(q_4) + \sin(q_2)\sin(q_4)\cos(q_1))\sin(\theta_1) \\
&+ l_5(-(((-\sin(q_1)\sin(q_3) + \cos(q_1)\cos(q_2)\cos(q_3))\cos(q_4) + \sin(q_2)\sin(q_4)\cos(q_1))\cos(q_5) \\
&+ (\sin(q_1)\cos(q_3) + \sin(q_3)\cos(q_1)\cos(q_2))\sin(q_5))\sin(q_6) \\
&+ ((-\sin(q_1)\sin(q_3) + \cos(q_1)\cos(q_2)\cos(q_3))\sin(q_4) - \sin(q_2)\cos(q_1)\cos(q_4))\cos(q_6))\cos \\
&(\theta_2) \\
&+ l_5((((-\sin(q_1)\sin(q_3) + \cos(q_1)\cos(q_2)\cos(q_3))\cos(q_4) + \sin(q_2)\sin(q_4)\cos(q_1))\cos(q_5) \\
&+ (\sin(q_1)\cos(q_3) + \sin(q_3)\cos(q_1)\cos(q_2))\sin(q_5))\cos(q_6) \\
&+ ((-\sin(q_1)\sin(q_3) + \cos(q_1)\cos(q_2)\cos(q_3))\sin(q_4) - \sin(q_2)\cos(q_1)\cos(q_4))\sin(q_6))\sin \\
&(\theta_2)
\end{aligned}$$

$p_y =$

$$\begin{aligned}
&l_2 \sin(q_1)\sin(q_2) + l_3(\sin(q_1)\cos(q_2)\cos(q_3) + \sin(q_3)\cos(q_1)) \\
&- l_4(-(\sin(q_1)\cos(q_2)\cos(q_3) + \sin(q_3)\cos(q_1))\sin(q_4) + \sin(q_1)\sin(q_2)\cos(q_4))\cos(\theta_1) \\
&+ l_4((\sin(q_1)\cos(q_2)\cos(q_3) + \sin(q_3)\cos(q_1))\cos(q_4) + \sin(q_1)\sin(q_2)\sin(q_4))\sin(\theta_1) \\
&+ l_5(-(((\sin(q_1)\cos(q_2)\cos(q_3) + \sin(q_3)\cos(q_1))\cos(q_4) + \sin(q_1)\sin(q_2)\sin(q_4))\cos(q_5) \\
&+ (\sin(q_1)\sin(q_3)\cos(q_2) - \cos(q_1)\cos(q_3))\sin(q_5))\sin(q_6) \\
&+ ((\sin(q_1)\cos(q_2)\cos(q_3) + \sin(q_3)\cos(q_1))\sin(q_4) - \sin(q_1)\sin(q_2)\cos(q_4))\cos(q_6))\cos(\theta_2) \\
&+ l_5((((\sin(q_1)\cos(q_2)\cos(q_3) + \sin(q_3)\cos(q_1))\cos(q_4) + \sin(q_1)\sin(q_2)\sin(q_4))\cos(q_5) \\
&+ (\sin(q_1)\sin(q_3)\cos(q_2) - \cos(q_1)\cos(q_3))\sin(q_5))\cos(q_6) \\
&+ ((\sin(q_1)\cos(q_2)\cos(q_3) + \sin(q_3)\cos(q_1))\sin(q_4) - \sin(q_1)\sin(q_2)\cos(q_4))\sin(q_6))\sin(\theta_2)
\end{aligned}$$

$p_z =$

$$\begin{aligned}
&l_1 + l_2\cos(q_2) - l_3\sin(q_2)\cos(q_3) - l_4(\sin(q_2)\sin(q_4)\cos(q_3) + \cos(q_2)\cos(q_4))\cos(\theta_1) \\
&+ l_4(-\sin(q_2)\cos(q_3)\cos(q_4) + \sin(q_4)\cos(q_2))\sin(\theta_1) \\
&+ l_5(-((-\sin(q_2)\cos(q_3)\cos(q_4) + \sin(q_4)\cos(q_2))\cos(q_5) - \sin(q_2)\sin(q_3)\sin(q_5))\sin(q_6) \\
&+ (-\sin(q_2)\sin(q_4)\cos(q_3) - \cos(q_2)\cos(q_4))\cos(q_6))\cos(\theta_2) \\
&+ l_5(((-\sin(q_2)\cos(q_3)\cos(q_4) + \sin(q_4)\cos(q_2))\cos(q_5) - \sin(q_2)\sin(q_3)\sin(q_5))\cos(q_6) \\
&+ (-\sin(q_2)\sin(q_4)\cos(q_3) - \cos(q_2)\cos(q_4))\sin(q_6))\sin(\theta_2)
\end{aligned}$$

Έπειτα θα υπολογίσουμε την ιακωβιανή μήτρα. Προτιμήσαμε να χρησιμοποιήσουμε τον τύπο με τις μερικές παραγωγίσεις, ο οποίος ακουλουθεί την παρακάτω φόρμουλα:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}.$$

410 × 143

Στην δική μας περίπτωση παραγωγίζουμε τα p_x, p_y, p_z κάθε φορά ως προς κάθε ένα απο τα q_1 εως q_7. Επομένως, η ιακωβιανή που προκύπτει είναι 3 γραμμών και 7 στηλών δηλαδή 3x7. Ο πίνακας J είναι και αυτός πολύ ογκώδης και γι' αυτό το λόγο δεν κρίνουμε απαραίτητο να τον δείξουμε. Ενδεικτικά παρουσιάζεται η συνάρτηση σε γλώσσα python που χρησιμοποιήθηκε για την εξαγωγή του J:

```
def Jacobian(v_str, f_list, f1_list, f2_list):
    vars = sym.symbols(v_str)
    f =(f_list)
    f1 = (f1_list)
    f2 = (f2_list)
    #print (f)
    J = sym.zeros(len(vars))
    for j, s in enumerate(vars):
        J[0,j] = sym.diff(f, s)

    for j, s in enumerate(vars):
        J[1,j] = sym.diff(f1, s)

    for j, s in enumerate(vars):
        J[2,j] = sym.diff(f2, s)

    return J
```

```
J = Jacobian('q1 q2 q3 q4 q5 q6 q7', px,py,pz)
```

1η Υποεργασία : (Παρακολούθηση τροχιάς)

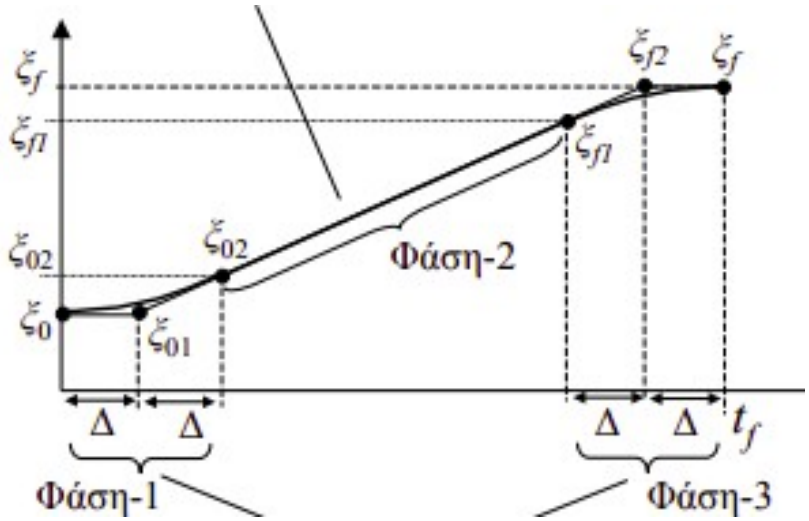
Για την συγκρεκριμένη υποεργασία θέλουμε να εξάγουμε τις γενικευμένες ταχύτητες των αρθρώσεων οι οποίες είναι αναγκαίες για την παρακολούθηση της τροχίας. Πιο συγκεκριμένα θα χρησιμοποιήσουμε τον τύπο:

$$(q)'_{(1)} = J^{+} * (p_{id})'$$

Ξεκινάμε υπολογίζοντας την ψευδοανάστροφη της ιακωβιανής χρησιμοποιώντας την εντολή pinv της βιβλιοθήκης numpy.linalg. Έπειτα, για να βρούμε τις γενικευμένες ταχύτητες της πρώτης υποεργασίας πρέπει να σχεδιάσουμε την τροχία ώστε να μπορούμε να βρούμε τις ταχύτητες που αποκτά το end effector κατα την ακολούθηση της.

Σχεδιασμός τροχιάς:

Επιλέξαμε να χωρίσουμε την τροχιά μας σε 3 φάσεις οπώς φαίνεται στην παρακάτω εικόνα:



Για τις φάσεις 1 και 3 χρησιμοποιήθηκαν πολυώνυμα 5ου βαθμού, ενώ για η φάση 2 χαρακτηρίζεται απο σταθερή ταχύτητα. Πιο συγκριμένα η φάση 1 μπορεί να ονομαστεί ως φάση επιτάχυνσης, η φάση 2 ως φάση σταθερής ταχύτητας και η φάση 3 ως φάση επιβράδυνσης. Με τα πολυώνυμα 5ου βαθμού επιτυγχάνουμε την συνέχεια τόσο ως προς την ταχύτητα όσο και ως προς την επιτάχυνση-επιβράδυνση.
Αρχικά ορίσαμε την κίνηση απο το σημείο Pa εως το σημείο Pb να διαρκεί 6 sec και έτσι η συνολική μας κίνηση διαρκεί 12 sec. Έπειτα ορίζουμε τις οριακές συνθήκες για την θέση, την ταχύτητα, καθώς και για την επιτάχυνση όπως φαίνεται παρακάτω:

```
px0 = start_x #startx=0.6043
py0 = start_y #starty=-0.2
pz0 = start_z #startz=0.1508

px1 = start1_x
py1 = -0.07
pz1 = start1_z

px2 = start2_x
py2 = 0.07
pz2 = start2_z

px3 = 0.6043
py3 = 0.2
pz3 = 0.1508
```

```
vx0 = 0
vy0 = 0
vz0 = 0

vx1 = (px2 - px1)/T2
vy1 = (py2 - py1)/T2
vz1 = (pz2 - pz1)/T2

vx2 = vx1
vy2 = vy1
vz2 = vz1

vx3 = 0
vy3 = 0
vz3 = 0
```

```
gx0 = 0
gy0 = 0
gz0 = 0

gx1 = 0
gy1 = 0
gz1 = 0

gx2 = 0
gy2 = 0
gz2 = 0

gx3 = 0
gy3 = 0
gz3 = 0
```

Οι οριακές συνθήκες για την θέση είναι ακριβώς αντίστοιχες στην περίπτωση που πηγαίνουμε απο το σημείο Pb στο Pa, όπως φαίνεται παρακάτω:

```
px0 = start_x
py0 = start_y
pz0 = start_z

px1 = start1_x
py1 = 0.07
pz1 = start1_z

px2 = start2_x
py2 = -0.07
pz2 = start2_z

px3 = 0.6043
py3 = -0.2
pz3 = 0.1508
```

Όπως φαίνεται, επιλέξαμε οι 3 φάσεις να έχουν την ίδια διάρκεια και να καλύπτουν σχεδόν την ίδια απόσταση στον άξονα y. Έχοντας ορίσει λοιπόν τα παραπάνω, αναλυτικά έχουμε:

Για την φάση 1:

Έχουμε το πολυώνυμο $P= a_0 + a_1 * t_1 + a_2 * t_1\text{^}2 + a_3 * t_1\text{^}3 + a_4 * t_1\text{^}4 + a_5 * t_1\text{^}5$ το οποίο παραγωγίζουμε 2 φορές και με αυτό το τρόπο κατασκευάζουμε τον πίνακα:

```
A1 = np.matrix([[1, 0, 0, 0, 0, 0],
                [0, 1, 0, 0, 0, 0],
                [0, 0, 1, 0, 0, 0],
                [1, T1, T1**2, T1**3, T1**4, T1**5],
                [0, 1, 2*T1, 3*(T1**2), 4*(T1**3), 5*(T1**4)],
                [0, 0, 2,  6*T1, 12*(T1**2), 20*(T1**3)]])
```

Επίσης κατασκευάζουμε και τους πίνακες με τις ζητούμενες καθε φορά τιμές ως εξής:

```
B1x = np.array([px0,    vx0,   gx0,     px1,     vx1,    gx1])
B1y = np.array([py0,    vy0,   gy0,     py1,     vy1,    gy1])
B1z = np.array([pz0,    vz0,   gz0,     pz1,     vz1,    gz1])
```

Λύνοντας λοιπόν τη γραμμίκη εξίσωση $A_1 * X = B$ παίρνουμε τους ζητούμενους συντελεστές a_i, i=0,1…,5. Έχοντας πλέον τους συντελεστές κατασκευάζουμε πάλι το πολυώνυμο χρησιμοποιώντας αυτούς τόσο για την επιθυμητή θέση (p desired) όσο και για την επιθυμητή ταχύτητα (v desired). Οι εξισώσεις φαίνονται παρακάτω:

```
pdx1 = ax[0] + ax[1]*t1 + ax[2] * \
    (t1**2) + ax[3]*(t1**3) + ax[4]*(t1**4) + ax[5]*(t1**5)
pdy1 = ay[0] + ay[1]*t1 + ay[2] * \
    (t1**2) + ay[3]*(t1**3) + ay[4]*(t1**4) + ay[5]*(t1**5)
pdz1 = az[0] + az[1]*t1 + az[2] * \
    (t1**2) + az[3]*(t1**3) + az[4]*(t1**4) + az[5]*(t1**5)

vdx1 = ax[1] + 2*ax[2]*t1 + 3*ax[3] * \
    (t1**2) + 4*ax[4]*(t1**3) + 5*ax[5]*(t1**4)
vdy1 = ay[1] + 2*ay[2]*t1 + 3*ay[3] * \
    (t1**2) + 4*ay[4]*(t1**3) + 5*ay[5]*(t1**4)
vdz1 = az[1] + 2*az[2]*t1 + 3*az[3] * \
    (t1**2) + 4*az[4]*(t1**3) + 5*az[5]*(t1**4)
```

Για την φάση 2:

Εδώ τα πράγματα είναι πολύ πιο απλά καθώς δεν χρησιμοποιούμε κάποιο πολυώνυμο και επειδή μιλάμε για την φάση σταθερής ταχύτητας οι μόνες προσαρμογές που χρειάζεται να γίνουν φαίνονται παρακάτω:

```
pdx2 = px1 + vx1*t2
pdy2 = py1 + vy1*t2
pdz2 = pz1 + vz1*t2

vdx2 = vx1   #constant velocities
vdy2 = vy1
vdz2 = vz1
```

Για την φάση 3:

Χρησιμοποιείται ακριβώς η ίδια διαδικασία με την φάση 1 και έτσι οι σχέσεις για την ταχύτητα και την θέση στις οποίες καταλήγουμε είναι οι εξής:

```
pdx3 = bx[0] + bx[1]*t3 + bx[2] * \
    (t3**2) + bx[3]*(t3**3) + bx[4]*(t3**4) + bx[5]*(t3**5)
pdy3 = by[0] + by[1]*t3 + by[2] * \
    (t3**2) + by[3]*(t3**3) + by[4]*(t3**4) + by[5]*(t3**5)
pdz3 = bz[0] + bz[1]*t3 + bz[2] * \
    (t3**2) + bz[3]*(t3**3) + bz[4]*(t3**4) + bz[5]*(t3**5)

vdx3 = bx[1] + 2*bx[2]*t3 + 3*bx[3] * \
    (t3**2) + 4*bx[4]*(t3**3) + 5*bx[5]*(t3**4)
vdy3 = by[1] + 2*by[2]*t3 + 3*by[3] * \
    (t3**2) + 4*by[4]*(t3**3) + 5*by[5]*(t3**4)
vdz3 = bz[1] + 2*bz[2]*t3 + 3*bz[3] * \
    (t3**2) + 4*bz[4]*(t3**3) + 5*bz[5]*(t3**4)
```

Ελέγχοντας λοιπόν κάθε φορά σε ποιό κομμάτι της κίνησης είμαστε κράταμε στα desired velocities και πλέον αυτά αποτελούν το p_{id} που είχαμε ορίσει στην αρχική μας σχέση για την πρώτη υποεργασία.
Επομένως, έχοντας υπολογίσει την ψευδοανάστροφη της ιακωβιανής μήτρας μπορούμε κάθε στιγμή να βρούμε τις γενικευμένες ταχύτητες $(q)'_{(1)}$ οι οποίες συντελούν στην παρακολούθηση της τροχιάς.

2η Υποεργασία: (αποφυγή εμποδίων)

Για την συγκρεκριμένη υποεργασία θέλουμε να εξάγουμε τις γενικευμένες ταχύτητες των αρθρώσεων οι οποίες είναι αναγκαίες για την αποφυγή των 2 κυλινδρικών εμποδίων. Πιο συγκεκριμένα, θα χρησιμοποιήσουμε τον τύπο:

$(q)'_{(2)} = (I_n - J^+ * J) * q_{r2}$ όπου σαν q_{r2} ορίζουμε το γινόμενο $k_c * \xi$ όπου:

ξ : διάνυσμα κλίσης της συνάρτησης κριτηρίου
k_c : κέρδος της δεύτερης ρομποτικής υποεργασίας το οποίο προσδιορίσαμε πειραματικά
Σαν συνάρτηση κριτηρίου ορίζουμε την συνάρτηση:

$$c(q)=\sqrt{((x-a)^2 + (y-b)^2)}$$ όπου:

x,y : οι συντεταγμένες του κάθε κρίσημου σημείου που επιλέξαμε
a,b : οι αντίστοιχες συντεταγμένες των εμποδίων

Όπως φαίνεται απο την διάταξη που έχουμε, το ρομπότ μας δεν κινδυνεύει να χτυπήσει με κάποιον τρόπο στον άξονα z και επομένως αναγάγουμε το πρόβλημα σε πρόβλημα 2 διαστάσεων. Οι συντεταγμένες του κέντρου των εμποδίων μας είναι γνωστές και μπορούμε κάθε στιγμή να τις ξέρουμε. Έστω ότι :

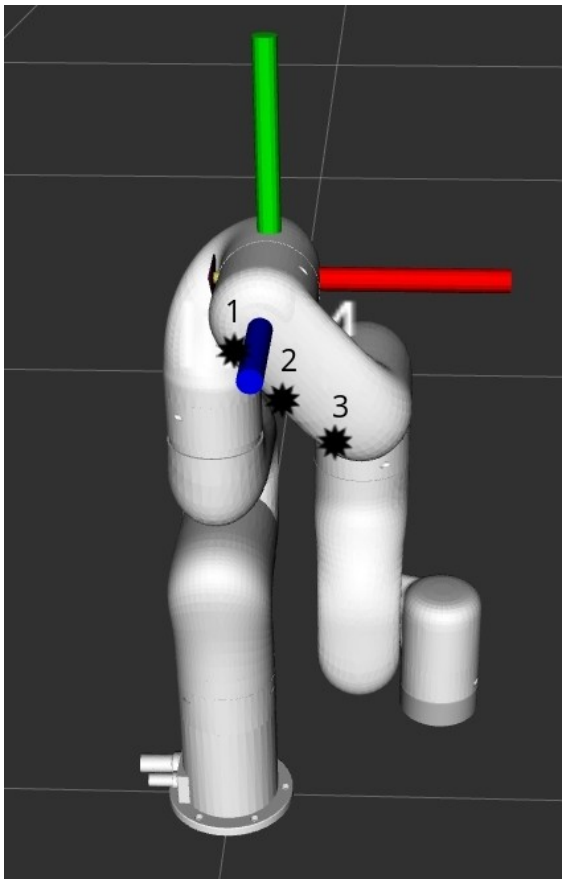
obs1_pos_x, obs1_pos_y : είναι οι συντεταγμένες του κέντρου του πράσινου εμποδίου
obs2_pos_x, obs2_pos_y : είναι οι συντεταγμένες του κέντρου του κόκκινου εμποδίου

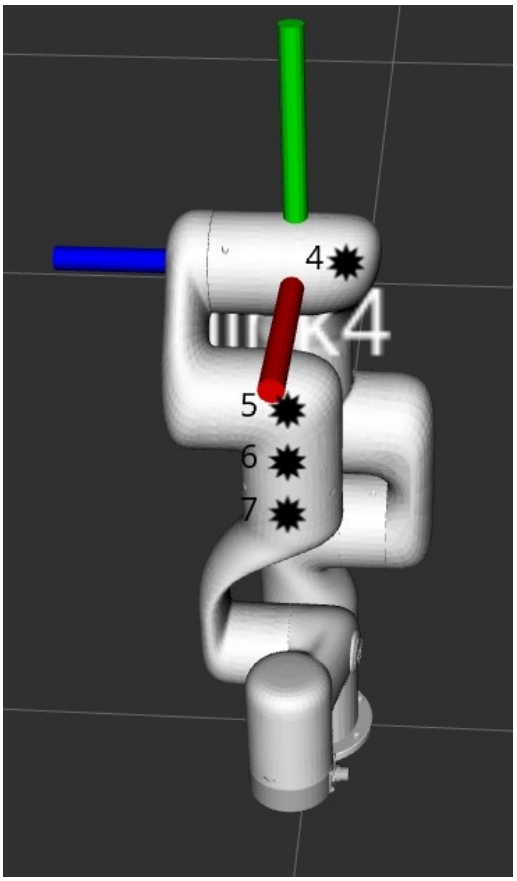
Έχοντας αυτες τις μεταβλητές, προσθέτουμε την ακτίνα του πράσινου εμποδίου (επειδή βρίσκεται στα αρνητικά y) και αφαιρούμε την ακτίνα του κόκκινου εμποδιού (επείδη βρίσκετα στα θετικά y) απο την συντεταγμένη y ώστε να έχουμε τις συντεγμένες του σημείου του εξωτερικού φλοιού που βρίκεται στην ίδια ευθεία με το κέντρο. Εκτελώντας την παραπάνω διεργασία παίρνουμε 2 set συντεταγμλενων για τα 2 εμπόδια όπως φαίνεται παρακάτω:

```
a1  = obs1_pos_x
b1  = obs1_pos_y + 0.05
a2  = obs2_pos_x
b2  = obs2_pos_y - 0.05
```

όπου:
(a_1,b_1) → συντεταγμένες πράσινου εμποδίου
(a_2,b_2) → συντεταγμένες κόκκινου εμποδίου
Όπως είναι λογικό ανάλογα τη θέση του κρίσιμου σημείου κάθε φορά χρησιμοποιούμε το αντίστοιχο σετ συντεταγμένων από το οποίο το ρομπότ διατρέχει μεγαλύτερο κίνδυνο.

Ορισμός κρίσιμων σημείων:
Παρατηρώντας την διάταξη του ρομπότ καταλήξαμε στον εντοπισμό 7 κρίσιμων σημείων. Αναλυτικά τα σημεία φαίνονται στις παρακάτω φωτογραφίες:





Όπως φαίνεται παραπάνω, όλοι οι πίνακες για τα παραπάνω σημεία μπορούν να υπολογιστούν χρησιμοποιώντας τον πίνακα A^0_4 και κάποια γεωμετρικά χαρακτηριστικά.

Πιο συγκεκριμένα:

Για τα σημεία 1, 2, 3: Ξέροντας την θέση που έχει τοποθετηθεί το πλαίσιο της 4ης άρθρωσης μετακινήσαμε το πράσινο εμπόδιο (έχοντας ακίνητο το robot) έτσι ώστε να ακουμπήσει την πλευρά που υπάρχουν τα σημεία 1, 2, 3. Γνωρίζοντας λοιπόν (από το gazebo) και τη θέση του κέντρου του εμποδίου βρήκαμε πόσο πρέπει να μετακινήσουμε το πλαίσιο 4 κατά z ώστε να το τοποθετήσουμε στο κέντρο της πλευράς που θέλουμε(μετακίνηση κατά 0.1062m). Έπειτα στρίψαμε το πλαίσιο κατά 45 μοίρες ως προς z ώστε ο άξονας x να είναι παράλληλος με την πλευρά που θέλουμε. Ύστερα παρατηρήσαμε ότι δεν θέλουμε τα σημεία μας να είναι στο κέντρο της πλευράς αλλά στο κάτω μέρος της. Αυτό συμβαίνει λόγω της κλίσης που παίρνει το ρομπότ ώστε να αποφύγει το πράσινο εμπόδιο και επομένως συμπαιρένουμε ότι αυτά τα σημεία κινδυνεύουν περισσότερο. Τέλος, για να πραγματοποιήσουμε αυτή τη κίνηση μετατοπιζόμαστε κατά τα αρνητικά του y όσο είναι η ακτίνα της άρθρωσης 4 (μετακίνηση κατά 0.063m).
Συνοπτικά, ο πίνακας που θα μας δώσει τη θέση του σημείου 1 είναι ο εξής:

$$L_1 = A^0_4 * Tra(z, 0.1062) * Rot(z, 45^o) * Tra(y, -0.063)$$

Τα σημεία 2, 3 δεν είναι παρά μια έξτρα μετατόπιση κατά τον άξονα x. Χρησιμοποιώντας πάλι το εμπόδιο παρατηρούμε ότι αυτή η πλευρά είναι μεγαλύτερη από 10 cm και επομένως οι πίνακες των σημείων 2, 3 προκύπτουν ως εξής:

$$L_2 = L_1 * Tra(x, 0.05)$$
$$L_3 = L_2 * Tra(x, 0.05)$$

Για το σημείο 4: Ξέροντας την θέση που έχει τοποθετηθεί το πλαίσιο της 4ης άρθρωσεις μετακινήσαμε το κόκκινο εμπόδιο έτσι ώστε να ακουμπήσει την πλευρά που είναι το σημείο 4. Επομένως, μετατοπίζοντας ως προς τον άξονα z κατά την διαφορά της απόστασης έχουμε:

$$L_4 = A^0_4 * Tra(z, -0.0676)$$

Για τα σημεία 5, 6, 7: Ξέροντας πλέον το σημείο 3 για να βρούμε το σημείο 5 κάνουμε μία μετατόπιση απο τον πίνακα L_3 ως προς τον άξονα z. Έπειτα ακολουθώντας παρόμοια διαδικασία με αυτή που ακολουθήθηκε για τα πρώτα 3 σημεία για να βρούμε την θέση των σημείων 6, 7 κάνουμε μία στροφή ως προς τον z κατα τις υπόλοιπες 45 μοίρες και μετά 1 μετατόπιση ως προς τον άξονα x κατά 0.05m για το σημείο 6 και κατά 0.1m για το σημείο 7. Οι πίνακες των σημειών είναι οι εξής:

$$L_5 = L_3 * Tra(z, -0.1473)$$
$$L_6 = L_5 * Rot(z, 45^o) * Tra(x, 0.05)$$
$$L_7 = L_6 * Tra(x, 0.05)$$

Έχοντας πλέον όλα τα x, y των κρίσιμων σημείων καθώς και τις θέσεις των εμποδίων μπορούμε να εφαρμόσουμε το κριτήριο. Κάθε φορά επιλέγουμε ποιο απο τα 2 σετ συντεταγμένων των εμποδίων θα χρησιμοποιήσουμε καθώς τα σημεία 1, 2, 3 κινδυνεύουν μόνο απο το πράσινο εμπόδιο, ενώ τα σημεία 4, 5, 6, 7 κινδυνεύουν από το κόκκινο εμπόδιο.

Υπολογισμός των ξ:

Ορίζουμε ως ξ το διάνυσμα κλίσης της συνάρτησης κριτηρίου c(q) δηλαδή:

$$\boldsymbol{\xi = - \partial (c(q))/\partial q}$$

Επειδή έχουμε 7 στροφικές αρθρώσεις και κατα συνέπεια 7 q το ξ θα είναι ένας πίνακας 7 στοιχείων. Ακόμα, επείδη επιλέξαμε 7 κρίσιμα σημεία, κάθε ένα απο αυτά τα κρίσιμα σημεία θα παράγει έναν πίνακα ξ_1 έως ξ_7. Ενδεικτικά θα παρουσιάσουμε την διαδικασία για το τέταρτο σημείο καθώς οι πράξεις ειναι πολύ μεγάλες και χαοτικές. Για τη διαδικασία παραγώγισης χρησιμοποιήθηκε η συνάρτηση :

```
def Deriv(v_str, function):
    vars = sym.symbols(v_str)
    f = (function)
    #print (f)
    J = sym.zeros(len(f))
    for i, fi in enumerate(f):
            J[i] = sym.diff(fi, vars[0])

    return J
```

Αρχικά λοιπόν, υπολογίζουμε τον πίνακα L_4 και εκτυπώνουμε τα στοιχεία $L_4[0,3]$ και $L_4[1,3]$ ώστε να παρούμε τα x, y του σημείου:

```
print(L4[0,3])
print('\n')
print(L4[1,3])
```

```
l2*sin(q2)*cos(q1) + l3*(-sin(q1)*sin(q3) + cos(q1)*cos(q2)*cos(q
3)) - 0.0676*sin(q1)*cos(q3) - 0.0676*sin(q3)*cos(q1)*cos(q2)


l2*sin(q1)*sin(q2) + l3*(sin(q1)*cos(q2)*cos(q3) + sin(q3)*cos(q
1)) - 0.0676*sin(q1)*sin(q3)*cos(q2) + 0.0676*cos(q1)*cos(q3)
```

Έπειτα καλώντας την συνάρτηση Deriv παίρνουμε κάθε φορά την μερική παράγωγο της ολικής συνάρτησης ως προς το πρώτο στοίχειο που δίνεται ως όρισμα v_str. Ακολουθώντας αυτή τη διαδικασία για το σημείο 4 προκύπτουν τα παρακάτω ξ:

```
        ksi1[3] = ((-2*self.l2*np.sin(q1)*np.sin(q2) + 2*self.l3*(-np.sin(q1)*np.cos(q2)*np.cos(q3) -
np.sin(q3)*np.cos(q1)) + 0.1352*np.sin(q1)*np.sin(q3)*np.cos(q2) - 0.1352*np.cos(q1)*np.cos(q3))*(-a2
+ self.l2*np.sin(q2)*np.cos(q1) + self.l3*(-np.sin(q1)*np.sin(q3) + np.cos(q1)*np.cos(q2)*np.cos(q3))
- 0.0676*np.sin(q1)*np.cos(q3) - 0.0676*np.sin(q3)*np.cos(q1)*np.cos(q2))/2 +
(2*self.l2*np.sin(q2)*np.cos(q1) + 2*self.l3*(-np.sin(q1)*np.sin(q3) +
np.cos(q1)*np.cos(q2)*np.cos(q3)) - 0.1352*np.sin(q1)*np.cos(q3) -
0.1352*np.sin(q3)*np.cos(q1)*np.cos(q2))*(-b2 + self.l2*np.sin(q1)*np.sin(q2) +
self.l3*(np.sin(q1)*np.cos(q2)*np.cos(q3) + np.sin(q3)*np.cos(q1)) -
0.0676*np.sin(q1)*np.sin(q3)*np.cos(q2) + 0.0676*np.cos(q1)*np.cos(q3))/2)/np.sqrt((-a2 +
self.l2*np.sin(q2)*np.cos(q1) + self.l3*(-np.sin(q1)*np.sin(q3) + np.cos(q1)*np.cos(q2)*np.cos(q3)) -
0.0676*np.sin(q1)*np.cos(q3) - 0.0676*np.sin(q3)*np.cos(q1)*np.cos(q2))**2 + (-b2 +
self.l2*np.sin(q1)*np.sin(q2) + self.l3*(np.sin(q1)*np.cos(q2)*np.cos(q3) + np.sin(q3)*np.cos(q1)) -
0.0676*np.sin(q1)*np.sin(q3)*np.cos(q2) + 0.0676*np.cos(q1)*np.cos(q3))**2)

        ksi2[3] = ((2*self.l2*np.sin(q1)*np.cos(q2) - 2*self.l3*np.sin(q1)*np.sin(q2)*np.cos(q3) +
0.1352*np.sin(q1)*np.sin(q2)*np.sin(q3))*(-b2 + self.l2*np.sin(q1)*np.sin(q2) +
self.l3*(np.sin(q1)*np.cos(q2)*np.cos(q3) + np.sin(q3)*np.cos(q1)) -
0.0676*np.sin(q1)*np.sin(q3)*np.cos(q2) + 0.0676*np.cos(q1)*np.cos(q3))/2 +
(2*self.l2*np.cos(q1)*np.cos(q2) - 2*self.l3*np.sin(q2)*np.cos(q1)*np.cos(q3) +
0.1352*np.sin(q2)*np.sin(q3)*np.cos(q1))*(-a2 + self.l2*np.sin(q2)*np.cos(q1) + self.l3*(-
np.sin(q1)*np.sin(q3) + np.cos(q1)*np.cos(q2)*np.cos(q3)) - 0.0676*np.sin(q1)*np.cos(q3) -
0.0676*np.sin(q3)*np.cos(q1)*np.cos(q2))/2)/np.sqrt((-a2 + self.l2*np.sin(q2)*np.cos(q1) + self.l3*(-
np.sin(q1)*np.sin(q3) + np.cos(q1)*np.cos(q2)*np.cos(q3)) - 0.0676*np.sin(q1)*np.cos(q3) -
0.0676*np.sin(q3)*np.cos(q1)*np.cos(q2))**2 + (-b2 + self.l2*np.sin(q1)*np.sin(q2) +
self.l3*(np.sin(q1)*np.cos(q2)*np.cos(q3) + np.sin(q3)*np.cos(q1)) -
0.0676*np.sin(q1)*np.sin(q3)*np.cos(q2) + 0.0676*np.cos(q1)*np.cos(q3))**2)

        ksi3[3] = ((2*self.l3*(-np.sin(q1)*np.cos(q3) - np.sin(q3)*np.cos(q1)*np.cos(q2)) +
0.1352*np.sin(q1)*np.sin(q3) - 0.1352*np.cos(q1)*np.cos(q2)*np.cos(q3))*(-a2 +
self.l2*np.sin(q2)*np.cos(q1) + self.l3*(-np.sin(q1)*np.sin(q3) + np.cos(q1)*np.cos(q2)*np.cos(q3)) -
0.0676*np.sin(q1)*np.cos(q3) - 0.0676*np.sin(q3)*np.cos(q1)*np.cos(q2))/2 + (2*self.l3*(-
np.sin(q1)*np.sin(q3)*np.cos(q2) + np.cos(q1)*np.cos(q3)) - 0.1352*np.sin(q1)*np.cos(q2)*np.cos(q3) -
0.1352*np.sin(q3)*np.cos(q1))*(-b2 + self.l2*np.sin(q1)*np.sin(q2) +
self.l3*(np.sin(q1)*np.cos(q2)*np.cos(q3) + np.sin(q3)*np.cos(q1)) -
0.0676*np.sin(q1)*np.sin(q3)*np.cos(q2) + 0.0676*np.cos(q1)*np.cos(q3))/2)/np.sqrt((-a2 +
self.l2*np.sin(q2)*np.cos(q1) + self.l3*(-np.sin(q1)*np.sin(q3) + np.cos(q1)*np.cos(q2)*np.cos(q3)) -
0.0676*np.sin(q1)*np.cos(q3) - 0.0676*np.sin(q3)*np.cos(q1)*np.cos(q2))**2 + (-b2 +
self.l2*np.sin(q1)*np.sin(q2) + self.l3*(np.sin(q1)*np.cos(q2)*np.cos(q3) + np.sin(q3)*np.cos(q1)) -
0.0676*np.sin(q1)*np.sin(q3)*np.cos(q2) + 0.0676*np.cos(q1)*np.cos(q3))**2)

        ksi4[3] = 0
        ksi5[3] = 0
        ksi6[3] = 0
        ksi7[3] = 0
```

Όπως φαίνεται απο την φωτογραφία, κάθε ξ αποθηκεύεται στην 3η θέση (επειδή είναι το τέταρτο σημείο και ξεκινήσαμε απο το 0) 7 διαφορετικών πινάκων. Το νούμερο του κάθε πίνακα δείχνει τον δείκτη της μεταβλητής ως προς την οποία έγινε η παραγώγιση.

Παρατηρούμε πως αφού όλα τα κρίσημα σημεία εξαρτώνται από τον πίνακα A^0_4, άρα και από q1, q2, q3, q4, (το παραπάνω μόνο απο q1, q2, q3) οι μερικές παράγωγοι ως προς q5, q6, q7 θα είναι μηδενικές.

Εκτελώντας την παραπάνω διαδικασία και για τα 7 σημεία, προσθέτουμε όλα τα ξ που έχουν προκύπτει απο την παραγώγιση με το ίδιο q μεταξύ τους και τέλος φτιάχνουμε έναν πίνακα με αυτά τα παραγώμενα ξ των οποίων και χρησιμοποιούμε ως το τελικό ξ που φαίνεται στην εξίσωση για την δεύτερη υποεργασία.

```
for i in range(0,7):
    ksi_1 = ksi_1 + ksi1[i]
    ksi_2 = ksi_2 + ksi2[i]   #where ksij[i] => j shows the qj over which we
    ksi_3 = ksi_3 + ksi3[i]   #take the derivative and i shows the critical
    ksi_4 = ksi_4 + ksi4[i]   #point number
    ksi_5 = ksi_5 + ksi5[i]
    ksi_6 = ksi_6 + ksi6[i]
    ksi_7 = ksi_7 + ksi7[i]

ksis = np.matrix([[ksi_1, ksi_2, ksi_3, ksi_4, ksi_5, ksi_6, ksi_7]])
```

Έχοντας πλέον γνωστό το ξ και ορίζοντας το κέρδος της διεργασίας ίσο με 10 η εξίσωση :

$(q)'_{(2)} = k_c * (I_n - J^+ * J) * \xi$ είναι γνωστή.

Για την συνολική διεργασία :

Γνωρίζοντας τις γενικευμένες ταχύτητες για την εκτέλεση τόσο της πρώτης όσο και της δεύτερης υποεργασίας καταλήγουμε στην γενική εξίσωση :

$$\mathbf{(q)' = J^+ * (p_{id})' + k_c * (I_n - J^+ * J) * \xi}$$

Τέλος, ολοκληρώνοντας την παραπάνω εξίσωση στο χρόνο παίρνουμε τις γωνίες που πρέπει να έχουν οι αρθρώσεις για την επίτευξη της συνολικής εργασίας.

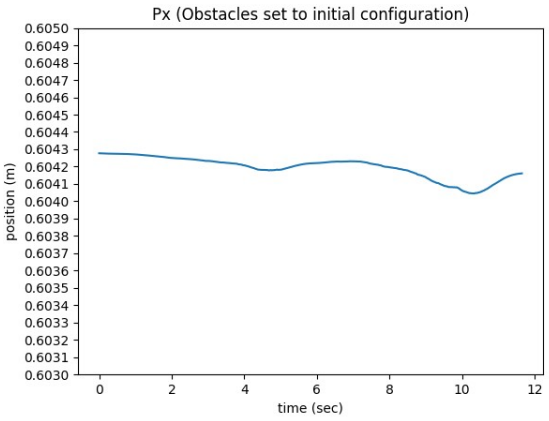
Προσομοίωση:

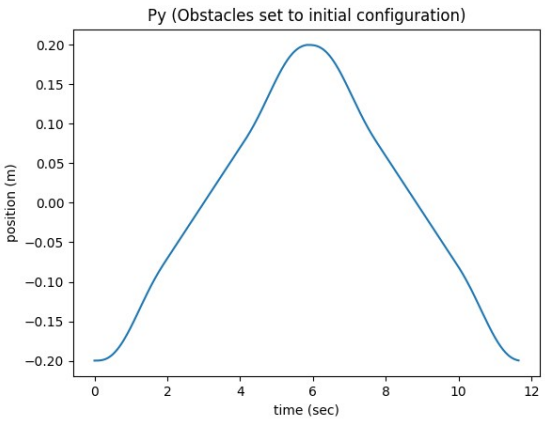
Το πρώτο πράγμα που καλούμαστε να φτίαξουμε τρέχοντας την προσομοίωση είναι να μετακινήσουμε το ρομπότ απο το configuration state (y=0) στην θέση Pa ώστε να ξεκινήσει η κανονική του τροχία. Για την επίτευξη αυτού ορίζουμε την μεταβλητή ‘first_time’ ίση με 1, η οποία παραμένει 1 μέχρι το ρομπότ να πάει απο την αρχική του κατάσταση στο άκρο Pa. Έπειτα για ολή την υπόλοιπη διάρκεια της κίνησης η μεταβλητή αυτή γίνεται 0 και το ρομπότ πλέον πηγαίνει συνεχώς απο το άκρο Pa στο Pb και αντίστροφα.
Επίσης, για την αλλαγή της κίνησης της τροχιάς χρησιμοποιούμε μία ακομα μεταβλητή ‘direction’, η οποία είναι ίση με 1 όταν κινούμαστε από το σημείο Pa προς το Pb και ίση με 0 οταν κινούμαστε απο το σημείο Pb προς το Pa.

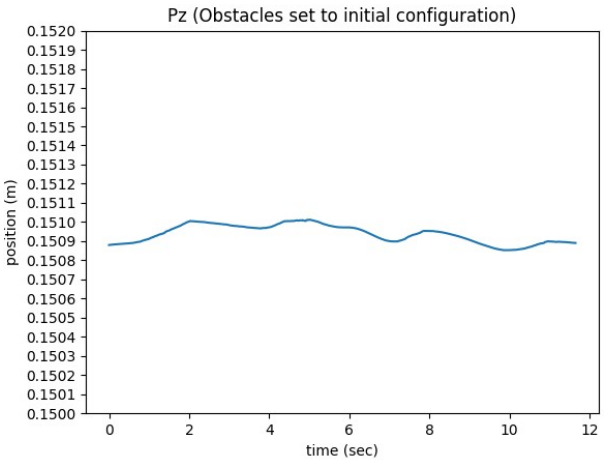
Για να γίνουν ευκολότερα κατανοητά τα αποτελέσμα που εξήγαμε απο την προσομοίωση θα χωρίσουμε τις πιθανές κινήσης των εμποδίων σε 3 καταστάσεις. Όλα τα διαγράμματα που θα δείξουμε παρακάτω αφορούν την ολόκληρη κίνηση. Σε κάθε κατάσταση, οι γραφικές παραστάσεις αφορούν την θέση του τελικού στοιχείου δράσης, το αντίστοιχο σφάλμα που προκύπτει από την ιδανική τροχιά, την ταχύτητα του τελικού σημείου δράσης, καθώς και την απόσταση που διατηρούν τα κρίσιμα σημεία απο τα εμπόδια.
Για όλες τις προσομοιώσεις επιλέχθηκε rate=1000Hz.

<u>Κατάσταση 1</u> – Εμπόδια ακίνητα στο αρχικό configuration

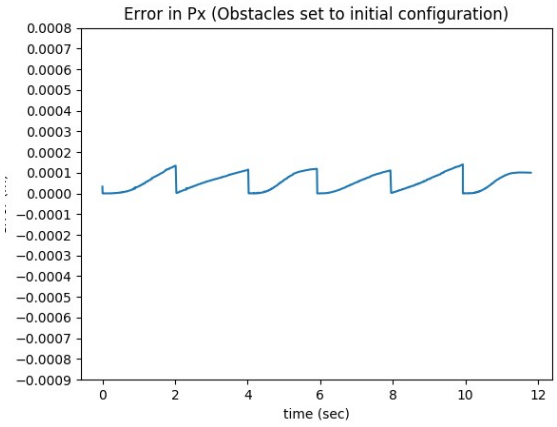
•Θέση τελικού στοιχείου δράσης:

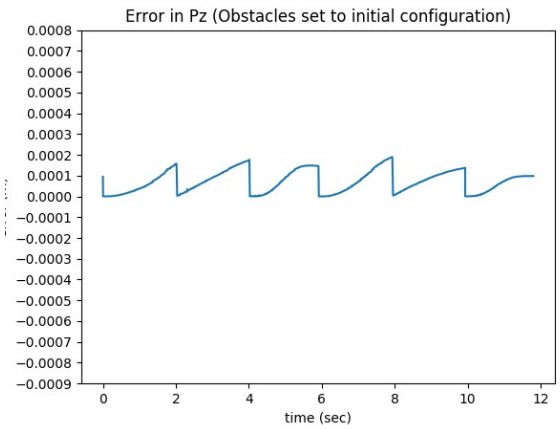




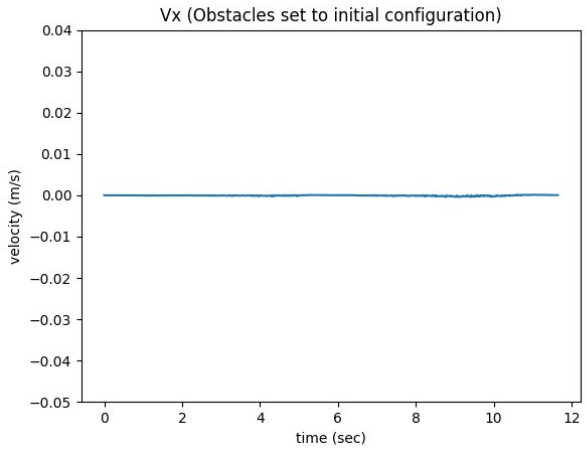


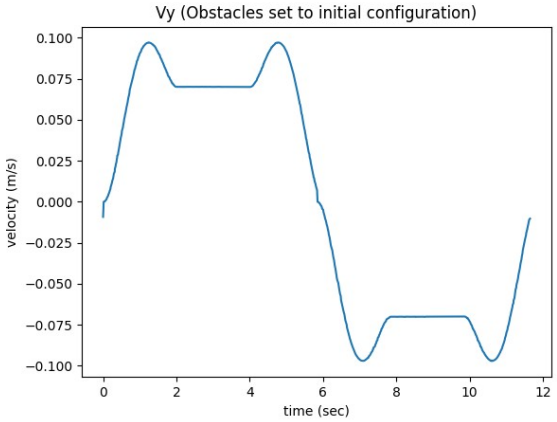
•Σφάλμα θέσης τελικού στοιχείου δράσης στις σταθερές διευθύνσεις x, z:

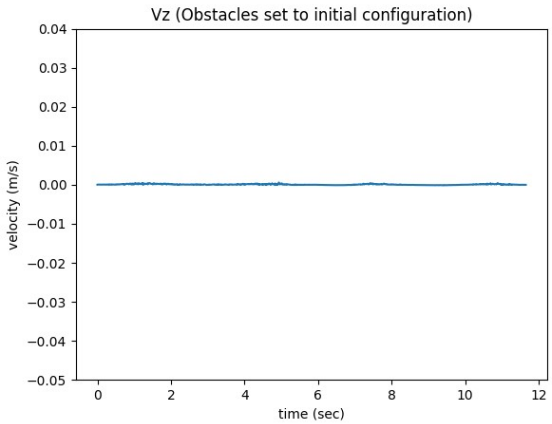




•Ταχύτητα τελικού στοιχείου δράσης:

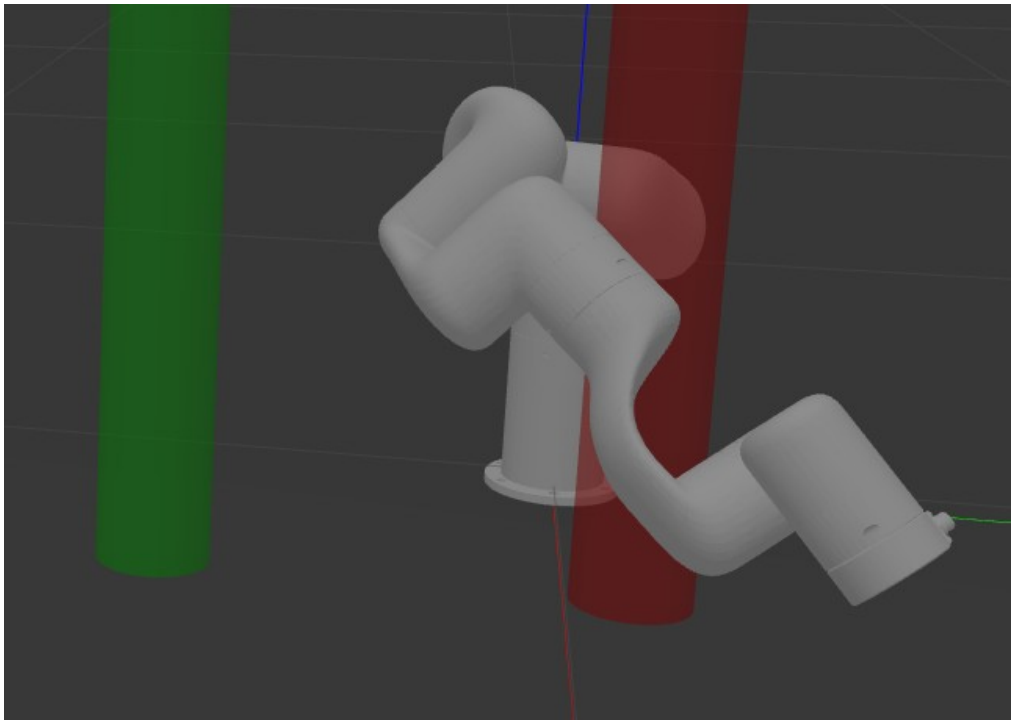




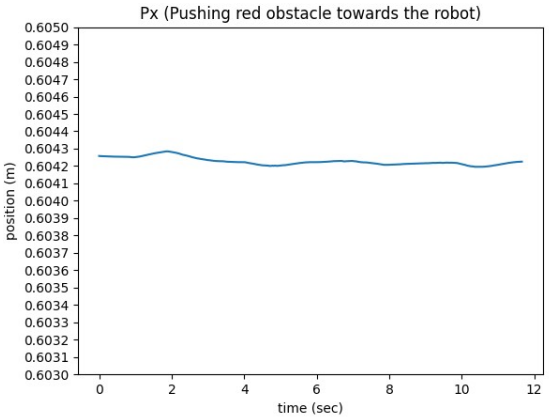


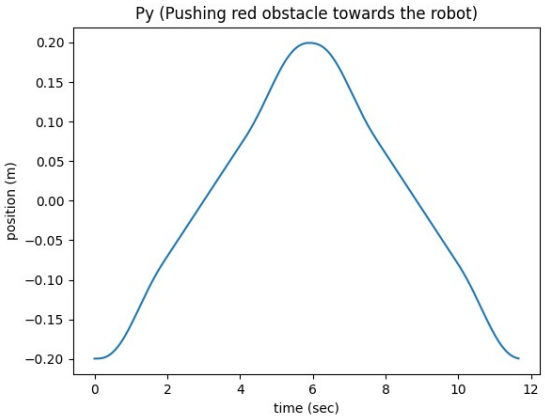
<u>Κατάσταση 2</u> – Κουνώντας σταδιακά τα εμπόδια έτσι ώστε το κόκκινο να πλησιάσει τον άξονα x στην κοντινότερη δυνατή θέση (κέντρο κόκκινου εμποδίου με y=0.054m), όπου όλες οι υποεργασίες εκτελούνται με επιτυχία.

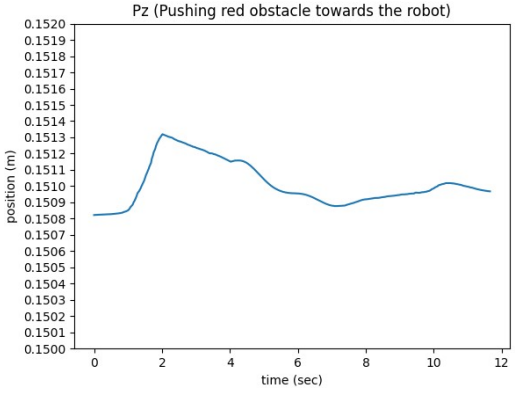
Παρακάτω η φωτογραφία με τη διάταξη του ρομποτικού βραχίωνα κατά την ακραία θέση του κόκκινου εμποδίου:



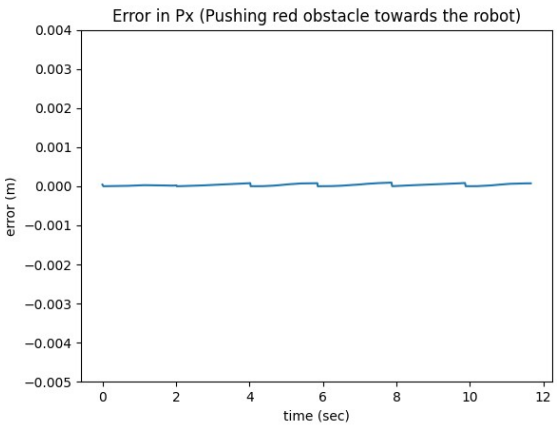
•Θέση τελικού στοιχείου δράσης:

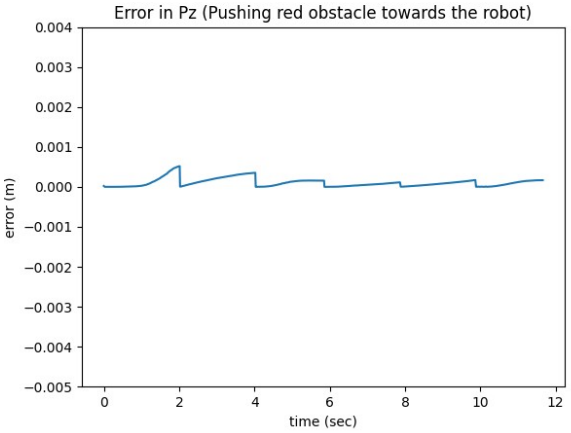




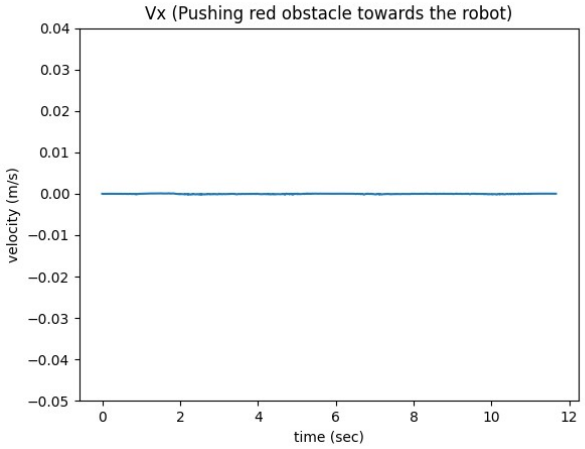


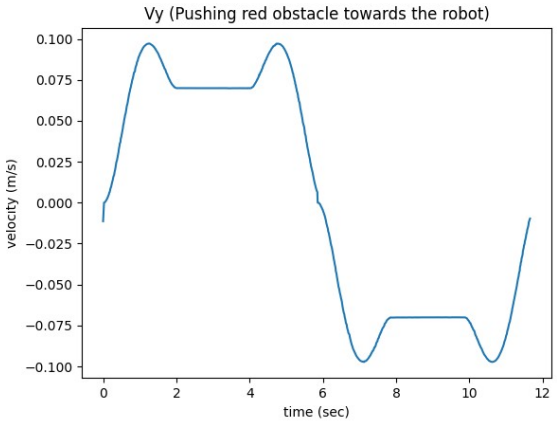
•Σφάλμα θέσης τελικού στοιχείου δράσης στις σταθερές διευθύνσεις x,z:

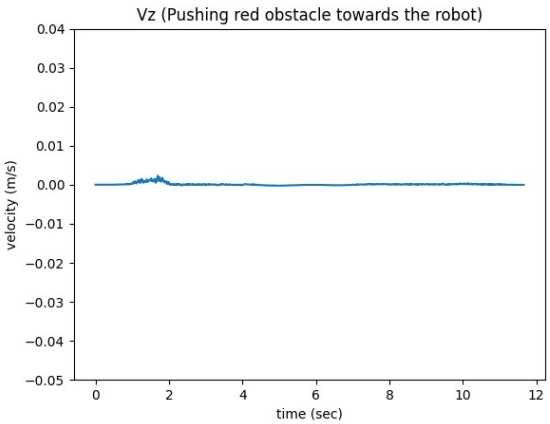




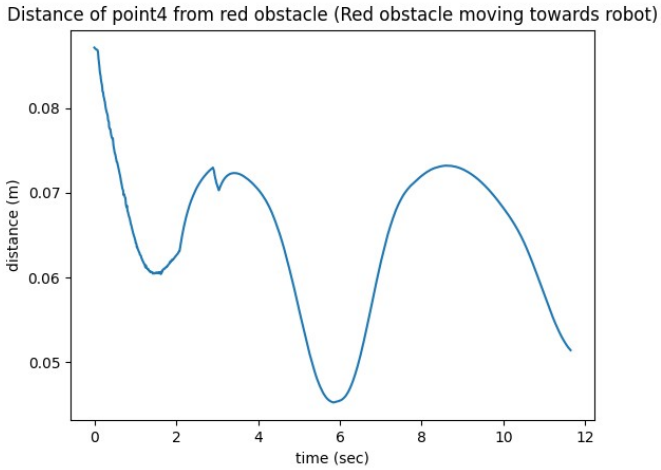
•Ταχύτητα τελικού στοιχείου δράσης:





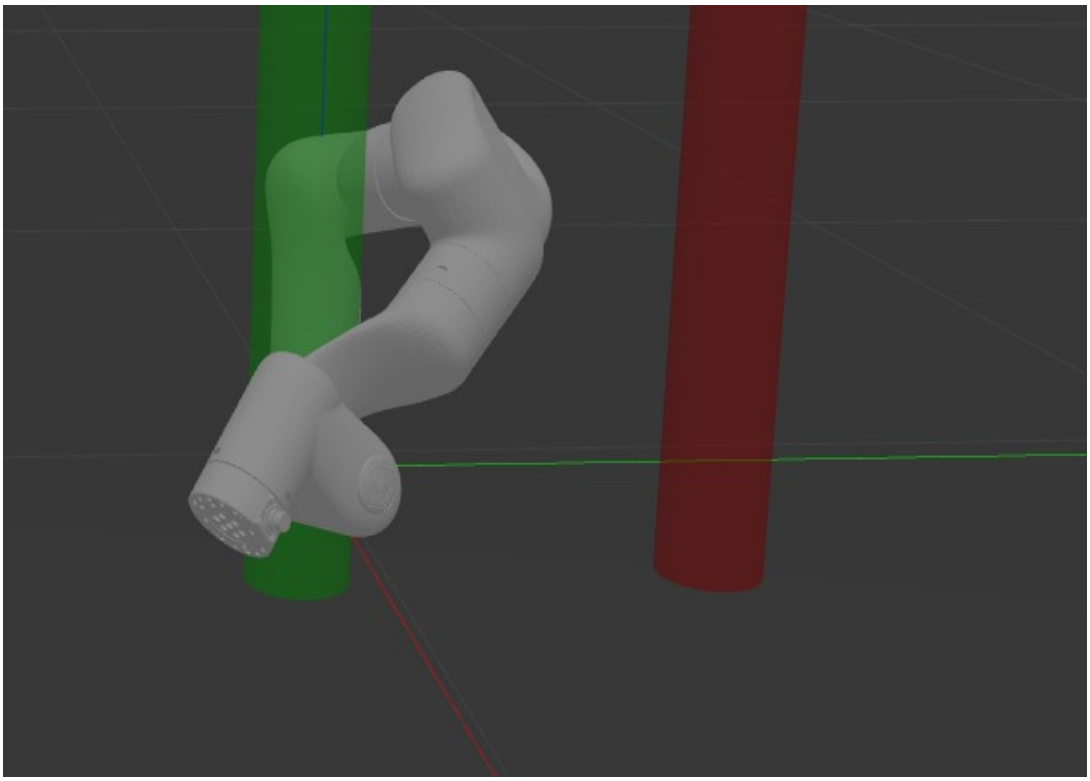


•Απόσταση κρίσιμου σημείου L4 από κόκκινο εμπόδιο (από σημείο της εξωτερικής επιφάνειας):

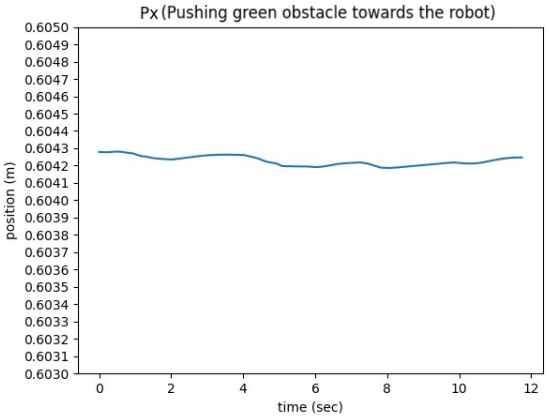


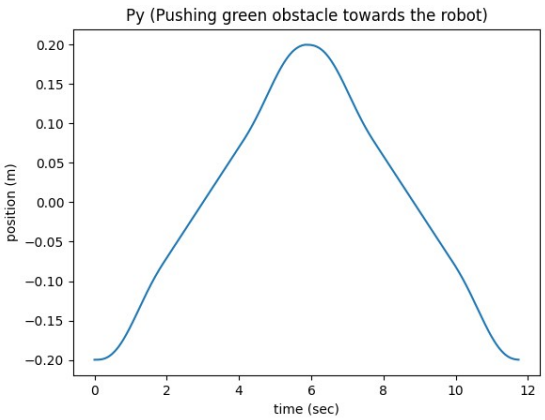
<u>Κατάσταση 3</u> – Κουνώντας σταδιακά τα εμπόδια έτσι ώστε το πράσινο να πλησιάσει τον άξονα x στην κοντινότερη δυνατή θέση (κέντρο πράσινου εμποδίου με y=-0.077m), όπου όλες οι υποεργασίες εκτελούνται με επιτυχία.

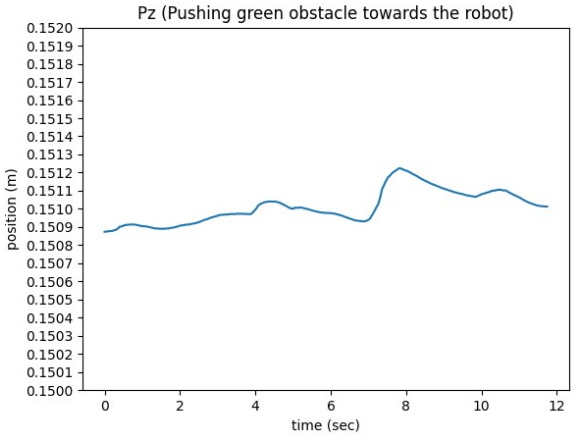
Παρακάτω η φωτογραφία με τη διάταξη του ρομποτικού βραχίωνα κατά την ακραία θέση του πράσινου εμποδίου:



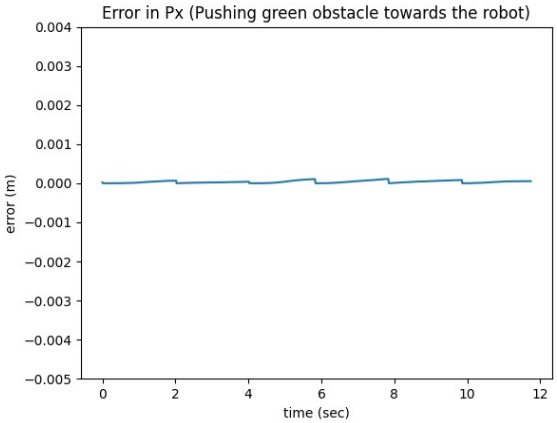
•Θέση τελικού στοιχείου δράσης:

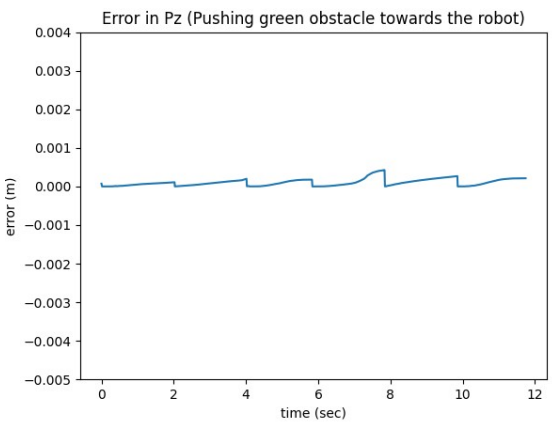




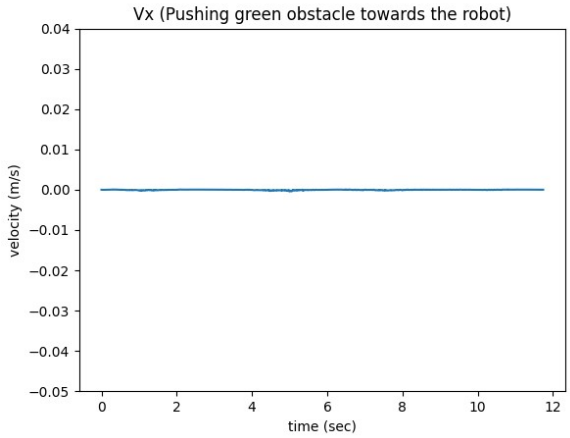


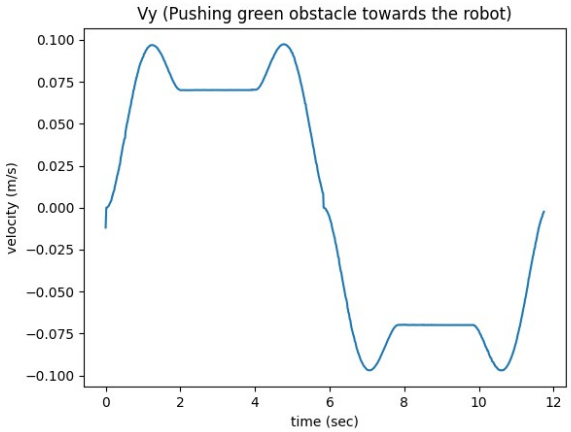
•Σφάλμα θέσης τελικού στοιχείου δράσης στις σταθερές διευθύνσεις x,z:

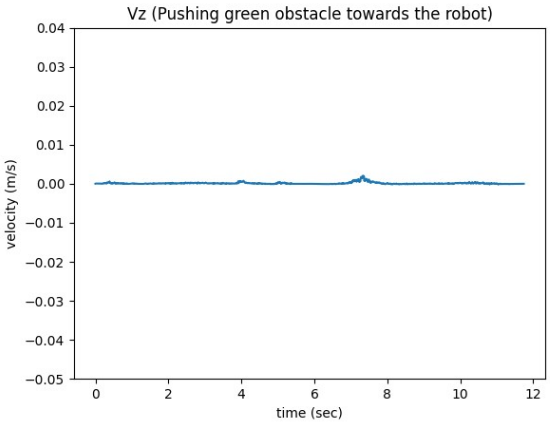




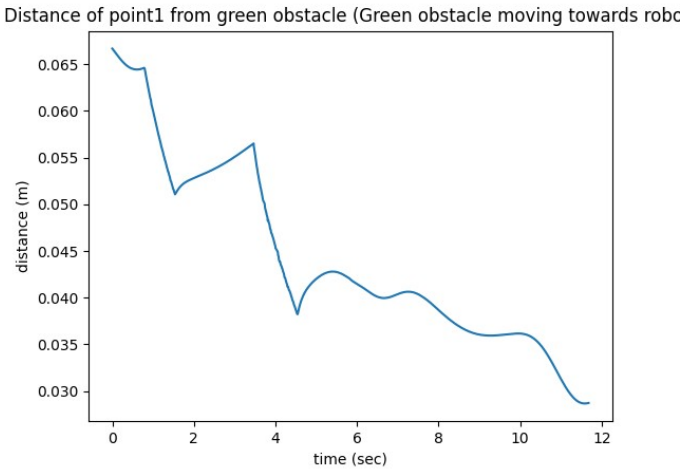
•Ταχύτητα τελικού στοιχείου δράσης:







•Απόσταση κρίσιμου σημείου L1 από πράσινο εμπόδιο (από σημείο της εξωτερικής επιφάνειας):



Παρατηρήσεις:

Σχετικά με τη θέση του τελικού σημείου δράσης, βλέπουμε ότι κατά τη διάρκεια που μετακινούμε τα εμπόδια έχουμε ένα αυξανόμενο σφάλμα, το οποίο μειώνεται σταδιακά μόλις σταθεροποιηθούν. Ωστόσο, το σφάλμα αυτό είναι αναμενόμενο, ενώ επίσης είναι ανάλογο της ταχύτητας που μετακινούμε τα εμπόδια, καθώς όσο πιο απότομα τα κουνάμε, τόσο μεγαλύτερο είναι το σφάλμα. Αυτό είναι απόλυτα λογικό, αφού σε απότομες μεταβολές το ρομπότ έχει λιγότερο χρόνο να 'αντιδράσει', συνεπώς και η κίνηση αποφυγής των εμποδίων είναι πιο απότομη. Πιο συγκεκριμένα:

- Κατάσταση 1: Καθώς τα εμπόδια παραμένουν σε σταθερές θέσεις, το ρομπότ προσαρμόζεται πολύ γρήγορα για την αποφυγή τους, με αποτέλεσμα το σφάλμα στις διευθύνσεις x,z να είναι της τάξης των 0.0003m στη χειρότερη περίπτωση.

- Καταστάσεις 2,3: Σε αυτές τις περιπτώσεις, το σφάλμα αυξάνεται κατά τη διάρκεια κίνησης των εμποδίων (ανάλογο της ταχύτητας κίνησής τους) και εξομαλύνεται καθώς τα εμπόδια ακινητοποιούνται. Το σφάλμα θέσης του τελικού στοιχείου δράσης στη διεύθυνση του x εξακολουθεί να είναι της τάξης των 0.0003-0.0004m στη χειρότερη περίπτωση, ενώ το σφάλμα στη διεύθυνση του z, είναι της τάξης των 0.0005-0.0006m.

- Οι παραπάνω τιμές σφαλμάτων είναι εξαρτώμενες από την ταχύτητα κίνησης των εμποδίων, και μπορεί να διαφέρουν ανάλογα με τον τρόπο που ο χρήστης χειρίζεται τα εμπόδια.

Για λόγους εξασφάλισης της επιτυχούς ολοκλήρωσης της 1ης ρομποτικής υποεργασίας, ακόμα και όταν τα εμπόδια βρίσκονται σε ακραίες θέσεις (όπως στα παραπάνω στιγμιότυπα), δώσαμε ένα περιθώριο σφάλματος μισού χιλιοστού από τις πραγματικές θέσεις των τελικών σημείων Pa, Pb.

Σχετικά με την ταχύτητα του τελικού σημείου δράσης, βλέπουμε ότι όπως περιμέναμε, η ταχύτητα στους άξονες x και z είναι σχεδόν μηδενική και παρουσιάζει κάποια spikes, πράγμα το οποίο είναι λογικό αφού υπάρχει μικρό σφάλμα στις διευθύνσεις x,z.
Παρατηρούμε επίσης, πως η ταχύτητα στον y άξονα έχει συνέχεια στο χρόνο και οι τρεις φάσεις της τροχιάς είναι ευδιάκριτες (επιτάχυνση, σταθερή ταχύτητα, επιβράδυνση).

Επιπλέον, σχετικά με τις αποστάσεις των παραπάνω κρίσιμων σημείων από τα αντίστοιχα εμπόδια, βλέπουμε ότι μειώνονται όσο πλησιάζει το κάθε σημείο το κοντινότερο σε αυτό εμπόδιο, ωστόσο διατηρείται μια απόσταση ασφαλείας σε κάθε περίπτωση.

Τέλος, από όλα τα παραπάνω συμπαιρένουμε ότι τα ρομπότ με πλεονάζοντες βαθμούς ελευθερίας έχει το πλεονέκτηματα ότι μπορεί να υλοποιήσει παραπάνω από μία διεργασίες ταυτόχρονα, πράγμα το οποίο κρίνεται απαραίτητο για την πλειονότητα των σύγχρονων ρομποτικών εργασιών. Ωστόσο, πολύ σημαντική είναι η επιλογή του κέρδους κάθε υποεργασίας, όπως η επιλογή του Kc για την αποφυγή των εμποδίων, καθώς ο χρήστης καλείται να βρει τη χρυσή τομή έτσι ώστε όλες οι υποεργασίες να εκτελούνται αρμονικά.

Όλες οι μέθοδοι που ακολουθήσαμε για τον υπολογισμό της Ιακωβιανής μήτρας, καθώς και των ξ(1 έως 7) κάθε κρίσιμου σημείου, βρίσκονται στο αρχείο jupyter notebook 'robo.ipynb' που επισυνάπτεται μαζί με το πακέτο που υλοποιήσαμε. Το έχουμε μετατρέψει και σε 'robo.py' σε περίπτωση που θέλετε απλά να το διαβάσετε.

Επίσης, στο αρχείο 'controller.py' έχουμε σχολιάσει πέρα από τις απαραίτητες επεξηγήσεις, το κομμάτι για την παραγωγή των plots που χρησιμοποιήσαμε στην παραπάνω αναφορά.