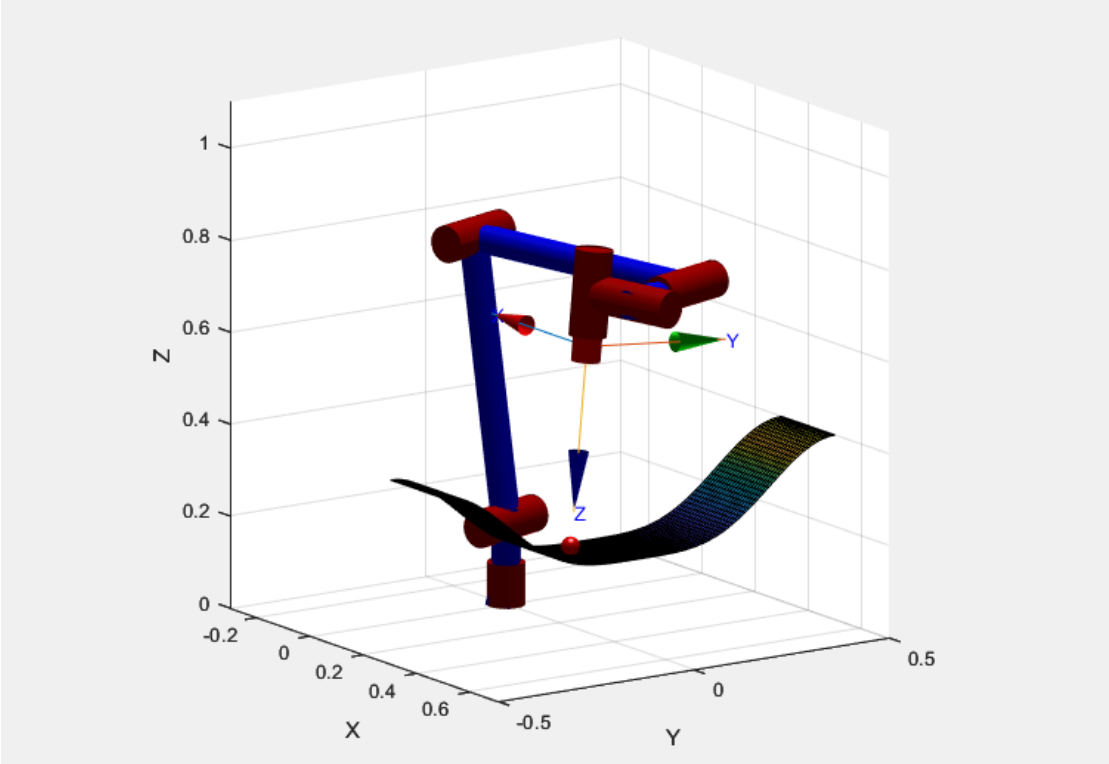
ΕΡΓΑΣΙΑ ΡΟΜΠΟΤΙΚΗΣ

Παρακολούθηση κίνησης σφαίρας σε καμπυλωτή επιφάνεια μέσω του ρομποτικού βραχίονα Ur10e_6



Ονοματεπώνυμο: Παπούλιας Μιχαήλ

ΑΕΜ: 10204

email: mpapouli@ece.auth.gr

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών ΑΠΘ

Ιούνιος 2023

Τμήμα Α

1) Κατασκευάζουμε το μοντέλο του ρομπότ μας μέσω της εντολής robot = mdl_ur10e() η οποία δημιουργεί ένα αντικείμενο τύπου mdl_ur10e() με ονομασία “robot”. Οι αρχικές θέσεις των αρθρώσεων δίνονται ίσες με q1 = [−0.140 −1.556 −1.359 1.425 −1.053 −1.732] rad. Μέσω της εντολής g_oe = robot.fkine(q1).T υλοποιούμε το ευθύ κινηματικό του βραχίονα, σύμφωνα με το οποίο από τις (γνωστές) θέσεις των αρθρώσεων μεταβαίνουμε στη θέση και τον προσανατολισμό του άκρου.

Το αποτέλεσμα του matlab δίνει τον επιθυμητό ομογενή μετασχηματισμό g_oe:

$$g_0e = \begin{bmatrix} -1 & 0.0003 & 0.0004 & 0.4 \\ 0.0001 & 0.8661 & -0.4999 & -0.2905 \\ -0.0005 & -0.4999 & -0.8661 & 0.8111 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

από τον οποίο προκύπτει αμέσως ότι ο ζητούμενος προσανατολισμός R_oe και η ζητούμενη θέση p_oe του άκρου e ως προς την αρχή 0 είναι:

$$R_{0e} = \begin{bmatrix} -1 & 0.0003 & 0.0004 \\ 0.0001 & 0.8661 & -0.4999 \\ -0.0005 & -0.4999 & -0.8661 \end{bmatrix} \quad \text{και}$$

$$p_{0e} = [0.4 \quad -0.2905 \quad 0.8111]^T$$

Μπορούμε να επιβεβαιώσουμε το παραπάνω αποτέλεσμα, εάν μελετήσουμε τον πίνακα παραμέτρων του ρομπότ για κάθε άρθρωση, που δίνεται από την εντολή robot

```
UR10 [Universal Robotics]:: 6 axis, RRRRRR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|     0.1807|          0|     1.5708|          0|
|  2|         q2|          0|    -0.6127|          0|          0|
|  3|         q3|          0|   -0.57155|          0|          0|
|  4|         q4|    0.17415|          0|     1.5708|          0|
|  5|         q5|    0.11985|          0|    -1.5708|          0|
|  6|         q6|    0.11655|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

Φαίνεται ότι για την πρώτη άρθρωση θα ισχύει:

$$g_{01} = g_r(z, q1) \cdot g_p(z, 0.1807) \cdot g_r(x, 1.5708)$$

όπου το $g_r(z, q1)$ είναι η στροφή γύρω από τον άξονα z κατά γωνία q1, διατηρώντας σταθερή τη θέση, ενώ το $g_p(z, 0.1807)$ είναι η μετατόπιση κατά τη διεύθυνση του άξονα z κατά 0.1807 μονάδες μήκους, διατηρώντας σταθερό (μοναδιαίο) προσανατολισμό. Παρόμοια, έχουμε:

$$g_{12} = g_r(z, q2) \cdot g_p(x, -0.6127)$$

$$g_{23} = g_r(z, q3) \cdot g_p(x, -0.57155)$$

$$g_{34} = g_r(z, q4) \cdot g_p(z, 0.17415) \cdot g_r(x, 1.5708)$$

$$g_{45} = g_r(z, q5) \cdot g_p(z, 0.11985) \cdot g_r(x, -1.5708)$$

$$g_{56} = g_r(z, q6) \cdot g_p(z, 0.11655)$$

Η τελική σύνθεση του μετασχηματισμού goe (ο οποίος ταυτίζεται με τον go6) υπολογίζεται από τη σχέση:

$$g_{oe} = g_{01} \cdot g_{12} \cdot g_{23} \cdot g_{34} \cdot g_{45} \cdot g_{56}$$

το αποτέλεσμα του οποίου θα είναι ακριβώς το ίδιο με αυτό που υπολογίστηκε παραπάνω.

2) Προκειμένου να σχεδιάσουμε κατάλληλο σήμα ελέγχου Q_dot, ώστε το άκρο του βραχίονα να παρακολουθεί την κινούμενη σφαίρα με τον ζητούμενο προσανατολισμό, θα εφαρμόσουμε κινηματικό έλεγχο για το άκρο.

Θα περιγράψουμε τη φιλοσοφία του κώδικα, ο κορμός του οποίου αποτελείται από μια for loop, μέσα στην οποία γίνονται οι υπολογισμοί για κάθε διακριτή χρονική στιγμή Ts = 2ms. Αρχικά καλούμε την εντολή [p_cb, v_cb , w_cb] = Wsp.sim_ball(Ts) σε κάθε κύκλο της for loop, η οποία επιστρέφει τη θέση, τη μεταφορική ταχύτητα και τη γωνιακή ταχύτητα, αντίστοιχα, του πλαισίου {B} ως προς το πλαίσιο της κάμερας {C}. Υπολογίζουμε την αρχική γωνία θ_0 μεταξύ του άξονα y του πλαισίου {B} και του άξονα y του πλαισίου {C}, γύρω από τον (κοινό κατά προσανατολισμό) άξονα x. Μας δίνεται ότι την t=0, ο άξονας y του {B} έχει κατεύθυνση

$[0 \;\; 0.9351 \;\; -0.3543]^T$. Το άνυσμα αυτό έχει μοναδιαίο μέτρο. Η ζητούμενη γωνία θ_0 υπολογίζεται ως:

$$|tan(\theta_0)| = \frac{|y_{B\_z}|}{|y_{B\_y}|} = \frac{0.3543}{0.9351} = 0.3788 =>$$

$$=> |\theta_0| = atan(0.3788) \Rightarrow |\theta_0| = 0.3621\, rad$$

Η αλγεβρική της τιμή θα είναι $\theta_0 = -0.3621\, rad$, ώστε να ικανοποιούνται οι κανόνες των δεξιόστροφων πλαισίων.

Γνωρίζοντας κάθε χρονική στιγμή τη μεταφορική ταχύτητα w_cb, μπορούμε να υπολογίζουμε σε κάθε διακριτή στιγμή Ts τις νέες γωνίες θ_{cb} από τη σχέση:

$$\theta_{cb}(i+1) = \theta_{cb}(i) + w_{cb}(i) \cdot T_s$$

Οι γωνίες αυτές χρειάζονται, ώστε να βρούμε στη συνέχεια τον προσανατολισμό R_cb κάθε χρονική στιγμή. Γνωρίζοντας συνεχώς και τη σχετική θέση p_cb, μέσω της κλίσης της sim_ball, μπορούμε να συνθέσουμε τον ομογενή μετασχηματισμό g_cb για κάθε διακριτό Ts. Οι υπόλοιποι μετασχηματισμοί g_oc και g_be είναι σταθεροί και γνωστοί. Έτσι, μπορούμε να βρούμε τον επιθυμητό προσανατολισμό του άκρου g_oe κάθε χρονική στιγμή Ts, ο οποίος θα υπολογίζεται ως:

$$g_{oe\_d} = g_{oc} \cdot g_{cb} \cdot g_{be}$$

Ο πραγματικός προσανατολισμός, όμως, κάθε χρονική στιγμή, θα δίνεται από την εντολή του ευθέως κινηματικού g_oe = robot.fkine(Q), όπου Q ο πίνακας που περιέχει τις θέσεις των αρθρώσεων εκείνη τη στιγμή.

Ο πραγματικός και ο επιθυμητός προσανατολισμός του {e} ως προς το {O} αρχικά θα διαφέρουν και στόχος μας είναι η κατά το δυνατόν ταύτισή τους (με κάποιο μη μηδενικό ανεκτό σφάλμα, προφανώς).

Ταυτόχρονα υπολογίζουμε και τον πραγματικό προσανατολισμό g_be_real κάθε χρονική στιγμή (ώστε να τον εκφράσουμε αργότερα υπό μορφή ισοδύναμου άξονα-γωνίας) που θα δίνεται ως:

$$g_{be\_real} = g_{cb}^{-1} \cdot g_{oc}^{1} \cdot g_{oe}$$

Ο κινηματικός έλεγχος που θα σχεδιάσουμε για το άκρο θα έχει τη μορφή:

$$u = \begin{bmatrix} \dot{p}_d \\ w_d \end{bmatrix} - K \cdot \begin{bmatrix} e_p \\ e_l \end{bmatrix}$$

όπου $e_p = p - p_d$ είναι το σφάλμα θέσης μεταξύ p_{oe} και p_{oe_d}, και e_l το σφάλμα προσανατολισμού μεταξύ R_{oe} και R_{oe_d}.

Εκφράζουμε υπό μορφή Quaternion το σφάλμα προσανατολισμού κάθε χρονική στιγμή:

$$R_e = R_{oe} \cdot R_{oe\_d}^{-1}$$

Υπολογίζουμε το ίχνος Trace αυτού του πίνακα:

$$Tr\{R_e\} = R_e(1,1) + R_e(2,2) + R_e(3,3)$$

Η ισοδύναμη έκφραση άξονας-γωνία είναι:

$$\theta_e = cos^{-1}\left(\frac{Tr\{R_e\} - 1}{2}\right) \ \ και \ \ k_e = \frac{1}{2 \cdot sin(\theta_e)} \cdot \begin{bmatrix} R_e(3,2) - R_e(2,3) \\ R_e(1,3) - R_e(3,1) \\ R_e(2,1) - R_e(1,2) \end{bmatrix}$$

Επιλέγουμε, λοιπόν, το σφάλμα προσανατολισμού e_l να είναι:

$$e_l = \theta_e \cdot k_e$$

Μπορούμε πλέον να υπολογίσουμε αναλυτικά την είσοδο ελέγχου $u \in \mathbb{R}^6$, ορίζοντας τα κέρδη K (K_p για θέση, K_v για προσανατολισμό)

Το ζητούμενο σήμα Q_dot, κάθε χρονική στιγμή Ts, θα δίνεται από τη σχέση:

$$Q_{dot} = J^{-1}(Q) \cdot u$$

όπου $J(Q)$ είναι η Ιακωβιανή του βραχίονα και υπολογίζεται μέσω της εντολής J = robot.jacob0(Q) στο matlab.

Στη συνέχεια, πρέπει να περάσουμε το σήμα Q_dot που μόλις υπολογίσαμε από μία συνάρτηση κορεσμού, ώστε να ικανοποιούνται κάθε χρονική στιγμή τα όρια ταχυτήτων:

$$|Q\_dot\_max| = [\frac{2\pi}{3}, \frac{2\pi}{3}, \pi, \pi, \pi, \pi] \ (rad/sec)$$

Υπολογίζουμε ακόμη τις επιταχύνσεις Q_dot_dot των αρθρώσεων κάθε στιγμή, μέσω της σχέσης:

$$Q\_dot\_dot(i-1) = (Q\_dot(i) - Q\_dot(i-1)) \ / \ Ts$$

Φιλτράρουμε το σήμα Q_dot_dot μέσω της συνάρτησης κορεσμού, ώστε να ικανοποιούνται και τα όρια επιταχύνσεων των αρθρώσεων:

$$|Q\_dot\_dot\_max| = [250\ 250\ 250\ 250\ 250\ 250] \ (rad/sec^2)$$

Επομένως, απαιτείται εκ νέου υπολογισμός των ταχυτήτων των αρθρώσεων, ο οποίος θα δίνεται ως:

$$Q\_dot(i) = Q\_dot(i-1) + Q\_dot\_dot(i-1) * Ts$$

Τέλος, πριν εισέλθουμε στη νέα επανάληψη της for loop, υπολογίζουμε τις επόμενες θέσεις των αρθρώσεων του βραχίονα μέσω της σχέσης:

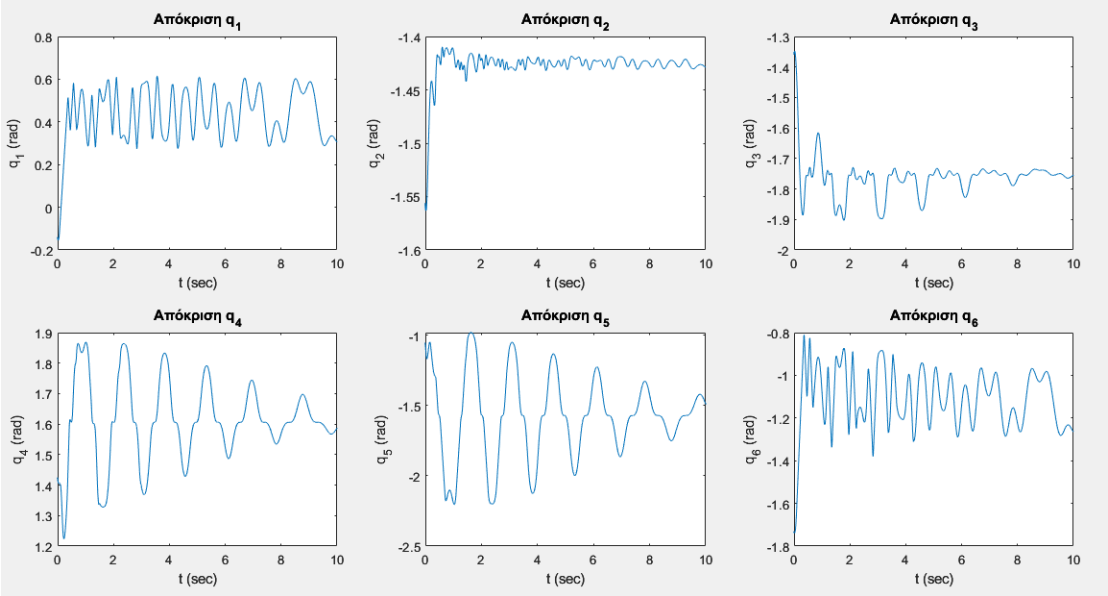
$$Q(i+1) = Q(i) + Q\_dot(i) * Ts$$

Προσομοιώνουμε κινηματικά το ρομποτικό σύστημα στο περιβάλλον matlab με χρήση του robotics toolbox. Στις παραμέτρους που αναφέρθηκαν παραπάνω, δίνουμε τις εξής τιμές: $K_p = 200, \ K_v = 50, \quad T_s = 0.002$

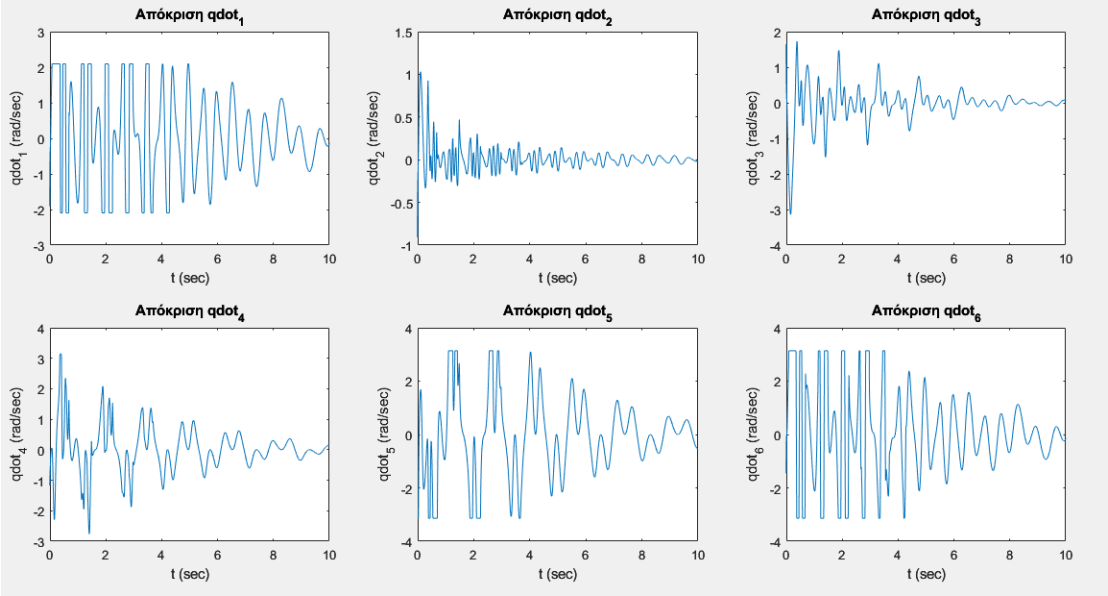
ενώ αρχικοποιούμε το συνολικό χρόνο κίνησης ως $t = 0: T_s:10$. Δηλαδή η προσομοίωση θα ολοκληρώνεται όταν ο χρόνος φτάσει τα 10 sec.

Τα ζητούμενα διαγράμματα είναι η θέση του άκρου του βραχίονα p_be ως προς το πλαίσιο {B}, ο προσανατολισμός R_be σε μορφή ισοδύναμου άξονα/γωνίας καθώς και οι αποκρίσεις θέσης, ταχύτητας και επιτάχυνσης όλων των αρθρώσεων.
Τα αποτελέσματα της προσομοίωσης φαίνονται παρακάτω:

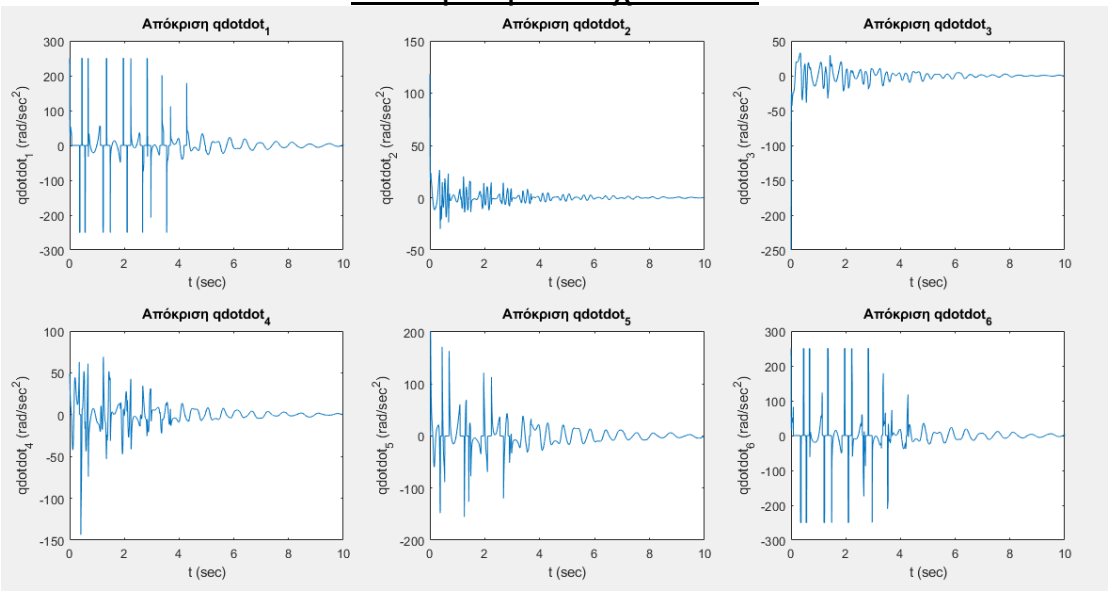
Απόκριση θέσεων



Απόκριση ταχυτήτων



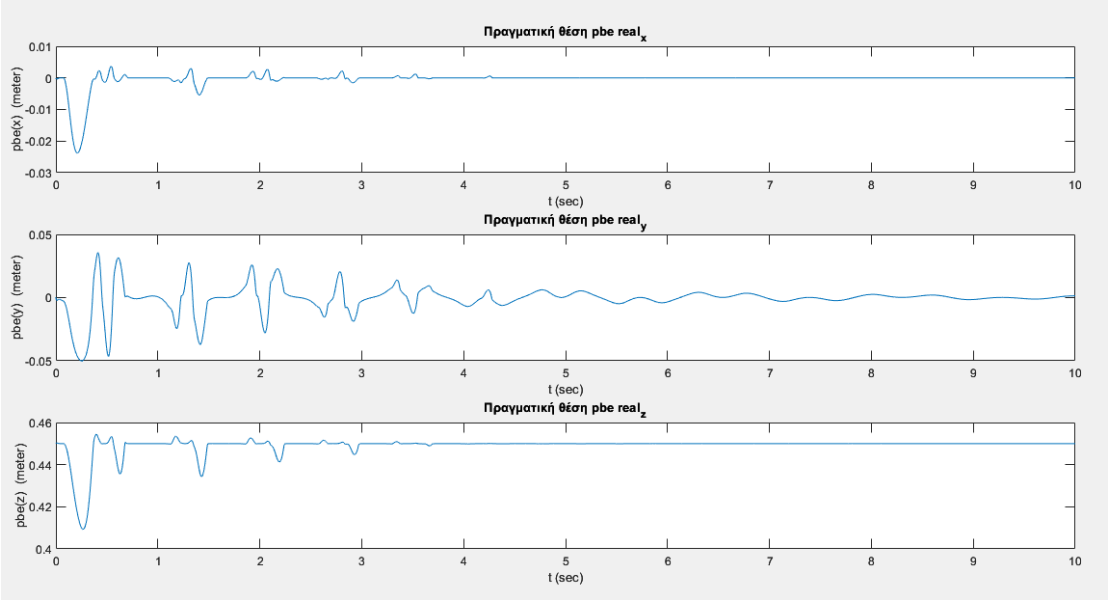
Απόκριση επιταχύνσεων



Από τα παραπάνω διαγράμματα φαίνεται ότι πληρούνται τόσο τα όρια ταχυτήτων όσο και τα όρια επιταχύνσεων για όλες τις αρθρώσεις.

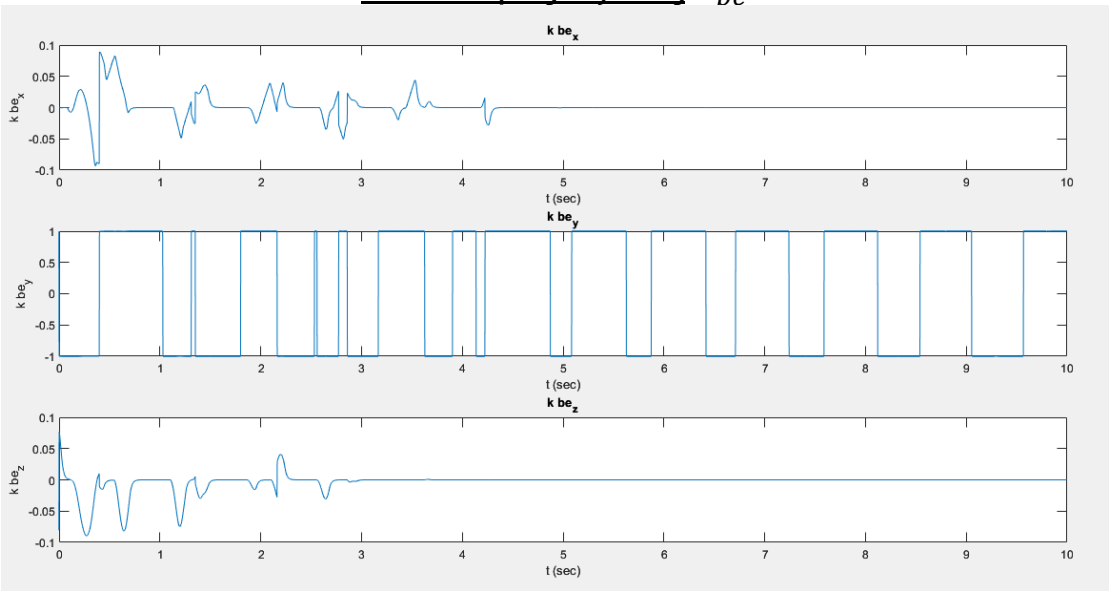
Για τη ζητούμενη σχετική θέση p_be στη διάρκεια του χρόνου, έχουμε το εξής γράφημα:

Σχετική θέση p_be

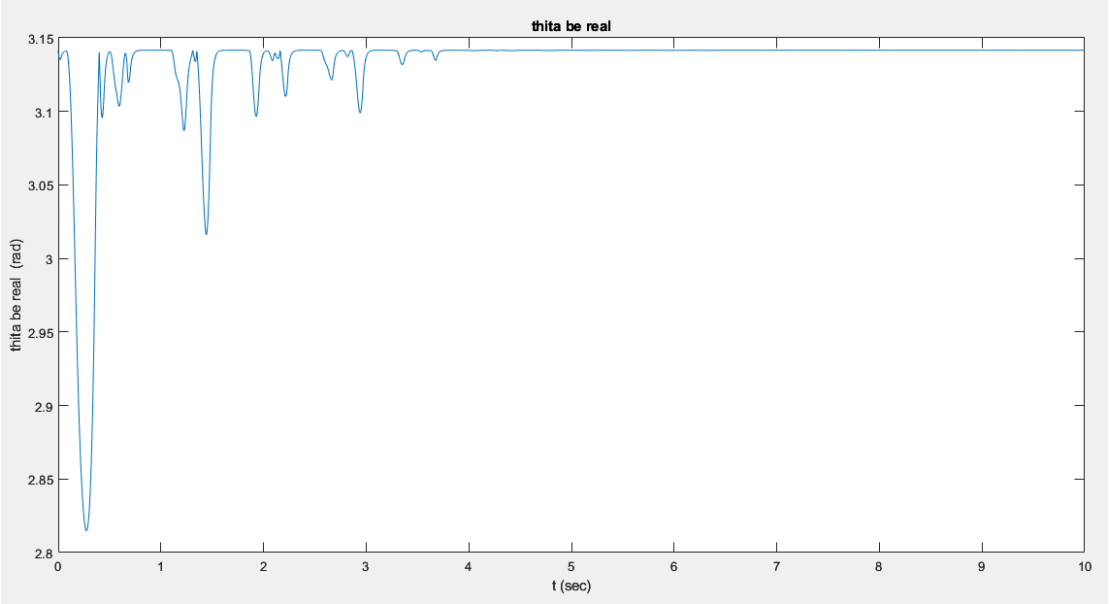


Όσον αφορά τον προσανατολισμό R_be σε μορφή ισοδύναμου άξονα/γωνίας:

Ισοδύναμος άξονας k_{be}



Ισοδύναμη γωνία θ_{be}



Παρατηρούμε ότι με την πάροδο του χρόνου, τόσο η σχετική θέση p_be, όσο και ο σχετικός προσανατολισμός R_be (υπό μορφή ισοδύναμου άξονα/γωνίας) συγκλίνουν στις επιθυμητές τους τιμές. Δηλαδή, όσο περνάει ο χρόνος, τόσο μεγαλώνει και η ζητούμενη ακρίβεια παρακολούθησης της σφαίρας από το άκρο του βραχίονα.

Τμήμα Β

Τοποθετώντας τώρα μία αρπάγη στο άκρο e του βραχίονα, επιθυμούμε να πιάσουμε την κινούμενη σφαίρα με τη ζητούμενη σχετική θέση και προσανατολισμό. Θεωρούμε ότι το πλαίσιο της αρπάγης ταυτίζεται με το πλαίσιο e του άκρου του βραχίονα.

Προκειμένου να κλείσουν τα δάκτυλα της αρπάγης, θα πρέπει να διατηρηθεί αυτός ο σχετικός μετασχηματισμός g_be τουλάχιστον για 1 sec.

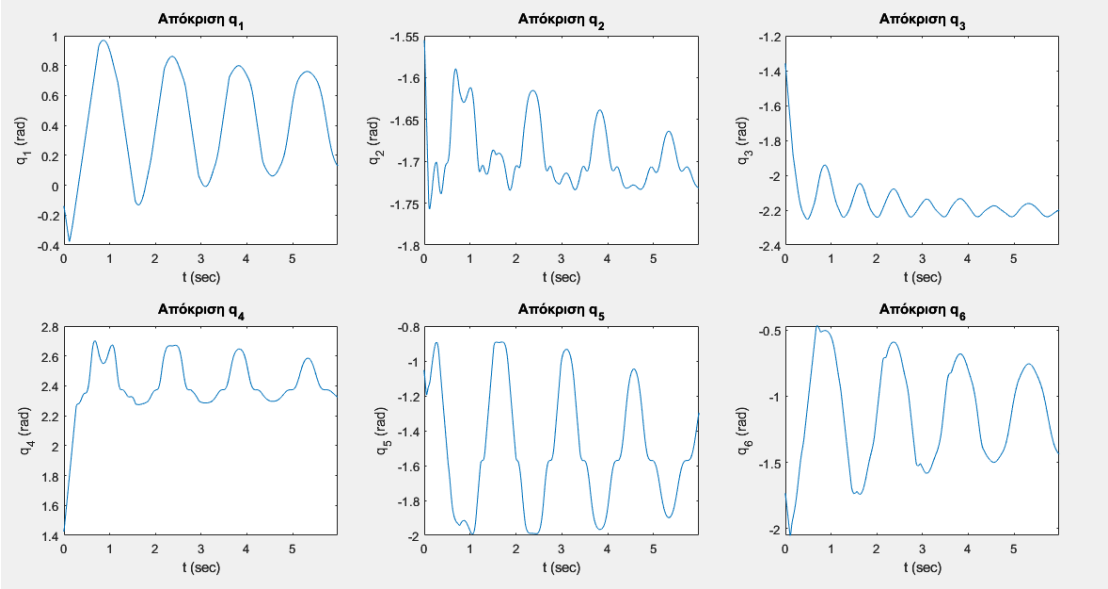
Η φιλοσοφία του κώδικα εδώ είναι ακριβώς η ίδια με αυτήν που ακολουθήθηκε στο Τμήμα Α, με μικρές μόνο τροποποιήσεις. Ζητείται να διατηρηθεί ο σχετικός προσανατολισμός g_be τουλάχιστον για 1 sec. Αυτό σημαίνει ότι για συνεχόμενες 500 επαναλήψεις (500 * Ts = 500 * 0.002 = 1sec), θα πρέπει το σφάλμα θέσης και προσανατολισμού του {e} ως προς το {B} να είναι μικρότερο από μία ανεκτή τιμή (την οποία θα ορίσουμε εμείς). Έτσι, στη θέση της for loop τοποθετούμε τώρα μία while(condition), όπου η συνθήκη είναι αυτή που ορίσαμε μόλις.

Με το που ικανοποιηθεί η συνθήκη, θα είμαστε βέβαιοι ότι η αρπάγη έχει πιάσει την κινούμενη σφαίρα, και ο αλγόριθμος σταματάει.

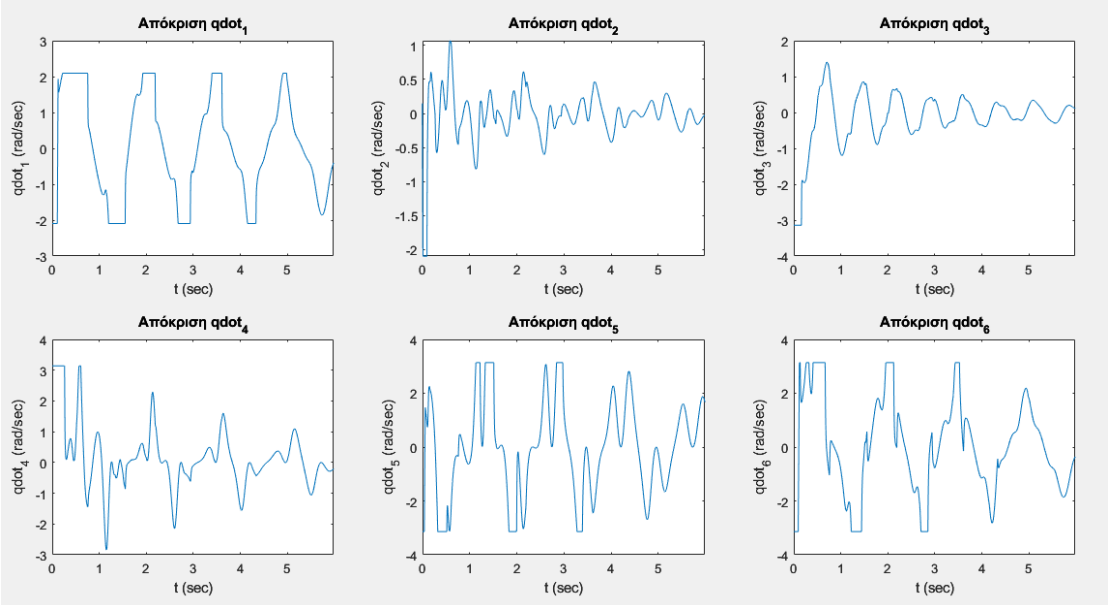
Δίνουμε αυτή τη φορά τις εξής τιμές: $K_p = 200$ και $K_v = 100$, ενώ ορίζουμε το ανεκτό σφάλμα να είναι: *tol_error = 0.002*.

Τα γραφήματα που ζητούνται εδώ είναι ακριβώς τα ίδια με τα ζητούμενα του Τμήματος Α. Παίρνουμε λοιπόν:

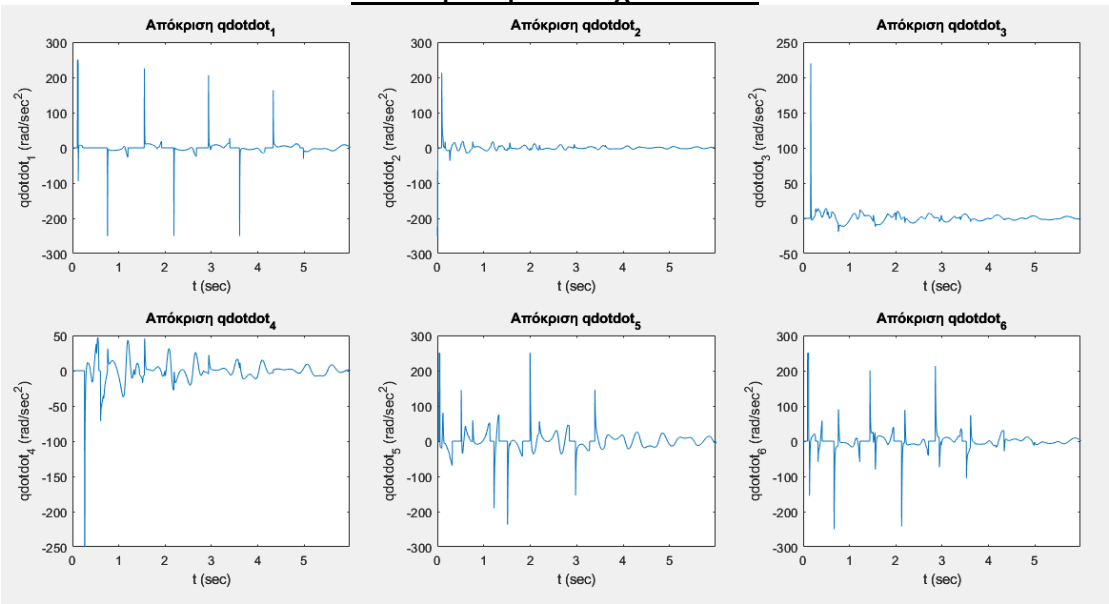
Απόκριση θέσεων



Απόκριση ταχυτήτων



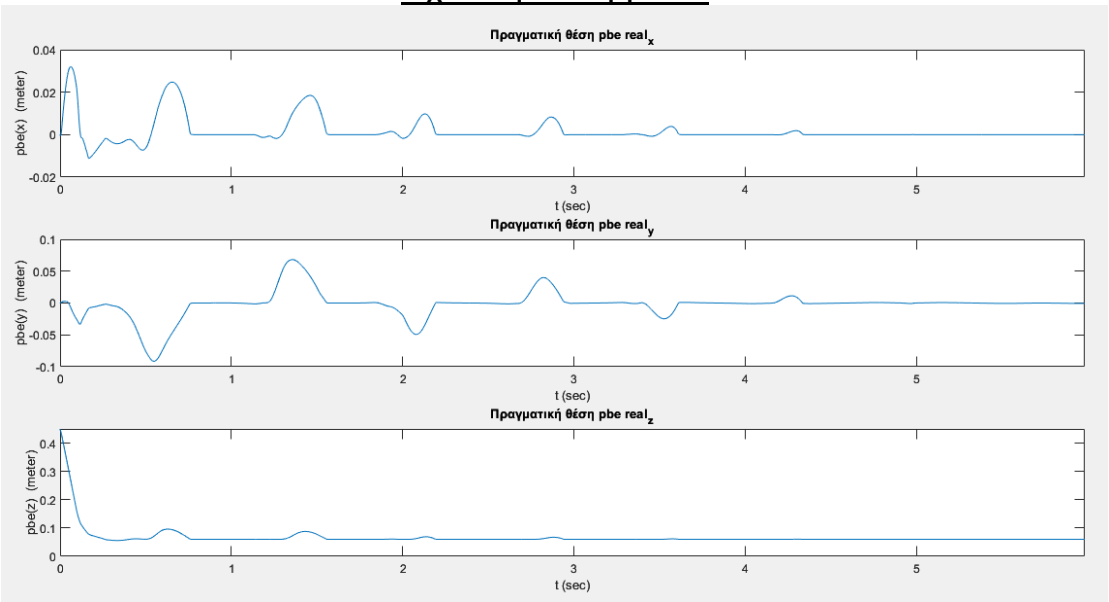
Απόκριση επιταχύνσεων



Προφανώς κι εδώ πληρούνται τόσο τα όρια ταχυτήτων όσο και τα όρια επιταχύνσεων για όλες τις αρθρώσεις.

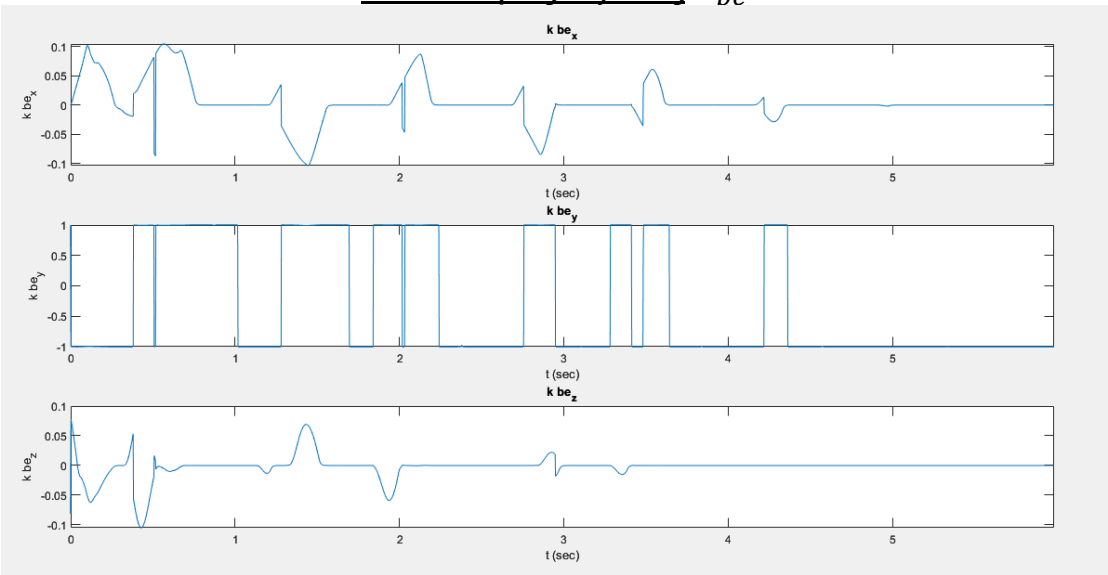
Για τη ζητούμενη σχετική θέση p_be στη διάρκεια του χρόνου, έχουμε:

Σχετική θέση p_be

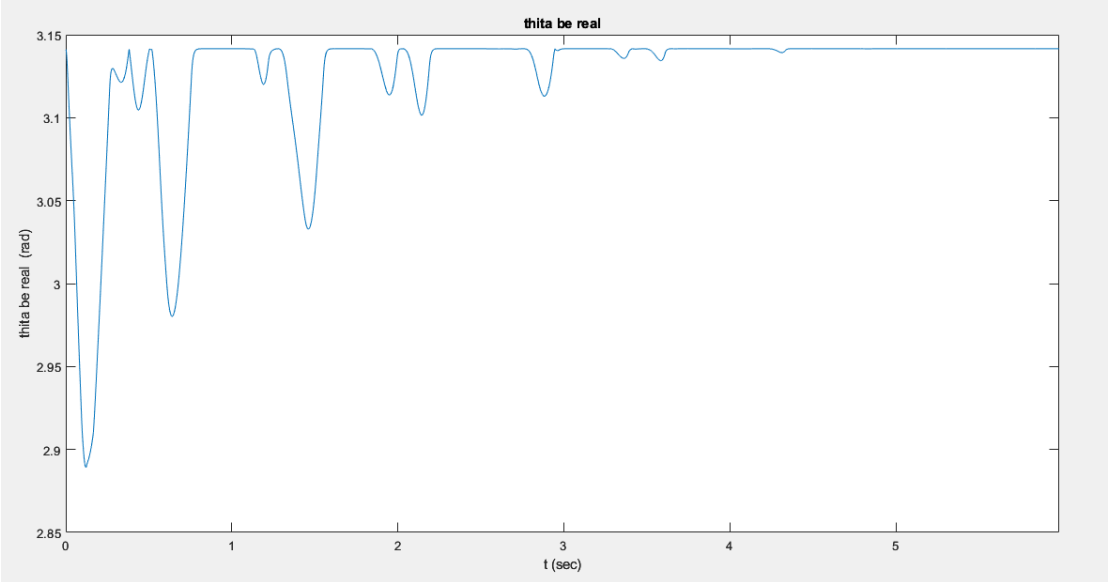


Όσον αφορά τον προσανατολισμό R_be σε μορφή ισοδύναμου άξονα/γωνίας:

Ισοδύναμος άξονας k_{be}



Ισοδύναμη γωνία θ_{be}



Ίδια ακριβώς συμπεράσματα κι εδώ, τόσο η σχετική θέση p_be, όσο και ο σχετικός προσανατολισμός R_be (υπό μορφή ισοδύναμου άξονα/γωνίας) συγκλίνουν στις επιθυμητές τους τιμές με την πάροδο του χρόνου.

Μάλιστα, η αρπάγη καταφέρνει να πιάσει τη σφαίρα τη χρονική

στιγμή t = 5.984 sec. Αυτό σημαίνει ότι από την t = 4.984 sec και μετά, το σφάλμα θέσης και προσανατολισμού θα είναι διαρκώς μικρότερα από την τιμή του *tol_error*.

Η οπτικοποίηση της κίνησης του βραχίονα και της σφαίρας γίνονται στο τέλος των γραφημάτων στο περιβάλλον matlab, όπως ακριβώς ζητείται.