Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Προχωρημένες Τεχνικές Συστημάτων Αυτομάτου Ελέγχου

2η Εργαστηριακή Άσκηση *Matlab – Simulink*

Νικόλαος Λάππας
Α.Μ.: 03121098

1 Εισαγωγή

Η παρούσα εργαστηριακή άσκηση αφορά την μελέτη της συμπεριφοράς ενός *drone* με 4 έλικες, το οποίο έχει την δυνατότητα να κινείται προς τις κατευθύνσεις x, y και z καθώς και να περιστραφεί γύρω από αυτούς κατά γωνία φ, ϑ και ψ αντίστοιχα. Για να το πετύχει αυτό, οι έλικες του drone λειτουργούνε συνδυαστικά και κυρίως σε ζευγέρια (ανά δυο οι αντιδιαμετρικοί έλικες) και παράγουν ωθήσεις που δίνονται από τον τύπο

$$f_i = b\Omega_i^2, i = 1, 2, 3, 4$$

με αποτέλεσμα να δημιουργείται ροπή κατάλληλη να κινήσει (περιστροφικά) το σύστημα γύρω από τον επιθυμητό άξονα. Οι κινήσεις, τόσο μεταφορικές όσο και περιστροφικές μπορούνε να μοντελοποιηθούνε αρκετά ικανοποιητικά από τις ακόλουθες σχέσεις:

- Η μεταβλητή ελέγχου

  $$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$

  αντιπροσωπεύει την συνολιική δύναμη που ασκείται στο drone από τον αέρα λόγω της συνολικής ώθησης από τους έλικες. Όταν αυξηθεί η δύναμη αυτή και γίνει μεγαλύτερη από το βάρος του *drone*, ίσο με (mg), τότε το *drone* θα μεταφερθεί προς τα πάνω κατά μήκος του άξονα z, ενώ αν η συνολική δύναμη μειωθεί, θα έχουμε μεταφορά προς την κατεύθυνση του βάρους.

- Η μεταβλητή ελέγχου (*roll*)

  $$U_2 = b(\Omega_4^2 - \Omega_2^2)$$

  αντιπροσωπεύει την διαφορά των δυνάμεων μεταξύ του έλικα 2 και του έλικα 4 του *drone*. Καθορίζει, δηλαδή, την περιστροφή γύρω από τον άξονα x. Αν οι δυο δυνάμεις δεν είναι ίσες, θα έχουμε ροπή περιστροφής γύρω από τον άξονα που προαναφέραμε.

- Η μεταβλητή ελέγχου (*pitch*)

  $$U_3 = b(\Omega_3^2 - \Omega_1^2)$$

  αντιπροσωπεύει την διαφορά των δυνάμεων μεταξύ του έλικα 1 και του έλικα 3 του *drone*. Καθορίζει, δηλαδή, την περιστροφή γύρω από τον άξονα y. Αν οι δυο δυνάμεις δεν είναι ίσες, θα έχουμε ροπή περιστροφής γύρω από τον άξονα που προαναφέραμε.

- Τέλος, η μεταβλητή (*yaw*)

  $$U_4 = d(\Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2)$$

  καθορίζει την περιστροφή γύρω από τον άξονα z.

Στις παραπάνω σχέσεις να αναφέρουμε ότι το b αποτελεί την παράμετρο για τον συντελεστή άνωσης, ενώ το d την παράμετρο για τον συντελεστή αντίστασης. Να επισημάνουμε, επίσης, ότι υπάρχει μια μέγιστη ταχύτητα Ω_i^2 για τον κάθε έλικα που αντιστοιχεί σε 15.000 στροφές το λεπτό.

2 Μαθηματική Μοντελοποίηση

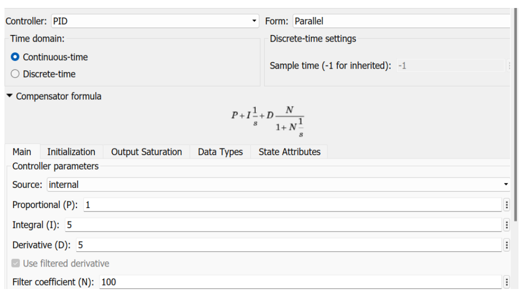
Στο μοντέλο που θα χρησιμοποιήσουμε για την μοντελοποίηση της κίνησης του *drone* στο *Matlab* έχουμε κάνει σημαντικές απλοποιήσεις και παραδοχές προκειμένου να γίνει πιο προσιτός ο έλεγχος. Το μοντέλο στο χώρο κατάστασης που θα χρησιμοποιήσουμε έχει προέλθει από τις γωνίες *Euler* φ, ϑ, ψ και τους αντίστοιχους ρυθμούς μεταβολής τους γύρω από τους άξονες, και από τις συνεταγμένες της θέσης x, y, z και τις αντίστοιχες ταχύτητες. Αυτές οι μεταβλητές κατάστασης μας έδωσαν τις ακόλουθες εξισώσεις στις οποίες βασίστηκε το μοντέλο μας στον χώρο κατάστασης:

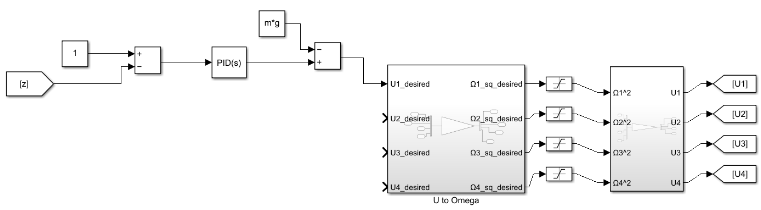
- $\frac{d^2\phi}{dt^2} = \frac{d\theta}{dt}\frac{d\psi}{dt}\left[\frac{I_{yy}-I_{zz}}{I_{xx}}\right] + \frac{l}{I_{xx}}U_2$
- $\frac{d^2\theta}{dt^2} = \frac{d\phi}{dt}\frac{d\psi}{dt}\left[\frac{I_{zz}-I_{xx}}{I_{yy}}\right] + \frac{l}{I_{yy}}U_3$
- $\frac{d^2\psi}{dt^2} = \frac{d\theta}{dt}\frac{d\phi}{dt}\left[\frac{I_{xx}-I_{yy}}{I_{zz}}\right] + \frac{l}{I_{zz}}U_4$
- $\frac{d^2\psi}{dt^2} = \left(c_\phi s_\theta c_\psi + s_\phi s_\psi\right)\frac{1}{m}U_1$
- $\frac{d^2\psi}{dt^2} = \left(-c_\phi s_\theta s_\psi + s_\phi c_\psi\right)\frac{1}{m}U_1$
- $\frac{d^2\psi}{dt^2} = -g + \left(c_\theta c_\phi\right)\frac{1}{m}U_1$

Σε αυτές τις σχέσεις l η απόσταση του κέντρου κάθε έλικα από το κέντρο μάζας, I_{xx}, I_{yy}, I_{zz} είναι οι ροπές αδράνειας γύρω από τους άξονες x, y, z αντίστοιχα και $c_\phi, s_\phi, c_\theta, s_\theta, c_\psi, s_\psi$ είναι τα συνημίτονα και ημίτονα αντίστοιχα των γωνιών $Euler$.

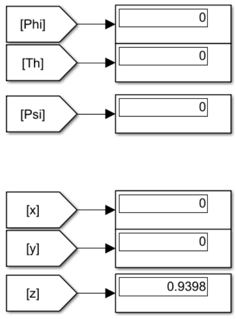
3 *Cascaded PID* Έλεγχος

Ερώτηση **1.** Στο συγκεκριμένο ερώτημα καλούμαστε να μηδενίσουμε τις γωνιακές θέσεις ϕ και θ, άρα πρακτικά να περιορίσουμε το *drone* να εκτελεί μόνο κατακόρυφη μεταφορική κίνηση στον άξονα z. Το ύψος θέλουμε να έχει μια τιμή αναφοράς της επιλογής μας, και υποθέτουμε αρχικές τιμές πολύ κοντά στο μηδέν. Μένει, λοιπόν, η ώθηση U_1 η οποία θα επιδρά συνδυαστικά με το βάρος του *drone* και θα το κατευθύνει να εκτελεί κατακόρυφη περιστροφική και μεταφορική κίνηση. Υποθέτουμε όμως ότι η περιστροφή γύρω από τον άξονα z (*yaw*) δεν μας επηρεάζει στο συγκεκριμένο ερώτημα. Χρησιμοποιούμε το μοντέλο που μας έχει δοθεί στην εκφώνηση και προσθέτουμε έναν *PID* ελεγκτή προκειμένου να δώσουμε μια τιμή αναφοράς για την κατακόρυφη θέση z του *drone*. Ακολουθούν οι παράμετροι για τον ελεγκτή καθώς και τα υπόλοιπα *blocks* που προσθέσαμε για να ελέγξουμε το U_1.

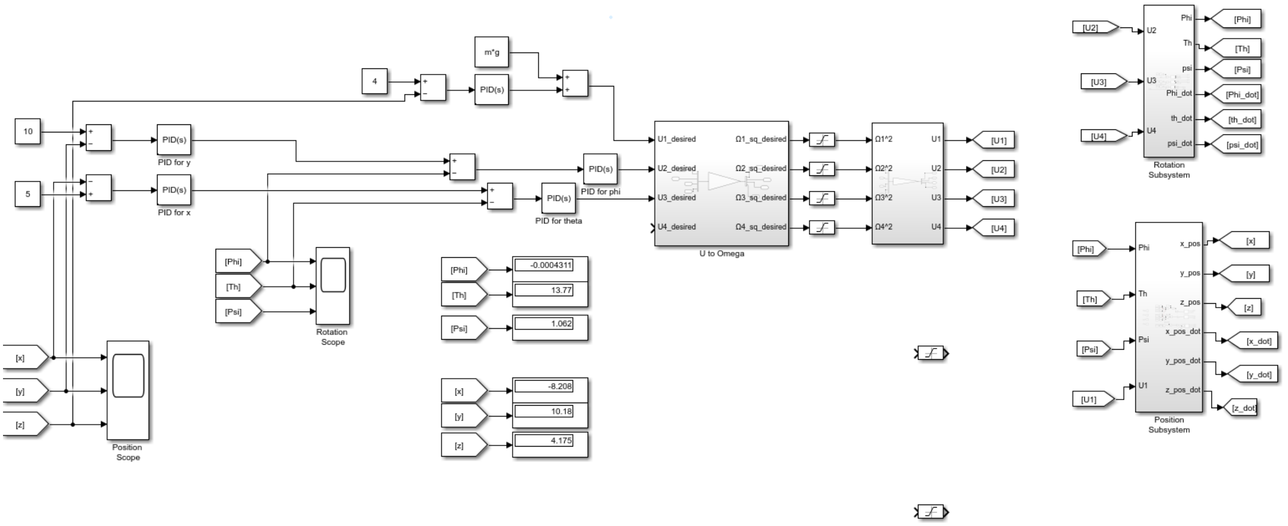




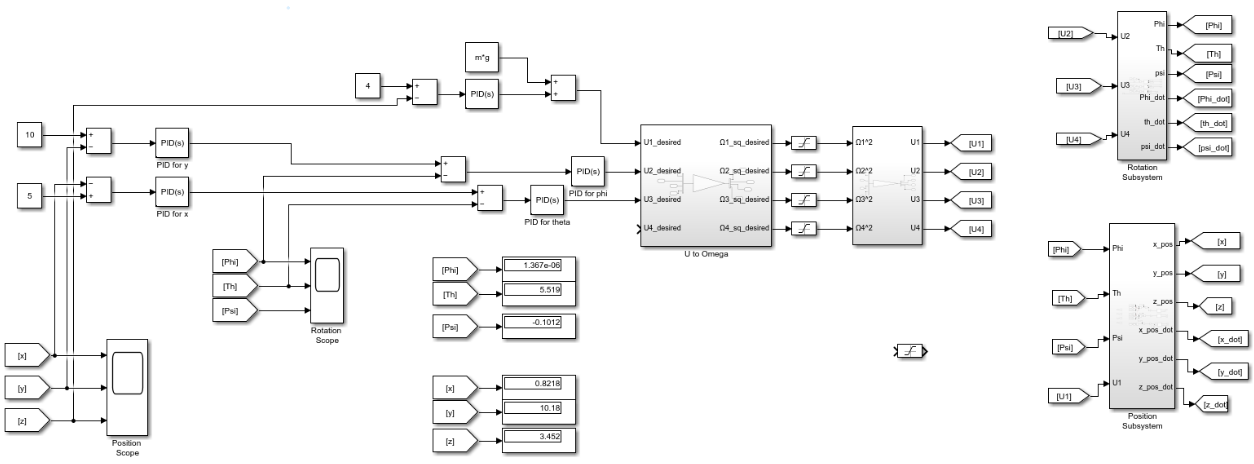
Με αυτές τις τιμές και με την αναφορά στον άξονα z ίση με 1, παίρνουμε την ακόλουθη τιμή μετά την προσομοίωση.



Ερώτηση **2.** Αφού καταφέραμε να σταθεροποιήσουμε σε μια συγκεκριμένη θέση κατά z το *drone* μας, καλούμαστε να υλοποιήσουμε *cascaded PID* ελεγκτές, οι οποίοι να κάνουν πλέον τις θέσεις x, y αλλά και z να έχουν επιθυμητές τιμές. Αυτό είναι κάτι πιο περίπλοκο, διότι ελέγχουμε τέσσερεις μόνο παραμέτρους, τις $\Omega_1, \Omega_2, \Omega_3, \Omega_4$, ενώ έχουμε έξι μεταβλητές που πρέπει να προσδιορίσουμε κατάλληλα, τις $x, y, z, \theta, \phi, \psi$. Για να το καταφέρουμε αυτό χρειάζεται να φτιάξουμε *cascaded PID* ελεγκτές για τις μεταβλητές κατάστασης x, ϕ και y, θ διότι η μια επηρεάζει την άλλη. Συγκεκριμένα, το outer loop θα περιέχει την διαφορά των x, y αντίστοιχα από τις επιθυμητές τους τιμές, διαφορά την οποία θα περάσουμε στην συνέχεια από τον φωλιασμένο PID ελεκγτή για να ελέγξουμε την επιθυμητή τιμή των θ και ϕ αντίστοιχα. Τα inner loops των ελεγκτών προσέξαμε να έχουν μεγαλύτερες τιμές στις παραμέτρους του ελεγκτή, καθώς η δυναμική τους πρέπει να ανταποκρίνεται και να προσαρμόζεται γρηγορότερα από αυτή των outer loops. Μια προσπάθεια να υλοποιηθεί αυτό το μοντέλο ακολουθεί μέσω Simulink:



Σε αυτή την δοκιμή βλέπουμε ότι οι θέσεις y και z έχουν προσεγγίσει σε ικανοποιητικό βαθμό τις επιθυμητές θέσεις που έχουμε εμείς ορίσει, ίσες με 10 και 4 αντίστοιχα. Ωστόσο, η παράμετρος x ίση με -8.208 αποκλίνει κατά πολύ από την επιθυμητή τιμή ίση με 5. Για να βελτιώσουμε αυτή την απόκλιση ξεκινήσαμε να πειράζουμε τις παραμέτρους του inner PID (PID for theta), και μόλις είδαμε ότι προσεγγίζουν μια καλή τιμή σε σχέση με την αναφορά, τότε περνάμε στον έλεγχο του outer PID (PID for x). Μια προσέγγιση μαζί με τις νέες τιμές για τα δυο αυτά PID είναι η ακόλουθη:



Proportional (P): 0.85

Integral (I): 0.1

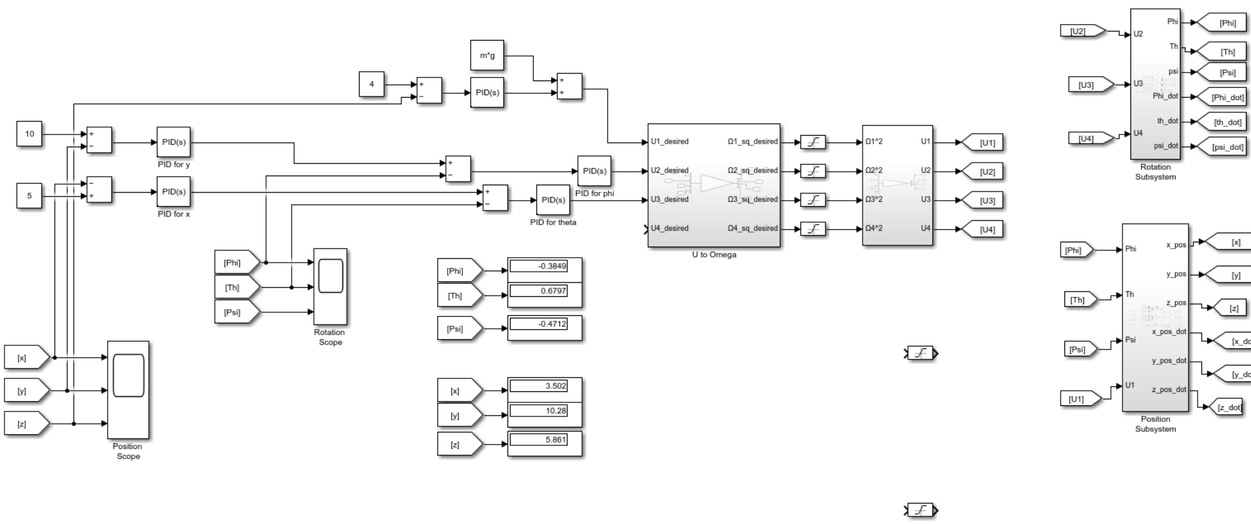
Derivative (D): 0.6

Proportional (P): 1.1

Integral (I): .01

Derivative (D): .3

Εδώ βλέπουμε ότι το x αποκτά καλύτερη προσέγγιση, αλλά ακόμα απέχει από την επιθυμητή τιμή που του έχουμε ορίσει. Πειράζοντας έτσι σιγά σιγά τις τιμές των PID, βλέπαμε ότι επηρεάζονταν και οι άλλες τιμές και όχι μόνο αυτή που πρακτικά ελέγχαμε. Αυτό έκανε τον έλεγχο ελαφρώς περίπλοκο, και μας έβαλε σε μια διαδικασία παρακολούθησης του τι συμβαίνει με κάθε μεταβολή προκειμένου να αλλάζουμε στοχευμένα τις μεταβλητές. Μικρή αλλαγή σε ορισμένες από αυτές τις παραμέτρους μπορούσε να αλλάξει εντελώς την κατάσταση του συστήματος. Τελικά, μετά από αρκετές αλλαγές κυρίως στις παραμέτρους που επηρέαζαν το x και το z, καταλήξαμε σε ένα αρκετά ικανοποιητικό μοντέλο. Αυτό φαίνεται αμέσως μετά μαζί με τις παραμέτρους για κάθε ελεγκτή.



Best Approach to stabilize the drone to a predefined position

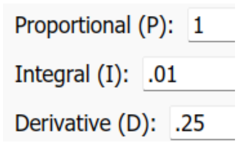


Figure 1: *
(a) X-axis

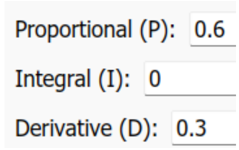


Figure 2: *
(b) Y-axis

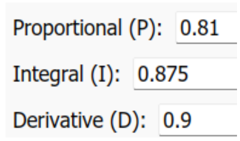


Figure 3: *
(c) Z-axis

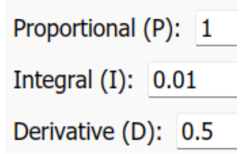


Figure 4: *
(d) ϕ

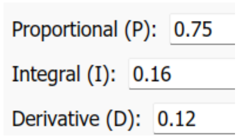


Figure 5: *
(e) θ

Figure 6: PID parameters for each axis and angle

Ερώτηση **3.** Εδώ καλούμαστε να υλοποιήσουμε *cascaded PID* ελεγκτές, οι οποίοι να κάνουν τις θέσεις x, y, z καθώς και τον προσανατολισμό ψ να έχουν επιθυμητές τιμές. Θα πρέπει να προσέξουμε να κατασκευάσουμε ένα βοηθητικό υποσύστημα στο Simulink, το οποίο θα μας μεταφέρει από το ορθοκανονικό σύστημα βάσης στο αντίστοιχο σύστημα που θα έχει 'δημιουργήσει' η όποια περιστροφική κίνηση κατά x ή κατά y.

4 Έλεγχος στον Χώρο Κατάστασης

Ερώτηση **4.** Ζητείται ο υπολογισμός των σημείων ισορροπίας του συστήματος. Δίνονται οι ακόλουθες εξισώσεις μέσω της δεύτερης βιβλιογραφίας της εκφώνησης:

$$f(X,U) = \begin{bmatrix} x_2 \\ p_1 x_4 x_6 - p_2 x_4 \Omega_d + p_3 U_2 \\ x_4 \\ p_4 x_2 x_6 + p_5 x_2 \Omega_d + p_6 U_3 \\ x_6 \\ p_7 x_4 x_2 + p_8 U_4 \\ x_8 \\ \left(c_\phi s_\theta c_\psi + s_\phi s_\psi\right) \frac{1}{m} U_1 \\ x_{10} \\ \left(c_\phi s_\theta s_\psi + s_\phi c_\psi\right) \frac{1}{m} U_1 \\ x_{12} \\ -g + \left(c_\phi c_\theta\right) \frac{1}{m} U_1 \end{bmatrix}$$

Μαθηματικά αυτό γίνεται μηδενίζοντας τις χρονικές παραγώγους της μεταφορικής όσο και της περιστροφικής κίνησης του *drone*. Αυτό πρακτικά έχει σαν αποτέλεσμα την ακινητοποίηση του *drone* σε μια σταθερή θέση χωρίς δυνατότητα μεταφοράς ή περιστροφής σε άλλη κατεύθυνση, παρά μόνο επίτρεψη περιστροφής κατά z. Μηδενισμός της γωνίας ϕ συνεπάγεται ακινητοποίηση γύρω από τον άξονα x, μηδενισμός της γωνίας θ συνεπάγεται ακινητοποίηση γύρω από τον άξονα y, ενώ το ψ μπορεί να πάρει οποιαδήποτε σταθερή τιμή επιτρέποντας στροφή γύρω από τον z. Μετά από πράξεις καταλήγουμε ότι τα U_2, U_3, U_4 θα είναι μηδενικά προκειμένου να μην υπάρχει καμία μεταφορική ή περιστροφική κίνηση και ότι το U_1 θα ισούται με το βάρος του *drone*, δηλαδή ίσο με mg, προκειμένου να αλληλοαναιρούνται και να μην υπάρχει κατακόρηφη μετατόπιση. Τα αποτελέσματα αυτά επαληθεύονται, επίσης, αν ανατρέξουμε στις μεταβλητές κατάστασης του μοντέλου που κατασκευάσαμε στο *Matlab* και προέκυψαν από τις παραπάνω εξισώσεις που καταγράψαμε. Συγκεκριμένα, το U_1 επηρεάζει το $\frac{dz}{dt}$ και κατεπέκταση το z, το U_4 επηρεάζει το $\frac{d\psi}{dt}$ και κατεπέκταση το ψ και τα U_2, U_3 έχουν αλυσίδα επιρροής μήκους τέσσερα, και συγκεκριμένα επηρεάζουν αντίστοιχα τα $\left\{\frac{d\phi}{dt}, \phi, \frac{dx}{dt}, x\right\}$ και $\left\{\frac{d\theta}{dt}, \theta, \frac{dy}{dt}, y\right\}$.

Ερώτηση **5.** Ζητείται η γραμμικοποίηση γύρω από ένα σημείο ισορροπίας με το ψ να είναι μηδέν κατά προτίμηση για ευκολία. Η γραμμικοποίηση είναι σημαντική για να μελετήσουμε το σύστημά μας, διότι αυτό περιέχει και μη γραμμικούς όρους που εξαρτώνται από τις γωνίες Euler καθώς και τις ταχύτητες (και τις δυνάμεις) που ασκούνται από τις έλικες. Για την γραμμικοποίηση, λοιπόν, χρειάζεται να υπολογίσουμε τον Ιακωβιανό πίνακα, δηλαδή τους πίνακες $A = \frac{df}{dx}$ και $B = \frac{df}{du}$, σε ένα σημείο ισορροπίας της επιλογής μας. Συγκεκριμένα, γραμμικοποιήσαμε γύρω από το σημείο ισορροπίας $x_{eq} = [0;0;0;0;0;0;0;0;0;0;0;0]$, το οποίο αντιπροσωπεύει την κατάσταση 'hover', και προκειμένου να διατηρήσει την αιώρηση το drone, θα πρέπει η δύναμη που ασκείται στον άξονα των z να αντισταθμίζει το βάρος. Άρα, ορίσαμε $U_1 = m \times g$. Ακολουθούν οι πίνακες A και B:

Editor - part3_lab2.m | Variables - A_lin

A_lin

12x12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0
8	0	0	0	9.8100	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0
10	0	9.8100	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0

Editor - part3_lab2.m

A_lin | B_lin

12x4 double

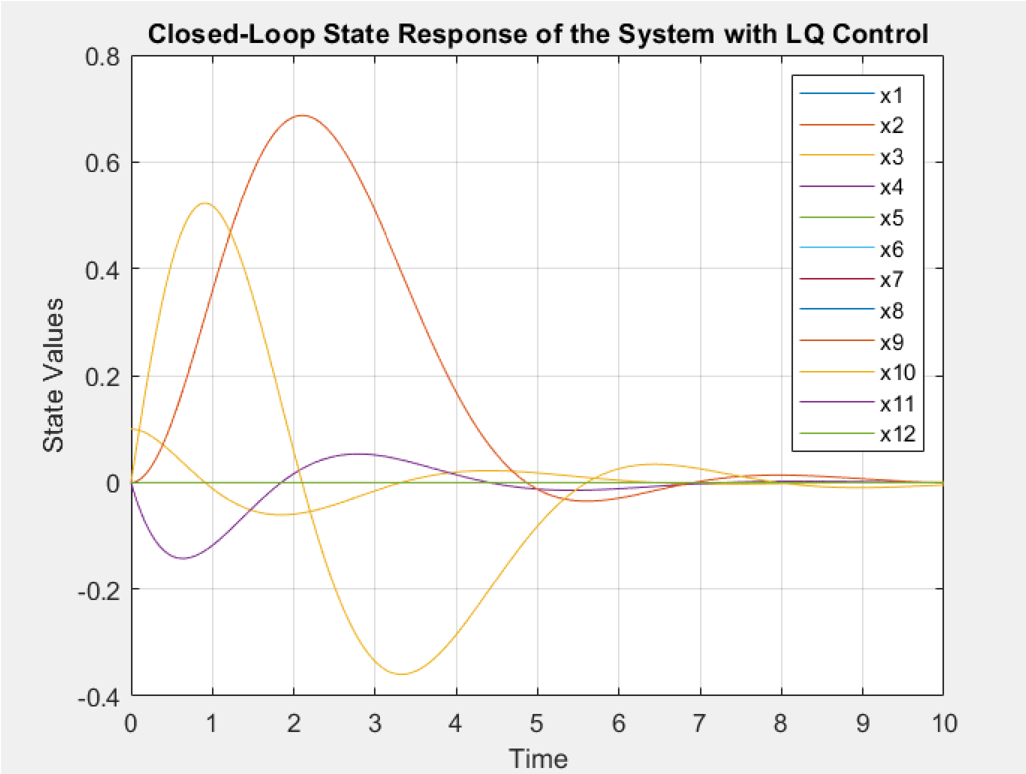
	1	2	3	4
1	0	0	0	0
2	0	19.1449	0	0
3	0	0	0	0
4	0	0	19.1449	0
5	0	0	0	0
6	0	0	0	35.2858
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	1.2500	0	0	0

Ερώτηση **6.** Ζητείται να υπολογίσουμε τον πίνακα μεταφοράς από τις εισόδους στις εξόδου, θεωρώντας ως εξόδους τις μεταβλητές κατάστασης x, y, z, ψ. Για τον σκοπό αυτό ορίζουμε τις μήτρες C, με βάση τις παραπάνω μεταβλητές κατάστασης και D, και ύστερα με την εντολή sys δημιουργούμε το γραμμικοποιημένο σύστημα. Παίρνουμε τις ακόλουθες τέσσερις συναρτήσεις μεταφοράς:

```
Transfer Function:

tf_system =

  From input 1 to output...
   1:  0

   2:  0

       1.25
   3:  ----
       s^2

   4:  0

  From input 2 to output...
   1:  0

       187.8
   2:  -----
        s^3

   3:  0

   4:  0

  From input 3 to output...
       187.8
   1:  -----
        s^3

   2:  0

   3:  0

   4:  0

  From input 4 to output...
   1:  0

   2:  0

   3:  0

       35.29
   4:  -----
        s^2
```

Ερώτηση **7.** Τώρα καλούμαστε να σχεδιάσουμε έναν ελεγκτή στον χώρο κατάστασης με LQ έλεγχο. Πρόκειται δηλαδή για πρόβλημα βέλτιστου ελέγχου για το οποίο πρέπει να υπολογίσουμε τους πίνακες Q, R και να λύσουμε την αλγεβρική εξίσωση Ricatti. Δεν τοποθετούμε πόλους για να πετύχουμε τις επιθυμητές θέσεις των ιδιοτιμών, αλλά ρυθμίζουμε κατάλληλα το Q και το R προκειμένου να πάρουμε απευθείας το βέλτιστο κέρδος K_{LQ}. Το Q επενεργεί πάνω στο x, και είναι συνήθως διαγώνιος τετραγωνικός πίνακας διάστασης όση και το x με θετικά στοιχεία πάνω στην διαγώνιο. Καθορίζει τις ποινές στις αποκλίσεις των μεταβλητών κατάστασης $X = [x_1, \ldots, x_{12}]$, όσο πιο μεγάλη δηλαδή η τιμή σε μια θέση, (i,i), της διαγωνίου στον πίνακα Q, τόσο μεγαλύτερη έμφαση δίνει ο ελεγκτής στην μείωση του σφάλματος από το μηδέν του αντίστοιχου στοιχείου x_i. Παρομοίως, το R επενεργεί πάνω στο διάνυσμα εισόδου, επιβάλλει δηλαδή ποινές στις εισόδους ελέγχου $U = [U_1, \ldots, U_4]$ και είναι συνήθως διαγώνιος και αυτός. Μεγαλύτερες τιμές στον R κάνουν τον ελεγκτή να κάνει πιο ομαλές ενέργειες ελέγχου.

Παρακάτω ακολουθούν ορισμένα plot του αποτελέσματος που έδωσε αυτή η προσέγγιση:



Αυτό έγινε με Q και R για την icare:

$$Q = \text{diag}\left([0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01]\right)$$

$$R = \text{diag}\left([100, 100, 100, 100]\right)$$

Και αρχική κατάσταση:

$$x_0 = \left([0.1, 0, 0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0]\right)$$

Ερώτηση **8.** Θέλουμε να ελέγξουμε αν το σύστημά μας μπορεί να ακολουθήσει την διαδρομή που μας δίνεται. Για να προσεγγίσουμε το πρόβλημά μας, επειδή το σύστημα μας είναι παρατηρήσιμο, χρησιμοποιήσαμε ελεγκτή ανάδρασης της μορφής:

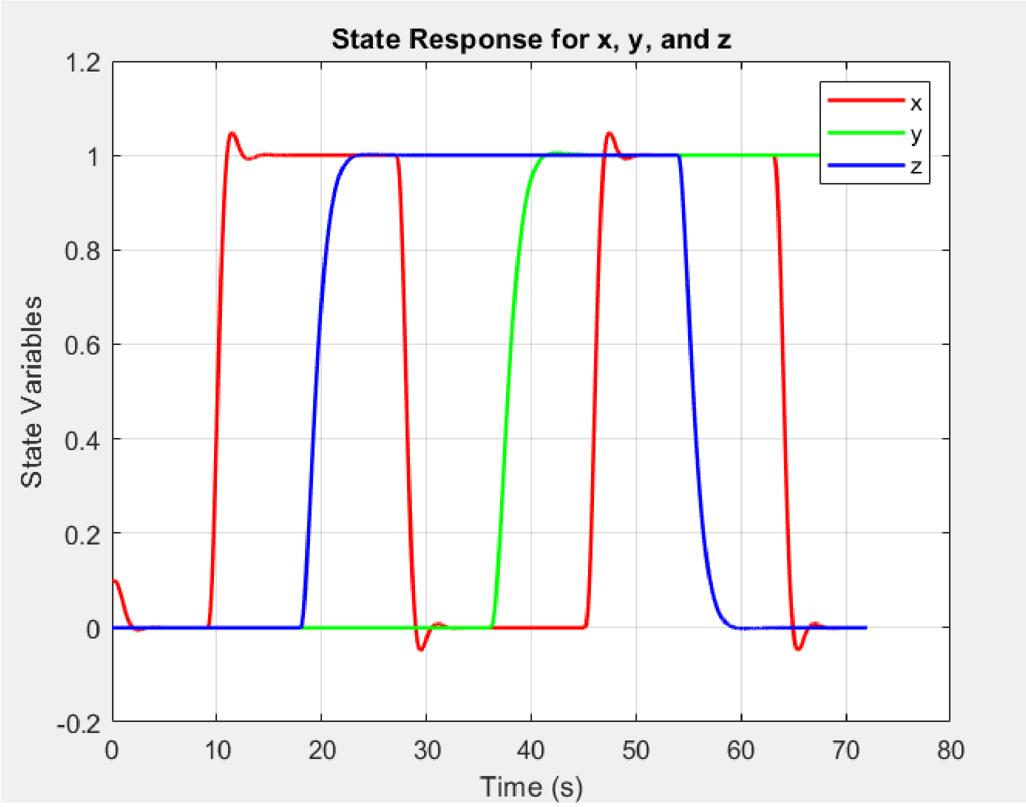
$$u = -K_{LQR}\left(x - x_{ref} + u_{ref}\right)$$

στην οποία αν αντικαταστήσουμε το γραμμικοποιημένο μας σύστημα, ήτοι

$$\frac{dx}{dt} = A_{lin}x + B_{lin}u$$

θα προκύψει το u_{ref}. Εκεί έχουμε χρησιμοποιήσει τον ψευδοαντίστροφο του πίνακα B_{lin}, διότι δεν έχει αντίστροφο. Με αυτόν τον τρόπο υπολογίζουμε διαδοχικά το u_{ref} για κάθε επιθυμητή νέα θέση x_{ref} του drone.

Η προσομοίωση επιλέξαμε να διαρκέσει 9 δευτερόλεπτα η καθεμία, και τα αποτελέσματα που πήραμε φαίνονται παρακάτω.



Τα σημεία που επιθυμούμε να μεταφέρουμε το drone, είναι μια τριάδα (x, y, z), τριάδα η οποία αντιστοιχεί στις θέσεις 7, 9 και 11 του πίνακα waypoints. Αυτό γιατί έτσι έχει οριστεί το διάνυσμα $x0$ να περιέχει τα στοιχεία x_0, y_0, z_0 σε αυτές τις συγκεκριμένες θέσεις. Ακολουθεί μια πιο κομψή εκτύπωση των σημείων και της διαδρομής που ακολούθησε το drone.

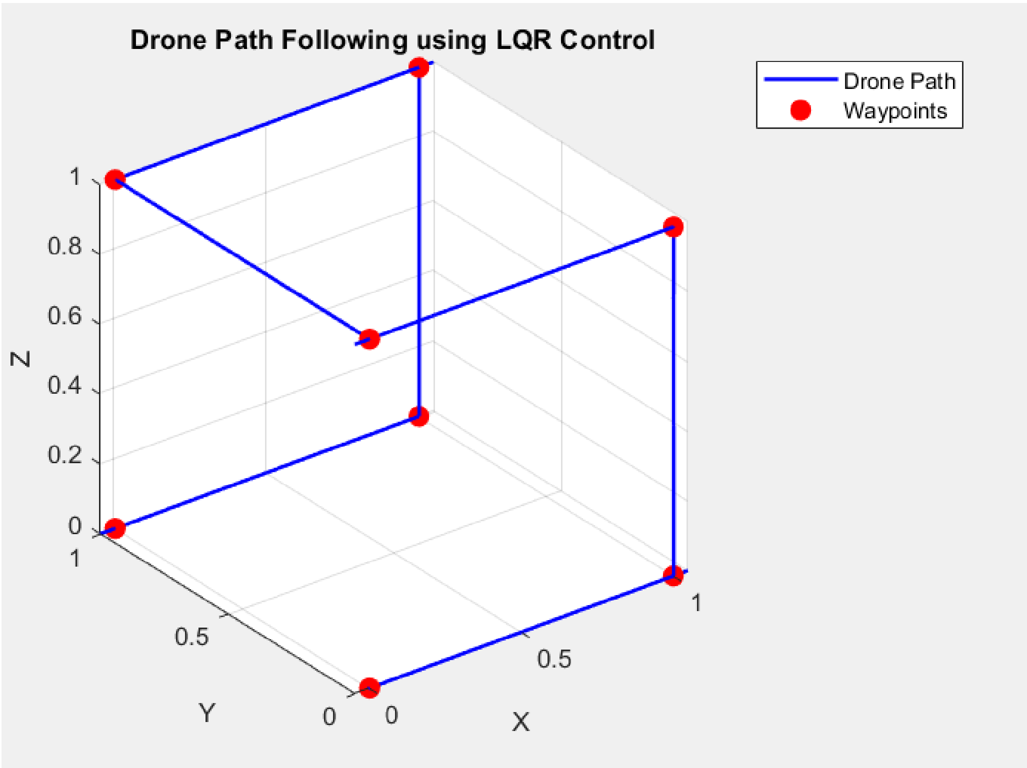


Figure 7: Desired trajectory of the drone

Ακολουθεί το script με τον κώδικα που χρησιμοποίησα έχοντας μεταφράσει τα σχόλια στα αγγλικά σε σχέση με το script που επισυνάπτω στο αρχείο .zip:

```
m=0.8;
g=9.81;
l=0.3;
Ixx=15.67e-3;
Iyy=15.67e-3;
Izz=28.34e-3;
b=192.32e-7;
d=4e-7;
Jr=0;%6e-5;

p1=(Iyy-Izz)/Ixx;
p2=Jr/Ixx;
p3=l/Ixx;
p4=(Izz-Ixx)/Iyy;
p5=Jr/Iyy;
p6=l/Iyy;
p7=(Ixx-Iyy)/Izz;
p8=1/Izz;

Omega_sq_2_U = [b   b   b   b;
                0  -b   0   b;
               -b   0   b   0;
                d  -d   d  -d];

U_to_Omega_sq = inv(Omega_sq_2_U);

phi_0=0;
phi_dot_0=0;
th_0=0;
th_dot_0=0;
psi_0=0;
psi_dot_0=0;

x_0=0.1;
x_dot_0=0;
y_0=0;
y_dot_0=0;
z_0=0;
z_dot_0=0;

U_max=15000/60*2*pi;
U_Sq_max=U_max^2;
U1_steady_State=g*m;
```

Listing 1: MATLAB Script: Given Parameters

```
% Question 5
syms x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12
syms U1 U2 U3 U4
syms phi theta psi

x1_dot = x2;
x2_dot = p1*x4*x6 + p3*U2;
x3_dot = x4;
x4_dot = p4*x2*x6 + p6*U3;
x5_dot = x6;
x6_dot = p7*x2*x4 + p8*U4;
x7_dot = x8;
x8_dot = (cos(x1)*sin(x3)*cos(x5) + sin(x1)*sin(x5))*(1/m)*U1;
x9_dot = x10;
x10_dot = (-cos(x1)*sin(x3)*sin(x5) + sin(x1)*cos(x5))*(1/m)*U1;
```

```
x11_dot = x12;
x12_dot = -g + (cos(x3)*cos(x1))*(1/m)*U1;

% Define State Vector X
X = [x1; x2; x3; x4; x5; x6; x7; x8; x9; x10; x11; x12];
% Define Input Vector
U = [U1; U2; U3; U4];
% State Equations of the Drone
f = [x1_dot; x2_dot; x3_dot; x4_dot; x5_dot; x6_dot; x7_dot; x8_dot; x9_dot;
    x10_dot; x11_dot; x12_dot];

% Calculate the linearized model
A = jacobian(f, X);
B = jacobian(f, U);

% Define Equilibrium Point
x_eq = zeros(12, 1); % We chose x_eq = 0
u_eq = [m*g; 0; 0; 0]; % U1=mg is required to achieve hover

% Substitute the equilibrium points and convert to double
A_lin = double(subs(A, [X; U], [x_eq; u_eq]));
disp('Linearized matrix A:');
disp(A_lin);
B_lin = double(subs(B, [X; U], [x_eq; u_eq]));
disp('Linearized matrix B:');
disp(B_lin);

% Question 6
% Define outputs (x7, x9, x11, x5) as specified in the problem statement
C = [0 0 0 0 0 0 1 0 0 0 0 0;  % x7
     0 0 0 0 0 0 0 0 1 0 0 0;  % x9
     0 0 0 0 0 0 0 0 0 0 1 0;  % x11
     0 0 0 0 1 0 0 0 0 0 0 0]; % x5
D = zeros(4, 4);

% Define State Space System
sys_state = ss(A_lin, B_lin, C, D);

% Transfer Function
tf_system = tf(sys_state);
disp('Transfer Function:');
tf_system

% Question 7
% Calculate the controllability matrix and rank
Co = ctrb(A_lin, B_lin);
rank_Co = rank(Co);  % Should be less than full rank (12) if uncontrollable
   states exist

% Define Q and R matrices for the LQR
%Q = diag([1 1 1 1 1 1 1 1 1 1 1 1]);
%R = diag([1 1 1 1]);
Q = diag([0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,
   0.01]);
R = diag([100, 100, 100, 100]);
%Q = diag([10,1,10,1,10,1,50,1,50,1,50,1]);
%R = diag([0.1, 0.1, 0.1, 0.1]);

% Solve the algebraic Riccati equation for matrix P
[X, K_LQ, ~] = icare(A_lin, B_lin, Q, R);

% Create the Closed-Loop System
A_cl = A_lin - (B_lin * K_LQ);
```

```
eig_values = eig(A_cl);

C_cl = eye(size(A_lin));
D_cl = zeros(size(B_lin));

sys_cl = ss(A_cl, B_lin, C_cl, D_cl);

% Simulate the Closed-Loop System
t = 0:0.005:10;
x0 = [0.1; 0; 0.1; 0; 0; 0; 0; 0; 0; 0; 0; 0];  % Initial conditions

% Simulate the closed-loop system with initial conditions
[y, t, x] = initial(sys_cl, x0, t);

figure;
plot(t, x);
title('Closed-Loop State Response of the System with LQ Control');
xlabel('Time');
ylabel('State Values');
legend('x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10', 'x11', '
    x12');
grid on;

%%
% Question 8
% Parameters for simulation
sim_duration = 9; % Total simulation duration in seconds
dt = 0.01; % Time step for each iteration

% Define time vector for each iteration and total simulation time
t_iter = 0:dt:sim_duration;
num_steps = length(t_iter);

positions = zeros(num_steps * size(waypoints, 1), 3); % Store positions
index = 1;

% Reference states for each iteration
waypoints = [
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 0 0 0 0 0;
    0 0 0 0 0 0 1 0 0 0 1 0;
    0 0 0 0 0 0 0 0 0 0 1 0;
    0 0 0 0 0 0 0 0 1 0 1 0;
    0 0 0 0 0 0 1 0 1 0 1 0;
    0 0 0 0 0 0 1 0 1 0 0 0;
    0 0 0 0 0 0 0 0 1 0 0 0;
];

% Moderate values for Q_LQR and R_LQR to improve numerical stability
Q_LQR = diag([100, 1, 1, 1000, 1, 10, 1000, 100, 1, 1, 1, 1]);
R_LQR = diag([1, 10, 1000, 1]);

% Solve the Riccati equation for LQR gain matrix
[X, K_LQR, ~] = icare(A_lin, B_lin, Q_LQR, R_LQR);

% Initial state
x0 = [phi_0; phi_dot_0; th_0; th_dot_0; psi_0; psi_dot_0; x_0; x_dot_0; y_0;
    y_dot_0; z_0; z_dot_0];

% Prepare arrays for storing simulation results
x_all = [];
t_all = [];
```

```
% Loop through each reference trajectory in waypoints
for j = 1:size(waypoints, 1)
    x_ref = waypoints(j, :)';

    % Compute steady-state input for reference state
    u_ref = -pinv(B_lin) * (A_lin * x_ref);

    % Initialize state matrix for the current iteration
    x = zeros(num_steps, length(x0));
    x(1, :) = x0';

    % Time vector for the current iteration
    t_current = t_iter + (j - 1) * sim_duration;

    % Simulate system with discrete time steps
    for i = 1:num_steps - 1
        % Calculate control input with reference tracking
        u = -K_LQR * (x(i, :)' - x_ref) + u_ref;

        % Update state using Euler integration
        x_dot = A_lin * x(i, :)' + B_lin * u;
        x(i + 1, :) = x(i, :) + x_dot' * dt;

        % Store the position for plotting
        positions(index, :) = x(i, [7, 9, 11])';
        index = index + 1;
    end

    % Append results of the current iteration
    x_all = [x_all; x];
    t_all = [t_all, t_current];

    % Use final state of the current iteration as the initial state for the next
    x0 = x(end, :)';
end

% Plot results
figure;
plot(t_all, x_all(:, 7), 'r', 'LineWidth', 1.5); % x-state
hold on;
plot(t_all, x_all(:, 9), 'g', 'LineWidth', 1.5); % y-state
plot(t_all, x_all(:, 11), 'b', 'LineWidth', 1.5); % z-state
xlabel('Time (s)');
ylabel('State Variables');
title('State Response for x, y, and z');
legend('x', 'y', 'z');
grid on;
hold off

% Remove unused positions
positions = positions(1:index-1, :);

% Plotting the path
figure;
plot3(positions(:,1), positions(:,2), positions(:,3), 'b', 'LineWidth', 1.5);
hold on;
plot3(waypoints(:,7), waypoints(:,9), waypoints(:,11), 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r');
grid on;
xlabel('X'); ylabel('Y'); zlabel('Z');
title('Drone Path Following using LQR Control');
```

```
legend('Drone Path', 'Waypoints');
hold off
```

Listing 2: MATLAB Script

Πηγή εργασίας: https://journals.sagepub.com/doi/10.1177/0142331215608427