Έλεγχος Drone

Μάθημα: Προχωρημένες Τεχνικές Συστημάτων Αυτομάτου Ελέγχου

Στυλιανός Μπούλιαρης - Μαλατέστας
ΑΜ: 03119214
Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

11 Νοεμβρίου 2024

Contents

1 Εισαγωγή

Στην άσκηση αυτή θα ασχοληθούμε με τον έλεγχο ενός Drone (quadrotor). Συγκεκριμένα θα δοκιμάσουμε διαφορετικούς τρόπους ελέγχου, όπως PID, Cascaded PID και έλεγχο στο χώρο κατάστασης χρησιμοποιώντας LQR. Στο τέλος θα δοκιμάσουμε να δώσουμε μια πορεία στο Drone με τους ελεγκτές που έχουμε σχεδιάσει, και θα δούμε πως ανταποκρίνεται το σύστημα μας. Το Drone μας φαίνεται στο παρακάτω σχήμα:

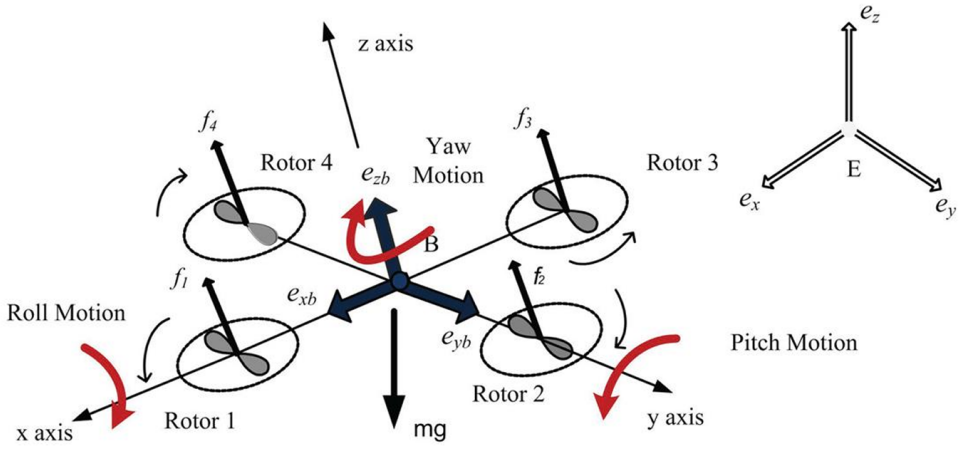

Figure 1: Quadrotor Drone System

Το drone μας έχει 6 βαθμούς ελευθερίας, 3 στροφές roll, pitch, yaw (φ, θ, ψ) και 3 μεταφορές (x, y, z). Προφανώς η ταχύτητα περιστροφής των κινητήρων (άρα και το lift force που δημιουργεί ο κάθε κινητήρας), επηρεάζουν τις μεταβλητές κατάστασης του drone που μόλις περιγράψαμε.

2 Μοντελοποίηση Συστήματος

2.1 Περιγραφή Περιστροφικής Κίνησης

Το drone έχει 4 έλικες. Οι δύο γυρνούν σύμφωνα με τη φορά των δεικτών του ρολογιού (οι έλικες από τον άξονα που περνάει ο x' άξονας) και οι άλλοι δύο αντίστροφα. Έτσι σε λειτουργία hovering, που μένει σταθερό σε ένα σημείο, η συνολική στροφορμή των ελίκων είναι 0. Η ώθηση που δίνει ο κάθε έλικας ($i = 1, 2, 3, 4$) είναι ανάλογη του τετραγώνου της γωνιακής ταχύτητάς του και δίνεται από τη σχέση:

$$f_i = b\Omega_i^2 \tag{1}$$

Για να στύψει το drone γύρω από τον άξονα των x (κίνηση roll, αλλαγή της γωνίας φ) αρχικά ελαττώνεται η ταχύτητα έλικα 2 και αυξάνει του 4, κάτι που δημιουργεί μια ροπή γύρω από τον άξονα των x, η οποία στη συνέχεια δημιουργεί μια γωνιακή ταχύτητα (προφανώς για να σταματήσει αυτή η γωνιακή κίνηση θα χρειαστεί μια αντίθετη ροπή). Αντίστοιχα η στροφή γύρω από τον y άξονα επιτυγχάνεται με τη διαφορά των ταχυτήτων των ελίκων 1, 3. Η περιστροφή γύρω από τον z άξονα γίνεται μέσω της αύξησης της ταχύτητας δύο αντιδιαμετρικών ελίκων και την μείωση στο άλλο ζευγάρι (η ροπή επιτυγχάνεται μέσω της αντίστασης drag στην κίνηση των ελίκων).

2.2 Περιγραφή Μεταφορικής Κίνησης

Προκειμένου να κινηθεί το drone στον άξονα z, θα πρέπει η συνολική δύναμη στον άξονα αυτόν να είναι μεγαλύτερη του βάρους. Η δύναμη περιγράφεται από τη σχέση:

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

Για να υπάρξει κίνηση στους άξονες x και y, πρώτα το drone πρέπει να στραφεί γύρω από τους άξονες y (pitch) και x (roll) αντίστοιχα, προκειμένου να εμφανιστεί μια συνιστώσα της συνολικής δύναμης U_1 στον άξονα που θέλουμε να υπάρξει κίνηση. Επομένως οι παρακάτω σχέσεις εκφράζουν:

$$U_2 = b(\Omega_4^2 - \Omega_2^2), \quad \text{Το } U_2 \text{ ελέγχει την κίνηση στον άξονα x} \tag{3}$$

$$U_3 = b(\Omega_3^2 - \Omega_1^2), \quad \text{Το } U_3 \text{ ελέγχει την κίνηση στον άξονα y} \tag{4}$$

$$U_4 = d(\Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2), \quad \text{Το } U_4 \text{ ελέγχει την στροφή γύρω από z} \tag{5}$$

Σημειώστε ότι τα U_1, U_2, U_3 εξαρτώνται από το συντελεστή άνωσης b ενώ το U_4 από το συντελεστή αντίστασης d.

2.3 Μαθηματική Μοντελοποίηση στο Χώρο Κατάστασης

Οι μεταβλητές κατάστασης που χρειαζόμαστε για να περιγράψουμε το σύστημα δεν είναι άλλες παρά τις γωνίες Euler φ, θ, ψ γύρω από τους άξονες, καθώς και οι μεταφορές x, y, z στους άξονες αυτούς. Οι εξισώσεις κατάστασης για τις στροφικές κινήσεις είναι οι εξής:

$$\ddot{\phi} = \dot{\theta}\dot{\psi}[\frac{I_{yy} - I_{zz}}{I_{xx}}] + \frac{1}{I_{xx}}U_2 \tag{6}$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}[\frac{I_{yy} - I_{xx}}{I_{yy}}] + \frac{1}{I_{yy}}U_3 \tag{7}$$

$$\ddot{\psi} = \dot{\theta}\dot{\phi}[\frac{I_{xx} - I_{yy}}{I_{zz}}] + \frac{1}{I_{zz}}U_4 \tag{8}$$

Η κίνηση στους άξονες x, y, z δίνεται από τις εξισώσεις:

$$\ddot{x} = (c_\phi s_\theta c_\psi + s_\phi s_\psi)\frac{1}{m}U_1 \tag{9}$$

$$\ddot{y} = (-c_\phi s_\theta s_\psi + s_\phi c_\psi)\frac{1}{m}U_1 \tag{10}$$

$$\ddot{z} = -g + (c_\phi c_\theta)\frac{1}{m}U_1 \tag{11}$$

Για την συνέχεια της άσκησης, παίρνουμε τις τιμές των μεταβλητών που βρίσκονται στις παραπάνω εξισώσεις σύμφωνα με τον πίνακα (1). Θα θεωρήσουμε επιπλέον πως κάθε έλικα μπορεί να περιστραφεί με μέγιστη ταχύτητα $\Omega_{max} = 15000RPM$

Table 1: Physical parameters of the quadrotor system.

Parameter	**Symbol**	**Value**
Total weight of the vehicle	m	0.800 kg
Gravitational acceleration	g	9.81 $\mathrm{m\,s^{-2}}$
Arm length of the vehicle	l	0.3 m
Moment of inertia along x-axis	I_{xx}	$15.67 \times 10^{-3}\,\mathrm{kg\,m^2}$
Moment of inertia along y-axis	I_{yy}	$15.67 \times 10^{-3}\,\mathrm{kg\,m^2}$
Moment of inertia along z-axis	I_{zz}	$28.34 \times 10^{-3}\,\mathrm{kg\,m^2}$
Thrust factor	b	$192.32 \times 10^{-7}\,\mathrm{N\,s^2}$
Drag factor	d	$4.003 \times 10^{-7}\,\mathrm{N\,m\,s^2}$

3 Cascaded PID Έλεγχος

3.1 Εξάλειψη γωνιών ϕ, θ και διατήρηση του drone σε σταθερό ύψος

Στόχος μας είναι να σταθεροποιήσουμε τις γωνίες ϕ, θ στο 0 όσο διατηρούμε το ύψος του drone σε συγκεκριμένη επιθυμητή τιμή. Επιλέξαμε σαν αρχικές συνθήκες:

Table 2: Initial Conditions

Variable Name	Symbol	Value
Initial ϕ	ϕ_0	$0.02rad$
Initial θ	θ_0	$0.02rad$
Initial z	z_0	$0m$

Κατασκευάζουμε λοιπόν έναν PID ελεγκτή για κάθε μία από τις εισόδους U_1, U_2, U_3, καθώς αυτές είναι υπεύθυνες για τις κινήσεις που θέλουμε να ελέγξουμε. Μετά από δοκιμές, καταλήγουμε πως τα ιδανικά κέρδη για τους ελεγκτές είναι:

$$[K_p = 1, K_i = 0.5, K_d = 4], \quad \text{Για το } U_1 \tag{12}$$

$$[K_p = 2, K_i = 0, K_d = 0.35], \quad \text{Για το } U_2, U_3 \tag{13}$$

Παρακάτω βλέπουμε την απόκριση του drone με τη χρήση των παραπάνω ελεγκτών, όταν θέτουμε σαν επιθυμητή τιμή για το $z = 1m$

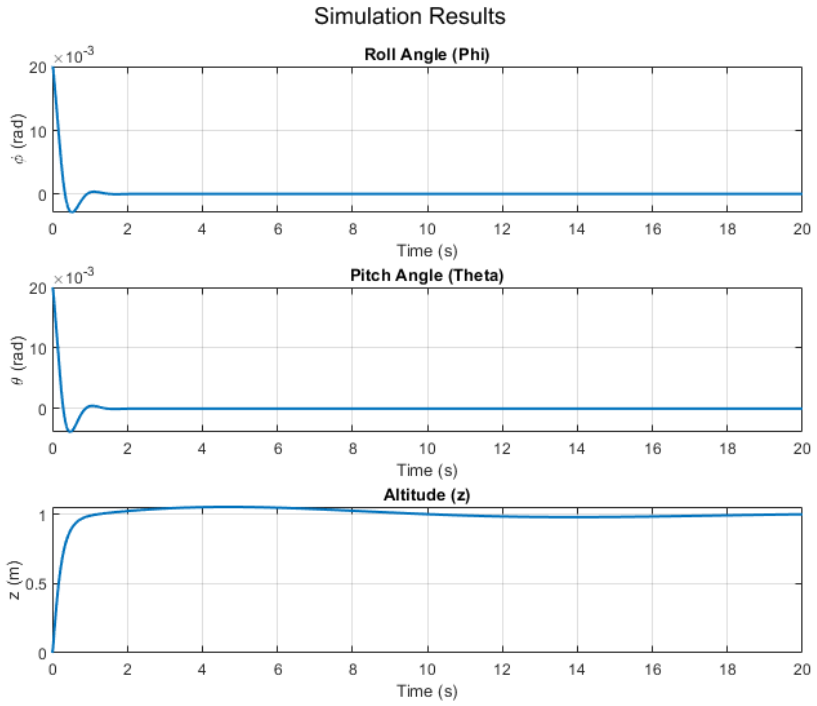

Figure 2: Response with gains specified in (12), (13), (14) for $z_d = 1$

Βλέπουμε λοιπόν πως το σύστημα μας αποκρίνεται αρκετά γρήγορα, και σταθεροποιείται στη μόνιμη κατάσταση όπως θέλουμε.

3.2 Έλεγχος x, y, z σε επιθυμητές τιμές, χρησιμοποιώντας Cascaded PID

Στόχος τώρα είναι το drone μας να φτάνει σε ένα επιθυμητό σημείο, με συντεταγμένες x, y, z στον καρτεσιανό χώρο. Για να το κάνουμε αυτό, θα σχεδιάσουμε 2 Cascaded PID ελεγκτές για τη θέση x και y και θα χρησιμοποιήσουμε τον ελεγκτή που σχεδιάσαμε προηγουμένως για τη σταθεροποίηση του drone σε συγκεκριμένο z. Οι συντεταγμένες του σημείου που θέλουμε να φτάσουμε είναι:

Table 3: Desired Coordinates

Variable Name	**Symbol**	**Value**
Desired x	x_d	$1m$
Desired y	y_d	$4m$
Desired z	z_d	$4m$

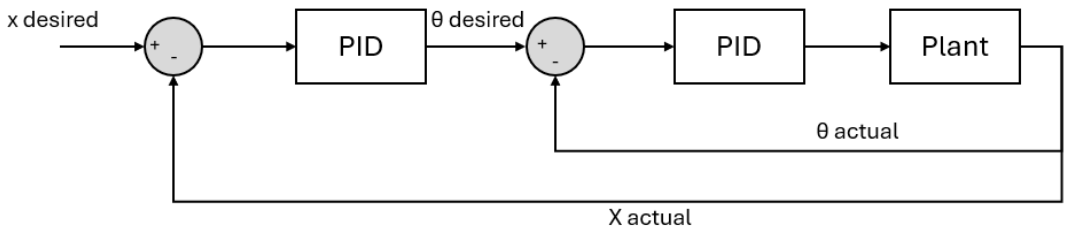

Figure 3: Block diagram of Cascaded PID for θ

Κρίθηκε σκόπιμο να περιμένουμε να σταθεροποιηθεί το drone στο ύψος στο οποίο θέλουμε πρώτα, και μετά να δώσουμε εντολή να μεταβεί στα επιθυμητά x και y. Μετά από δοκιμές τα gains που χρησιμοποιήθηκαν στους Cascaded PID ελεγκτές είναι:

$$[K_p = 1, K_i = 0, K_d = 0.7], \quad \text{Για τον υπολογισμό desired γωνίας} \tag{14}$$

$$[K_p = 2.95, K_i = 0.5, K_d = 2.6], \quad \text{Για τον έλεγχο των κινητήρων} \tag{15}$$

Επιπλέον είναι σημαντικό να αναφέρουμε πως ο απευθείας έλεγχος των γωνιών ϕ, θ χρησιμοποιώντας τις global συντεταγμένες x και y δεν λειτουργεί, καθώς ακόμη δεν κάνουμε έλεγχο στη γωνία ψ, επομένως, όταν στρέφεται το drone, οι άξονες του παύουν να συμπίπτουν με το αδρανειακό σύστημα global σύστημα αναφοράς.

Ο μετασχηματισμός στο σύστημα αναφοράς του drone γίνεται ως εξής:

$$x_{drone} = xcos(\psi) + ysin(\psi) \tag{16}$$

$$y_{drone} = ycos(\psi) - xsin(\psi) \tag{17}$$

Παρακάτω βλέπουμε την απόκριση του drone με τη χρήση του συστήματος ελέγχου που περιγράψαμε. Σημαντικό είναι να αναφέρουμε ότι έχουμε περιορίσει τις εξόδους των PID που υπολογίζουν τις επιθυμητές γωνίες ανάλογα με την απόκλιση της θέσης στο διάστημα $\phi, \theta \in [-15, 15]$, με στόχο την αποφυγή μεγάλων target γωνιών, που πιθανώς να έστελνε σε αστάθεια το σύστημα.

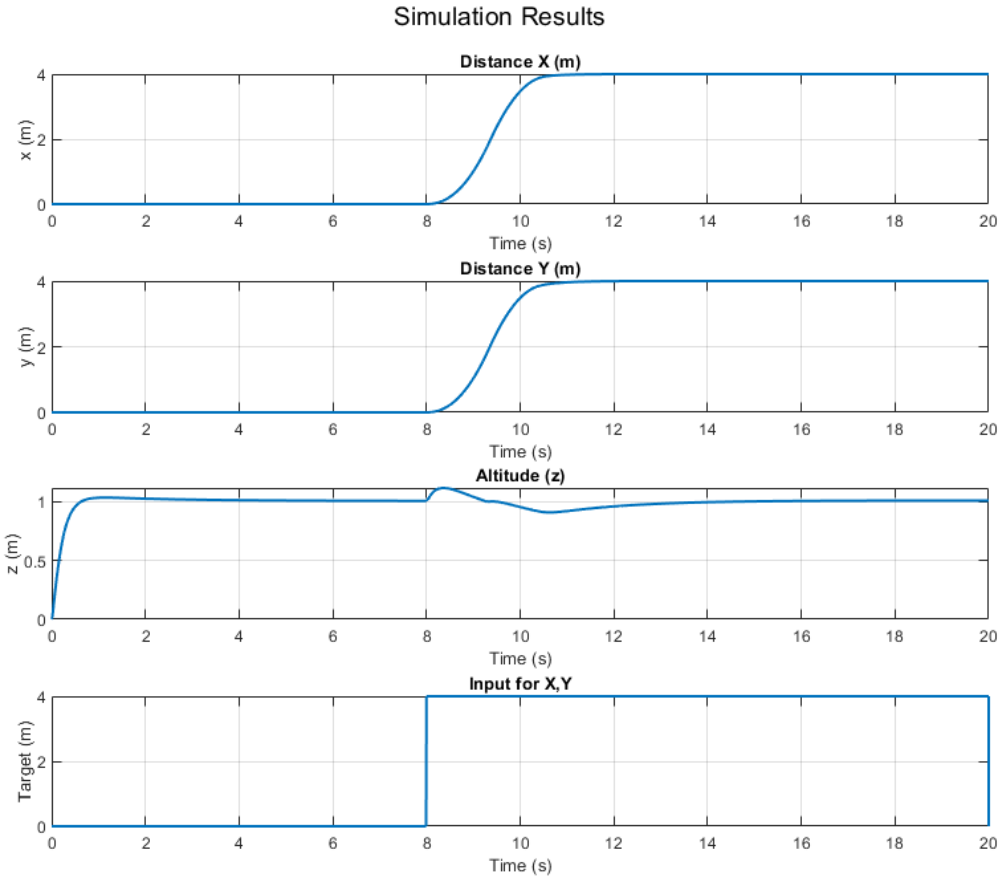

Figure 4: Drone position & Controller input

Παρατηρούμε πως όταν το drone αρχίζει να κινείται στους άξονες x και y το ύψος κάνει overshoot που δεν είναι εύκολο να γίνει compensated γρήγορα, καθώς δε μπορούμε να ασκήσουμε δύναμη προς τα κάτω, πάρα μόνο, να μειώσουμε την άνωση, αφήνοντας τη βαρύτητα να επαναφέρει το overshoot. Ωστώσο παρατηρήσαμε πως το συγκεκριμένο overshoot κατά την αρχική επιτάχυνση καθώς και το undershoot στην επιβράδυνση συνδέονται άμεσα με τα όρια των γωνιών που επιτρέπουμε στο drone. Έτσι τα όρια γωνιών που συζητήσαμε παραπάνω, αποτελούν ένα συμβιβασμό μεταξύ ταχύτητας απόκρισης και σφάλματος στο θέση z

Στη συνέχεια βλέπουμε την απόκριση γωνιών του drone. Στην αρχή ο στόχος είναι να σταθεροποιηθεί σε ύψος $z = 1$, που δεν επηρεάζει τις γωνίες. Στα 8 δευτερόλεπτα που το drone έχει πλέον σταθεροποιηθεί ξεκινάει να ακολουθεί τους στόχους για τις συντεταγμένες x και y. Περιμένουμε λοιπόν να παρατηρήσουμε μεταβολή των γωνιών ϕ, θ.

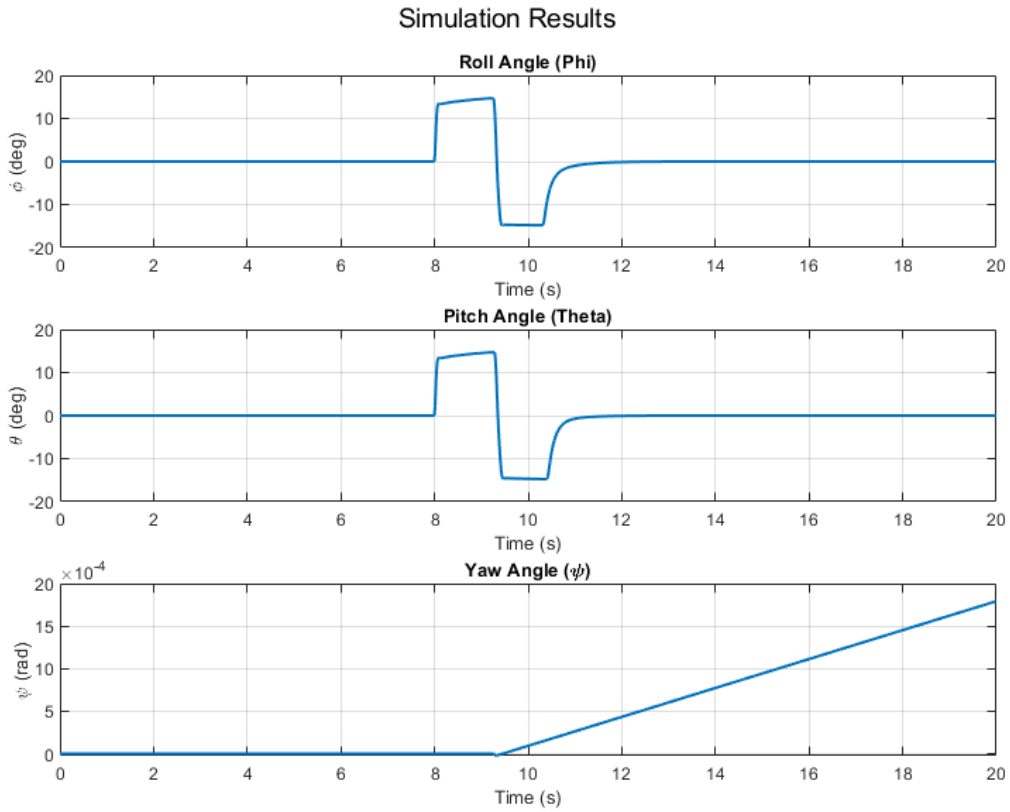

Figure 5: Drone Euler angles

Παρατηρούμε πως η γωνία yaw (ψ) δεν μεταβάλεται και παραμένει πρακτικά σταθερή, παρά το γεγονός ότι δε την ελέγχουμε. Αυτό οφείλεται στο ότι οι στόχοι και οι ελεγκτές για τις γωνίες ϕ, θ είναι ίδιοι, επομένως δίνοντας την ίδια εντολή ταχύτητας, οι ροπές yaw ακυρώνονται. Τα διαγράμματα ταχυτήτων κάθε έλικα φαίνονται στο διάγραμμα 7.

Αξίζει επίσης να αναφέρουμε πως παρότι χρησιμοποιήθηκε ο ίδιος ελεγκτής όπως και στο 3.1, τα κέρδη του χρειάστηκαν προσαρμογή, ώστε να μειωθεί το μέγιστο σφάλμα στην μεταβατική κατάσταση. Τα νέα κέρδη φαίνονται παρακάτω:

$$[K_p = 2, K_i = 0.1, K_d = 4.4], \quad \text{Για το } U_1 \tag{18}$$

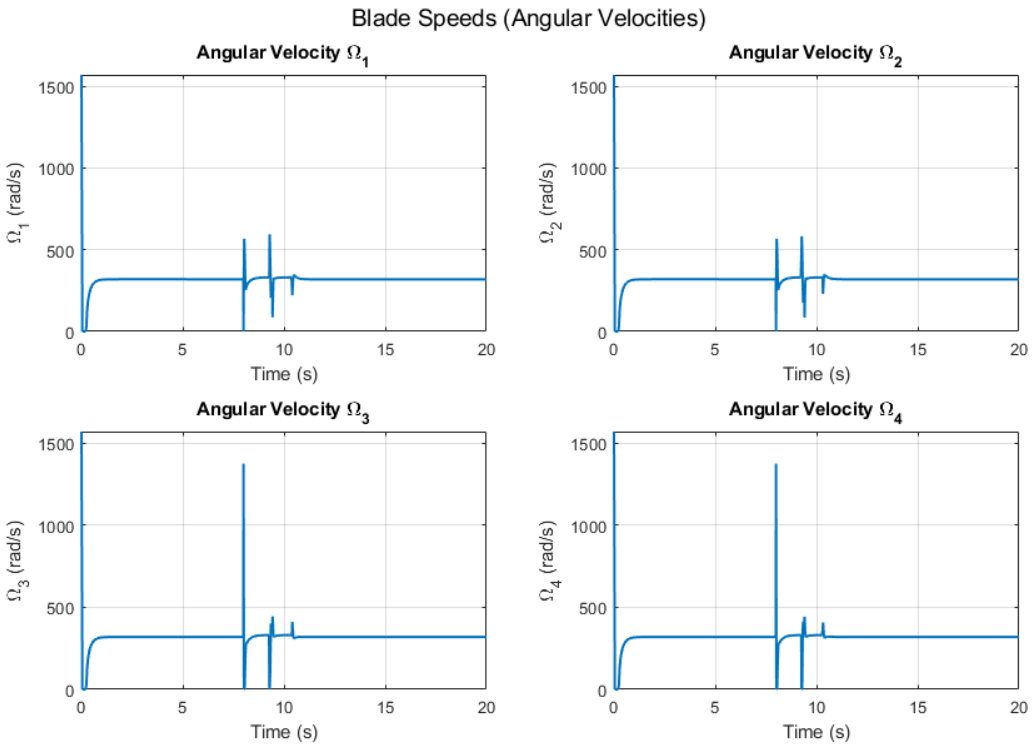

Figure 6: Drone angular velocities

3.3 Έλεγχος x, y, z καθώς και ψ σε επιθυμητές τιμές, χρησιμοποιώντας Cascaded PID

Στόχος τώρα είναι να κάνουμε τον έλεγχο του ερωτήματος 3.2 ενώ ταυτόχρονα ελέγχουμε και τη γωνία yaw (ψ). Για να το πετύχουμε αυτό θα κρατήσουμε το controller concept που περιγράφηκε προηγουμένως, και θα υλοποιήσουμε εκ νέου έναν απλό PID που θα ελέγχει το yaw. Στην περίπτωση που χρειαστεί, θα γίνουν και αλλαγές στους άλλους ελεγκτές για να βελτιωθεί η συνδυασμένη απόδοση του συστήματος.

Μετά από δοκιμές καταλήγουμε στα παρακάτω κέρδη για τους ελεγκτές. Παρατηρούμε πως χρειάστηκαν πολύ μικρές αλλαγές και όχι σε όλους τους ελεγκτές. Ακόμη, για τον έλεγχο της yaw γωνίας χρησιμοποιήθηκε ο ίδιος ελεγκτής που είναι υπεύθυνος για τον έλεγχο των κινητήρων στους Cascaded PID.

$$[K_p = 2, K_i = 0.1, K_d = 4.4], \quad \text{Για το } U_1 \tag{19}$$

$$[K_p = 1, K_i = 0, K_d = 1.05], \quad \text{Για τον υπολογισμό desired γωνίας} \tag{20}$$

$$[K_p = 2.95, K_i = 0.5, K_d = 2.6], \quad \text{Για τον έλεγχο των κινητήρων} \tag{21}$$

$$[K_p = 2.95, K_i = 0.5, K_d = 2.6], \quad \text{Για το } U_4 \tag{22}$$

Παρακάτω φαίνεται η απόκριση του συστήματος. Αξίζει να σημειωθεί πως αυτή τη τώρα οι στόχοι για την θέση x και y, ορίζονται με χρονική διαφορά 1 δευτερόλεπτο, με σκοπό να δυσκολέψει ακόμη περισσότερο ο ρόλος του ελεγκτή του yaw.

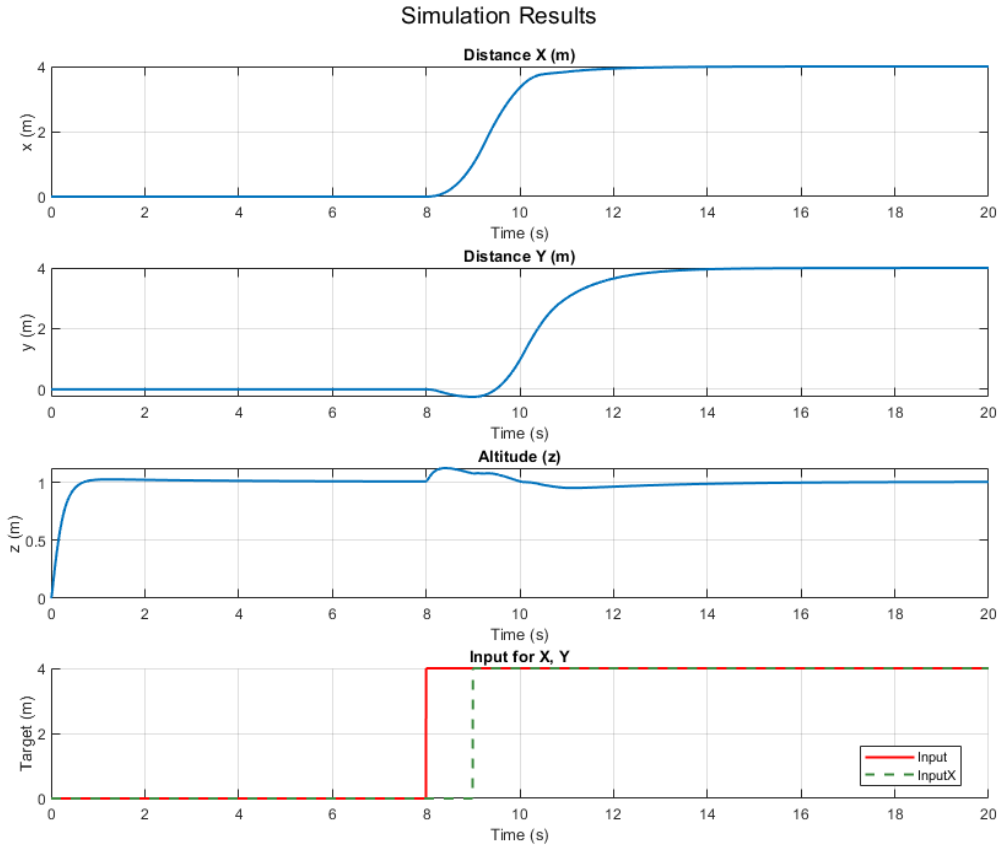

Figure 7: Drone position & Controller input

Παρατηρούμε πως παρά την αυξημένη πολυπλοκότητα του προβλήματος ελέγχου, οι ελεγκτές μας καταφέρνουν να οδηγήσουν το drone αποτελεσματικά στα σημεία ισορροπίας που θέτουμε ως στόχους. Η επίδραση του ελεγκτή της γωνίας yaw είναι αντιληπτή στο διάγραμμα 9. Συγκεκριμένα βλέπουμε την ταχύτητα των ελίκων να αυξάνεται και να μειώνεται με τέτοιο τρόπο ώστε να μην επηρεάζεται καθόλου το ύψος, δίνοντας την κατάλληλη ταχύτητα στο drone ώστε να ακολουθήσει την επιθυμητή γωνία. Όταν τελικά ο στόχος της γωνίας σταματάει να μεταβάλεται, παρατηρούμε την αντίθετη συμπεριφορά από το σύστημα ώστε αυτό να πάψει να στρέφεται.

Στο διάγραμμα 8 φαίνεται πως ο εκλεκτής μας ανταποκρίνεται πολύ γρήγορα και με ελάχιστο overshoot.

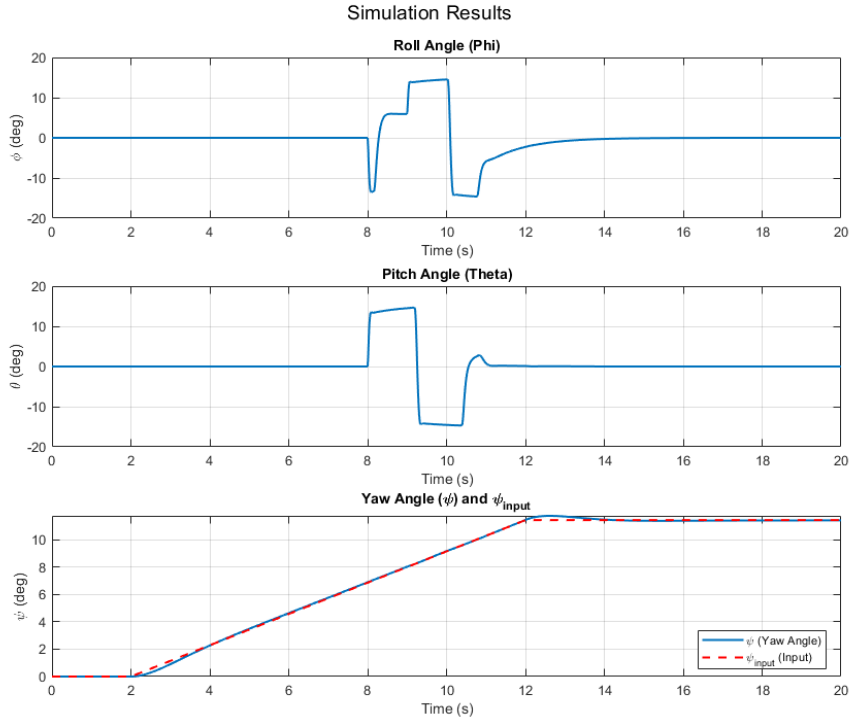

Figure 8: Drone Euler angles

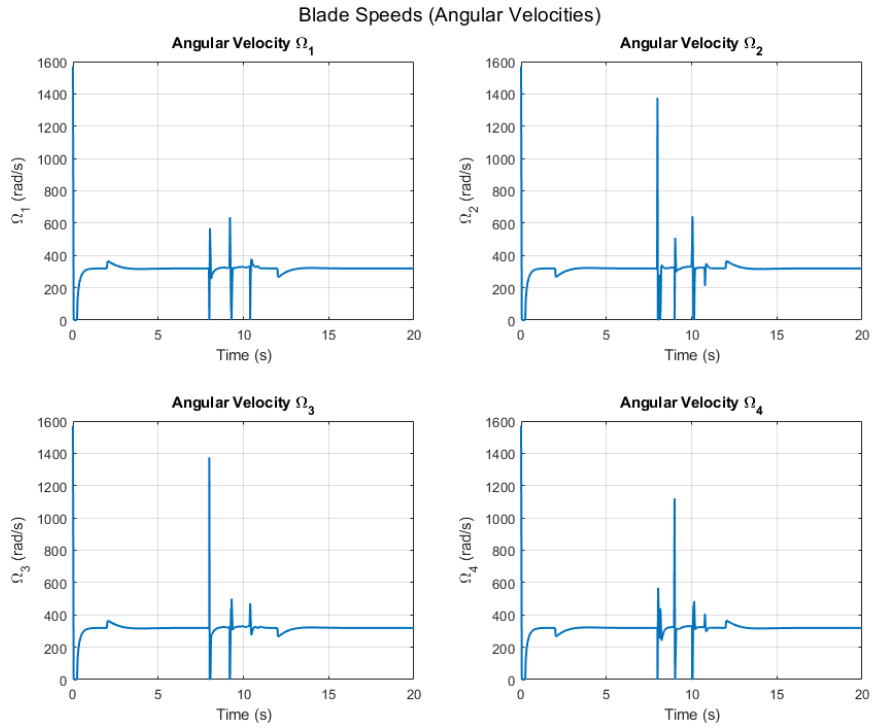

Figure 9: Drone angular velocities

4 Έλεγχος στο χώρο κατάστασης

4.1 Υπολογισμός σημείων ισορροπίας

Για τον υπολογισμό των σημείων ισορροπίας αρκεί να απαιτήσω να ισχύει η σχέση στην οποία ορίζουμε ώς s το state vector $[x, y, z, \phi, \theta, \psi]$. Επίσης είναι φανερό πως για να ισορροπεί το σύστημα οι γωνίες θ και ϕ πρέπει να είναι 0. Αν αυτό δεν ίσχυε, και το drone είχε ύψος $z > 0$, τότε δεν θα βρισκόταν σε ισορροπία.

$$\frac{ds}{dt} = 0 \tag{23}$$

Οι εξισώσεις κατάστασης λοιπόν, θεωρώντας $\theta = 0$ και $\phi = 0$ γράφονται ώς εξής:

$$\ddot{x} = 0 \tag{24}$$

$$\ddot{y} = 0 \tag{25}$$

$$\ddot{z} = 0 \longrightarrow U_1 = mg \tag{26}$$

$$\ddot{\phi} = 0 \tag{27}$$

$$\ddot{\theta} = 0 \tag{28}$$

$$\ddot{\psi} = 0 \tag{29}$$

Συμπεραίνουμε λοιπόν πως οποιοδήποτε σημείο στον καρτεσιανό χώρο μπορεί να αποτελέσει σημείο ισορροπίας, αρκεί η συνολική δύναμη ώθησης να είναι ίση με το βάρος του οχήματος.

4.2 Γραμμικοποίηση γύρω από σημείο ισορροπίας

Αρχικά, προκειμένου να γραμμικοποιήσουμε το συγκεκριμένο σύστημα, πρέπει να το εκφράσουμε σαν σύστημα διαφορικών εξισώσεων πρώτου βαθμού. Η δυναμική του συστήματος λοιπόν γράφεται ως εξής:

$$\begin{aligned}
\dot{x}_1 &= x_2, \\
\dot{x}_2 &= p_1 x_4 x_6 + p_2 U_2, \\
\dot{x}_3 &= x_4, \\
\dot{x}_4 &= p_3 x_2 x_6 + p_4 U_3, \\
\dot{x}_5 &= x_6, \\
\dot{x}_6 &= p_5 x_2 x_4 + p_6 U_4, \\
\dot{x}_7 &= x_8, \\
\dot{x}_8 &= \left(c_\phi s_\theta c_\psi + s_\phi s_\psi\right) \frac{1}{m} U_1,
\end{aligned}$$

$$
\begin{aligned}
\dot{x}_9 &= x_{10}, \\
\dot{x}_{10} &= \left(-c_\phi s_\theta s_\psi + s_\phi c_\psi\right) \frac{1}{m} U_1, \\
\dot{x}_{11} &= x_{12}, \\
\dot{x}_{12} &= -g + \left(c_\theta c_\phi\right) \frac{1}{m} U_1,
\end{aligned}
$$

Επιλέγουμε να γραμμικοποιήσουμε γύρω από το σημείο ισορροπίας με καρτεσιανές συντεταγμένες $(x, y, z) = (1, 1, 1)$ και γωνίες Euler $(\phi, \theta, \psi) = (0, 0, 0)$. Καθώς λοιπόν που το σύστημα μας έχει πλέον την κατάλληλη μορφή και γνωρίζουμε τα σημεία που μας ενδιαφέρουν, μπορούμε να υπολογίσουμε τους πίνακες A και B του γραμμικού συστήματος πρώτης τάξης από τη σχέση:

$$\dot{x} = \frac{\partial f(x,u)}{\partial x}(x - x_{ss}) + \frac{\partial f(x,u)}{\partial u}(u - u_{ss}) \tag{30}$$

$$A = \frac{\partial f(x,u)}{\partial x}, \quad B = \frac{\partial f(x,u)}{\partial u} \tag{31}$$

Προκύπτει λοιπόν ότι οι πίνακες A και B είναι ίσοι με:

$$
A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\tag{32}
$$

$$
B = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 63.8162 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 63.8162 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 35.2858 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1.25 & 0 & 0 & 0
\end{bmatrix}
$$

4.3 Υπολογισμός πίνακα μεταφοράς από την είσοδο στην έξοδο (με έξοδους x, y, z, ψ)

Το σύστημα μας είναι ένα MIMO σύστημα (Multiple Input Multiple Output). Συνεπώς δεν ορίζεται μια μοναδική συνάρτηση μεταφορας, αλλά ένας πίνακας συναρτήσεων μεταφοράς από τις εισόδους στις εξόδους. Υπολογίζοντας αυτό τον πίνακα, για το γραμμικοποιημένο σύστημα, μπορούμε να κατανοήσουμε πως επηρεάζει η κάθε είσοδος την έξοδο, καθώς και ποιές είσοδοι επηρεάζουν ποιές εξόδους, κάνοντας ευκολότερο το tuning process κατά τον σχεδιασμό ελεγκτών.
Χρησιμοποιώντας τις κατάλληλες συναρτήσεις της MATLAB για να αποφύγουμε λάθη στις πράξεις, υπολογίσαμε τον πίνακα μεταφοράς:

$$G = \begin{bmatrix} \frac{1.25}{s^2} & 0 & 0 & 0 \\ 0 & \frac{626.04}{s^4} & 0 & 0 \\ 0 & 0 & \frac{626.04}{s^4} & 0 \\ 0 & 0 & 0 & \frac{35.2858}{s^2} \end{bmatrix}$$

Για τον παραπάνω πίνακα μεταφοράς ισχύει:

$$\begin{bmatrix} z \\ y \\ x \\ \psi \end{bmatrix} = \begin{bmatrix} \frac{1.25}{s^2} & 0 & 0 & 0 \\ 0 & \frac{626.04}{s^4} & 0 & 0 \\ 0 & 0 & \frac{626.04}{s^4} & 0 \\ 0 & 0 & 0 & \frac{35.2858}{s^2} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

Παρατηρούμε πως ο πίνακας μεταφοράς είναι διαγώνιος. Αυτό συμβαίνει καθώς επιλέξαμε να γραμμικοποιήσουμε το σύστημα γύρω από γωνία yaw $\psi = 0$, το οποίο επιτρέπει decoupling των x και y. Βλέπουμε επίσης ότι η θέση στον άξονα x επηρεάζεται από το pitch angle, και αντίστοιχα η θέση στον y από το roll angle, δηλαδή η θέση στον x εξαρτάται από την περιστροφή γύρω από τον y και αντίστοιχα η θέση y από την περιστροφή στο x. Αυτό είναι μια χρήσιμη παρατήρηση να εκφραστεί και μαθηματικά, γιατί σε συστήματα για το οποία δεν έχουμε τοσο εύκολη εμπειρική κατανόηση, τέτοια πράγματα ίσως να μην είναι τόσο αυτονόητα. Παρατηρούμε επίσης, πως καθώς οι είσοδοι U_2, U_3 δεν επηρεάζουν άμεσα τις εξόδους, αλλά τις επηρεάζουν μέσω των γωνιών όπως συζητήσαμε προηγουμένως, χρειάζονται 4 ολοκληρωτές (s^4 στον παρονομαστή). Αντίθετα, U_1, U_4 επηρεάζουν άμεσα τις μεταβλητές τους οπότε χρειάζονται μόνο 2 ολοκληρωτές.

Προφανώς οι παρατηρήσεις αυτές ισχύουν για το γραμμικοποιημένο σύστημα, επομένως γίνεται αντιληπτό ότι το πραγματικό σύστημα δεν θα παρουσιάζει ακριβώς αυτά τα χαρακτηριστικά.

4.4 Σχεδιασμός ελεκτή με LQR και δοκιμή σε συγκεκριμένη πορεία

Καθώς τώρα έχουμε γραμμικοποιήσει το σύστημα και έχουμε αποκτήσει κάποια κατανόηση για το πως συνδέονται οι έξοδοι με τις εισόδους, μπορούμε να χρησιμοποιήσουμε κλασικά εργαλεία για το σχεδιασμό γραμμικών ελεγκτών σε γραμμικά συστήματα στο χώρο κατάστασης. Συγκεκριμένα θα χρησιμοποιήσουμε έναν LQR, προκειμένου να υπολογίσουμε τα κέρδη του ελεγκτή.

Ας περιγράψουμε πρώτα όμως με ένα block diagram το νόμο ελέγχου. Προηγουμένως χρησιμοποιήσαμε 4 cascaded PID ελεγκτές, έναν για κάθε είσοδο. Τώρα λύνοντας την εξίσωση Riccati για επιλεγμένα Q και R matrices, θα καταλήξουμε σε έναν πίνακα κερδών και θα εφαρμόσουμε το νόμος ελέγχου που περιγράφεται από το παρακάτω block diagram.

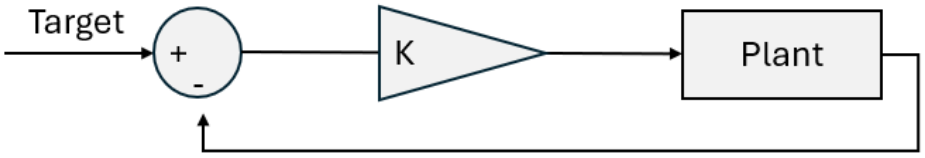

Figure 10: LQR Control law

Όπως και στο tuning των PID ελεγκτών, προκειμένου να βρούμε τους καταλληλότερους Q, R πίνακες ακολουθήσαμε μια trial and error διαδικασία. Μετά από πολλές δοκιμές όπου και πάλι ως στόχο είχαμε την γρήγορη απόκριση συστήματος καθώς και την ελαχιστοποίηση απόκλισης από την επιθυμητή κατάσταση καταλήξαμε στους παρακάτω πίνακες:

$$Q = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{33}$$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$$

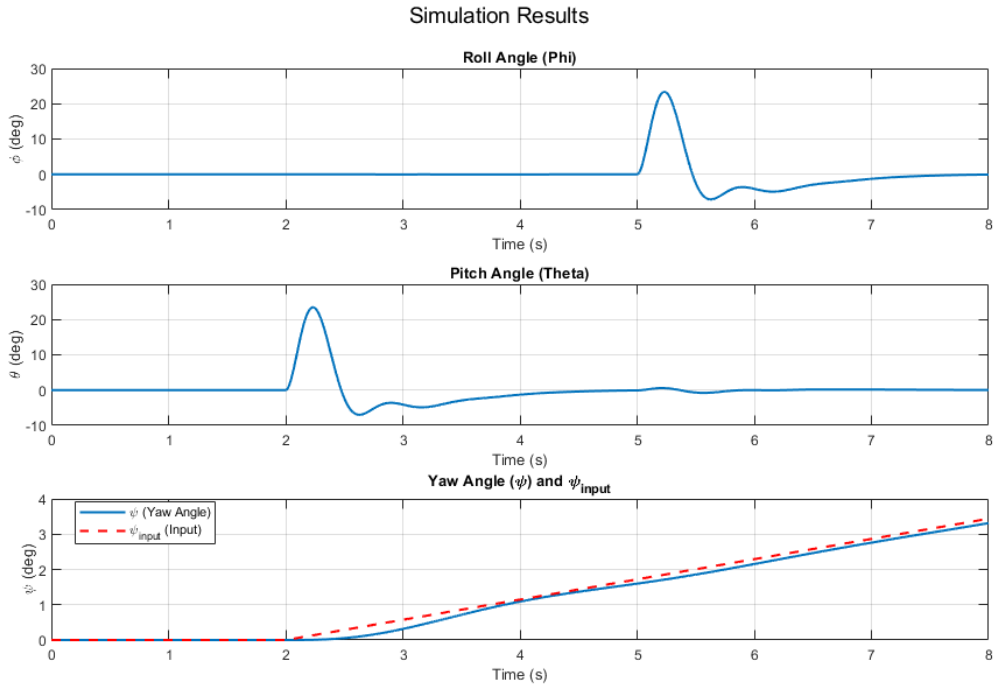

Figure 11: LQR controller Euler angle response

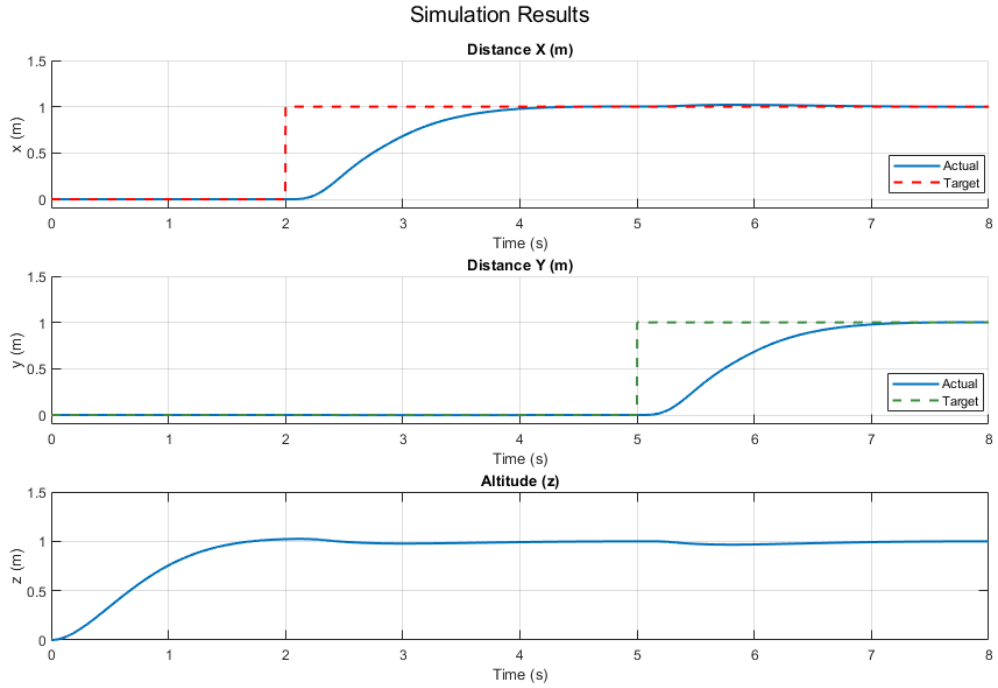

Figure 12: LQR Controller cartesian coordinates

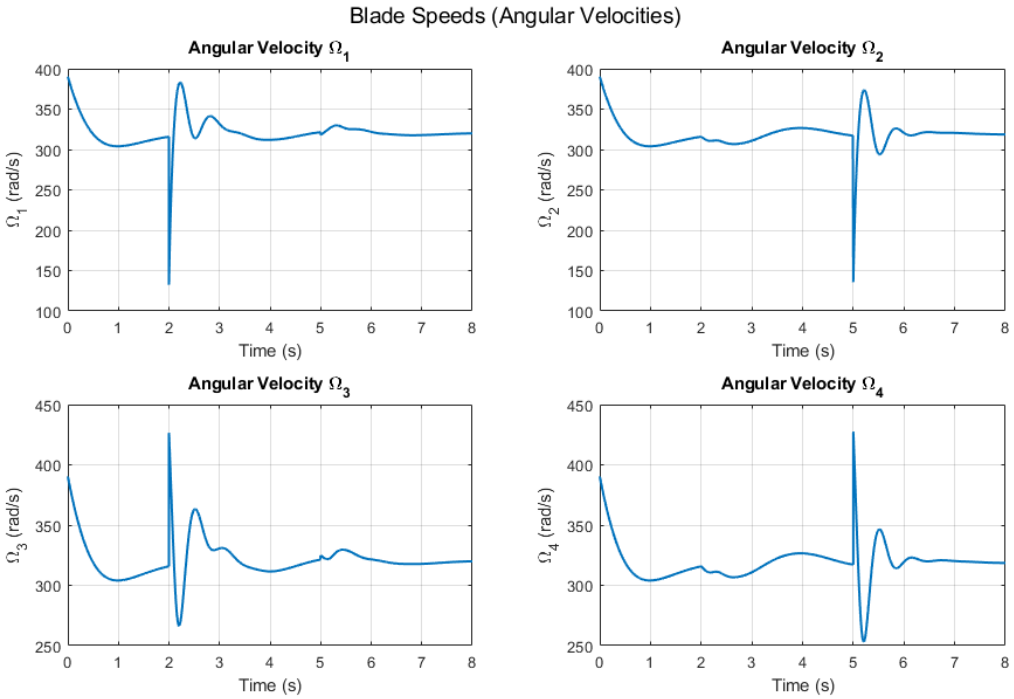

Figure 13: LQR Controller cartesian coordinates

Είναι φανερό πως ο controller που σχεδιάσαμε με τον LQR, αποδίδει καλύτερα σε σχέση με τους cascaded PID, τόσο και στο πόσο καλά καταφέρνει να ακολουθήσει τον στόχο, χωρίς overshoots, αλλά και από άποψη actuator effort. Αυτό το καταλαβαίνουμε από το διάγραμμα 13 καθώς οι αλλαγές στις ταχύτητες είναι πολύ πιο ομαλές, και δεν παρατηρούμε τόσο μεγάλα spikes στις ταχύτητες όπως αυτά που προκαλούν οι PID, που κάνει αυτή τη μέθοδο ελέγχου, και συγκεκριμένα τον ελεγκτή αυτό, προτιμότερη επιλογή για τον έλεγχο ενός physical drone. Σε ένα physical drone, δε θα μπορούμε να επιταχύνουμε με την ίδια ευκολία τους έλικες και επιπλέον, θα επωφεληθούμε από τη χαμηλότερη κατανάλωση ενέργειας.

Τώρα θα σχεδιάσουμε μια συγκεκριμένη τροχιά και θα ζητήσουμε από τον ελεγκτή του drone, να κινήσει το σύστημα πάνω σε αυτή την τροχιά.

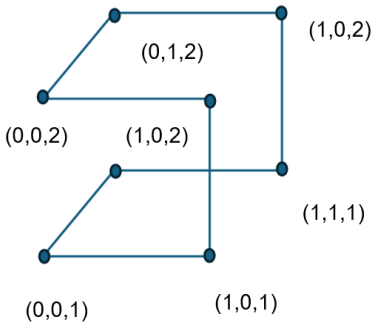

Figure 14: Drone desired trajectory

Παρακάτω φαίνεται η απόκριση του συστήματος στην τροχιά:

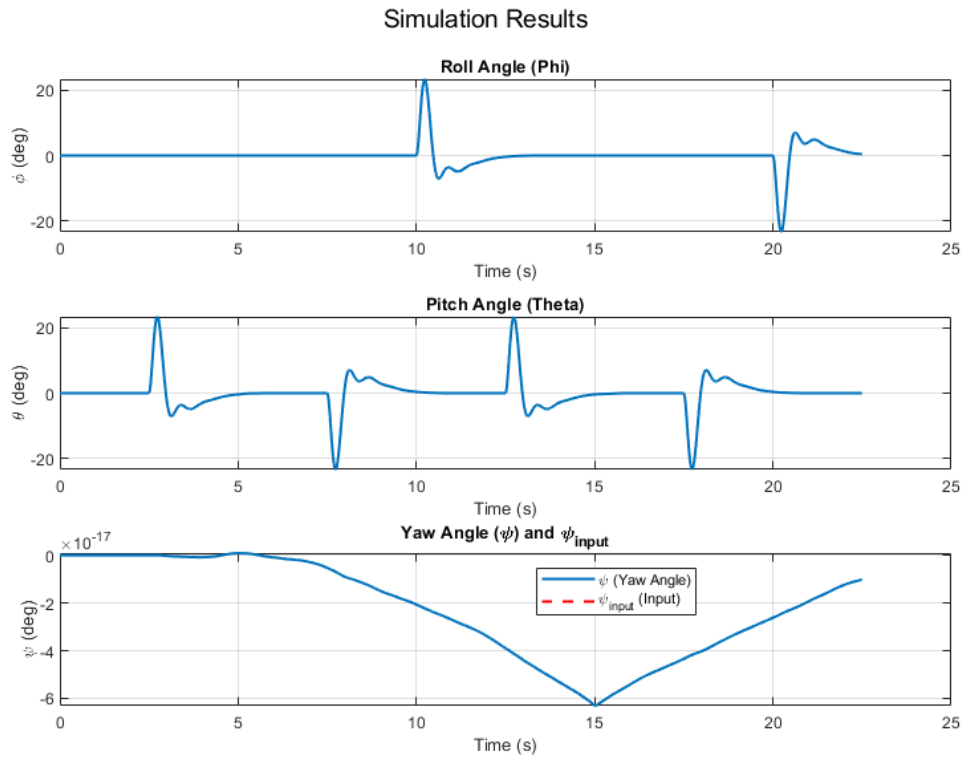

Figure 15: Drone Euler angles following trajectory described in figure 14

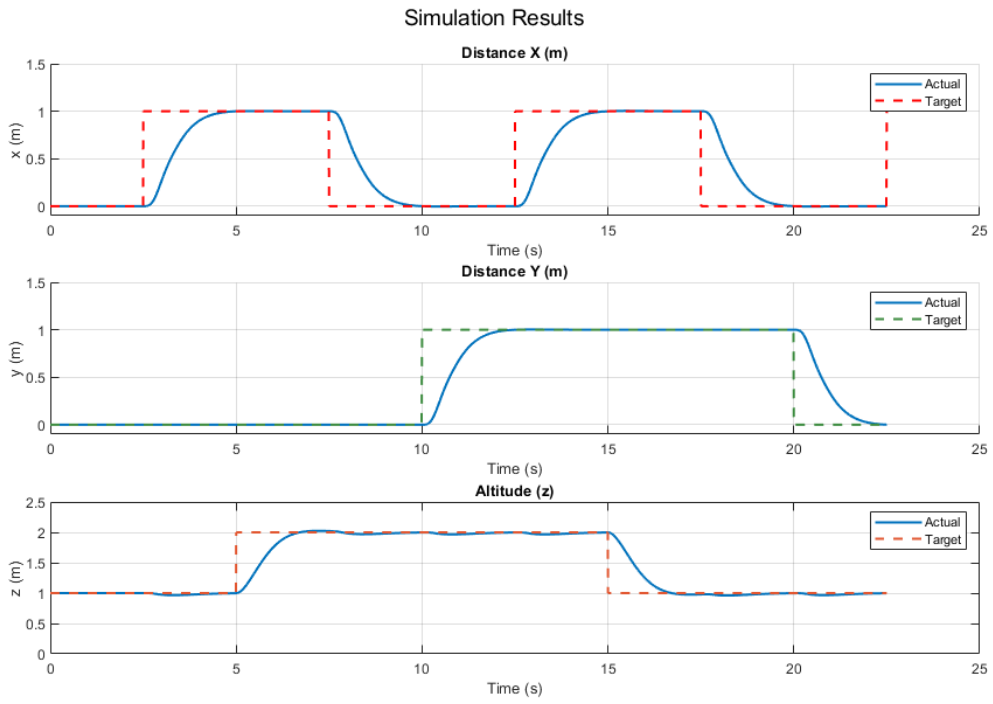

Figure 16: Drone response cartesian coordinates & coordinates of figure 14

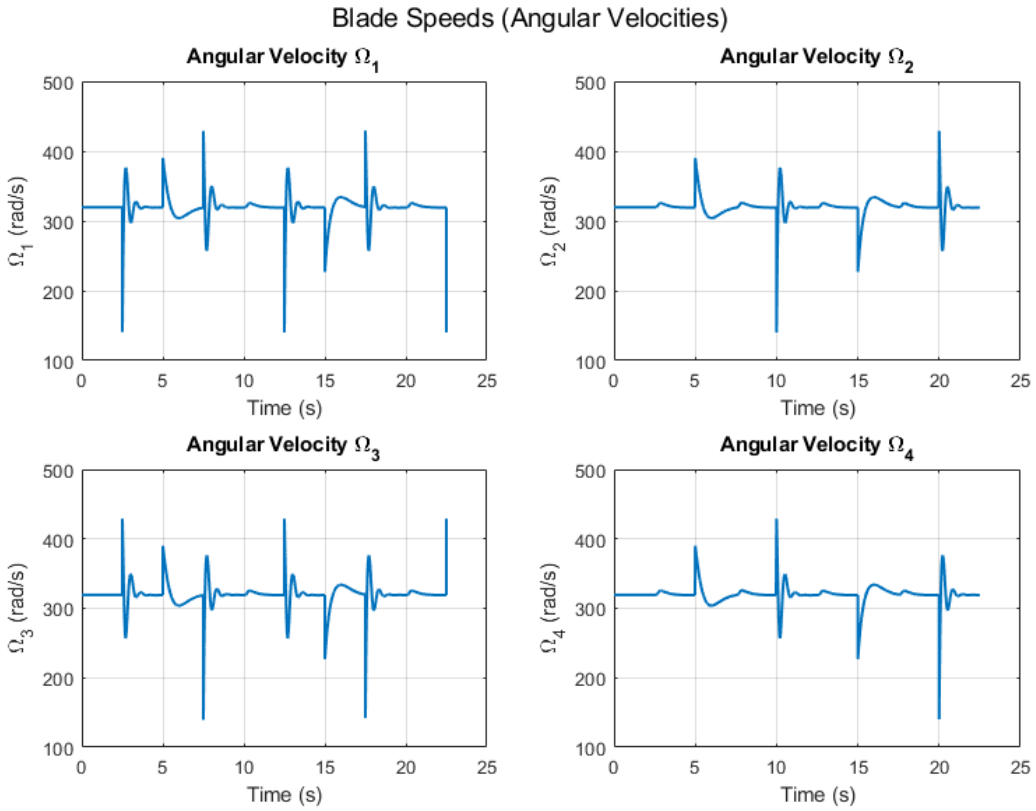

Figure 17: Drone blade speeds following described in figure 14

Παρατηρούμε πως ο ελεγκτής κρατάει το drone ικανοποιητικά στην τροχιά, χωρίς να έχει αισθητά overshoots η undershoots, και χωρίς να κάνει overwork τους κινητήρες των ελίκων. Στο διάγραμμα 18 φαίνεται ακριβώς η τροχιά που τελικά ακολουθεί το drone για να γίνει καλύτερα αντιληπτό το performance του ελεγκτή.

Γίνεται για ακόμη μια φορά αντιληπτό πως η προτιμότερη μέθοδος σχεδίασης ελεγκτών είναι μέσω LQR και επίλυσης της αντίστοιχης εξίσωσης Riccati, καθώς και το tuning είναι ευκολότερο, αλλά και η τελική επίδοση του συστήματος είναι καλύτερη. Ειδικά για ένα σύστημα όπως αυτό, όπου έχουμε πολλές μεταβλητές στο χώρο κατάστασης, όταν σχεδιάσαμε τους Cascaded PID ελεγκτές για τις μεταβλητές x, y, άρα 4 ελεγκτές, και ακόμη 2 ελεγκτές για τις μεταβλητές z, ψ, χρειάστηκε να κάνουμε tune 18 παραμέτρους. Στον LQR ελεγκτή είχαμε 16 παραμέτρους που χρειάστηκαν tuning, που λόγω της ανάθεσης βαρών στο διάνυσμα κατάστασης μέσω του πίνακα Q, κάνει το tuning ιδιαίτερα ευκολότερο απ' ότι του PID.

Σε κάθε περίπτωση, είναι απαραίτητο να αναφέρουμε το σημαντικό πλεονέκτημα της PID μεθόδου, που είναι ιδιαίτερα εύκολη υλοποίηση και προσαρμογή στο μοντέλο. Δε χρειάζεται καμία γραμμικοποίηση η γενικότερα προεπεξεργασία του συστήματος. Παραμένει ένας εύκολος και αξιόπιστος τρόπος να σχεδιάσουμε γρήγορα λειτουργικούς ελεγκτές.

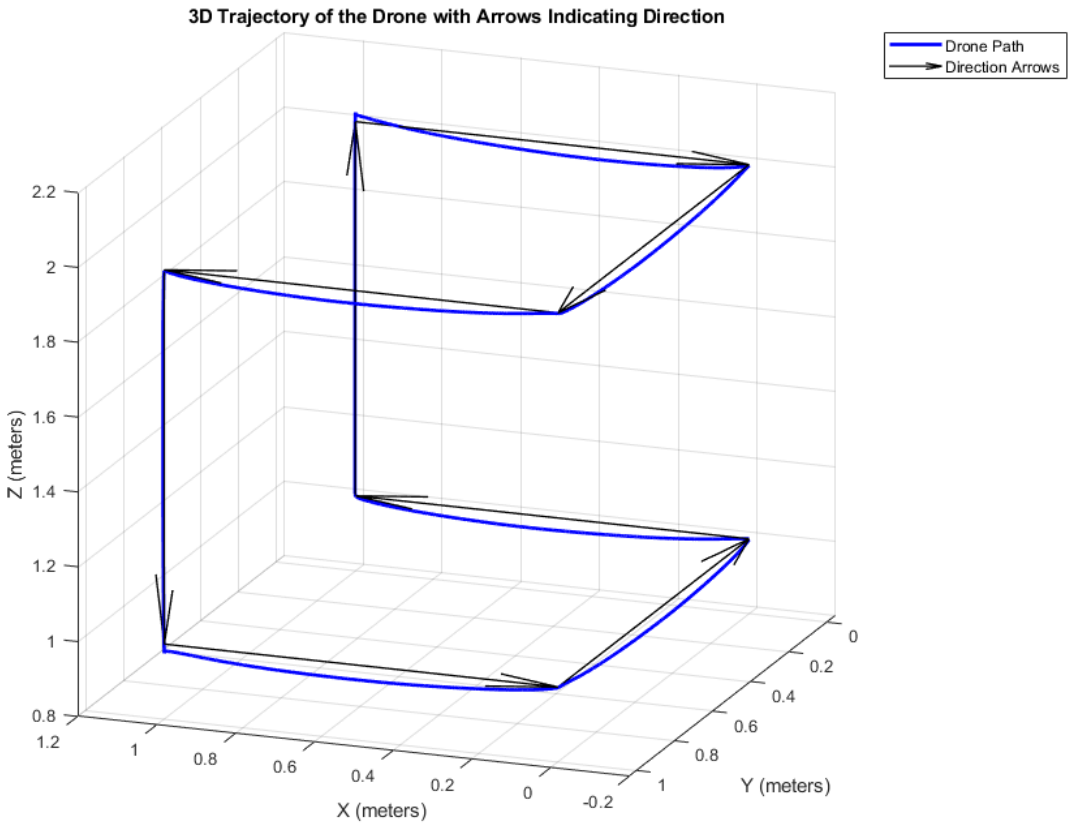

Figure 18: Drone trajectory response for reference trajectory in figure 14