Ρομποτικά Συστήματα ΙΙ: Εργασία 2

Ονοματεπώνυμο: Βραχωρίτη Αλεξάνδρα

Αριθμός Μητρώου: 1092793

12 Δεκεμβρίου 2025

1 Μοντελοποίηση συστήματος ενός planar quadrotor

Στην εκφώνηση δίνεται η δυναμική του συστήματος ως:

$$f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \frac{1}{m}(-\sin\theta(T + f_y) + \cos\theta f_x) \\ \frac{1}{m}(\cos\theta(T + f_y) + \sin\theta f_x) - g \\ \frac{\tau - f_\theta}{J} \end{bmatrix} \tag{1}$$

Στη συνέχεια, για τη διακριτοποίηση του συστήματος χρησιμοποιούμε τον integrator Runge-Kutta 4th Order και γίνεται ένας απλός έλεγχος με PD-controller, για τον οποίο έχουμε:

$a_{x_{desired}} = K_{px}(x_{desired} - x) + K_{dx}(\dot{x}_{desired} - \dot{x})$

$a_{y_{desired}} = K_{py}(y_{desired} - y) + K_{dy}(\dot{y}_{desired} - \dot{y})$

$a_{\theta_{desired}} = K_{p\theta}(\theta_{desired} - \theta) + K_{d\theta}(\dot{\theta}_{desired} - \dot{\theta})$.

Λύνοντας τις δύο τελευταίες εξισώσεις του διανύσματος δυναμικής ως προς T και τ αντίστοιχα, θα έχουμε:

$T = \frac{-\cos\theta f_y - \sin\theta f_x + m a_{y_{desired}} + mg}{\cos\theta}$

$\tau = f_\theta + J a_{\theta_{desired}}$.

Εδώ θα πρέπει να προσέξουμε ότι στην αρχική και τελική θέση το quadrotor έχει γωνία $\theta = 0$ και γωνιασκή ταχύτητα $\dot{\theta} = 0$, πράγμα το οποίο σημαίνει πως $a_{\theta_{desired}} = 0$ συνεχώς. Για να το αποφύγουμε κάνουμε μια προσέγγιση της γωνίας θ. Πάμε, λοιπόν, στην εξίσωση $\frac{1}{m}(-\sin\theta(T + f_y) + \cos\theta f_x)$ και θεωρούμε $f_x = f_y \approx 0$, $T \approx mg$ στο σημείο όπου το quadrotor θα κάνει hover και $\sin\theta \approx \theta$ για πολύ μικρές γωνίες. Έτσι, έχουμε $\theta \approx -\frac{a_x}{g}$. Με την κατάλληλη ρύθμιση των κερδών για τον έλεγχο κάθε μίας μεταβλητής, το quadrotor κάνει hover στο $\begin{bmatrix} x & y & \theta \end{bmatrix} = \begin{bmatrix} -0.5 & 0.5 & 0. \end{bmatrix}$, ξεκινώντας από το $\begin{bmatrix} x & y & \theta \end{bmatrix} = \begin{bmatrix} 0. & 0.042 & 0. \end{bmatrix}$. Αν ξεκινήσει από γωνία $\neq 0$ ή ζητήσουμε να κάνει hover σε γωνία $\neq 0$, τότε ο αλγόριθμος "σπάει" και δεν υπάρχει λύση.

2 Γραμμικοποίηση γύρω από ένα σταθερό σημείο και ο αλγόριθμος LQR

2.1 Γραμμικοποίηση

Για την γραμμικοποίηση του συστήματος γύρω από το $\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}$ και τον προσδιορισμό των πινάκων A, B χρειάζεται να προσδιορίσουμε τους πίνακες (αναλυτικά) $A_{continuous} = A_c = \frac{\partial f}{\partial \boldsymbol{x}_k}$ και $B_{continuous} = B_c = \frac{\partial f}{\partial \boldsymbol{u}_k}$ (με f συμβολίζουμε τα `dynamics()` και είναι διαφορετικό από τα f_x και f_y της εκφώνησης), οι οποίοι προκύπτουν:

$$A_c = \begin{bmatrix} 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. \\ 0. & 0. & -\frac{1}{m}(\cos\theta(T+f_y)+\sin(\theta)f_x) & \frac{\cos\theta}{m}df_x & -\frac{\sin\theta}{m}df_y & 0. \\ 0. & 0. & \frac{1}{m}(-\sin\theta(T+f_y)+\cos(\theta)f_x) & \frac{\sin\theta}{m}df_x & \frac{\cos\theta}{m}df_y & 0. \\ 0. & 0. & 0. & 0. & 0. & \frac{k_\theta}{J} \end{bmatrix}, \quad B_c = \begin{bmatrix} 0. & 0. \\ 0. & 0. \\ 0. & 0. \\ -\frac{1}{m}\sin\theta & 0. \\ \frac{1}{m}\cos\theta & 0. \\ 0. & \frac{1.}{J}] \end{bmatrix}.$$

Για την διακριτοποίηση του συστήματος χρησιμοποιούμε την αριθμητική μέθοδο ολοκλήρωσης Runge-Kutta 4ης τάξης, η οποία διέπεται από τις ακόλουθες εξισώσεις για τον υπολογισμό της δυναμικής του συστήματος σε κάθε επόμενο βήμα $(k+1)$:

$$\begin{aligned} f_1 &= f\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \\ f_2 &= f\left(\boldsymbol{x}_k + \frac{dt}{2} f_1, \boldsymbol{u}_k\right) \\ f_3 &= f\left(\boldsymbol{x}_k + \frac{dt}{2} f_2, \boldsymbol{u}_k\right) \\ f_4 &= f(\boldsymbol{x}_k + dt \cdot f_3, \boldsymbol{u}_k) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \frac{dt}{6}\left(f_1 + 2f_2 + 2f_3 + f_4\right) \end{aligned}$$

Παίρνοντας την 1η παράγωγο αυτής της μεθόδου και εφαρμόζοντας τον κανόνα της αλυσίδας, προσδιορίζουμε τους πίνακες $A_{discrete} = A_d = \frac{\partial f_{discrete}}{\partial \boldsymbol{x}_k}$ και $B_{discrete} = B_d = \frac{\partial f_{discrete}}{\partial \boldsymbol{u}_k}$. Πιο συγκεκριμένα, ορίζουμε τα διανύσματα καταστάσεων:

$$\begin{aligned} \boldsymbol{x}_2 &= \boldsymbol{x}_k + \frac{dt}{2} \cdot f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \boldsymbol{x}_3 &= \boldsymbol{x}_k + \frac{dt}{2} \cdot f(\boldsymbol{x}_2, \boldsymbol{u}_k) \\ \boldsymbol{x}_4 &= \boldsymbol{x}_k + dt \cdot f(\boldsymbol{x}_3, \boldsymbol{u}_k) \end{aligned}$$

Οι παράγωγοί τους ως προς $\boldsymbol{x}_k$ υπολογίζονται ως εξής:

$$\begin{aligned} \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{x}_k} &= I + \frac{dt}{2}\frac{\partial f}{\partial \boldsymbol{x}_k} \\ \frac{\partial \boldsymbol{x}_3}{\partial \boldsymbol{x}_k} &= I + \frac{dt}{2}\frac{\partial f}{\partial \boldsymbol{x}_2} \cdot \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{x}_k} \\ \frac{\partial \boldsymbol{x}_4}{\partial \boldsymbol{x}_k} &= I + dt \cdot \frac{\partial f}{\partial \boldsymbol{x}_3} \cdot \frac{\partial \boldsymbol{x}_3}{\partial \boldsymbol{x}_k} \end{aligned}$$

Αντίστοιχα για τις παραγώγους ως προς την είσοδο $\boldsymbol{u}_k$:

$$\begin{aligned} \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{u}_k} &= \frac{dt}{2}\frac{\partial f}{\partial \boldsymbol{u}_k} \\ \frac{\partial \boldsymbol{x}_3}{\partial \boldsymbol{u}_k} &= \frac{dt}{2}\left(\frac{\partial f}{\partial \boldsymbol{u}_2} + \frac{\partial f}{\partial \boldsymbol{x}_2} \cdot \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{u}_k}\right) \\ \frac{\partial \boldsymbol{x}_4}{\partial \boldsymbol{u}_k} &= \frac{dt}{2}\left(\frac{\partial f}{\partial \boldsymbol{u}_3} + \frac{\partial f}{\partial \boldsymbol{x}_3} \cdot \frac{\partial \boldsymbol{x}_3}{\partial \boldsymbol{u}_k}\right) \end{aligned}$$

Έτσι, καταλήγουμε στους πίνακες του γραμμικοποιημένου συστήματος:

$$\begin{aligned} A_d &= I + \frac{dt}{6}\left(\frac{\partial f}{\partial \boldsymbol{x}_k} + 2\frac{\partial f}{\partial \boldsymbol{x}_2} \cdot \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{x}_k} + 2\frac{\partial f}{\partial \boldsymbol{x}_3} \cdot \frac{\partial \boldsymbol{x}_3}{\partial \boldsymbol{x}_k} + \frac{\partial f}{\partial \boldsymbol{x}_4} \cdot \frac{\partial \boldsymbol{x}_4}{\partial \boldsymbol{x}_k}\right) \\ B_d &= \frac{dt}{6}\left(\frac{\partial f}{\partial \boldsymbol{u}_k} + 2\left(\frac{\partial f}{\partial \boldsymbol{u}_2} + \frac{\partial f}{\partial \boldsymbol{x}_2} \cdot \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{u}_k}\right) + 2\left(\frac{\partial f}{\partial \boldsymbol{u}_3} + \frac{\partial f}{\partial \boldsymbol{x}_3} \cdot \frac{\partial \boldsymbol{x}_3}{\partial \boldsymbol{u}_k}\right) + \left(\frac{\partial f}{\partial \boldsymbol{u}_2} + \frac{\partial f}{\partial \boldsymbol{x}_4} \cdot \frac{\partial \boldsymbol{x}_4}{\partial \boldsymbol{u}_k}\right)\right) \end{aligned}$$

2.2 Υλοποίηση του LQR

Ο αλγόριθμος ζητείται να υλοποιηθεί σε infinite horizon, άρα υπολογίζουμε τους πίνακες P και K, ώσπου ο πρώτος να είναι αρκετά κοντά στον πίνακα P_k, δηλαδή μέχρι ο αλγόριθμος να συγκλίνει.

Algorithm 1: Infinite Horizon LQR

Input: System matrices $\boldsymbol{A}, \boldsymbol{B}$, Cost matrices $\boldsymbol{Q}, \boldsymbol{R}$, Terminal $\boldsymbol{Q}_N$
Output: Optimal Gain $\boldsymbol{K}$, Cost Matrix $\boldsymbol{P}$

// Initialize cost matrix with terminal cost
$\boldsymbol{P}_k \leftarrow \boldsymbol{Q}_N$
// Iterate until convergence (Algebraic Riccati Equation)
while true **do**
    // Compute Gain and updated Cost Matrix
    $\boldsymbol{K} \leftarrow (\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}_k \boldsymbol{B})^{-1} \boldsymbol{B}^\top \boldsymbol{P}_k \boldsymbol{A}$
    $\boldsymbol{P} \leftarrow \boldsymbol{Q} + \boldsymbol{A}^\top \boldsymbol{P}_k (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})$
    // Check for convergence
    if $\|\boldsymbol{P}_k - \boldsymbol{P}\| < \epsilon$ **then**
        break
    // Update for next iteration
    $\boldsymbol{P}_k \leftarrow \boldsymbol{P}$
return $\boldsymbol{K}, \boldsymbol{P}$

Οι πίνακες A, B είναι σταθεροί και έχουν προκύψει από τη γραμμικοποίηση του συστήματος, όπως αυτή περιγράφηκε νωρίτερα.

2.3 Έλεγχος συστήματος με τον LQR και οπτικοποίηση

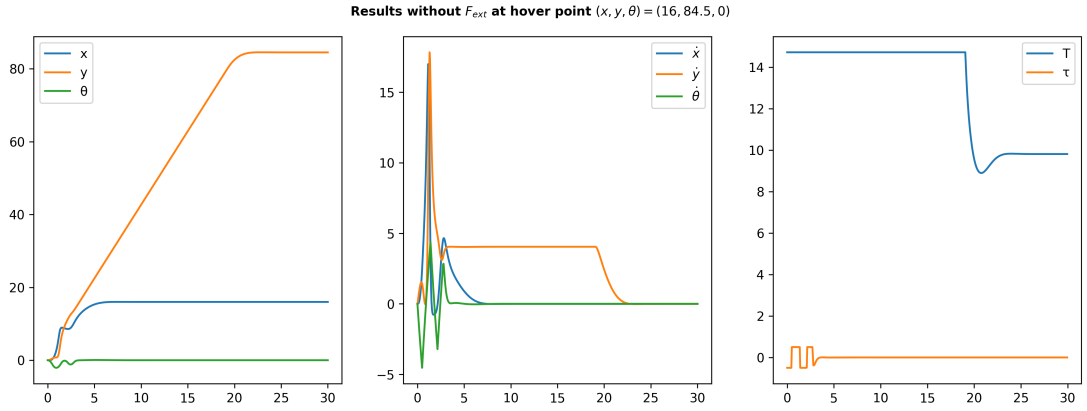
Αφού υπολογιστούν οι A, B γύρω από το $\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}$ και ο πίνακας K, ο νόμος ελέγχου γίνεται $\boldsymbol{u} = \bar{\boldsymbol{u}} - K(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}})$, όπου $\boldsymbol{x}$ το διάνυσμα της τωρινής κατάστασης, $\boldsymbol{x}_{\text{ref}}$ το επιθυμητό διάνυσμα κατάστασης και $\bar{\boldsymbol{u}}$ το διάνυσμα ελέγχου βάσει του οποίου γραμμικοποιούμε. Επιπλέον, μπορούμε να ασκήσουμε και μια εξωτερική δύναμη $\boldsymbol{F}_{ext}$ ενώ το quadrotor προσπαθεί να κάνει hover στο $\boldsymbol{x}_{\text{ref}}$, ώστε να βγάλουμε συμπεράσματα για τη συμπεριφορά του. Μέσω πειραμάτων μπορούμε να παρατηρούμε πως παρά τη διαταραχή (εντός καθορισμένων ορίων), το quadrotor φτάνει επιτυχώς στον στόχο, με timestep $dt = 0.05$!

2.4 Πόσο μακριά από το σημείο γραμμικοποίησης είναι σε θέση να ελέγξει το σύστημα ο ελεγκτής LQR;

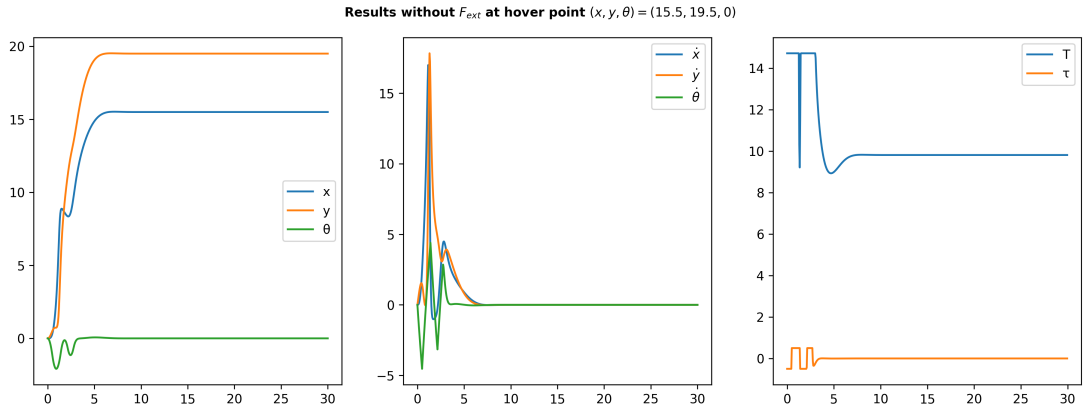
Το πιο απομακρυσμένο σημείο από το $\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}$ στο οποίο μπορεί να φτάσει το quadrotor και ο LQR ελεγκτής να συνεχίζει να μπορεί να ελέγξει το σύστημα ποικίλλει ανάλογα με την μέγιστη ταχύτητα που του επιτρέπουμε να αποκτήσει ως προς τον οριζόντιο και τον κάθετο άξονα. Μπορούμε, λοιπόν, να κάνουμε διάφορα πειράματα με βάση τα όρια ταχυτήτων και να εξάγουμε τα αντίστοιχα συμπεράσματα.

$\boldsymbol{F}_{ext} = (F_x\ F_y\ F_\theta)^\top$	$\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}$	$\lvert v_x\rvert_{max},\ \lvert v_y\rvert_{max}$	$(x, y, \theta)_{max}$
$\mathbf{0}_{3\times1}$	$\mathbf{0}_{6\times1}$, $(mg\ 0)^\top$	85	$(\pm16,\ 84.5,\ 0)$
		20	$(\pm15.5,\ 19.5,\ 0)$

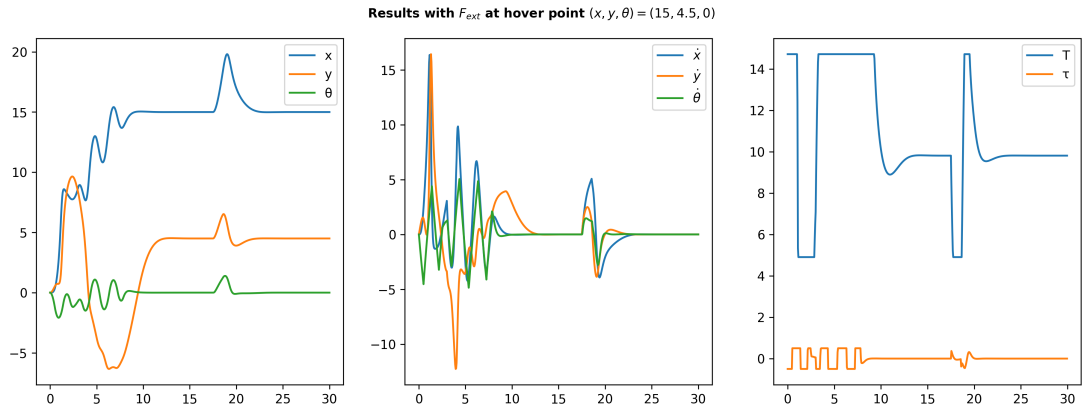
Τώρα θα ασκήσουμε δύο διαφορετικές εξωτερικές δυνάμεις στο quadrotor. Η πρώτη θα ασκηθεί πριν προλάβει να φτάσει στον στόχο $\boldsymbol{x}_{\text{ref}}$ και η δεύτερη ενώ είναι στον στόχο. Τα αποτελέσματα φαίνονται στην Εικόνα 3, από την οποία διαπιστώνουμε ότι ο LQR ελεγκτής μπορεί να επαναφέρει το quadrotor στον στόχο, με το πιο απομακρυσμένο σημείο για το οποίο λειτουργεί να είναι το $(\pm15,\ 4.5\ , 0)$, με όριο ταχυτήτων το 20. Αν με τις δυνάμεις $\boldsymbol{F}_{ext1}$ και $\boldsymbol{F}_{ext2}$ που έχουμε θεωρήσει προσπαθήσουμε να οδηγήσουμε το quadrotor πιο μακριά, ο LQR δε θα καταφέρει να βρει λύση στο πρόβλημα και συγκεκριμένα θα προσπαθήσει από την αρχή να θέσει ακαριαία πολύ μεγάλες τιμές στις δυνάμεις για να μηδενίσει το σφάλμα, με αποτέλεσμα οι ταχύτητες να απειριστούν.



Εικόνα 1. Αποτελέσματα προσομοίωσης για μηδενική εξωτερική δύναμη και όριο ταχύτητας το 85. Το quadrotor φτάνει ως το σημείο $(16,\ 84.5\ , 0)$.



Εικόνα 2. Αποτελέσματα προσομοίωσης για μηδενική εξωτερική δύναμη και όριο ταχύτητας το 20. Το quadrotor φτάνει ως το σημείο $(15.5,\ 19.5\ , 0)$.



Εικόνα 3. Αποτελέσματα προσομοίωσης για εφαρμογή εξωτερικών δυνάμεων $\mathbf{F}_{ext1} = \begin{bmatrix} -25 & -10 & 0.02 \end{bmatrix}$, $\mathbf{F}_{ext2} = \begin{bmatrix} 9 & 10 & 0.05 \end{bmatrix}$. Το quadrotor φτάνει ως το σημείο $(15,\ 4.5\ , 0)$.

3 Έλεγχος μέσω Convex Model Predictive Control

3.1 Γενικός ορισμός του Convex MPC

$$\begin{aligned} \min_{\{\boldsymbol{x}_{1:H},\, \boldsymbol{u}_{1:H-1}\}} \quad & \sum_{k=1}^{H-1} \left(\frac{1}{2}\boldsymbol{x}_k^\top \boldsymbol{Q}_k \boldsymbol{x}_k + \frac{1}{2}\boldsymbol{u}_k^\top \boldsymbol{R}_k \boldsymbol{u}_k\right) + \frac{1}{2}\boldsymbol{x}_H^\top \boldsymbol{P}_H \boldsymbol{x}_H \\ \text{s.t.} \quad & \boldsymbol{x}_{k+1} = \mathbf{A}\boldsymbol{x}_k + \mathbf{B}\boldsymbol{u}_k, \\ & \boldsymbol{u}_k \in \mathcal{U}. \end{aligned} \tag{2}$$

3.2 Υλοποίηση του αλγορίθμου

3.2.1 Η συνάρτηση `mpc_controller()`

Algorithm 2: MPC Controller

Input: Current state $\boldsymbol{x}$, linearization point $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$, reference point $(\boldsymbol{x}_{\text{ref}}, \boldsymbol{u}_{\text{ref}})$
Output: Optimal control input $\boldsymbol{u}_{\text{opt}}$

// Update dynamics error vector based on current state mismatch
$\boldsymbol{d}_{0:N} \leftarrow -\boldsymbol{A}(\boldsymbol{x} - \bar{\boldsymbol{x}})$

// Update linear cost term $\boldsymbol{h}$ for the horizon
for $k \leftarrow 0$ **to** $N_h - 2$ **do**
  $idx_u \leftarrow k(M+N) : k(M+N) + M$
  $idx_x \leftarrow k(M+N) + M : (k+1)(M+N)$
  $\boldsymbol{h}[idx_u] \leftarrow -\boldsymbol{R}(\boldsymbol{u}_{\text{ref}} - \bar{\boldsymbol{u}})$
  $\boldsymbol{h}[idx_x] \leftarrow -\boldsymbol{Q}(\boldsymbol{x}_{\text{ref}} - \bar{\boldsymbol{x}})$
$\boldsymbol{h}[-N:] \leftarrow -\boldsymbol{P}_H(\boldsymbol{x}_{\text{ref}} - \bar{\boldsymbol{x}})$

// Update and solve the QP problem
qp.update$(\boldsymbol{H}, \boldsymbol{h}, \boldsymbol{G}, \boldsymbol{d}, \boldsymbol{C}, \boldsymbol{l}, \boldsymbol{u})$
qp.solve$(\boldsymbol{x}_{\text{prev}}, \boldsymbol{y}_{\text{eq}}, \boldsymbol{z}_{\text{ineq}})$

// Extract first control input deviation and compute final control
$\delta\boldsymbol{u} \leftarrow \boldsymbol{x}_{\boldsymbol{prev}}[0 : M]$
$\boldsymbol{u}_{\text{opt}} \leftarrow \delta\boldsymbol{u} + \bar{\boldsymbol{u}}$

return $\boldsymbol{u}_{opt}$

Με βάση τη γενική σχέση (2) και με δεδομένο ότι το σημείο γραμμικοποίησης $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$ και το σημείο εκκίνησης $(\bar{\boldsymbol{x_0}}, \bar{\boldsymbol{u_0}})$ διαφέρουν, θα πρέπει να εργαστούμε με deviations. Έτσι, ορίζουμε:

$$J = \sum_{k=1}^{H-1} \left(\frac{1}{2}(\boldsymbol{x}_k - \boldsymbol{x}_{\text{ref}})^\top \boldsymbol{Q}(\boldsymbol{x}_k - \boldsymbol{x}_{\text{ref}}) + \frac{1}{2}(\boldsymbol{u}_k - \boldsymbol{u}_{\text{ref}})^\top \boldsymbol{R}_k(\boldsymbol{u}_k - \boldsymbol{u}_{\text{ref}})\right) + \frac{1}{2}(\boldsymbol{x}_H - \boldsymbol{x}_{\text{ref}})^\top \boldsymbol{P}_H(\boldsymbol{x}_H - \boldsymbol{x}_{\text{ref}}) \tag{3}$$

Θεωρώντας ότι

$\Delta\boldsymbol{x}_k = \boldsymbol{x}_k - \bar{\boldsymbol{x}} \Rightarrow \boldsymbol{x}_k = \Delta\boldsymbol{x}_k + \bar{\boldsymbol{x}}$

$\Delta\boldsymbol{u}_k = \boldsymbol{u}_k - \bar{\boldsymbol{u}} \Rightarrow \boldsymbol{u}_k = \Delta\boldsymbol{u}_k + \bar{\boldsymbol{u}}$

και αντικαθιστώντας στην (3), προκύπτει

$$
\begin{aligned}
J = \sum_{k=1}^{H-1} \big(& \frac{1}{2}(\Delta \boldsymbol{x}_k + \bar{\boldsymbol{x}} - \boldsymbol{x}_{\text{ref}})^\top \boldsymbol{Q}(\Delta \boldsymbol{x}_k + \bar{\boldsymbol{x}} - \boldsymbol{x}_{\text{ref}}) + \\
& \frac{1}{2}(\Delta \boldsymbol{u}_k + \bar{\boldsymbol{u}} - \boldsymbol{u}_{\text{ref}})^\top \boldsymbol{R}(\Delta \boldsymbol{u}_k + \bar{\boldsymbol{u}} - \boldsymbol{u}_{\text{ref}})\big) + \\
& \frac{1}{2}(\Delta \boldsymbol{x}_H + \bar{\boldsymbol{x}} - \boldsymbol{x}_{\text{ref}})^\top \boldsymbol{P}_H(\Delta \boldsymbol{x}_H + \bar{\boldsymbol{x}} - \boldsymbol{x}_{\text{ref}}) \Rightarrow \\
J = \sum_{k=1}^{H-1} \big(& \frac{1}{2}(\Delta \boldsymbol{x}_k - (\boldsymbol{x}_{\text{ref}} - \bar{\boldsymbol{x}}))^\top \boldsymbol{Q}(\Delta \boldsymbol{x}_k - (\boldsymbol{x}_{\text{ref}} - \bar{\boldsymbol{x}})) + \\
& \frac{1}{2}(\Delta \boldsymbol{u}_k - (\boldsymbol{u}_{\text{ref}} - \bar{\boldsymbol{u}}))^\top \boldsymbol{R}(\Delta \boldsymbol{u}_k - (\boldsymbol{u}_{\text{ref}} - \bar{\boldsymbol{u}}))\big) + \\
& \frac{1}{2}(\Delta \boldsymbol{x}_H - (\boldsymbol{x}_{\text{ref}} - \bar{\boldsymbol{x}}))^\top \boldsymbol{P}_H(\Delta \boldsymbol{x}_H - (\boldsymbol{x}_{\text{ref}} - \bar{\boldsymbol{x}})) \Rightarrow \\
J = \sum_{k=1}^{H-1} \big(& \frac{1}{2}(\Delta \boldsymbol{x}_k - \Delta \boldsymbol{x}_{\text{ref}})^\top \boldsymbol{Q}(\Delta \boldsymbol{x}_k - \Delta \boldsymbol{x}_{\text{ref}}) + \\
& \frac{1}{2}(\Delta \boldsymbol{u}_k - \Delta \boldsymbol{u}_{\text{ref}})^\top \boldsymbol{R}(\Delta \boldsymbol{u}_k - \Delta \boldsymbol{u}_{\text{ref}})\big) + \\
& \frac{1}{2}(\Delta \boldsymbol{x}_H - \Delta \boldsymbol{u}_{\text{ref}})^\top \boldsymbol{P}_H(\Delta \boldsymbol{x}_H - \Delta \boldsymbol{u}_{\text{ref}})
\end{aligned}
$$

Αναλύοντας τους δύο πρώτους όρους του αθροίσματος προκύπτουν

$$
\begin{aligned}
\frac{1}{2}(\Delta \boldsymbol{x}_k - \Delta \boldsymbol{x}_{\text{ref}})^\top \boldsymbol{Q}(\Delta \boldsymbol{x}_k - \Delta \boldsymbol{x}_{\text{ref}}) &= \frac{1}{2}\Delta \boldsymbol{x}_k^\top \boldsymbol{Q}\Delta \boldsymbol{x}_k - \Delta \boldsymbol{x}_{\text{ref}}^\top \boldsymbol{Q}\Delta \boldsymbol{x}_k + \frac{1}{2}\Delta \boldsymbol{x}_{\text{ref}}^\top \boldsymbol{Q}\Delta \boldsymbol{x}_{\text{ref}} \\
\frac{1}{2}(\Delta \boldsymbol{u}_k - \Delta \boldsymbol{u}_{\text{ref}})^\top \boldsymbol{R}(\Delta \boldsymbol{u}_k - \Delta \boldsymbol{u}_{\text{ref}}) &= \frac{1}{2}\Delta \boldsymbol{u}_k^\top \boldsymbol{R}\Delta \boldsymbol{u}_k - \Delta \boldsymbol{u}_{\text{ref}}^\top \boldsymbol{R}\Delta \boldsymbol{u}_k + \frac{1}{2}\Delta \boldsymbol{u}_{\text{ref}}^\top \boldsymbol{R}\Delta \boldsymbol{u}_{\text{ref}}
\end{aligned}
$$

τα οποία πορούμε να γράψουμε ως

$$
\begin{aligned}
\frac{1}{2}(\Delta \boldsymbol{x}_k - \Delta \boldsymbol{x}_{\text{ref}})^\top \boldsymbol{Q}(\Delta \boldsymbol{x}_k - \Delta \boldsymbol{x}_{\text{ref}}) &= \frac{1}{2}\Delta \boldsymbol{x}_k^\top \boldsymbol{Q}\Delta \boldsymbol{x}_k + \frac{1}{2}\Delta \boldsymbol{x}_{\text{ref}}^\top \boldsymbol{Q}\Delta \boldsymbol{x}_{\text{ref}} + (-\boldsymbol{Q}\Delta \boldsymbol{x}_{\text{ref}})^\top \Delta \boldsymbol{x}_k \\
\frac{1}{2}(\Delta \boldsymbol{u}_k - \Delta \boldsymbol{u}_{\text{ref}})^\top \boldsymbol{R}(\Delta \boldsymbol{u}_k - \Delta \boldsymbol{u}_{\text{ref}}) &= \frac{1}{2}\Delta \boldsymbol{u}_k^\top \boldsymbol{R}\Delta \boldsymbol{u}_k + \frac{1}{2}\Delta \boldsymbol{u}_{\text{ref}}^\top \boldsymbol{R}\Delta \boldsymbol{u}_{\text{ref}} + (-\boldsymbol{R}\Delta \boldsymbol{u}_{\text{ref}})^\top \Delta \boldsymbol{u}_k
\end{aligned}
$$

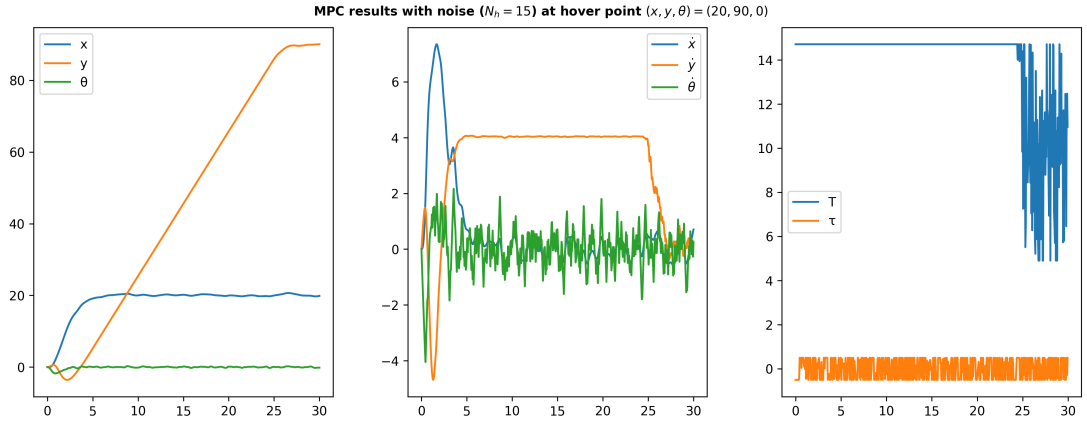
όπου οι γραμμικοί όροι $(-\boldsymbol{Q}\Delta \boldsymbol{x}_{\text{ref}})^\top$ και $(-\boldsymbol{R}\Delta \boldsymbol{u}_{\text{ref}})^\top$, όπως και ο όρος $(-\boldsymbol{P}_H\Delta \boldsymbol{x}_{\text{ref}})^\top$ καθορίζουν το διάνυσμα $\mathbf{h}$, δηλαδή το διάνυσμα σταθερού κόστους. Αντίστοιχα, ανάγοντας την δυναμική του συστήματος (2) σε διαφορές θα έχουμε $\Delta \boldsymbol{x}_{k+1} = \mathbf{A}\Delta \boldsymbol{x}_k + \mathbf{B}\Delta \boldsymbol{u}_k$ και λύνοντας το σύστημα $\mathbf{Gz} - \mathbf{d} = \mathbf{0}$, ορίζουμε το διάνυσμα $\mathbf{d}$ ως

$$
\mathbf{d} = \begin{bmatrix} -\mathbf{A}(\boldsymbol{x}_k - \bar{\mathbf{x}}) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}
$$

3.2.2 Επιλέγοντας horizon H και προσθέτοντας θόρυβο, πόσο μακριά από το σημείο γραμμικοποίησης είναι σε θέση να ελέγξει το σύστημα ο ελεγκτής MPC; Να συγκριθούν οι ελεγκτές LQP και MPC.

Για τα πειράματα επιλέγουμε $H = \Delta h = 15$ και θόρυβο με mean ίσο με μηδέν και covariance ίσο με $\begin{bmatrix}1 & 1 & 0.01 & 0.5 & 0.5 & 0.01\end{bmatrix}$. Επιπλέον, οι πίνακες Q, R, Q_H του MPC είναι ίδιοι με αυτούς του LQR για να μπορέσουμε να κάνουμε τη σύγκριση. Επίσης, διατηρούμε σταθερό τον χρόνο κατά τον οποίο εκτελείται η προσομοίωση στα 30 sec. Έτσι, το πιο απομακρυσμένο σημείο στο οποίο μπορεί να φτάσει το quadrotor και να κάνει hover είναι το $(x,\ y,\ \theta) = (20,\ 90,\ 0)$, όπως φαίνεται στην Εικόνα 4.

Από την Εικόνα 4 μπορούμε να παρατηρήσουμε τη συμπεριφορά του συστήματος με θόρυβο στις μετρήσεις και αλλάζοντας τα επίπεδά του να εξάγουμε τα αντίστοιχα συμπεράσματα. Συγκριτικά με τα αποτελέσματα που φαίνοται στην Εικόνα 1 μπορούμε να πούμε ότι ο LQR φτάνει πιο γρήγορα στο πιο απομακρυσμένο σημείο αναφοράς, το οποίο δε διαφέρει πολύ από αυτό στο οποίο μπορεί ο MPC να φτάσει στον ίδιο χρόνο. Δύο λόγοι που ο MPC καθυστερεί είναι οι εξής:



Εικόνα 4. Αποτελέσματα προσομοίωσης του MPC, με θόρυβο στις μετρήσεις του διανύσματος καταστάσεων, στο πιο απομακρυσμένο σημείο που καταφέρνει να φτάσει ($(x,\ y,\ \theta) = (20,\ 90,\ 0)$), για χρόνο προσομοίωσης 30 sec.

1. λαμβάνει υπόψη τους περιορισμούς στο διάνυσμα ελέγχου, γεγονός που τον οδηγεί στην επιλογή λύσης η οποία εξασφαλίζει την ομαλή λειτουργία του συστήματος, αλλά με πιο αργή απόκριση, ενώ ο LQR χωρίς γνώση των περιορισμών μπορεί να δίνει μεγάλες τιμές στην είσοδο $\boldsymbol{u}$ προκειμένου να μηδενίσει γρήγορα το σφάλμα, δηλαδή να φτάσει τον στόχο. Επειδή όμως εμείς "κόβουμε" τις τιμές της εισόδου με απλό clamp το σύστημα καταφέρνει να φτάσει τελικά στη λύση και μάλιστα γρήγορα. Στην περίπτωση του LQR (Εικόνα 1) η τιμή της δύναμης T, αν δε την περιορίσουμε, προσπαθεί να γίνει πάρα πολύ μεγάλη, άρα και οι ταχύτητες λαμβάνουν πολύ μεγάλες τιμές συγκριτικά με αυτές από τον MPC (Εικόνα 4). Έχει ενδιαφέρον να σημειώσουμε ότι για το ίδιο επίπεδο θορύβου οι ταχύτητες του συστήματος απειρίζονται αμέσως με αποτέλεσμα ο LQR να μην μπορεί αν δώσει τελικά λύση.

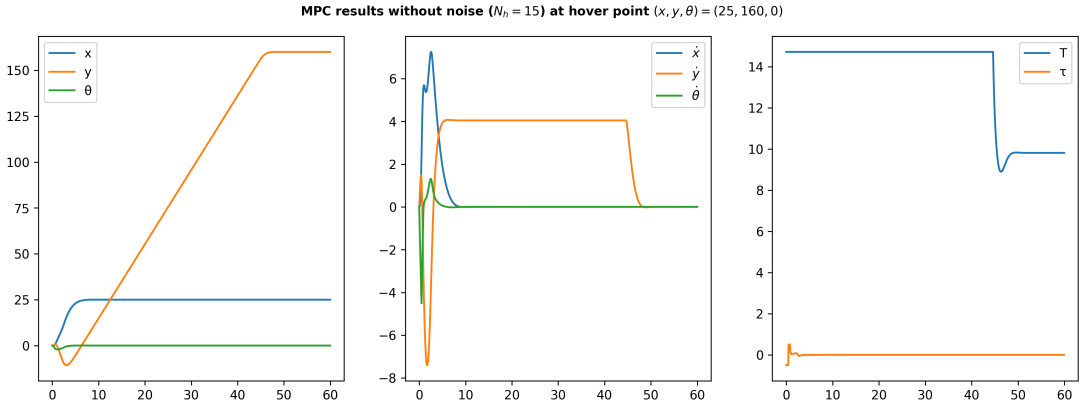
2. ο Infinite LQR που υλοποιήσαμε στην αρχή έχει άπειρο χρονικό ορίζοντα (infinite horizon) και υπολογίζει έτσι το βέλτιστο κέρδος K το οποίο χρησιμοποιεί σε κάθε βήμα. Αντίθετα, ο MPC υλοποιείται βάση του LQR πεπερασμένου χρονικού ορίζοντα (finite horizon), που σημαίνει πως υπολογίζει το βέλτιστο κέρδος για το συγκεκριμένο χρονικό παράθυρο, το οποίο δεν είναι απαραίτητα βέλτιστο και για κάθε επόμενη κατάσταση. Έτσι, μπορούν να προκύψουν υποβέλτιστες λύσεις, που όμως τον καθιστούν πιο σθεναρό, μιας και μπορεί να λειτουργεί καλά ακόμη και μακριά από το σημείο γραμμικοποίησης.

Κάνοντας διάφορες δοκιμές διαπιστώνουμε ότι ο MPC, για διπλάσιο χρόνο (60 sec) μπορεί να κάνει hover μέχρι και στο σημείο $(x,\ y,\ \theta) = (25,\ 160,\ 0)$. Προσαρμόζοντας τα x και y μπορεί να φτάσει ακόμη πιο μακριά από το σημείο γραμμικοποίησης. Ενδιαφέρον έχει ότι οι τιμές των ταχυτήτων δε ξεφεύγουν από το 7, κάτι που στον LQR δεν ισχύει. Οι τιμές των ταχυτήτων λαμβάνουν αρκετά μεγάλες τιμές για να καταφέρει να δώσει γρήγορα λύση για απομακρυσμένα σημεία.

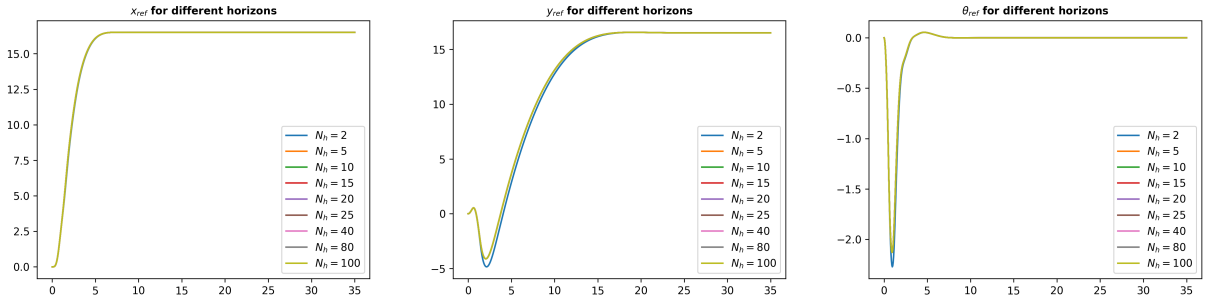
3.2.3 Πειραματιστείτε με διαφορετικά μήκη ορίζοντα. Είναι δυνατόν να εκτελέσετε εκκίνηση από οποιοδήποτε τυχαίο σημείο και προσανατολισμό; Μπορούμε να σταθεροποιηθούμε σε διαφορετικό σημείο;

Δοκιμάζοντας διάφορα μήκη ορίζοντα N_h εξάγουμε το συμπέρασμα ότι υπάρχουν συγκεκριμένα μήκη για τα οποία ο αλγόριθμος δίνει γρήγορα λύση και σχετικά μικρό κόστος. Όσο το N_h είναι μικρό (π.χ. $N_h = 2, 3, 4, 5$) το κόστος μεγαλώνει και είναι λογικό σύμφωνα με όσα εξηγήσαμε προηγουμένως στο 2. Επίσης, όταν το $N_h > 20$ ο αλγόριθμος καθυστερεί αισθητά να δώσει λύση, το κόστος της οποίας δε μειώνεται και σημαντικά. Οι βέλτιστες τιμές είναι $N_h = 10, 15, 20$. Από τα διαγράμματα της Εικόνας 6 μπορούμε να διαπιστώσουμε πως οι διαφορές δεν είναι μεγάλες, με εξαίρεση τα πιο μικρά N_h στα οποία διαφοροποιούνται λίγο οι ταχύτητες και οι δυνάμεις.

Διατηρώντας τώρα το $N_h = 15$ θα ελέγξουμε αν η αρχική κατάσταση $\boldsymbol{x}_0$ μπορεί να είναι οποιαδήποτε και όχι κοντά στο σημείο γραμμικοποίησης. Ωστόσο, πριν τις δοκιμές μπορούμε πούμε πως περιμένουμε ο αλγόριθμός, επειδή είναι γραμμικός MPC και συγκεκριμένα Convex MPC, δε μπορεί να ξεκινάει από ένα σημείο



Εικόνα 5. Αποτελέσματα προσομοίωσης του MPC, χωρίς θόρυβο, στο πιο απομακρυσμένο σημείο που καταφέρνει να φτάσει $((x,\ y,\ \theta) = (25,\ 160,\ 0))$, για χρόνο προσομοίωσης 60 sec.



Εικόνα 6. Αποτελέσματα προσομοίωσης του MPC, χωρίς θόρυβο, για διαφορετικές τιμές στο μήκος του ορίζοντα, στόχο το σημείο $(x,\ y,\ \theta) = (16,\ 16,\ 0)$, για χρόνο προσομοίωσης 35 sec.

οσοδήποτε μακριά θέλουμε, αλλά ούτε και να φτάσει πολύ πολύ μακριά από το σημείο γραμμικοποίησης, ειδικά αν απαιτούνται περίεργοι ελιγμοί για να πετύχει τον στόχο. Πράγματι, αν εκτελέσουμε δοκιμές για τυχαίες αρχικές καταστάσεις και σήματα ελέγχου $\boldsymbol{x}_0, \boldsymbol{u}_0$ το quadrotor δε μπορεί αντίστοιχα να φτάσει σε οποιονδήποτε στόχο $\boldsymbol{x}_{\text{ref}}, \boldsymbol{u}_{\text{ref}}$.

3.2.4 Ακολουθία 2D τροχιάς

Για την ακολουθία μίας απλής 2D τροχιάς δε μπορούμε να χρησιμοποιήσουμε τον MPC controller που φτιάξαμε πριν, διότι αυτός φροντίζει το σύστημα να σταθεροποιηθεί σε ένα σημείο, όχι να ακολουθήσει μια προκαθορισμένη τροχιά. Για τον σκοπό αυτό θα ορίσουμε την τροχιά βάσει των knot points, δηλαδή των σημείων εκείνων που ορίζουν τις υπο-τροχιές και μεταξύ των οποίων γίνεται η παρεμβολή με τα κυβικά πολυώνυμα. Πάμε να ορίσουμε τα κυβικά πολυώνυμα, την υπο-τροχιά μεταξύ δύο knot points, καθώς και το προφίλ ταχυτήτων του quadrotor γι' αυτή.

Οι συντελεστές για τα cubic splines δίνονται από τις εξισώσεις:

$$\begin{aligned}
\boldsymbol{c}_0 &= \boldsymbol{q}_k \\
\boldsymbol{c}_1 &= \dot{\boldsymbol{q}}_k \\
\boldsymbol{c}_2 &= \frac{3\boldsymbol{q}_{k+1}}{T_k^2} - \frac{3\boldsymbol{q}_k}{T_k^2} - \frac{2\dot{\boldsymbol{q}}_k}{T_k} - \frac{\dot{\boldsymbol{q}}_{k+1}}{T_k} \\
\boldsymbol{c}_3 &= -\frac{2\boldsymbol{q}_{k+1}}{T_k^3} + \frac{2\boldsymbol{q}_k}{T_k^3} + \frac{\dot{\boldsymbol{q}}_k}{T_k^2} + \frac{\dot{\boldsymbol{q}}_{k+1}}{T_k^2}
\end{aligned}$$

ενώ η τροχιά και το προφίλ ταχυτήτων της αντίστοιχα, δίνονται από τις σχέσεις:

$$\begin{aligned}
\boldsymbol{q}(t) &= \boldsymbol{c}_3 t^3 + \boldsymbol{c}_2 t^2 + \boldsymbol{c}_1 t + \boldsymbol{c}_0 \\
\dot{\boldsymbol{q}}(t) &= 3\boldsymbol{c}_3 t^2 + 2\boldsymbol{c}_2 t + \boldsymbol{c}_1
\end{aligned}$$

Algorithm 3: Trajectory Generation using Cubic Splines

Input: Knot points $\boldsymbol{X}_{ref} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{L-1}\}$, Durations $\boldsymbol{T} = \{T_0, T_1, \ldots, T_{L-1}\}$, dt
Output: Reference trajectory

// Initialize empty trajectory list
$\boldsymbol{ref}_{traj} \leftarrow \varnothing$

// Iterate through each segment between knot points
for $i \leftarrow 0$ **to** $L-1$ **do**
    // Define start and goal states
    $\boldsymbol{x}_s \leftarrow \boldsymbol{x}_i$
    $\boldsymbol{x}_g \leftarrow \boldsymbol{x}_{i+1}$
    $T \leftarrow \boldsymbol{T}_i$

    // Calculate cubic spline coefficients
    $\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4 \leftarrow$ calculate_coeffs$(\boldsymbol{x}_s, \boldsymbol{x}_g, \boldsymbol{T})$

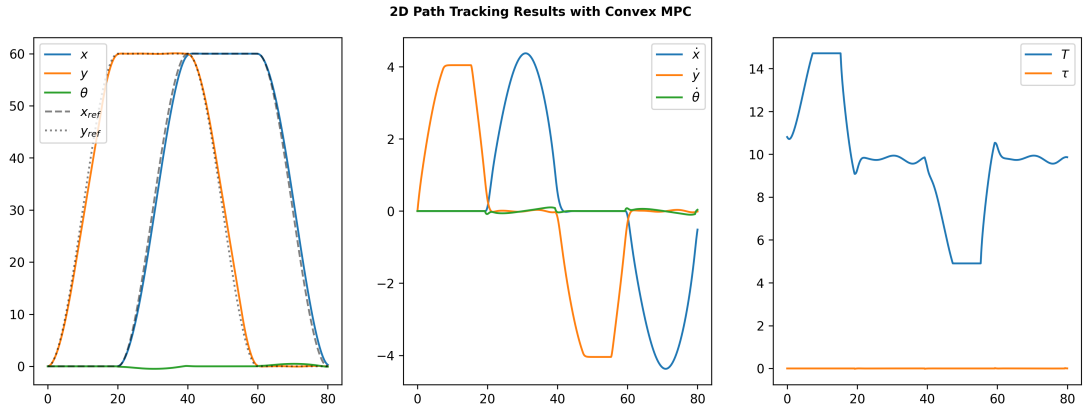
    // Generate points for current sub-trajectory
    for $k \leftarrow 0$ **to** *steps* **do**
        $t \leftarrow k \cdot dt$
        $\boldsymbol{q}(t) \leftarrow$ cubic_spline$(t, \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3)$
        $\dot{\boldsymbol{q}}(t) \leftarrow$ cubic_spline_dot$(t, \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3)$

// Padding
Append final state $\boldsymbol{x}_{L-1}$ to $\boldsymbol{x}_{traj}$ for N_h steps

return $\boldsymbol{x}_{traj}$

Τροποποίηση πρέπει να γίνει και στον MPC controller, καθώς δεν έχουμε πλέον ένα διάνυσμα καταστάσεων $\boldsymbol{x}_{\text{ref}}$, αλλά μία τροχιά αποτελούμενη από πολλά τέτοια διανύσματα. Ορίοντας, λοιπόν, τα knot points

$$\boldsymbol{x}_0 = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^\top \quad \boldsymbol{x}_1 = \begin{bmatrix}0 & 60 & 0 & 0 & 0 & 0\end{bmatrix}^\top \quad \boldsymbol{x}_2 = \begin{bmatrix}60 & 60 & 0 & 0 & 0 & 0\end{bmatrix}^\top$$
$$\boldsymbol{x}_3 = \begin{bmatrix}60 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^\top \quad \boldsymbol{x}_4 = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^\top$$



Εικόνα 7. Αποτελέσματα προσομοίωσης του MPC, χωρίς θόρυβο, για παρακολούθηση μίας τροχιάς σε σχήμα τετραγώνου.