Ρομποτικά Συστήματα ΙΙ: Εργασία 3

Ονοματεπώνυμο: Βραχωρίτη Αλεξάνδρα

Αριθμός Μητρώου: 1092793

13 Ιανουαρίου 2026

1 Μοντελοποίηση συστήματος ενός double pendulum

Στην εκφώνηση δίνεται η δυναμική του συστήματος ως:

$$f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} \tag{1}$$

και ορίζοντας $\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$ έχουμε

$$\dot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{u} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})) \tag{2}$$

όπου

$$M = \begin{bmatrix} l_1^2 m_1 + l_2^2 m_2 + l_1^2 m_2 + 2 l_1 m_2 l_2 \cos q_2 & l_2^2 m_2 + l_1 m_2 l_2 \cos q_2 \\ l_2^2 m_2 + l_1 m_2 l_2 \cos q_2 & l_2^2 m_2 \end{bmatrix} \in \mathbb{R}^{2\times 2}$$

$$C = \begin{bmatrix} -2\dot{q}_2 l_1 m_2 l_2 \sin q_2 & -\dot{q}_2 l_1 m_2 l_2 \sin q_2 \\ \dot{q}_1 l_1 m_2 l_2 \sin q_2 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}$$

$$G = \begin{bmatrix} -g m_1 l_1 \sin q_1 - g m_2 \big(l_1 \sin q_1 + l_2 \sin(q_1 + q_2)\big) \\ -g m_2 l_2 \sin(q_1 + q_2) \end{bmatrix} \in \mathbb{R}^{2\times 1}$$

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:

$\mathbf{q}_{error} = \mathbf{q}_{ref} - \mathbf{q}$

$\dot{\mathbf{q}}_{error} = \dot{\mathbf{q}}_{ref} - \dot{\mathbf{q}}$

$\ddot{\mathbf{q}} = K_p \cdot \mathbf{q}_{error} + K_d \cdot \dot{\mathbf{q}}_{error}$

Λύνοντας την εξίσωση (2) ως προς u θα έχουμε:

$\mathbf{u} = \mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})$.

Για τον έλεγχο της μοντελοποίησης του συστήματος έγινε μια προσομοίωση με τη βοήθεια του PD ελεγκτή, με αρχική κατάσταση $\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ και στόχο το $\mathbf{x}_0 = \begin{bmatrix} \pi/7 & \pi/7 & 0 & 0 \end{bmatrix}^{T}$. Τα αποτελέσματα φαίνονται στο αρχείο `1-modeling.mp4`.

2 Βελτιστοποίηση τροχιάς

Για την βελτιστοποίηση της τροχιάς του διπλού εκκρεμούς (Double Pendulum) θα χρησιμοποιηθεί η μέθοδος σημειακής προσαρμογής (collocation) μέσω του κανόνα Hermite-Simpson [1]. Αυτό σημαίνει πως η δυναμική του συστήματος ((1)) θα μετασχηματιστεί σε αλγεβρικούς περιορισμούς με τη χρήση πολυωνύμων τρίτου βαθμού (cubic polynomials), τα οποία θα πρέπει να ικανοποιούν τις εξισώσεις κίνησης σε συγκεκριμένα χρονικά σημεία (collocation points). Σε κάθε χρονικό διάστημα $[t_k, t_{k+1}]$, η κατάσταση $\mathbf{x}(t)$ προσεγγίζεται από ένα κυβικό πολυώνυμο της μορφής:

$$\mathbf{x}(t) = a_0 + a_1\, t + a_2\, t^2 + a_3\, t^3 \tag{3}$$

με παράγωγο

$$\dot{\mathbf{x}}(t) = a_1 + a_2\, t + a_3\, t^2 \tag{4}$$

Οι συντελεστές του πολυωνύμου υπολογίζονται από τις τιμές στα άκρα του διαστήματος, δηλαδή βάσει των $\begin{bmatrix}\mathbf{x}_k & \dot{\mathbf{x}}_k & \mathbf{x}_{k+1} & \dot{\mathbf{x}}_{k+1}\end{bmatrix}^T$, όπου dt είναι το χρονικό βήμα:

$\mathbf{x}_k = \mathbf{x}(0) = a_0$
$\dot{\mathbf{x}}_k = \dot{\mathbf{x}}(0) = a_1$
$\mathbf{x}_{k+1} = \mathbf{x}(dt) = a_0 + a_1\, dt + a_2\, t^2 + a_3\, t^3$
$\dot{\mathbf{x}}_{k+1} = \dot{\mathbf{x}}(dt) = a_1 + a_2\, dt + a_3\, dt^2$

Έτσι, καταλήγουμε στην εξίσωση:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{dt^2} & -\frac{2}{dt} & \frac{3}{dt^2} & -\frac{1}{dt} \\ \frac{2}{dt^3} & \frac{1}{dt^2} & -\frac{2}{dt^3} & \frac{1}{dt^2} \end{bmatrix} \begin{bmatrix} x(0) \\ \dot{x}(0) \\ x(dt) \\ \dot{x}(dt) \end{bmatrix}$$

Στο μέσο του χρονικού διαστήματος $[t_k, t_{k+1}]$, όπου $t_m = \frac{t_{k+1}-t_k}{2} = \frac{dt}{2}$, ορίζεται *collocation point*, όπου η κατάσταση $\mathbf{x}_m$ και η παράγωγός της $\dot{\mathbf{x}}_m$ προκύπτουν ως:

$$\begin{aligned}
(3) \Rightarrow \mathbf{x}(\frac{dt}{2}) = \mathbf{x}_m &= \mathbf{x}(0) + \dot{\mathbf{x}}(0)\frac{dt}{2} \\
&\quad + [-\frac{3}{dt^2}\mathbf{x}(0) - \frac{2}{dt}\dot{\mathbf{x}}(0) + \frac{3}{dt^2}\mathbf{x}(dt) - \frac{1}{dt}\dot{\mathbf{x}}(dt)](\frac{dt}{2})^2 \\
&\quad + [\frac{2}{dt^3}\mathbf{x}(0) + \frac{1}{dt^2}\dot{\mathbf{x}}(0) - \frac{2}{dt^3}\mathbf{x}(dt) + \frac{1}{dt^2}\dot{\mathbf{x}}(dt)](\frac{dt}{2})^3 \\
\Rightarrow \mathbf{x}_m &= \mathbf{x}_k + \dot{\mathbf{x}}_k\frac{dt}{2} \\
&\quad + [-\frac{3}{dt^2}\mathbf{x}_k - \frac{2}{dt}\dot{\mathbf{x}}_k + \frac{3}{dt^2}\mathbf{x}_{k+1} - \frac{1}{dt}\dot{\mathbf{x}}_{k+1}](\frac{dt}{2})^2 \\
&\quad + [\frac{2}{dt^3}\mathbf{x}_k + \frac{1}{dt^2}\dot{\mathbf{x}}_k - \frac{2}{dt^3}\mathbf{x}_{k+1} + \frac{1}{dt^2}\dot{\mathbf{x}}_{k+1}](\frac{dt}{2})^3
\end{aligned}$$

και καταλήγουμε στις εξισώσεις

$$\mathbf{x}_m = \frac{1}{2}(\mathbf{x}_k + \mathbf{x}_{k+1}) + \frac{dt}{8}[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})] \tag{5}$$

$$\dot{\mathbf{x}}_m = -\frac{3}{2dt}(\mathbf{x}_k - \mathbf{x}_{k+1}) - \frac{1}{4}[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})] \tag{6}$$

Η δυναμική του συστήματος ικανοποιείται όταν το σφάλμα ολοκλήρωσης (integration defect) $\Delta = \dot{\mathbf{x}}_m - \mathbf{f}(\mathbf{x}_m, \mathbf{u}_m)$ οδηγείται στο μηδέν. Έτσι, με τη βοήθεια της σχέσης (6) προκύπτει:

$$\Delta = -\frac{3}{2dt}(\mathbf{x}_k - \mathbf{x}_{k+1}) - \frac{1}{4}[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})] - \mathbf{f}(\mathbf{x}_m, \mathbf{u}_m) = 0$$

$$-3(\mathbf{x}_k - \mathbf{x}_{k+1}) - \frac{2dt}{4}[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})] - (2dt)\,\mathbf{f}(\mathbf{x}_m, \mathbf{u}_m) = 0$$

$$-(\mathbf{x}_k - \mathbf{x}_{k+1}) - \frac{dt}{6}[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})] - \frac{2dt}{3}\,\mathbf{f}(\mathbf{x}_m, \mathbf{u}_m) = 0$$

$$(\mathbf{x}_k - \mathbf{x}_{k+1}) + \frac{dt}{6}[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})] + \frac{4dt}{6}\,\mathbf{f}(\mathbf{x}_m, \mathbf{u}_m) = 0$$

Σύμφωνα με τα παραπάνω, οι δύο πρώτοι περιορισμοί του προβλήματός μας είναι:

$$(\mathbf{x}_k - \mathbf{x}_{k+1}) + \frac{dt}{6}[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) + 4\mathbf{f}(\mathbf{x}_m, \mathbf{u}_m)] = 0 \tag{7}$$

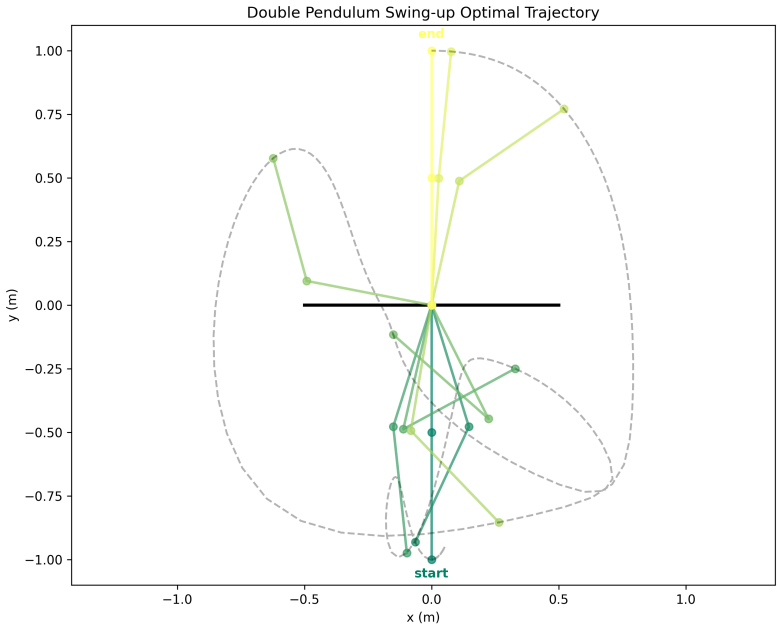
$$\mathbf{x}_m - \frac{1}{2}(\mathbf{x}_k + \mathbf{x}_{k+1}) - \frac{dt}{8}[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})] = 0 \tag{8}$$

Επιπλέον θα πρέπει να λάβουμε υπόψη δύο ακόμη περιορισμούς που θα εξασφαλίζουν την αρχική και τελική κατάσταση του συστήματος σύμφωνα με τα ζητούμενα:

$$\mathbf{x}_k = \mathbf{x}_0, \quad \texttt{για} \;\; k = 0 \tag{9}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_{final}, \quad \texttt{για} \;\; k = 2 \cdot \texttt{knot\_points} \tag{10}$$

Για την επίλυση του προβλήματος θα χρησιμοποιήσουμε τη βιβλιοθήκη `cyipopt`. Στόχος μας είναι το double pendulum να κάνει swing-up, με δεδομένο ότι $\mathbf{x}_0 = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^T$ και στόχος είναι το $\mathbf{x}_f = \begin{bmatrix}\pi & 0 & 0 & 0\end{bmatrix}^T$. Ο επιλυτής που θα χρησιμοποιήσουμε για την βελτιστοποίηση της τροχιάς θα χρησιμοποιεί τους περιορισμούς που ορίσαμε παραπάνω και επιπλέον χρειάζεται μια αρχική εκτίμηση του διανύσματος που περιλαμβάνει όλα τα διανύσματα καταστάσεων και ελέγχου, που συνολικά είναι $2 \cdot S - 1$, όπου $S = 101$ τα knot points. Για την εκτίμηση αυτή θα χρησιμοποιήσουμε τα cubic splines από το $\mathbf{x}_0$ στο $\mathbf{x}_m = \begin{bmatrix}-\pi/2 & 0 & 0 & 0\end{bmatrix}^T$ και από το $\mathbf{x}_m$ στο $\mathbf{x}_f$.



Εικόνα 2.1 Προσομοίωση της βέλτιστης τροχιάς που μπορεί να ακολουθήσει το double pendulum προκειμένου να φτάσει στον στόχο $\mathbf{x}_f$ (`end` με κίτρινο), ξεκινώντας από το $\mathbf{x}_0$ (`start` με πράσινο).

Παρακάτω παρουσιάζεται ο αλγόριθμος για την εύρεση της βέλτιστης τροχιάς.

Algorithm 1: Trajectory Optimization via Hermite-Simpson and `cyipopt` library

Input: Initial state $\boldsymbol{x}_0$, Intermediate state $\boldsymbol{x}_m$, Final state $\boldsymbol{x}_f$, Time step dt, Knot points S
Output: Optimized trajectory $\boldsymbol{z}^*$

// 1. Initial Guess Generation using Cubic Splines
$\boldsymbol{c}_{0a}, \boldsymbol{c}_{1a}, \boldsymbol{c}_{2a}, \boldsymbol{c}_{3a}, \boldsymbol{c}_{4a} \leftarrow$ calculate_coeffs$(\boldsymbol{x}_0, \boldsymbol{x}_m, \frac{2}{5}t_f)$
$\boldsymbol{c}_{0b}, \boldsymbol{c}_{1b}, \boldsymbol{c}_{2b}, \boldsymbol{c}_{3b}, \boldsymbol{c}_{4b} \leftarrow$ calculate_coeffs$(\boldsymbol{x}_m, \boldsymbol{x}_f, \frac{3}{5}t_f)$
$\boldsymbol{z}_{init} \leftarrow \varnothing$

for $k \leftarrow 0$ **to** $steps_a$ **do**
  $t \leftarrow k \cdot \frac{dt}{2}$
  $\boldsymbol{q}(t) \leftarrow$ cubic_spline$(t, \boldsymbol{c}_{0a}, \boldsymbol{c}_{1a}, \boldsymbol{c}_{2a}, \boldsymbol{c}_{3a})$
  $\dot{\boldsymbol{q}}(t) \leftarrow$ cubic_spline_dot$(t, \boldsymbol{c}_{0a}, \boldsymbol{c}_{1a}, \boldsymbol{c}_{2a}, \boldsymbol{c}_{3a})$
  $\boldsymbol{u} \leftarrow \mathbf{0}$
  Append $[\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{u}]$ to $\boldsymbol{z}_{\text{init}}$

for $k \leftarrow 0$ **to** $steps_b$ **do**
  $t \leftarrow k \cdot \frac{dt}{2}$
  $\boldsymbol{q}(t) \leftarrow$ cubic_spline$(t, \boldsymbol{c}_{0b}, \boldsymbol{c}_{1b}, \boldsymbol{c}_{2b}, \boldsymbol{c}_{3b})$
  $\dot{\boldsymbol{q}}(t) \leftarrow$ cubic_spline_dot$(t, \boldsymbol{c}_{0b}, \boldsymbol{c}_{1b}, \boldsymbol{c}_{2b}, \boldsymbol{c}_{3b})$
  $\boldsymbol{u} \leftarrow \mathbf{0}$
  Append $[\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{u}]$ to $\boldsymbol{z}_{\text{init}}$

// 2. Define Bounds
bounds $\leftarrow (\mathbf{l}, \mathbf{u})$

// 3. Compute Gradient and Jacobian
$\nabla c \leftarrow$ grad(cost_function)
$\boldsymbol{J_f} \leftarrow$ jacobian(constraints)

// 4. Nonlinear Programming (NLP) Solver
$\boldsymbol{z}^* \leftarrow$ minimize_ipopt(
  fun = cost,
  $\boldsymbol{x}_0 = \boldsymbol{z}_{init}$,
  bounds = $(\boldsymbol{l}, \boldsymbol{u})$,
  constraints = {
    fun : constraints,
    jac : $\boldsymbol{J_f}$ },
  jac = ∇c,
  options = {
    max_iter : 150,
    disp : 5,
    tol : 10^{-5}
  }
)

return $\boldsymbol{z}^*$

Το μοντέλο συγκλίνει επιτυχώς σε βέλτιστη λύση μετά από 65 επαναλήψεις, με την ακρίβεια των αποτελεσμάτων να είναι υψηλή (10^{-6}). Ο χρόνος επίλυσης επηρεάζεται κυρίως από το υπολογιστικό κόστος των συναρτήσεων του προβλήματος και την πυκνότητα του Jacobian πίνακα, γι' αυτό και χρειάστηκαν περίπου 146 sec, ενώ ο solver χρειάστηκε μόνο 110 sec.

3 Έλεγχος μέσω του Time-Varying LQR (TVLQR)

3.1 Υλοποίηση του αλγορίθμου TVLQR

Ο αλγόριθμος TVLQR αποτελεί επέκταση του κλασικού LQR για προβλήματα πεπερασμένου ορίζοντα (finite horizon). Η βασική του διαφοροποίηση έγκειται στο γεγονός ότι το σύστημα γραμμικοποιείται δυναμικά σε

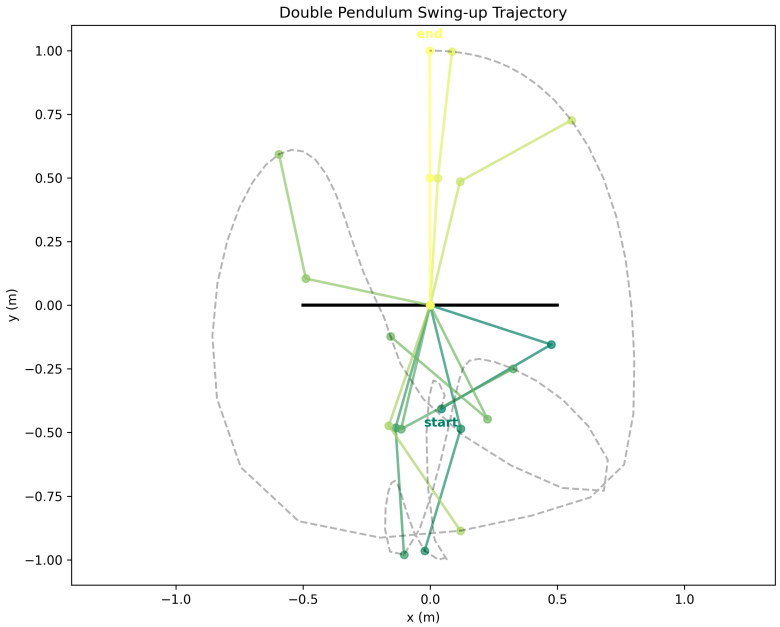
κάθε χρονικό βήμα k γύρω από την βέλτιστη τροχιά $(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)$, επιτρέποντας στον ελεγκτή να διαχειρίζεται μη γραμμικές δυναμικές.

Algorithm 2: Time-Varying LQR

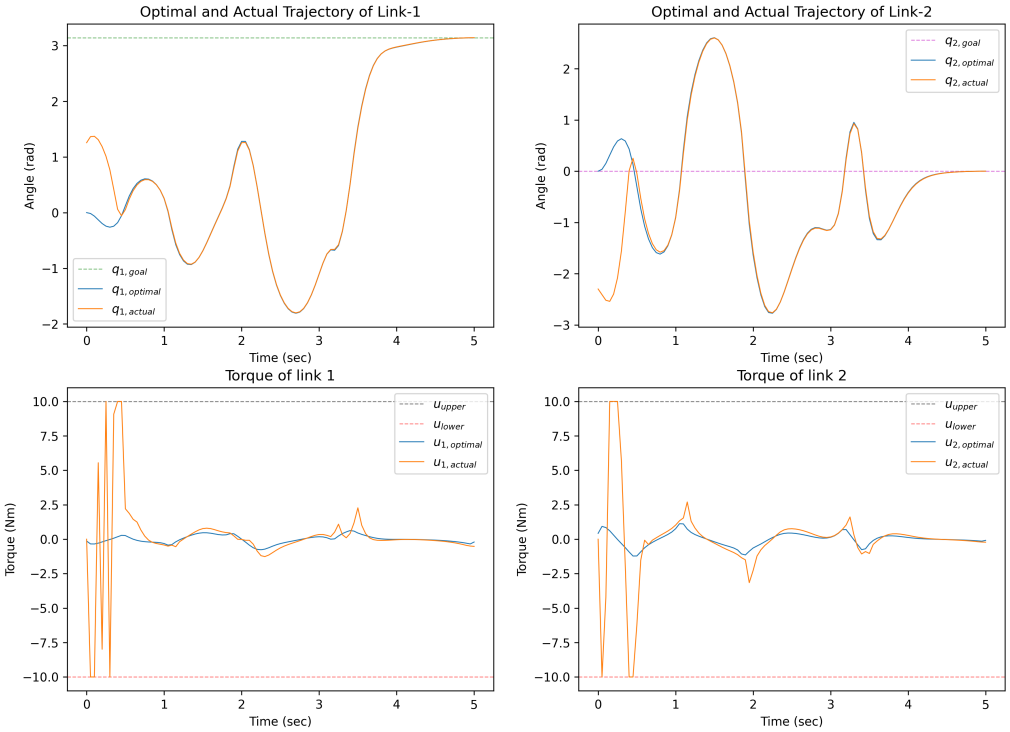
Input: Initial state $\boldsymbol{x}_0$, Reference trajectory $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$, Cost matrices $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{Q}_F$, Horizon S
Output: Predicted states $\boldsymbol{xs}$, Control sequence $\boldsymbol{us}$

// 1. Backward Pass: Riccati Recursion
$\boldsymbol{P}_{S-1} \leftarrow \boldsymbol{Q}_F$
for $k \leftarrow (S-2)$ **to** -1 **do**
  // Linearize system around $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$
  $\boldsymbol{A}_k, \boldsymbol{B}_k \leftarrow \text{linearize}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)$
  // Compute Optimal Gain and Value Function Matrix
  $\boldsymbol{K}_k \leftarrow (\boldsymbol{R} + \boldsymbol{B}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{B}_k)^{-1} \boldsymbol{B}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{A}_k$
  $\boldsymbol{P}_k \leftarrow \boldsymbol{Q} + \boldsymbol{A}_k^\top \boldsymbol{P}_{k+1} (\boldsymbol{A}_k - \boldsymbol{B}_k \boldsymbol{K}_k)$
// 2. Forward Pass: Simulation with Feedback Control
$\boldsymbol{x} \leftarrow \boldsymbol{x}_0$
for $k \leftarrow 0$ **to** $(S-1)$ **do**
  $\boldsymbol{u} \leftarrow \bar{\boldsymbol{u}}_k - \boldsymbol{K}_k (\boldsymbol{x} - \bar{\boldsymbol{x}}_k)$
  $\boldsymbol{x} \leftarrow \text{RK4}(\boldsymbol{x}, \boldsymbol{u})$
  $\boldsymbol{xs} \leftarrow \boldsymbol{x}$, $\boldsymbol{us} \leftarrow \boldsymbol{u}$
return $\boldsymbol{xs}, \boldsymbol{us}$

Η διαδικασία επίλυσης του προβλήματος περιλαμβάνει δύο στάδια: (α) το Backward Pass, όπου υπολογίζονται αναδρομικά οι πίνακες $\mathbf{P}_k$ και τα βέλτιστα κέρδη $\mathbf{K}_k$ βάσει των πινάκων κόστους $\mathbf{Q}$ και $\mathbf{R}$, και (β) το Forward Pass, όπου εφαρμόζεται ο νόμος ελέγχου ώστε να υπολογιστεί το νέο διάνυσμα κατάστασης. Ας εξετάσουμε παραδείγματα για τυχαίο αρχικό διάνυσμα κατάστασης $\mathbf{x}_0$, παρατηρώντας την τροχιά που διαγράφει κάθε άρθρωση του συστήματος.



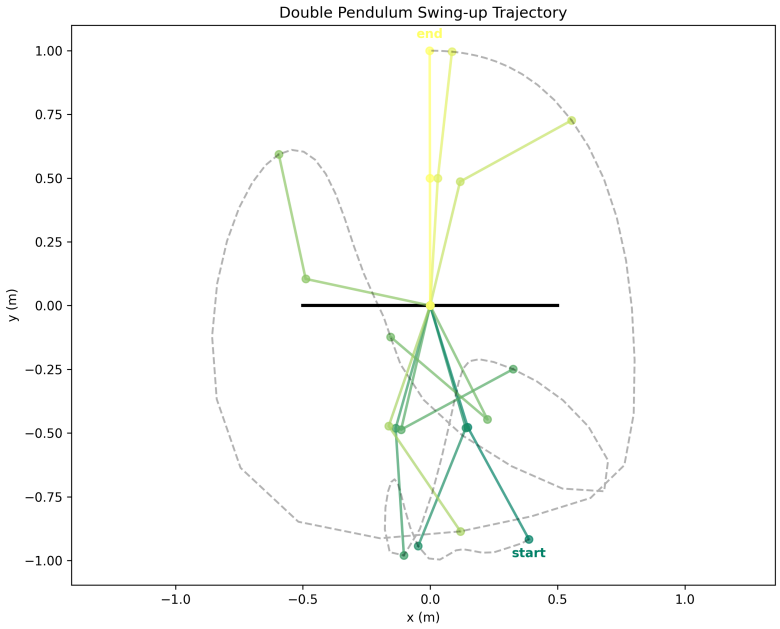
Εικόνα 3.1 Προσομοίωση της τροχιάς που διαγράφει το double pendulum για $\mathbf{x}_0 = [-\pi/2.5 \ -2.3 \ 3 \ -2]^T$.



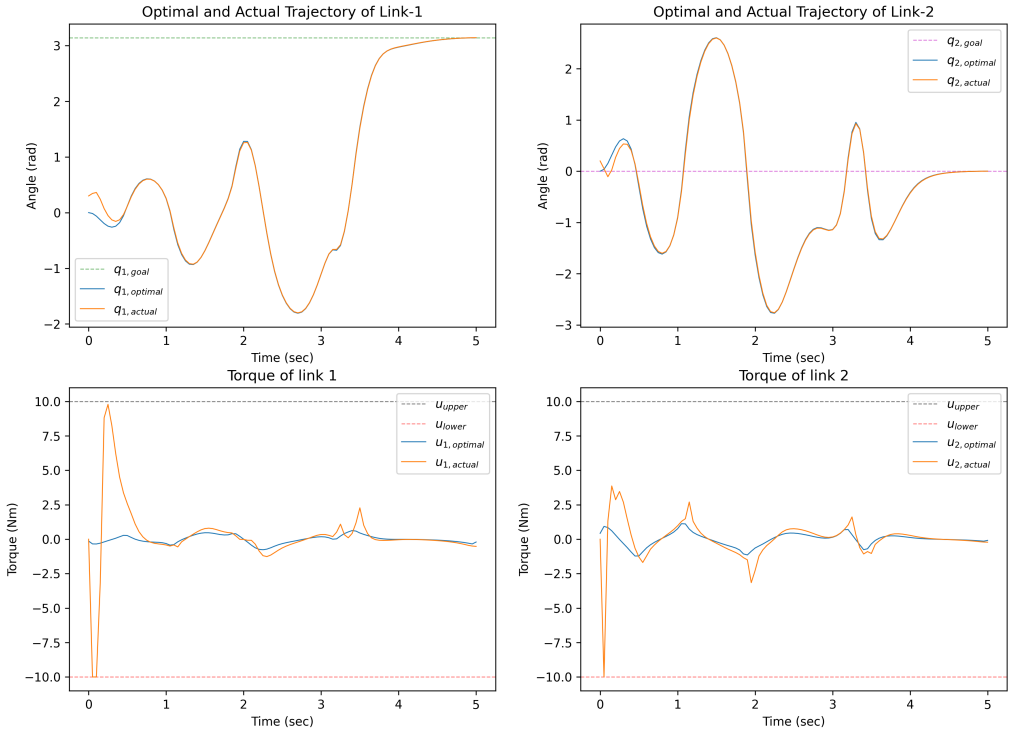
Διάγραμμα 3.1.1 Διαγράμματα της τροχιάς του ακρινού σημείου και της ροπής κάθε άρθρωσης για $\mathbf{x}_0 = [-\pi/2.5 \ \ -2.3 \ 3 \ \ -2]^T$.

Όπως μπορούμε να παρατηρήσουμε στο Διάγραμμα 3.1, λόγω της μεγάλης απόκλισης των γωνιών των αρθρώσεων από αυτές της βέλτιστης τροχιάς έχουμε μεγάλη απόκλιση $\Delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$, γι' αυτό ο αλγόριθμος βρίσκει λύση επιβάλλοντας τη μέγιστη δυνατή ροπή. Αυτό σημαίνει πως για να κάνει swing-up το double pendulum πρέπει είσοδος $\mathbf{u}$ να οδηγηθεί σε κορεσμό (saturation) προκειμένου το σφάλμα παρακολούθησης να εξαλειφθεί και τελικά το σύστημα να πετύχει τον στόχο ικανοποιώντας παράλληλα τις εξισώσεις που εκφράζουν τη δυναμική του συστήματος.

Στο παράδειγμα που ακολουθεί παρατηρούμε από το Διάγραμμα 3.2 ότι οι στην αρχή, που η απόκλιση των γωνιών είναι μεγαλύτερη, ο αλγόριθμος πάλι αναγκάζει το τους κινητήρες να λειτουργήσουν στο μέγιστο έως ότου καταφέρει να αποκτήσει την αναγκαία δύναμη και να κάνει swing-up. Ωστόσο, η προσπάθεια που καταβάλει είναι σαφώς μικρότερη συγκριτικά με πριν, όπου η το $\mathbf{x}_0$ δυσκόλευε περισσότερο τον ελεγκτή μας. Επιπλέον, παρατηρείται ότι η ταχύτητα σύγκλισης της πραγματικής τροχιάς προς τη βέλτιστη είναι αυξημένη, με το σφάλμα παρακολούθησης να μηδενίζεται πολύ γρήγορα μετά την αρχική ώθηση. Αυτό επιβεβαιώνει ότι ο TVLQR λειτουργεί πιο αποδοτικά βρισκόμαστε σε σημείο κοντά στην περιοχή γραμμικοποίησης.



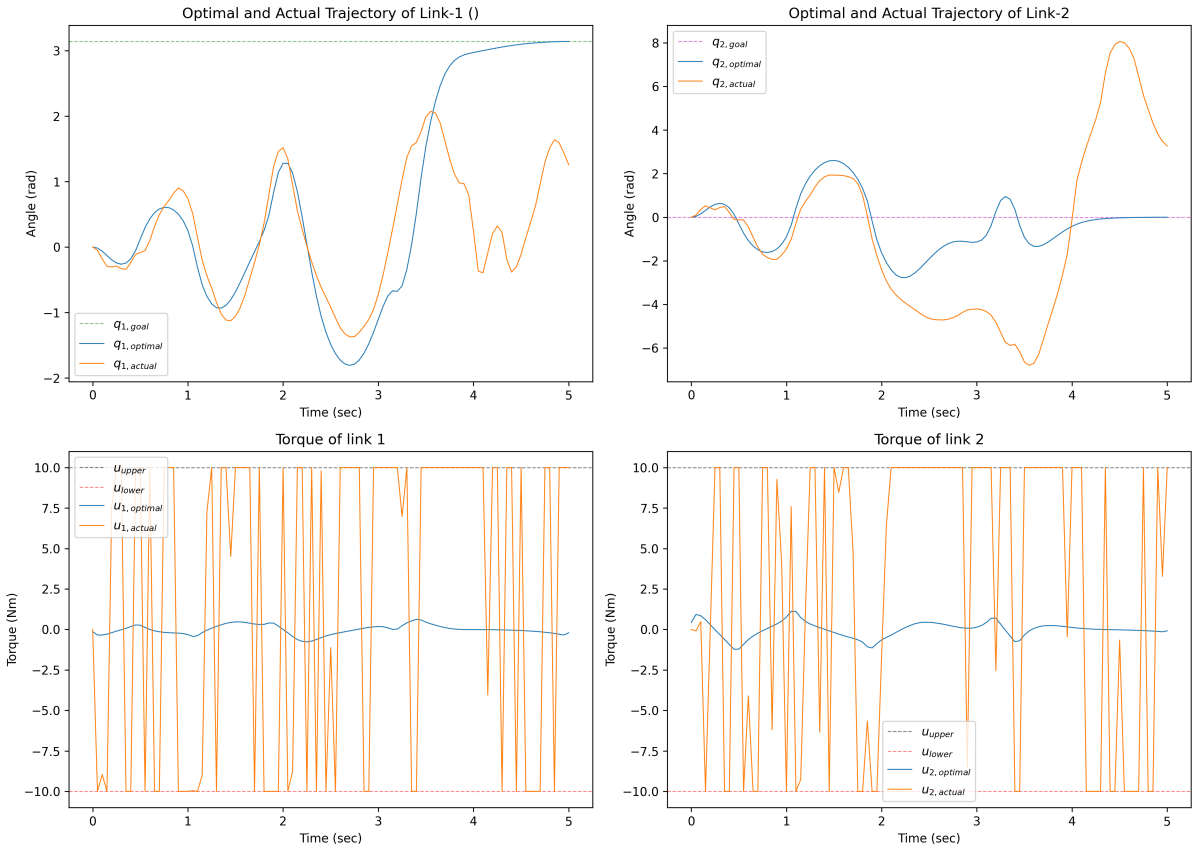
Εικόνα 3.1.2 Προσομοίωση της τροχιάς που διαγράφει το double pendulum για $\mathbf{x}_0 = [0.3\ 0.2\ 0\ 0]^T$.



Διάγραμμα 3.1.1 Διαγράμματα της τροχιάς του ακρινού σημείου και της ροπής κάθε άρθρωσης για $\mathbf{x}_0 = [0.3\ 0.2\ 0\ 0]^T$.

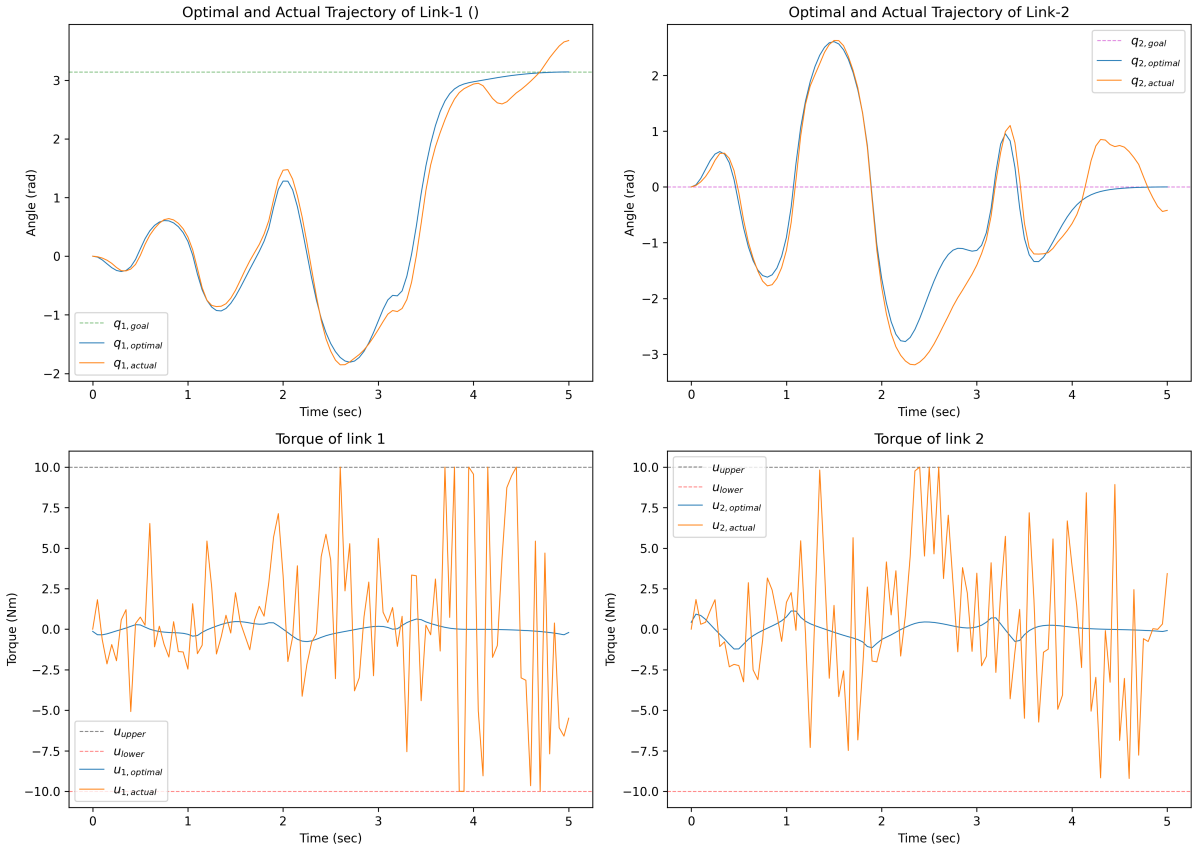
3.2 Προσθήκη θορύβου στις παρατηρήσεις (observations) του διανύσματος κατάστασης

Ας υποθέσουμε ότι $\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$. Η προσθήκη θορύβου στις παρατηρήσεις μοιάζει με την πρώτη περίπτωση που εξετάσαμε παραπάνω, μόνο που η απόκλιση επιμένει καθ' όλη τη διάρκεια της προσομοίωσης. Αυτό σημαίνει πως όσο μεγαλύτερο είναι το $\Delta\mathbf{x}$, τόσο περισσότερο ο αλγόριθμος θα δυσκολεύεται να δώσει λύση στο πρόβλημα. Για μικρά επίπεδα θορύβου δεν εμφανίζεται κάποιο ουσιαστικό πρόβλημα. Πάμε, λοιπόν, να δούμε τι συμβαίνει αν για παράδειγμα ο θόρυβος έχει covariance $\boldsymbol{\sigma}_1 = \begin{bmatrix} 0.9 & 0.9 & 1 & 1 \end{bmatrix}^T$.



Διάγραμμα 3.2.1 Διαγράμματα της τροχιάς του ακρινού σημείου και της ροπής κάθε άρθρωσης για covariance $\boldsymbol{\sigma}_1$.

Όσο μειώνουμε τον θόρυβο τα αποτελέσματα βελτιώνονται. Άλλωστε τα 0.9 rad είναι υπερβολική τιμή για θόρυβο σε γωνία, όπως και το 1 rad/sec για γωνιακή ταχύτητα. Αν για παράδειγμα έχουμε covariance $\boldsymbol{\sigma}_2 = \begin{bmatrix} 0.4 & 0.4 & 0.5 & 0.5 \end{bmatrix}^T$ η παρακολούθηση της βέλτιστης τροχιάς είναι πολύ καλύτερη. Ας παρατηρήσουμε το Διάγραμμα 3.2.2.



Διάγραμμα 3.2.2 Διαγράμματα της τροχιάς του ακρινού σημείου και της ροπής κάθε άρθρωσης για covariance $\boldsymbol{\sigma}_2$.

Τελικά, η αύξηση του θορύβου στις παρατηρήσεις επηρεάζει τον ελεγκτή με δύο τρόπους:

- εμφανίζεται μεγάλο σφάλμα παρακολούθησης (tracking error). Καθώς ο θόρυβος αυξάνεται, η πραγματική τροχιά ($\mathbf{q}_{actual}$) αποκλίνει σημαντικά από τη βέλτιστη ($\mathbf{q}_{optimal}$), με αποτέλεσμα ο ελεγκτής να λαμβάνει λανθασμένη πληροφόρηση για την κατάσταση του συστήματος και οι νόμοι ελέγχου να μην ανταποκρίνονται στην πραγματική ανάγκη που έχει το σύστημα και
- υπάρχει chattering στα $\mathbf{u}$. Οι ροπές ($\mathbf{u}_{actual}$) εμφανίζουν έντονες και γρήγορες ταλαντώσεις. Ο ελεγκτής προσπαθεί να διορθώσει τον τυχαίο θόρυβο που επιβάλουμε, αντί για το πραγματικό σφάλμα, και έτσι καταλήγουν οι εντολές ελέγχου των ενεργοποιητών να φτάνουν στον κόρο (actuator saturation).

Αναφορές

[1] F. Topputo and C. Zhang, "Survey of direct transcription for low-thrust space trajectory optimization with applications," *Abstract and Applied Analysis*, vol. 2014, no. Article ID 851720, pp. 1–15, 2014, review Article.