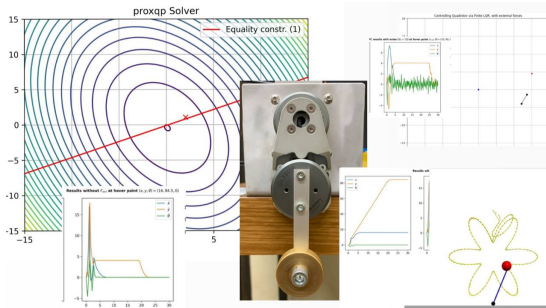


# Ρομποτικά Συστήματα II

Ανάλυση θεωρητικών γνώσεων και τεχνικών δυσκολιών κατά την εκπόνηση των εργασιών.



Αλεξάνδρα Βραχωρίτη - up1092793@ac.upatras.gr

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών,  
Πανεπιστήμιο Πατρών

# Homework 1 - Hopper Robot (1)

## Διακριτοποίηση του συστήματος

- Κανέννας υπολογιστής μπορεί να χειριστεί τον συνεχή χρόνο
- Explicit integration:  $\mathbf{z} = \mathbf{z}_t + f(\mathbf{z}) \cdot dt$ 
  - ▶ Forward Euler, Semi-Implicit Euler, Midpoint, Runge-Kutta 4th Order
- Implicit integration:  $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}$ ,  $\Delta \mathbf{x} = -(\frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}_k})^{-1} f(\mathbf{x}_k)$ 
  - ▶ Backward Euler, Trapezoidal Rule
- Για μεγάλο time step, από όλους τους explicit integrators, μόνο ο RK4 καταφέρνει να μην σπάσει, προσπαθώντας να παρακολουθήσει την τροχιά.

# Homework 1 - Hopper Robot (2)

## Μέθοδος Primal-Dual Augmented Lagrangian (1)

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{q}^T \mathbf{x}$$

$$\text{s.t. } \mathbf{A} \mathbf{x} - \mathbf{b} = \mathbf{0}$$

$$\mathbf{C} \mathbf{x} - \mathbf{d} \leq \mathbf{0}$$

- Μετατροπή του QP σε πρόβλημα χωρίς περιορισμούς με:
  - ▶ lagrange multipliers ( $\lambda, \mu$ ) και
  - ▶ quadratic penalty terms.
- Χρήση slack variables ( $s \geq 0$ )
  - ▶ μετατροπή των ανισοτικών περιορισμών σε ισοτικούς.
- Επαναληπτικός αλγόριθμος:
  - ▶ ανανέωση των μεταβλητών  $\mathbf{x}, \mathbf{s}, \lambda, \mu$  και
  - ▶ αύξηση του penalty  $\rho$  για αυστηρότερη τήρηση των περιορισμών.
- Ολοκλήρωση επαναληπτικού αλγορίθμου όταν τα stationarity και feasibility residuals πέσουν κάτω από το όριο ανοχής (π.χ.  $10^{-6}$ ).

# Homework 1 - Hopper Robot (3)

## Μέθοδος Primal-Dual Augmented Lagrangian (2)

Γιατί υπάρχουν και οι lagrange multipliers και penalty terms;

- Αν υπήρχαν μόνο τα penalty terms, θα έπρεπε  $\rho \rightarrow \infty$ , για να ικανοποιούνται οι περιορισμοί. Αυτό θα οδηγούσε σε αστάθεια του συστήματος:
  - ▶ για πολύ μεγάλο  $\rho$  ο επιλυτής θα "έβλεπε" μόνο τους περιορισμούς, αγνοώντας την αντικειμενική συνάρτηση.
- Αν υπήρχαν μόνο οι lagrange multipliers θα έπρεπε να βρεθεί το σαγματικό σημείο  $\lambda, \mu$ , το οποίο είναι αρκετά δύσκολο υπολογιστικά.

# Homework 1 - Hopper Robot (4)

## Έλεγχος του ρομπότ μέσω Quadratic Programming

- Δυναμική αρχικού συστήματος: μη-γραμμική
  - ▶ χρήση του explicit integrator RK4.
- Ορισμός της αντικειμενικής συνάρτησης:  $f(z) = \frac{1}{2}z^T Qz + q^T z$ 
  - ▶  $z = \begin{bmatrix} a & u \end{bmatrix}^T$
  - ▶  $Q = \begin{bmatrix} Q_1 & 0_2 \\ 0_2 & Q_2 \end{bmatrix}$
  - ▶  $q = \begin{bmatrix} -Q_1 a_{desired} \\ -Q_2 u_{desired} \end{bmatrix}$
- Ισοτικός περιορισμός:  $Az - b = 0$

# Homework 2 - Planar Quadrotor (1)

## Γραμμικοποίηση του συστήματος

Σύστημα: μη-γραμμικό

Ελεγκτής: Linear Quadratic Regulator (LQR)

Απαίτηση ελεγκτή: γραμμικό σύστημα  $\Delta \mathbf{x}_{k+1} = \mathbf{A}_d \Delta \mathbf{x}_k + \mathbf{B}_d \Delta \mathbf{u}_k$

- Πώς θα φτάσουμε να έχουμε ένα διακριτοποιημένο γραμμικό μοντέλο;
  - 1 Υπολογισμός των  $A_{continuous} = A_c = \frac{\partial f}{\partial \mathbf{x}_k}$ ,  $B_{continuous} = B_c = \frac{\partial f}{\partial \mathbf{u}_k}$ .
  - 2 Μετατροπή τους σε διακριτούς πίνακες  $\mathbf{A}_d$ ,  $\mathbf{B}_d$  εφαρμόζοντας τον κανόνα της αλυσίδας στα βήματα του RK4 integrator.

# Homework 2 - Planar Quadrotor (2)

## Ελεγκτής LQR (1)

- Infinite Horizon LQR:

- ▶ Στόχος: σταθεροποίηση (hover) στο  $(\mathbf{x}_{ref}, \mathbf{u}_{ref})$ .
- ▶ Βασική προϋπόθεση: γραμμικοποιημένο σύστημα γύρω από το  $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ .
- ▶ Μέθοδος: επαναληπτική επίλυση της εξίσωσης Riccati μέχρι τη σύγκλιση του πίνακα  $P$ , δηλαδή  $\|\mathbf{P} - \mathbf{P}_k\| < \epsilon$ .
- ▶ Αποτέλεσμα: **σταθερό** βέλτιστο κέρδος  $K$  για κάθε βήμα.

# Homework 2 - Planar Quadrotor (3)

## Ελεγκτής LQR (2)

- **Finite Horizon LQR:**

- ▶ Στόχος: σταθεροποίηση, σε καθορισμένο χρονικό παράθυρο.
- ▶ Βασική προϋπόθεση: γραμμικοποιημένο σύστημα γύρω από το  $(\bar{x}, \bar{u})$ .
- ▶ Μέθοδος: αναδρομική επίλυση της εξίσωσης Riccati με Backward Pass, από το τελικό βήμα  $N$  έως το 0.
- ▶ Αποτέλεσμα: **πίνακας** βέλτιστων κερδών  $K_s$ , ένα για κάθε χρονική στιγμή.

# Homework 2 - Planar Quadrotor (4)

## Σύγκριση των δύο ελεγκών

Πότε επιλέγω Infinite και πότε Finite Horizon LQR;

- **Infinite Horizon:** Επιλέγεται για ρύθμιση γύρω από ένα σταθερό σημείο ισορροπίας, καθώς το υπολογιστικό του κόστος είναι μικρότερο (σταθερό  $K$ ).
- **Finite Horizon:** Επιλέγεται σε γραμμικά, χρονικά εξαρτημένα συστήματα, όπως εδώ που πρέπει να γίνει trajectory tracking, άρα το  $x_{ref}$  αλλάζει συνεχώς.

# Homework 2 - Planar Quadrotor (5)

## LQR ως Quadratic Programming Πρόβλημα (1)

- Διάνυσμα Μεταβλητών:  $z = [u_1, x_2, u_2, \dots, x_K]^T$
- Αντικειμενική Συνάρτηση:  $\min_z \mathcal{J}(z) = \frac{1}{2} z^T H z$
- Περιορισμός:  $Gz - d = 0$
- Λύση: είναι ακριβής και προκύπτει από το γραμμικό ΚΚΤ σύστημα σε ένα μόνο βήμα:

$$\begin{bmatrix} H & G^T \\ G & 0 \end{bmatrix} \begin{bmatrix} z \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ d \end{bmatrix}$$

# Homework 2 - Planar Quadrotor (6)

## LQR ως Quadratic Programming Πρόβλημα (2)

Γιατί είναι χρήσιμη η παραπάνω μορφή;

- Το Planar Quadrotor έχει control limits.
- Ο απλός Infinite LQR δεν λαμβάνει υπόψη αυτά τα όρια.

Χρησιμοποιώντας την παραπάνω μορφή, λύνουμε Finite Horizon LQR ως QP με ρητούς περιορισμούς στα control inputs. Η διαδικασία ονομάζεται Model Predictive Control (MPC) και αναλύεται παρακάτω.

# Homework 2 - Planar Quadrotor (7)

## Convex Model Predictive Control (MPC) (1)

$$\begin{aligned} \min_{\mathbf{x}_{1:H}, \mathbf{u}_{1:H}} \quad & \sum_{k=1}^{H-1} \left[ \frac{1}{2} (\mathbf{x}_k - \mathbf{x}_{\text{ref}})^\top \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_{\text{ref}}) + \frac{1}{2} (\mathbf{u}_k - \mathbf{u}_{\text{ref}})^\top \mathbf{R}_k (\mathbf{u}_k - \mathbf{u}_{\text{ref}}) \right] \\ & + \frac{1}{2} (\mathbf{x}_H - \mathbf{x}_{\text{ref}})^\top \mathbf{P}_H (\mathbf{x}_H - \mathbf{x}_{\text{ref}}) \\ & \mathbf{x}_{k+1} = \mathbf{A} \mathbf{x}_k + \mathbf{B} \mathbf{u}_k, \quad k = 1, \dots, H-1 \\ & \mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max}, \quad k = 1, \dots, H-1 \end{aligned}$$

### • Μέθοδος:

- ▶ χρήση του  $\mathbf{P}_{\text{inf}}$  από τον Finite Horizon LQR, ως terminal cost και
- ▶ επίλυση ενός QP προβλήματος σε κάθε χρονικό βήμα, για ένα πεπερασμένο ορίζοντα  $H$ .

### • Που βασίζεται;

- ▶ στα deviations  $\Delta \mathbf{x}_k$ ,  $\Delta \mathbf{u}_k$ ,  $\Delta \mathbf{x}_{\text{ref}}$ ,  $\Delta \mathbf{u}_{\text{ref}}$  από το σημείο γύρω από το οποίο έχει γίνει γραμμικοποίηση.

# Homework 2 - Planar Quadrotor (8)

## Convex Model Predictive Control (MPC) (2)

Πιο αναλυτικά, έχουμε:

- Συνολική διάρκεια επίλυσης:  $t_f = 60 \text{ sec}$
- Χρονικό βήμα:  $dt = 0.05 \text{ sec}$
- Σύνολο βημάτων:  $S = \frac{t_f}{dt} = 1200$
- Χρονικό παράθυρο ορίζοντα:  $t_c = 0.75 \text{ sec}$
- Σύνολο βημάτων χρονικού παραθύρου:  $N_h = \frac{t_c}{dt} = 15$

Ο κώδικας εκτελεί 1200 επαναλήψεις με ένα κυλιόμενο παράθυρο 15 βημάτων. Σε κάθε επανάληψη επιλύεται ένα πρόβλημα βελτιστοποίησης για τα επόμενα 15 βήματα, εφαρμόζεται μόνο η πρώτη εντολή ελέγχου και ο ορίζοντας μετατοπίζεται (sliding window).

# Homework 2 - Planar Quadrotor (9)

Γιατί προβλέπουμε πολλαπλά βήματα στον MPC;

- Ο MPC δεν χρησιμοποιεί όλα τα control inputs που υπολόγισε, αλλά τα χρειάζεται για τη βελτιστοποίηση του πρώτου  $u$ .
- Η πρόβλεψη σε πολλαπλά βήματα επιτρέπει στον αλγόριθμο να λαμβάνει υπόψη τη μελλοντική συμπεριφορά του συστήματος και να αποφεύγει βραχυπρόθεσμες (myopic) αποφάσεις.
- Οι διαταραχές και τα σφάλματα μοντέλου αντιμετωπίζονται μέσω της επαναληπτικής επίλυσης του προβλήματος σε κάθε χρονικό βήμα (receding horizon).

# Homework 2 - Planar Quadrotor (10)

## Συμπεράσματα

- **Απόκριση συστήματος:**

- ▶ Ο MPC εμφανίζει **πιο αργή απόκριση** σε σχέση με τον LQR, λόγω των περιορισμών στα  $u$ .
- ▶ Ο LQR δίνει ακαριαία μεγάλες τιμές εισόδου για να μηδενίσει το σφάλμα.

- **Robustness:** Ο MPC αποδεικνύεται **πιο σθεναρός**, ιδιαίτερα σε σημεία μακριά από το  $(\bar{x}, \bar{u})$ , έχοντας αποδεκτές τιμές στις ταχύτητες και τις ροπές.

- **Επίδραση μεγέθους Horizon:**

- ▶ **Μικρό  $N_h$  ( $2 - 5$ ):** "μυωπική" συμπεριφορά.
- ▶ **Μεγάλο  $N_h$  ( $> 20$ ):** σημαντική καθυστέρηση χωρίς το κόστος να μειώνεται ιδιαίτερα.
- ▶ **Βέλτιστες τιμές ( $N_h = 10 - 20$ ):** δίνουν την καλύτερη δυνατή απόδοση.

# Homework 2 - Planar Quadrotor (11)

## Ακολουθία 2D τροχιάς

Για 2D τροχιά:

- 1 επιλογή  $N$  knot points,
- 2 χρήση **cubic splines** μεταξύ των knot points, για δημιουργία ομαλών sub-paths και κατ' επέκταση ταχυτήτων και επιταχύνσεων,
- 3 ορισμός χρονικής διάρκειας για κάθε τμήμα spline, ώστε να καθορίζεται η ταχύτητα του drone από το ένα knot point στο επόμενο.

# Homework 3 - Double Pendulum (1)

## Βελτιστοποίηση Τροχιάς - Κανόνας Hermite-Simpson

Αποτελεί

- μέθοδο σημειακής προσαρμογής (collocation)

και δουλεύει ως εξής:

- τα dynamics μετατρέπονται σε αλγεβρικούς περιορισμούς,
- οι περιορισμοί είναι πολυώνυμα 3ου βαθμού και
- τα πολυώνυμα πρέπει να ικανοποιούν τις εξισώσεις κίνησης στα collocation points.

Γιατί Hermite-Simpson και όχι Trapezoidal Rule;

- **Hermite-Simpson Rule:** χρησιμοποιεί κυβικά πολυώνυμα, μεγαλύτερη ακρίβεια με λιγότερα collocation points.
- **Trapezoidal Rule:** χρησιμοποιεί γραμμική παρεμβολή μεταξύ των σημείων, μικρότερη ακρίβεια με περισσότερα collocation points.

# Homework 3 - Double Pendulum (2)

## Βελτιστοποίηση Τροχιάς - Warm Start με Cubic Splines

- **Warm Start:** παρέχει μια καλή αρχική εκτίμηση για το optimizer, μειώνοντας τον αριθμό επαναλήψεων.
- **Υλοποίηση:**
  - ▶ χρήση cubic splines για την ένωση των states:  $\mathbf{x}_0 \rightarrow \mathbf{x}_m \rightarrow \mathbf{x}_f$ ,
  - ▶ υπολογισμός των συντελεστών  $c_0, c_1, c_2, c_3$  για κάθε τμήμα και
  - ▶ ο αλγόριθμος επιστρέφει τα  $(\mathbf{q}, \dot{\mathbf{q}})$  σε κάθε collocation point.
- **Αποτέλεσμα:** η αρχική εκτίμηση δημιουργεί μια ομαλή τροχιά που ο optimizer μπορεί να βελτιώσει εύκολα, αντί να ξεκινά από τυχαίες τιμές.

# Homework 3 - Double Pendulum (3)

## Time-Varying LQR (TVLQR)

Ο TVLQR σταθεροποιεί το σύστημα γύρω από τη βέλτιστη τροχιά  $z^* = (\bar{x}, \bar{u})$  που προέκυψε από το Trajectory Optimization.

- 1 **Γραμμικοποίηση:** σε κάθε βήμα  $k$ , υπολογίζονται οι πίνακες  $A_k = \left. \frac{\partial f}{\partial x} \right|_{\bar{x}_k, \bar{u}_k}$ ,  $B_k = \left. \frac{\partial f}{\partial u} \right|_{\bar{x}_k, \bar{u}_k}$ .
- 2 **Riccati Recursion (Backward Pass):** προσδιορισμός πίνακα βέλτιστων κερδών  $K_S$ .
- 3 **Νόμος Ελέγχου:**  $u_k = \bar{u}_k - K_k(x_k - \bar{x}_k)$ .

Πώς συμπεριφέρεται το σύστημα όταν υπάρχει θόρυβος;

Ο TVLQR μπορεί να αντισταθμίσει μικρές αποκλίσεις από τη βέλτιστη τροχιά, ακόμη και αν οι μετρήσεις έχουν θόρυβο, επαναφέροντας το σύστημα. Γι' αυτό πρέπει οι αποκλίσεις να παραμένουν μικρές, εντός της περιοχής γραμμικοποίησης.

# Homework 3 - Double Pendulum (4)

## Time-Varying LQR (TVLQR)

Τι μπορούμε να κάνουμε αν οι αποκλίσεις από το nominal trajectory είναι μεγάλες;

- 1 Υπολογίζουμε νέα γραμμικοποίηση γύρω από το τρέχον state και εφαρμόζουμε ξανά TVLQR.
- 2 Χρησιμοποιούμε MPC.
- 3 Δυναμική προσαρμογή των κερδών  $K_s$  ανάλογα με το τρέχον state.

Σας ευχαριστώ για την προσοχή σας!

Ερωτήσεις;