CS159 Lecture 2: Optimal Control

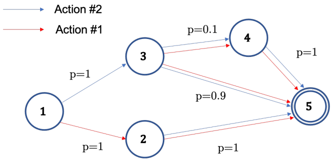
Ugo Rosolia

Caltech

Spring 2021

Adapted from Berkeley ME231
Original slide set by F. Borrelli, M. Morari, C. Jones

Summary of Last Lecture



- We discussed how to solve optimal control problem with discrete state and action spaces of the form

$$\pi^* = \arg\min_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t) | \pi \right].$$

- The solution can be computed exactly given a known model and state-action spaces of moderate size.
- Approximate dynamic programming can be used to reduce the computational complexity of syntehsis strategies.

Next Three Lectures

We will focus on Model Predictive Control (MPC) design. Our goals will be:

Next Three Lectures

We will focus on Model Predictive Control (MPC) design. Our goals will be:

- Theoretical analysis of safety and stability properties

Next Three Lectures

We will focus on Model Predictive Control (MPC) design. Our goals will be:

- Theoretical analysis of safety and stability properties
- Learn what makes control problems hard

Next Three Lectures

We will focus on Model Predictive Control (MPC) design. Our goals will be:

- Theoretical analysis of safety and stability properties
- Learn what makes control problems hard
- Get familiar with Python toolboxes for control synthesis (via HW problems)

Today's Class: Optimal Control Problem with Continuous State Spaces

Goal: Compute a control policy mapping continuous states to continuous control actions

$$\pi : \mathbb{R}^n \rightarrow \mathbb{R}^d.$$

We will consider different cases

- Linear Quadratic Regulator
- Constrained Linear Quadratic Regulator
- General control problem with nonlinear dynamics

Key Message: For problem with continuous state-action spaces computing an optimal trajectory is "easy", but computing a policy is hard when we have constraints !

Today's Class: Deterministic Problems

Computing the expected cost for problems with continuous state-action spaces is hard!

- Consider an MDP with where the state $x \in \mathcal{X} \subseteq \mathbb{R}^n$, the input $u \in \mathbb{R}^d$ and $x' \sim p(x'|x,u)$. Then for the function $V : \mathbb{R}^n \to \mathbb{R}$ we have that

$$\mathbb{E}[V(x')|x,u] = \int_{\mathcal{X}} V(x')p(x'|x,u)dx'$$

There are several methodologies to handle expected cost in continuous settings. These strategies build on the ideas that we will present in this class.

Today's Class: Deterministic Problems

Computing the expected cost for problems with continuous state-action spaces is hard!

- Consider an MDP with where the state $x \in \mathcal{X} \subseteq \mathbb{R}^n$, the input $u \in \mathbb{R}^d$ and $x' \sim p(x'|x, u)$. Then for the function $V : \mathbb{R}^n \to \mathbb{R}$ we have that

$$\mathbb{E}[V(x')|x, u] = \int_{\mathcal{X}} V(x')p(x'|x, u)dx'$$

- Consider an MDP with where the state $s \in \mathcal{S} = \{1, 2, \ldots\}$, the action $a \in \mathcal{A} = \{1, 2, \ldots\}$ and $s' \sim p(s'|s, a)$. Then for the function $V : \mathcal{S} \to \mathbb{R}$ we have that

$$\mathbb{E}[V(s')|s, a] = \sum_{\mathcal{S}} V(s')p(s'|s, a)$$

There are several methodologies to handle expected cost in continuous settings. These strategies build on the ideas that we will present in this class.

Table of Contents

Problem Formulation

Consider the discrete time system

$$x(k+1) = f(x(k), u(k))$$

where the state $x(k) \in \mathbb{R}^n$ and the input $u(k) \in \mathbb{R}^d$. The above system is subject to the constraints

$$g(x(k), u(k)) \leq 0, \forall k \geq 0. \tag{1}$$

Problem Formulation

Consider the discrete time system

$$x(k+1) = f(x(k), u(k))$$

where the state $x(k) \in \mathbb{R}^n$ and the input $u(k) \in \mathbb{R}^d$. The above system is subject to the constraints

$$g(x(k), u(k)) \leq 0, \forall k \geq 0. \tag{1}$$

Goal

Our goal is to find a control policy $\pi : \mathbb{R}^n \times \mathbb{N}_+ \to \mathbb{R}^d$ that maps states to controls, i.e., $u(k) = \pi(x(k), k)$. Furthermore, the policy π should guarantee that:

- State and input constraints (1) are satisfied.
- A used-defined cost function is minimized.

Optimal Control – Preliminaries

- In discrete-time optimal control problems the goal is to find a sequence of *predicted controls* $\boldsymbol{u}_{0\to N} = \{u_0, \ldots, u_{N-1}\}$ that for an initial conditions $x(0)$ minimizes a cost function while satisfying state and input constraints.
- The sequence of *predicted states* $\boldsymbol{x}_{0\to N} = \{x_0, \ldots, x_N\}$ is by

$$\begin{aligned} x_{k+1} &= f(x_k, u_k), \forall k \in \{0, \ldots, N-1\}, \\ x_0 &= x(0). \end{aligned}$$

The predicted states and inputs must satisfy

$$g(x_k, u_k) \le 0, \forall k \in \{0, \ldots, N-1\}$$

Optimal Control – Preliminaries

- The sequences of predicted states and inputs should minimize the following cost function:

$$\sum_{k=0}^{N-1} h(x_k, u_k) + q(x_N)$$

- Oftentimes it is required that the terminal predicted state x_N lays in a terminal set

$$x_N \in \mathcal{X}_N \subset \mathbb{R}^n.$$

Optimal Control – Problem Formulation

Consider the following Constrained Finite Time Optimal Control Problem (CFTOCP):

$$
\begin{aligned}
J^*_{0\to N}(x(0)) = \min_{\boldsymbol{u}_{0\to N}} \quad & \sum_{k=0}^{N-1} h(x_k, u_k) + q(x_N) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k), \forall k \in \{0, \dots, N-1\}, \\
& g(x_k, u_k) \leq 0, \forall k \in \{0, \dots, N-1\}, \\
& x_N \in \mathcal{X}_F, x_0 = x(0).
\end{aligned}
$$

- the optimal cost $J^*_{0\to N}(x_0)$ is the *value function* of the problem.
- the optimal sequence of actions is denoted as $\boldsymbol{u}^*_{0\to N} = \{u^*_0, \dots, u^*_{N-1}\}$.

Table of Contents

Linear Quadratic Optimal Control

We consider *linear* discrete-time time-invariant systems

$$x(k+1) = Ax(k) + Bu(k)$$

and *quadratic* cost functions

$$J_0(x_0, U_0) \triangleq x_N' P x_N + \sum_{k=0}^{N-1} [x_k' Q x_k + u_k' R u_k]. \qquad (2)$$

In this settings states and inputs are unconstrained.

We will solve the above LQR problem with following approaches:

1. *Batch Approach*, which yields a series of *numerical values* for the input
2. *Recursive Approach*, which uses Dynamic Programming to compute control *policies* or *laws*.

Unconstrained Finite Horizon Control Problem

Goal: Find a vector of inputs $U_0 = [u_0^\top, \dots, u_{N-1}^\top]^\top$ that minimizes the objective function

$$\begin{aligned} J_0^*(x(0)) \triangleq \min_{U_0} \quad & x_N' P x_N + \sum_{k=0}^{N-1} [x_k' Q x_k + u_k' R u_k] \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k,\ k = 0, \dots, N-1 \\ & x_0 = x(0) \end{aligned}$$

- $P \succeq 0$, with $P = P'$, is the *terminal* weight
- $Q \succeq 0$, with $Q = Q'$, is the *state* weight
- $R \succ 0$, with $R = R'$, is the *input* weight
- N is the horizon length
- Note that $x(0)$ is the current state, whereas $x_0, \dots, x_N$ and $u_0, \dots, u_{N-1}$ are *optimization variables* that are constrained to obey the system dynamics and the initial condition.

Table of Contents

Solution approach 1: Batch Approach (1/3)

Solution approach 1: Batch Approach (2/3)

We wrote the states sequence as $X_0 \triangleq \mathcal{S}^x x(0) + \mathcal{S}^u U_0$.
Now define Define

$$\overline{Q} \triangleq \text{blockdiag}(Q, \ldots, Q, P) \quad \text{and} \quad \overline{R} \triangleq \text{blockdiag}(R, \ldots, R)$$

Solution approach 1: Batch Approach (3/3)

We wrote the cost as

$$J_0(x(0), U_0) = U_0' H U_0 + 2x(0)' F U_0 + x(0)' \mathcal{S}^{x'} \overline{Q} \mathcal{S}^x x(0)$$

Final Result

- The problem is unconstrained
- Setting the gradient to zero:

$$U_0^*(x(0)) = \mathrm{K}x(0)$$

which implies

$$u_0^*(x(0)) = K_0 x(0), \ldots, u_{N-1}^*(x(0)) = K_{N-1} x(0)$$

which is a linear, open-loop controller function of the initial state $x(0)$.

Table of Contents

Recall: Finite Horizon LQR

- **Goal:** Find a sequence of inputs $U_0 \triangleq [u_0', \ldots, u_{N-1}']'$ that minimizes the objective function

$$\begin{aligned} J_0^*(x(0)) \triangleq \min_{U_0} \quad & x_N' P x_N + \sum_{k=0}^{N-1} [x_k' Q x_k + u_k' R u_k] \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k,\ k = 0, \ldots, N-1 \\ & x_0 = x(0) \end{aligned}$$

- $P \succeq 0$, with $P = P'$, is the *terminal* weight
- $Q \succeq 0$, with $Q = Q'$, is the *state* weight
- $R \succ 0$, with $R = R'$, is the *input* weight
- N is the horizon length
- Note that $x(0)$ is the current state, whereas $x_0, \ldots, x_N$ and $u_0, \ldots, u_{N-1}$ are *optimization variables* that are constrained to obey the system dynamics and the initial condition.

LQR – The Dynamic Programming Approach

Principle of Optimality

Let $\boldsymbol{x}^*_{0\to N} = \{x^*_0 = x_0, \ldots, x^*_N\}$ and $\boldsymbol{u}^*_{0\to N} = \{u^*_0, \ldots, u^*_{N-1}\}$ be the optimal sequences of state and input for the FTOCP $J^*_{0\to N}(x_0)$. Then we have that the sequences $\{x^*_k, \ldots, x^*_N\}$ and $\{u^*_k, \ldots, u^*_{N-1}\}$ are optimal for the FTOCP $J^*_{k\to N}(x^*_k), \forall k \in \{0, \ldots, N-1\}$.

LQR – The Dynamic Programming Approach

Principle of Optimality

Let $\boldsymbol{x}^*_{0\to N} = \{x^*_0 = x_0, \ldots, x^*_N\}$ and $\boldsymbol{u}^*_{0\to N} = \{u^*_0, \ldots, u^*_{N-1}\}$ be the optimal sequences of state and input for the FTOCP $J^*_{0\to N}(x_0)$. Then we have that the sequences $\{x^*_k, \ldots, x^*_N\}$ and $\{u^*_k, \ldots, u^*_{N-1}\}$ are optimal for the FTOCP $J^*_{k\to N}(x^*_k), \forall k \in \{0, \ldots, N-1\}$.

Given the optimal value function $J^*_{k+1\to N}$ at time k we can compute

$$
\begin{aligned}
J^*_{k\to N}(x(k)) = \min_{u_k} \quad & x(k)' Q x(k) + u_k' R u_k + J^*_{k+1\to N}(x_{k+1}) \\
\text{subject to} \quad & x_{k+1} = A x(k) + B u_k
\end{aligned}
$$

- $J^*_{k\to N}$ is often called the <u>*optimal cost-to-go*</u>.
- h is often called the <u>*stage cost*</u>.

Solution approach 2: Recursive Approach (1/3)

Solution approach 2: Recursive Approach (2/3)

We derived the cost:

$$J^*_{N-1}(x_{N-1}) = \min_{u_{N-1}} \{x'_{N-1}(A'P_N A + Q)x_{N-1} + u'_{N-1}(B'P_N B + R)u_{N-1} + 2x'_{N-1}A'P_N B u_{N-1}\}$$

Solution approach 2: Recursive Approach (3/3)

Substituting the optimal policy we have that

$$J^*_{N-1}(x_{N-1}) = x'_{N-1} P_{N-1} x_{N-1}\,,$$

where

$$P_{N-1} = A' P_N A + Q - A' P_N B (B' P_N B + R)^{-1} B' P_N A\,.$$

Final Result

- The problem is unconstrained
- Using the Dynamic Programming Algorithm we have

$$u_k^* = K_k x_k$$

which is a linear, time-varying state-feedback controller.

Final Result

- The problem is unconstrained
- Using the Dynamic Programming Algorithm we have

$$u_k^* = K_k x_k$$

  which is a linear, time-varying state-feedback controller.
- the optimal cost-to-go $k \to N$ is

$$J_k^*(x(k)) = x(k)' P_k x(k)$$

  which is a positive definite quadratic function of the state at time k
- K_k is computed by using P_{k+1}
- Each P_k is related to P_{k+1} by a recursive equation (Riccati Difference Equation)

The Bach Approach Vs Dynamic Programming Approach

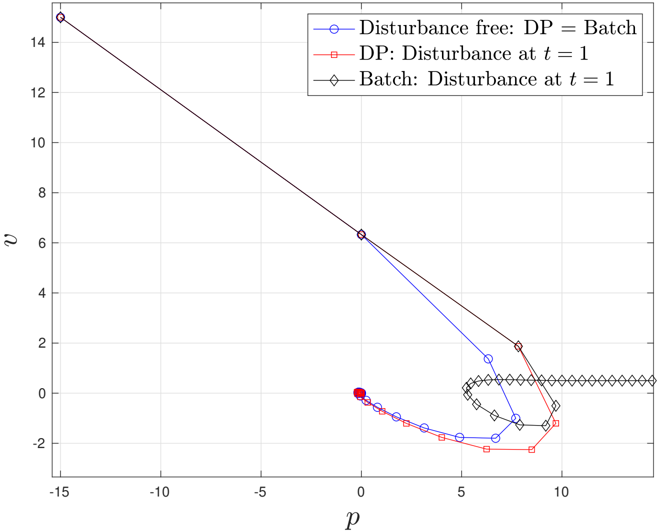
In the *batch approach* we compute the optimal sequence of actions $\boldsymbol{u}^*_{0\to N} = \{u^*_0, \dots, u^*_{N-1}\}$. Thus, the control policy is defined as

$$\pi(x(t), t) = u^*_t.$$

In the *dynamic programming approach* we compute the optimal value function $J^*_{k\to N}, \forall k \in \{0, \dots, N\}$. Thus, the control policy is defined as

$$\begin{aligned} \pi(x(t), t) = K_t x(t) = \arg\min_u \quad & h(x(t), u) + J^*_{t+1\to N}(x_{t+1}) \\ \text{subject to} \quad & x_{t+1} = Ax(t) + Bu_t \end{aligned}$$

Batch Vs Dynamic Programming



How about adding state and input constraints?

- Without any modification, both solution methods will break down when inequality constraints on x_k or u_k are added.

How about adding state and input constraints?

- Without any modification, both solution methods will break down when inequality constraints on x_k or u_k are added.
- The Batch Approach is far easier to adapt than the Recursive Approach when constraints are present: just perform a constrained minimization for the current state.

How about adding state and input constraints?

- Without any modification, both solution methods will break down when inequality constraints on x_k or u_k are added.
- The Batch Approach is far easier to adapt than the Recursive Approach when constraints are present: just perform a constrained minimization for the current state.
- Doing this at *every* time step within the time available, and then using only the first input from the resulting sequence, amounts to *receding horizon control*.

Table of Contents

Constrained Quadratic Linear Optimal Control

Consider the Constrained Finite Time Optimal Control Problem (CFTOCP):

$$
\begin{aligned}
J_0^*(x(0)) = \min_{U_0} \quad & \sum_{k=0}^{N-1} h(x_k, u_k) + x_N^\top Q_F x_N \\
\text{such that} \quad & x_{k+1} = Ax_k + Bu_k,\ k = 0, \ldots, N-1 \\
& x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0, \ldots, N-1 \\
& x_N \in \mathcal{X}_F \\
& x_0 = x(0)
\end{aligned}
\tag{3}
$$

where

- N is the time horizon.
- The state constraint set $\mathcal{X} = \{x \in \mathbb{R}^n : F_x x \leq b_x\}$.
- The input constraint set $\mathcal{U} = \{u \in \mathbb{R}^n : F_u x \leq b_u\}$.
- The terminal $\mathcal{X}_F = \{x \in \mathbb{R}^n : F_f x \leq b_f\}$.

Table of Contents

Quadratic Program with Substitution (1/4)

We re-write the CFTOCP as a Quadratic Program (QP) where the optimization variable is the vector of inputs $U_0 = [u_0^\top, \ldots, u_{N-1}^\top]$

- Starting with $x_0 = x(0)$, we have $x_1 = Ax(0) + Bu_0$, and $x_2 = Ax_1 + Bu_1 = A^2x(0) + ABu_0 + Bu_1$, by substitution for x_1, and so on. Continuing up to x_N we obtain:

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} I \\ A \\ \vdots \\ \vdots \\ A^N \end{bmatrix} x(0) + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix}$$

- The equation above can be represented as

$$X_0 \triangleq \mathcal{S}^x x(0) + \mathcal{S}^u U_0 \,. \tag{4}$$

Quadratic Program with Substitution (2/4)

- Define the cost matrices

  $$\overline{Q} \triangleq \text{blockdiag}(Q, \ldots, Q, Q_F) \quad \text{and} \quad \overline{R} \triangleq \text{blockdiag}(R, \ldots, R)$$

  Then the finite horizon cost function can be written as

  $$\begin{aligned} J_0(x(0), U_0) &= X_0' \overline{Q} X_0 + U_0^\top \overline{R} U_0 \\ &= (\mathcal{S}^x x(0) + \mathcal{S}^u U_0)^\top \overline{Q} (\mathcal{S}^x x(0) + \mathcal{S}^u U_0) + U_0^\top \overline{R} U_0 \\ &= U_0^\top H U_0 + 2 U_0 F^\top x(0) + (\mathcal{S}^x x(0))^\top \overline{Q} \mathcal{S}^x x(0), \end{aligned}$$

  where $H = \mathcal{S}^x \overline{Q} \mathcal{S}^x + \overline{R}$ and $F^\top = (\mathcal{S}^u)^\top \overline{Q} \mathcal{S}^x$.

Quadratic Program with Substitution (3/4)

- For all $k \in \{0, \ldots, N-1\}$, we have that the constraints $x_k \in \mathcal{X} = \{x \in \mathbb{R}^n \mid F_x x \leq b_x\}, u_k = \{x \in \mathbb{R}^d \mid F_u u \leq b_u\} \in \mathcal{U}$ and $x_N \in \mathcal{X}_F$ can be rewritten as

$$G_0 U_0 \leq E_0 x(0) + w_0$$

where

$$G_0 = \begin{bmatrix} F_u & 0 & \ldots & 0 \\ 0 & F_u & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & F_u \\ 0 & 0 & \ldots & 0 \\ F_x B & 0 & \ldots & 0 \\ F_x AB & F_x B & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ F_f A^{N-1} B & F_f A^{N-2} B & \ldots & F_f B \end{bmatrix}, E_0 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -A_x \\ -A_x A \\ -A_x A^2 \\ \vdots \\ -F_f A^N \end{bmatrix}, w_0 = \begin{bmatrix} b_u \\ b_u \\ \vdots \\ b_u \\ b_x \\ b_x \\ b_x \\ \vdots \\ b_f \end{bmatrix}$$

Quadratic Program with Substitution (4/4)

Given the quantities defined in the previous slides we can write the CFTOCP as

$$\begin{aligned} J_0^*(x(0)) = \min_{U_0} \quad & U_0^\top H U_0 + 2 U_0 F^\top x(0) \\ \text{subject to} \quad & G_0 U_0 \leq E_0 x(0) + w_0. \end{aligned}$$

Table of Contents

Quadratic Program with Substitution (1/4)

We re-write the FCTOCP as a Quadratic Program (QP) where the optimization variables are the vectors of inputs $U_0 = [u_0^\top, \dots, u_{N-1}^\top]^\top$ and states $X_0 = [x_1^\top, \dots, x_N^\top]^\top$.

- The dynamic constraints can be rewritten as

$$G_{0,\mathrm{eq}} \begin{bmatrix} X_0 \\ U_0 \end{bmatrix} = E_{0,\mathrm{eq}} x(0).$$

where

$$G_{0,\mathrm{eq}} = \begin{bmatrix} I & & & & & -B & & & \\ -A & I & & & & & -B & & \\ & -A & I & & & & & -B & \\ & & \ddots & \ddots & & & & & \ddots \\ & & & -A & I & & & & & -B \end{bmatrix}$$

and $E_{0,\mathrm{eq}} = [A^\top, 0, \dots, 0]^\top$.

Quadratic Program without Substitution (2/4)

- Define the matrices $G_{0,\text{in}}$, $w_{0,\text{in}}$ and $E_{0,\text{in}}$ be defined as follows:

$$G_{0,\text{in}} = \begin{bmatrix} 0 & & \dots & & 0 & 0 & & \dots & & 0 \\ F_x & & & & & 0 & & & & \\ & F_x & & & & & 0 & & & \\ & & \ddots & & & & & \ddots & & \\ & & & F_x & & & & & 0 & \\ & & & & F_f & & & & & 0 \\ 0 & & & & & F_u & & & & \\ & 0 & & & & & F_u & & & \\ & & \ddots & & & & & \ddots & & \\ & & & 0 & & & & & F_u & \\ & & & & 0 & & & & & F_u \end{bmatrix}$$

$$E_{0,\text{in}} = [-A_x^\top, 0, \dots, 0]^\top,$$

$$w_{0,\text{in}} = [b_x^\top, \dots, b_x^\top, b_f^\top, b_u^\top, \dots, b_u^\top]^\top$$

Quadratic Program without Substitution (3/4)

- Let $G_{0,\mathrm{in}}$, $w_{0,\mathrm{in}}$ and $E_{0,\mathrm{in}}$ be defined as in the previous slide. Then we have that for all $k \in \{0, \ldots, N-1\}$ the state $x_k \in \mathcal{X}$, the input $u_k \in \mathcal{U}$ and $x_N \in \mathcal{X}_F$ if and only if

$$G_{0,\mathrm{in}} \begin{bmatrix} X_0 \\ U_0 \end{bmatrix} \leq E_{0,\mathrm{in}} x(0) + w_{0,\mathrm{in}}.$$

Quadratic Program without Substitution (4/4)

Given the quantities defined in the previous slides we can write the CFTOCP as

$$
\begin{aligned}
J_0^*(x(0)) = \min_{U_0, X_0} \quad & [X_0^\top, U_0^\top] \begin{bmatrix} \overline{Q} & 0 \\ 0 & \overline{R} \end{bmatrix} \begin{bmatrix} X_0 \\ U_0 \end{bmatrix} \\
\text{subject to} \quad & G_{0,\mathrm{in}} \begin{bmatrix} X_0 \\ U_0 \end{bmatrix} \leq E_{0,\mathrm{in}} x(0) + w_{0,\mathrm{in}} \\
& G_{0,\mathrm{eq}} \begin{bmatrix} X_0 \\ U_0 \end{bmatrix} = E_{0,\mathrm{eq}} x(0).
\end{aligned}
$$

where

$$
\overline{Q} \triangleq \text{blockdiag}(Q, \dots, Q, Q_F) \quad \text{and} \quad \overline{R} \triangleq \text{blockdiag}(R, \dots, R).
$$

Constrained Linear Quadratic Optimal Control – Summary

Batch Approaches

We presented two batch approaches to compute a sequence of optimal control action $\boldsymbol{u}^*_{0\to N} = [u^*_0, \ldots, u^*_{N-1}]$. Given $\boldsymbol{u}^*_{0\to N} = [u^*_0, \ldots, u^*_{N-1}]$ a policy for the CLQR can be defined as

$$\pi(x(t), t) = u^*_t.$$

Constrained Linear Quadratic Optimal Control – Summary

Batch Approaches

We presented two batch approaches to compute a sequence of optimal control action $\boldsymbol{u}^*_{0\to N} = [u^*_0, \ldots, u^*_{N-1}]$. Given $\boldsymbol{u}^*_{0\to N} = [u^*_0, \ldots, u^*_{N-1}]$ a policy for the CLQR can be defined as

$$\pi(x(t), t) = u^*_t.$$

Dynamic Programming Approach

Recall that the dynamic programming recursion is defined as

$$\begin{aligned} \pi(x(t), t) = \arg\min_{u_i} \quad & h(x(t), u_i) + J^*_{t+1}(x_{t+1}) \\ \text{such that} \quad & x_{t+1} = Ax(t) + Bu_t \\ & x_{t+1} \in \mathcal{X}, u_t \in \mathcal{U}. \end{aligned}$$

In order to solve the above recursion, we need to explicitly compute the function $J^*_{t+1} : \mathbb{R}^n \to \mathbb{R}$.

Table of Contents

Optimal Control – Problem Formulation

Consider the following Constrained Finite Time Optimal Control Problem (CFTOCP):

$$
\begin{aligned}
J^*_{0\to N}(x(0)) = \min_{\boldsymbol{u}_{0\to N}} \quad & \sum_{k=0}^{N-1} h(x_k, u_k) + q(x_N) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k), \forall k \in \{0, \ldots, N-1\}, \\
& g(x_k, u_k) \leq 0, \forall k \in \{0, \ldots, N-1\}, \\
& x_N \in \mathcal{X}_F, x_0 = x(0).
\end{aligned}
$$

- the optimal cost $J^*_{0\to N}(x_0)$ is the *value function* of the problem.
- the optimal sequence of actions is denoted as $\boldsymbol{u}^*_{0\to N} = \{u^*_0, \ldots, u^*_{N-1}\}$.

Optimal Control – The Batch Approach

The FTOCP can be reformulated as a the following Non-Linear Program (NLP):

$$
\begin{aligned}
J^*_{0\to N}(x(0)) = \min_{\boldsymbol{u}_{0\to N}, \boldsymbol{x}_{0\to N}} \quad & \sum_{k=0}^{N-1} h(x_k, u_k) + q(x_N) \\
\text{subject to} \quad & x_1 = f(x(0), u_0), \\
& \vdots \\
& x_N = f(x_{N-1}, u_{N-1}), \\
& g(x_k, u_k) \leq 0, \forall k \in \{0, \dots, N-1\}, \\
& x_N \in \mathcal{X}_F, x_0 = x(0).
\end{aligned}
$$

where the sequences $\boldsymbol{u}_{0\to N}$ and $\boldsymbol{x}_{0\to N}$ are optimization variables. The above problem can be recast as an NLP, which can be solved with off-the-shelf solvers:

- Not all NLPs are the same. Some are easy to solve!
- Smoothness allows us to leverage gradient-based strategies
- Most solvers are based on iterative linearization techniques

The Bach Approach Vs Dynamic Programming Approach

In the *batch approach* we compute the optimal sequence of actions $\boldsymbol{u}^*_{0\to N} = \{u^*_0, \ldots, u^*_{N-1}\}$. Thus, the control policy is defined as

$$\pi(x(t), t) = u^*_t.$$

In the *dynamic programming approach* we compute the optimal value function $J^*_{k\to N}, \forall k \in \{0, \ldots, N\}$. Thus, the control policy is defined as

$$\begin{aligned} \pi(x(t), t) = \arg\min_u \quad & h(x(t), u) + J^*_{t+1\to N}(x_{t+1}). \\ \text{subject to} \quad & x_{t+1} = f(x(t), u) \\ & g(x(t), u) \leq 0. \end{aligned}$$

Table of Contents

Constrained Nonlinear Optimal Control – Linearization (1/2)

Next we consider the following problem:

$$
\begin{aligned}
J^*_{0\to N}(x_0) = \min_{\boldsymbol{u}_{0\to N}} \quad & \sum_{k=0}^{N-1} (x_k^\top Q x_k + u_k^\top R u_k) + x_N^\top Q_f x_N \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k), \forall k \in \{0, \dots, N-1\}, \\
& x_k \in \mathcal{X}, u_k \in \mathcal{U}, \forall k \in \{0, \dots, N-1\}, \\
& x_N \in \mathcal{X}_F, x_0 = x(0),
\end{aligned}
$$

where the cost function and the constraint sets are convex, but the system dynamics are defined by the nonlinear function $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$. We are going to approximate a solution to the above problem by iteratively linearizing the system dynamics.

Constrained Nonlinear Optimal Control – Linearization (2/2)

Notice that the system dynamics may be linearized around a state-input pair $(\bar{x}, \bar{u})$ as follows:

$$x_{k+1} = f(x, u) \approx f(\bar{x}, \bar{u}) + \nabla_x f(\bar{x}, \bar{u})(x - \bar{x}) + \nabla_u f(\bar{x}, \bar{u})(u - \bar{u}),$$

when $||x - \bar{x}||_2 \leq \epsilon$ and $||u - \bar{u}||_2 \leq \epsilon$.

Constrained Nonlinear Optimal Control – Linearization (2/2)

Notice that the system dynamics may be linearized around a state-input pair $(\bar{x}, \bar{u})$ as follows:

$$x_{k+1} = f(x, u) \approx f(\bar{x}, \bar{u}) + \nabla_x f(\bar{x}, \bar{u})(x - \bar{x}) + \nabla_u f(\bar{x}, \bar{u})(u - \bar{u}),$$

when $||x - \bar{x}||_2 \leq \epsilon$ and $||u - \bar{u}||_2 \leq \epsilon$.

Now define the matrices

$$A(\bar{x}, \bar{u}) = \nabla_x f(\bar{x}, \bar{u}), B(\bar{x}, \bar{u}) = \nabla_u f(\bar{x}, \bar{u})$$
$$\text{and } C(\bar{x}, \bar{u}) = f(\bar{x}, \bar{u}) - \nabla_x f(\bar{x}, \bar{u})\bar{x} - \nabla_u f(\bar{x}, \bar{u})\bar{u}.$$

Constrained Nonlinear Optimal Control – Linearization (2/2)

Notice that the system dynamics may be linearized around a state-input pair $(\bar{x}, \bar{u})$ as follows:

$$x_{k+1} = f(x, u) \approx f(\bar{x}, \bar{u}) + \nabla_x f(\bar{x}, \bar{u})(x - \bar{x}) + \nabla_u f(\bar{x}, \bar{u})(u - \bar{u}),$$

when $||x - \bar{x}||_2 \leq \epsilon$ and $||u - \bar{u}||_2 \leq \epsilon$.

Now define the matrices

$$A(\bar{x}, \bar{u}) = \nabla_x f(\bar{x}, \bar{u}), B(\bar{x}, \bar{u}) = \nabla_u f(\bar{x}, \bar{u})$$
$$\text{and } C(\bar{x}, \bar{u}) = f(\bar{x}, \bar{u}) - \nabla_x f(\bar{x}, \bar{u})\bar{x} - \nabla_u f(\bar{x}, \bar{u})\bar{u}.$$

Then we have that

$$x_{k+1} = A(\bar{x}, \bar{u})x + B(\bar{x}, \bar{u})u_k + C(\bar{x}, \bar{u})$$

Table of Contents

Sequential Quadratic Program – Preliminaries

Defined the constrained LQR problem around the of states and inputs sequences $U_j^0 = [u_0^j, \dots, u_{N-1}^j]$ and $X_0^j = [x_0^j, \dots, x_N^j]$:

$$
\begin{aligned}
\min_{\boldsymbol{u}_{0\to N}} \quad & \sum_{k=0}^{N-1} (x_k^\top Q x_k + u_k^\top R u_k + \alpha_1 ||x_k - x_k^j||_2^2 + \alpha_2 ||u_k - u_k^j||_2^2) \\
& \qquad + x_N^\top Q_f x_N \\
\text{subject to} \quad & x_{k+1} = A_k x_k + B_k u_k + C_k, \forall k \in \{0, \dots, N-1\}, \\
& x_k \in \mathcal{X}, u_k \in \mathcal{U}, \forall k \in \{0, \dots, N-1\}, \\
& x_N \in \mathcal{X}_F, x_0 = x(0),
\end{aligned}
\tag{5}
$$

where the matrices $A_k = A(x_k^j, u_k^j)$, $B_k = B(x_k^j, u_k^j)$ and $C_k = C(x_k^j, u_k^j)$ are computed linearizing the system dynamics, and cost terms $\alpha_1 ||x_k - x_k^j||_2^2$ and $\alpha_2 ||u_k - u_k^j||_2^2$ are used to limit the linearization error.

Sequential Quadratic Program – The algorithm

Given an initial guess of states $X_0^0 = [x_0^0, \ldots, x_N^0]$ and inputs $U_0^0 = [u_0^0, \ldots, u_{N-1}^0]$, we defined the following algorithm initialized for $j = 0$:

Sequential Quadratic Program – The algorithm

Given an initial guess of states $X_0^0 = [x_0^0, \ldots, x_N^0]$ and inputs $U_0^0 = [u_0^0, \ldots, u_{N-1}^0]$, we defined the following algorithm initialized for $j = 0$:

1. Set $A_k = A(x_k^j, u_k^j)$, $B_k = B(x_k^j, u_k^j)$ and $C_k = A(x_k^j, u_k^j)$ for all $k \in \{0, \ldots, N-1\}$.

Sequential Quadratic Program – The algorithm

Given an initial guess of states $X_0^0 = [x_0^0, \ldots, x_N^0]$ and inputs $U_0^0 = [u_0^0, \ldots, u_{N-1}^0]$, we defined the following algorithm initialized for $j = 0$:

1. Set $A_k = A(x_k^j, u_k^j)$, $B_k = B(x_k^j, u_k^j)$ and $C_k = A(x_k^j, u_k^j)$ for all $k \in \{0, \ldots, N-1\}$.
2. Solve the convex optimization problem (5) and let $\{x_k^*\}_{k=0}^{N}$ and $\{u_k^*\}_{k=0}^{N-1}$ be the optimal sequences of states and inputs, respectively.

Sequential Quadratic Program – The algorithm

Given an initial guess of states $X_0^0 = [x_0^0, \ldots, x_N^0]$ and inputs $U_0^0 = [u_0^0, \ldots, u_{N-1}^0]$, we defined the following algorithm initialized for $j = 0$:

1. Set $A_k = A(x_k^j, u_k^j)$, $B_k = B(x_k^j, u_k^j)$ and $C_k = A(x_k^j, u_k^j)$ for all $k \in \{0, \ldots, N-1\}$.
2. Solve the convex optimization problem (5) and let $\{x_k^*\}_{k=0}^{N}$ and $\{u_k^*\}_{k=0}^{N-1}$ be the optimal sequences of states and inputs, respectively.
3. If $||x_k^* - x_k^k|| \leq \epsilon$ for all $k \in \{0, \ldots, N\}$ and $||u_k^* - u_k^k|| \leq \epsilon$ for all $k \in \{0, \ldots, N-1\}$ then terminate. Otherwise, set $x_k^{j+1} = x_k^*$ and $u_k^{j+1} = u_k^*$ for all $k \in \{0, \ldots, N-1\}$, $j = j + 1$ and go to 1.

Sequential Quadratic Program – The algorithm

Given an initial guess of states $X_0^0 = [x_0^0, \ldots, x_N^0]$ and inputs $U_0^0 = [u_0^0, \ldots, u_{N-1}^0]$, we defined the following algorithm initialized for $j = 0$:

1. Set $A_k = A(x_k^j, u_k^j)$, $B_k = B(x_k^j, u_k^j)$ and $C_k = A(x_k^j, u_k^j)$ for all $k \in \{0, \ldots, N-1\}$.
2. Solve the convex optimization problem (5) and let $\{x_k^*\}_{k=0}^N$ and $\{u_k^*\}_{k=0}^{N-1}$ be the optimal sequences of states and inputs, respectively.
3. If $||x_k^* - x_k^k|| \leq \epsilon$ for all $k \in \{0, \ldots, N\}$ and $||u_k^* - u_k^k|| \leq \epsilon$ for all $k \in \{0, \ldots, N-1\}$ then terminate. Otherwise, set $x_k^{j+1} = x_k^*$ and $u_k^{j+1} = u_k^*$ for all $k \in \{0, \ldots, N-1\}$, $j = j + 1$ and go to 1.

- A similar strategy may be used to linearize cost and constraints
- The parameters α_1 and α_2 may be adapted
- It is not required to solve Problem (5) to optimality

Summary

We discussed the difference between strategies to solve finite time optimal control problems:

- the batch approach which is used to compute a sequence of open-loop actions.
- the dynamic programming approach which is used to compute a control policy mapping states to actions.

We considered different cases:

- General control problem with nonlinear dynamics
- Linear Quadratic Regulator
- Constrained Linear Quadratic Regulator

Key Message: For problem with continuous state-action spaces computing an optimal trajectory is “easy”, but computing a policy is hard when we have constraints !

What is next?

We will show how to compute control policies for constrained optimal control problems with continuous state-action spaces.

We will focus on guaranteeing that

- State and input constraints are satisfied.
- The closed-loop system is stable.
- The control policy is optimal.

First, we will show the control design when the system dynamics are known. Then, we will discuss iterative learning strategies which are similar to the policy iteration approach that we saw in Lecture #1.