Introduction to Model Predictive Control

Lectures 9-10: Optimal Control

Francesco Borrelli

University of California at Berkeley,
Mechanical Engineering Department

ME190M-Fall 2009

Summarizing...

Need

1. A discrete-time model of the system (Matlab, Simulink)
2. A state observer
3. Set up an Optimization Problem (Matlab, MPT toolbox/Yalmip)
4. Solve an optimization problem (Matlab/Optimization Toolbox, NPSOL)
5. Verify that the closed-loop system performs as desired (avoid infeasibility/stability)
6. Make sure it runs in real-time and code/download for the embedded platform

Summarizing...

Need

1. A discrete-time model of the system (Matlab, Simulink)
2. A state observer
3. Set up an Optimization Problem (Matlab, MPT toolbox/Yalmip)
4. Solve an optimization problem (Matlab/Optimization Toolbox, NPSOL)
5. Verify that the closed-loop system performs as desired (avoid infeasibility/stability)
6. Make sure it runs in real-time and code/download for the embedded platform

General Problem Formulation

Consider the nonlinear time-invariant system

$$x(t+1) = g(x(t), u(t)), \tag{1}$$

subject to the constraints

$$h(x(t), u(t)) \leq 0 \tag{2}$$

at all time instants $t \geq 0$. $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ are the state and input vector.

- Define the following *performance objective* or *cost function*

$$J_{0\to N}(x_0, U_{0\to N}) \triangleq p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \tag{3}$$

N is the time *horizon*, x_k is the state at time k obtained by starting from $x_0 = x(0)$ and applying the input sequence $u_0, \ldots, u_{k-1}$. to the system model $x_{k+1} = g(x_k, u_k)$

General Problem Formulation

- $U_{0\to N} \triangleq [u_0', \ldots, u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq mN$ is the vector of future inputs. $q(x_k, u_k)$ and $p(x_N)$ are referred to as *stage cost* and *terminal cost*, respectively.

Consider the constrained finite time optimal control (CFTOC) problem.

$$
\begin{aligned}
J^*_{0\to N}(x_0) = \quad & \min_{U_{0\to N}} \quad && J_{0\to N}(x_0, U_{0\to N}) \\
& \text{such that} && x_{k+1} = g(x_k, u_k),\ k = 0, \ldots, N-1 \\
& && h(x_k, u_k) \leq 0,\ k = 0, \ldots, N-1 \\
& && x_N \in \mathcal{X}_f \\
& && x_0 = x(0)
\end{aligned}
\tag{4}
$$

General Problem Formulation

- $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a *terminal region* that we want the system states to reach at the end of the horizon.
- The optimal cost $J^*_{0\to N}(x_0)$ is also called *value function*.
- There might be several input vectors $U^*_{0\to N}$ which yield the minimum

$$J^*_{0\to N}(x_0) = J_{0\to N}(x_0, U^*_{0\to N})$$

In this case we will define one of them as the minimizer $U^*_{0\to N}$.

Solution via Batch Approach

Write the equality constraints appearing in (4) explicitly

$$\begin{aligned} x_1 &= g(x(0), u_0) \\ &\vdots \\ x_N &= g(x_{N-1}, u_{N-1}) \end{aligned} \tag{5}$$

then the optimal control problem

$$\begin{aligned} J^*_{0\to N}(x_0) = \quad & \min_{U_{0\to N}} \quad && p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \\ & \text{such that} && x_1 = g(x_0, u_0) \\ & && x_2 = g(x_1, u_1) \\ & && \vdots \\ & && x_N = g(x_{N-1}, u_{N-1}) \\ & && h(x_k, u_k) \leq 0, \;\; k = 0, \ldots, N-1 \\ & && x_N \in \mathcal{X}_f \\ & && x_0 = x(0) \end{aligned} \tag{6}$$

is a general nonlinear programming problem with variables $u_0, \ldots, u_{N-1}$ and $x_1, \ldots, x_N$.

Solution via Batch Approach

As an alternative eliminate the state variables and equality constraints (5) by successive substitution so that we are left with $u_0, \ldots, u_{N-1}$ as the only decision variables.

$$\begin{aligned} x_2 &= g(x_1, u_1) \\ x_2 &= g(g(x(0), u_0), u_1). \end{aligned} \tag{7}$$

- The solution of the nonlinear programming problem is a sequence of present and future inputs $U^*_{0\to N} = [u_0^{*'}, \ldots, u_{N-1}^{*'}]'$ determined for the particular initial state $x(0)$.
- Except when the state equations are linear this successive substitution may become complex.

Solution via Recursive Approach (Unconstrained)

The recursive approach, Bellman's dynamic programming technique, rests on a simple idea,

The *principle of optimality*

For a trajectory $x_0, x_1^*, \ldots, x_N^*$ to be optimal, the trajectory starting from any intermediate point x_j^*, i.e. $x_j^*, x_{j+1}^*, \ldots, x_N^*$, $0 \leq j \leq N-1$, must be optimal.

Suppose that the fastest route from Los Angeles to Boston passes through Chicago. The principle of optimality formalizes the obvious fact that the Chicago to Boston portion of the route is also the fastest route for a trip that starts from Chicago and ends in Boston.

Solution via Recursive Approach (Unconstrained)

Define the cost over the reduced horizon from j to N

$$J_{j\to N}(x_j, u_j, u_{j+1}, \ldots, u_{N-1}) \triangleq p(x_N) + \sum_{k=j}^{N-1} q(x_k, u_k), \tag{8}$$

also called the j-step *cost-to-go*. Then the *optimal cost-to-go* $J^*_{j\to N}$ is

$$\begin{aligned} J^*_{j\to N}(x_j) \triangleq \quad & \min_{u_j, u_{j+1}, \ldots, u_{N-1}} \quad J_{j\to N}(x_j, u_j, u_{j+1}, \ldots, u_{N-1}) \\ & \text{such that} \qquad\qquad x_{k+1} = g(x_k, u_k),\ k = j, \ldots, N-1 \end{aligned} \tag{9}$$

Note that the optimal cost-to-go $J^*_{j\to N}(x_j)$ depends only on the initial state x_j.

Solution via Recursive Approach

The principle of optimality implies that the optimal cost-to-go $J^*_{j-1\to N}$ from time $j-1$ to the final time N can be found by minimizing the sum of the stage cost $q(x_{j-1}, u_{j-1})$ and the optimal cost-to-go $J^*_{j\to N}(x_j)$ from time j onwards:

Dynamic Programming

$$J^*_{j-1\to N}(x_{j-1}) = \min_{u_{j-1}} \quad q(x_{j-1}, u_{j-1}) + J^*_{j\to N}(x_j) \quad \text{such that} \quad x_j = g(x_{j-1}, u_{j-1}) \tag{10}$$

Here the only decision variable left for the optimization is u_{j-1}, the input at time $j-1$. All the other inputs $u^*_j, \ldots, u^*_{N-1}$ have already been selected optimally to yield the optimal cost-to-go $J^*_{j\to N}(x_j)$.

Solution via Recursive Approach: Comments

- The dynamic programming problem is appealing because it can be stated compactly and because at each step the optimization takes place over one element u_j of the optimization vector only.
- However need to construct the optimal cost-to-go $J^*_{N-j}(x_j)$
- In a few special cases we know the type of function and we can find it efficiently.
- In general, a "brute force" constructs $J_{j-1\to N}$ by griding the state space and computing the optimal cost-to-go function on each grid point.
  The complexity of constructing the cost-to-go function in this manner increases rapidly with the dimension of the state space ("curse of dimensionality").

Notation

For the sake of simplicity we will use the following shorter notation

$$J_j^*(x_j) \triangleq J_{j\to N}^*(x_j),\ j = 0, \dots, N$$
$$J_\infty^*(x_0) \triangleq J_{0\to\infty}^*(x_0)$$
$$U_0 \triangleq U_{0\to N}$$

Linear Quadratic Optimal Control

Linear Quadratic Optimal Control

Consider a linear and time-invariant system

$$x(t+1) = Ax(t) + Bu(t) \tag{11}$$

Define the following quadratic cost function over a finite horizon of N steps

$$J_0(x_0, U_0) \triangleq x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \tag{12}$$

where we denote with $U_0 \triangleq [u_0, \ldots, u_{N-1}] \in \mathbb{R}^s$, $s \triangleq mN$ the decision vector containing all future inputs.

Linear Quadratic Optimal Control

Consider the finite time optimal control problem

Finite Time Linear Quadratic Regulator

$$\begin{aligned} J_0^*(x(0)) = \quad & \min_{U_0} \quad && J_0(x(0), U_0) \\ & \text{such that} \quad && x_{k+1} = Ax_k + Bu_k, \ k = 0, 1, \ldots, N-1 \\ & && x_0 = x(0). \end{aligned} \tag{13}$$

In (13) we assume that the state penalty is positive semi-definite $Q = Q' \succeq 0$, $P = P' \succeq 0$ and the input penalty is positive definite $R = R' \succ 0$.

Solution via Batch Approach

Eliminate all intermediate states by successive substitution to obtain

$$\underbrace{\begin{bmatrix} x(0) \\ x_1 \\ \vdots \\ \vdots \\ x_N \end{bmatrix}}_{\mathcal{X}} = \underbrace{\begin{bmatrix} I \\ A \\ \vdots \\ \vdots \\ A^N \end{bmatrix}}_{\mathcal{S}^x} x(0) + \underbrace{\begin{bmatrix} 0 & \dots & \dots & 0 \\ B & 0 & \dots & 0 \\ AB & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A^{N-1}B & \dots & \dots & B \end{bmatrix}}_{\mathcal{S}^u} \begin{bmatrix} u_0 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix}. \tag{14}$$

Here all future states are explicit functions of the present state $x(0)$ and the future inputs $u_0, u_1, u_2, \dots$ only. Rewrite this expression compactly as

$$\mathcal{X} = \mathcal{S}^x x(0) + \mathcal{S}^u U_0 \tag{15}$$

Using the same notation the objective function can be rewritten as

$$J(x(0), U_0) = \mathcal{X}' \bar{Q} \mathcal{X} + U_0{}' \bar{R} U_0 \tag{16}$$

where $\bar{Q} = \text{blockdiag}\{Q, \cdots, Q, P\}, \bar{Q} \succeq 0$, and $\bar{R} = \text{blockdiag}\{R, \cdots, R\}, \bar{R} \succ 0$.

Solution via Batch Approach

Substituting (15) into the objective function (16) yields

$$\begin{aligned} J_0(x(0), U_0) &= (\mathcal{S}^x x(0) + \mathcal{S}^u U_0)' \bar{Q} (\mathcal{S}^x x(0) + \mathcal{S}^u U_0) + U_0' \bar{R} U_0 \\ &= U_0' \underbrace{(\mathcal{S}^{u\prime} \bar{Q} \mathcal{S}^u + \bar{R})}_{H} U_0 + 2x'(0) \underbrace{(\mathcal{S}^{x\prime} \bar{Q} \mathcal{S}^u)}_{F} U_0 + \\ &\quad x'(0) \underbrace{(\mathcal{S}^{x\prime} \mathcal{S}^x)}_{Y} x(0) \\ &= U_0' H U_0 + 2x(0)' F U_0 + x'(0) Y x(0) \end{aligned} \tag{17}$$

This yields the optimal vector of future inputs

Finite Time LQR Solution

$$U_0^* = -H^{-1} F' x(0) \tag{18}$$

Note that optimal vector of future inputs U_0^* is a linear function of the initial state $x(0)$.

Solution via Recursive Approach

Define the optimal cost $J_j^*(x_j)$ for the $N-j$ step problem starting from state x_j by

$$J_j^*(x_j) \stackrel{\Delta}{=} \min_{u_j,\cdots,u_{N-1}} x_N' P x_N + \sum_{k=j}^{N-1} \left[x_k' Q x_k + u_k' R u_k\right]$$

By principle of optimality

$$J_{N-1}^*(x_{N-1})) = \min_{u_{N-1}} x_N' P_N x_N + x_{N-1}' Q x_{N-1} + u_{N-1}' R u_{N-1} \tag{19}$$

$$\begin{aligned} x_N &= A x_{N-1} + B u_{N-1} \\ P_N &= P \end{aligned} \tag{20}$$

P_j expresses the optimal cost-to-go ${x_j}' P_j x_j$ from time j to the end of the horizon N. Specifically if $j = N$ this is simply P.

Solution via Recursive Approach

Substitute system dynamics into the objective function

$$\begin{aligned} J^*_{N-1}(x_{N-1}) \quad = \quad & \min_{u_{N-1}} \{ x'_{N-1}(A'P_NA + Q)x_{N-1} \\ & +2x'_{N-1}A'P_NBu_{N-1} \\ & +u'_{N-1}(B'P_NB + R)u_{N-1} \} . \end{aligned} \tag{21}$$

and thus

$$u^*_{N-1} = \underbrace{-(B'P_NB + R)^{-1}B'P_NA}_{F_{N-1}} x_{N-1} \tag{22}$$

and the one-step optimal cost-to-go

$$J^*_{N-1}(x_{N-1}) = x'_{N-1}P_{N-1}x_{N-1}, \tag{23}$$

where we have defined

$$P_{N-1} = A'P_NA + Q - A'P_NB(B'P_NB + R)^{-1}B'P_NA \tag{24}$$

Solution via Recursive Approach

At the next stage, consider the two-step problem from time $N-2$ forward:

$$J^*_{N-2}(x_{N-2}) = \min_{u_{N-2}} x'_{N-1} P_{N-1} x_{N-1} + \left[x'_{N-2} Q x_{N-2} + u'_{N-2} R u_{N-2} \right] \tag{25}$$

$$x_{N-1} \quad = \quad Ax_{N-2} + Bu_{N-2} \tag{26}$$

We recognize that the $N-2$ stage problem has the same form as the N-1 problem, Therefore

$$u^*_{N-2} = \underbrace{-(B' P_{N-1} B + R)^{-1} B' P_{N-1} A}_{F_{N-2}} x_{N-2} \tag{27}$$

The optimal two-step cost-to-go is

$$J^*_{N-2}(x_{N-2}) = x'_{N-2} P_{N-2} x_{N-2}, \tag{28}$$

where we defined

$$P_{N-2} = A' P_{N-1} A + Q - A' P_{N-1} B (B' P_{N-1} B + R)^{-1} B' P_{N-1} A \tag{29}$$

Solution via Recursive Approach

Continuing in this manner, at some arbitrary time k the optimal control action is

Finite Time LQR Feedback Solution

$$\begin{aligned} u^*(k) &= -(B'P_{k+1}B + R)^{-1}B'P_{k+1}Ax(k), \\ &= F_k x(k), \qquad \text{for } k = 0, \ldots, N-1, \end{aligned} \tag{30}$$

where

$$P_k = A'P_{k+1}A + Q - A'P_{k+1}B(B'P_{k+1}B + R)^{-1}B'P_{k+1}A \tag{31}$$

Equation (31) (called *Discrete Time Riccati Equation* or *Riccati Difference Equation* - RDE) is initialized with $P_N = P$ and is solved backwards, i.e., starting with P_N and solving for P_{N-1}, etc.

Note that the optimal control action $u^*(k)$ is obtained in form of a feedback law as a linear time-varying function of the measured state $x(k)$ at time k.

Infinite Horizon Problem

$$J^*_\infty(x(0)) = \min_{u(0),u(1),\ldots} \sum_{k=0}^{\infty} \left[x_k' Q x_k + u_k' R u_k\right] \tag{32}$$

We can initialize the RDE with the terminal cost matrix $P_0 = Q$

$$P_k = A' P_{k+1} A + Q - A' P_{k+1} B (B' P_{k+1} B + R)^{-1} B' P_{k+1} A \tag{33}$$

and solve it backwards for $k \to -\infty$. Assume that the iterations converge to a solution P_∞. Such P_∞ would then satisfy the *Algebraic Riccati Equation* (ARE)

$$P_\infty = A' P_\infty A + Q - A' P_\infty B (B' P_\infty B + R)^{-1} B' P_\infty A. \tag{34}$$

Infinite Horizon Problem

Then the optimal feedback control law is

Infinite Time LQR Solution

$$u^*(k) = \underbrace{-(B'P_\infty B + R)^{-1}B'P_\infty A}_{F_\infty} x(k), \qquad k = 0, \cdots, \infty \tag{35}$$

Controller (35) is the *Linear Quadratic Regulator* (LQR)

Matlab Command

$$[F_\infty] = lqr(SYS, Q, R)$$

To Remember

1. 2 ways of solving batch approach
2. Idea of Dynamic Programming
3. Linear Quadratic Regulator (LQR) Problem and role of Q, R matrices
4. Finite Time LQR Solution $u^*(k) = F_k x(k)$, $k = 0, \ldots, N-1$
5. Infinite Time LQR Solution $u^*(k) = F_\infty x(k)$
6. Infinite Time LQR Matlab command (LQR)

Homework 2/1

Consider the Ball and Plate System

- The control goal is to bring the ball from any admissible position to the origin of the state space (position and speed of the plate equal to zero and position and speed of the ball equal to zero).
- Substitute the state feedback discrete time controller given in the past homework $u(k) = Kx(k)$ (remember is sampling $\Delta T = 30ms$) with a discrete time LQR controller with same sampling time. (Hint: First discretize the continuous time system and then use "LQR" command with chosen Q and R weighting matrices).
- Simulate the closed loop system (DT controller and CT LTI system in state space form) when the system starts from the initial condition $x(0) = [20, 0, 0, 0]$ (plate and ball at rest and ball at distance 20cm from the origin)
- Study the effect of Q and R on the closed-loop behavior.

Homework 2/2

Consider the vehicle model (for fuel consumption), assume zero grade

- Use the simulink files you have already.
- Take **a piece of** the average speed profile provided on b-Space (101 highway). The length of this piece (distance interval) will affect the complexity of your problem.
- For such distance interval, design a (open loop) controller which regulates the wheel force $F_w(k)$ which minimizes fuel consumption and satisfies the following constraints.
  - Over the considered distance interval assuming the car cannot speed up (or slow down) more than (less than) 20% of the average speed profile.
  - The completion time for the chosen travel segment is less than $\alpha\%$ of the one of the average speed profile.
- Provide the fuel consumption (and completion time) for the considered distance interval when the car travels at the average speed profile. Compare this with the fuel consumption (and completion time) if the car travels at the speed profile computed by the optimizer for some α.