

Linear Quadratic Optimal Control

Dr. Niket S. Kaisare

Department of Chemical Engineering
Indian Institute of Technology – Madras

January 22, 2009

Outline of the Lecture Series

- Part A: Deterministic System
 - Introduction to LQ control
 - Least Squares vs. Dynamic Programming solution
 - State vs. Output Feedback
- Part B: Stochastic System
- Part C: Constrained Linear Quadratic Control
- Part D: Extensions and Approximate Solutions
 - Multi-parametric Programming
 - Approximate Dynamic Programming
 - Iterative Dynamic Programming

Introduction

Consider the following linear system:

$$x(k+1) = Ax(k) + Bu(k) \quad (1a)$$

$$y(k) = Cx(k) \quad (1b)$$

- The objective of regulation is to drive the system to the origin starting at any arbitrary state $x(0) \in \mathcal{X}$.
- For a linear controller

$$u(k) = -L(k)x(k) \quad (2)$$

- Thus, the linear system with linear controller is:

$$x(k+1) = (A - BL(k))x(k) \quad (3)$$

Stability

Output feedback controller (2) is stable if all the eigenvalues of $(A - BL)$ lie within the unit disk

Definition

An optimal control involves obtaining a set of manipulated variable moves such that a certain performance criterion (called *objective function*) is minimized or maximized, subject to appropriate constraints.

Some points to note:

- Optimal control is the truly best you can do, given an objective function
- “Best” depends on the choice of the objective function
- Often, solving optimal control problem exactly requires too much effort. Linear Quadratic Optimal Control is an exception

Linear Quadratic Optimal Control

- The performance criterion for typical control applications is chosen to be quadratic

$$V_p \triangleq \sum_{k=0}^{p-1} [x^T(k)Qx(k) + u^T(k)Ru(k)] + x^T(p)Q_t x(p) \quad (4)$$

- Q and R are positive definite; Q_t is positive semi-definite
- Q provide relative importance to the errors in various states
- R account for the cost of implementing input moves
- Q_t weights the importance of the final state $x(p)$
- Finite and infinite horizon problems:

$p = \infty \quad \Rightarrow \quad$ Infinite Horizon Problem
otherwise \Rightarrow Finite Horizon Problem

Some Terminologies and Classification

- Finite vs. Infinite Horizon
- Open-loop Optimal vs. Optimal Feedback
 - Open-loop Optimal Feedback Control (OLOFC):
Finding a sequence of input moves, $u(0), u(1), \dots, u(p-1)$, that minimizes the objective function for a given $x(0)$.
 - Optimal Feedback Control (OFC) involves finding the input $u(k)$ as a function of any system state $x(k) \in \mathcal{X}$.
 - OLOFC and OFC problems are same for deterministic system; OFC is more accurate for stochastic case.
- State Feedback vs. Output Feedback Case
 - **State feedback:** $u(k) = f(x(k))$
 - **Output feedback:** $u(k) = \mathcal{F}(y(k))$

Linear Quadratic Optimal Control

Part-A: Deterministic Case

Dr. Niket S. Kaisare

Department of Chemical Engineering
Indian Institute of Technology – Madras

January 22, 2009

Least Squares Solution

Open-Loop Optimal Feedback Control

We can recursively use (1a) as:

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k) = A(Ax(k-1) + Bu(k-1)) \\ &= A^2x(k-1) + Bu(k) + ABu(k-1) \\ &\vdots \\ &= A^{k+1}x(0) + \left(Bu(k) + ABu(k-1) + \dots + A^k Bu(0) \right)\end{aligned}$$

Thus, we can write

$$\underbrace{\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(p) \end{bmatrix}}_{\mathcal{X}} = \underbrace{\begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^p \end{bmatrix}}_{S^x} x(0) + \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ A^{p-1}B & A^{p-2}B & \dots & B \end{bmatrix}}_{S^u} \underbrace{\begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(p-1) \end{bmatrix}}_{\mathcal{U}} \quad (5)$$

- System equation

$$\mathcal{X} = \mathcal{S}^x x(0) + \mathcal{S}^u \mathcal{U}$$

- Quadratic cost function

$$\begin{aligned} V_p &\triangleq \sum_{k=0}^{p-1} \left[x^T(k) Q x(k) + u^T(k) R u(k) \right] + x^T(p) Q_t x(p) \\ &= \mathcal{X}^T \Gamma^x \mathcal{X} + \mathcal{U}^T \Gamma^u \mathcal{U} \end{aligned}$$

$$\Gamma^x = \text{blockdiag}\{Q, \dots, Q, Q_t\}; \Gamma^u = \text{blockdiag}\{R, \dots, R\}$$

- Optimal Cost

$$\begin{aligned} J_p(x(0)) = \min_{\mathcal{U}} &\left\{ x^T(0) \mathcal{S}^{xT} \Gamma^x \mathcal{S}^x x(0) + \right. \\ &\left. \mathcal{U}^T \left[\mathcal{S}^{uT} \Gamma^x \mathcal{S}^u + \Gamma^u \right] \mathcal{U} + 2x^T(0) \mathcal{S}^{xT} \Gamma^x \mathcal{S}^u \mathcal{U} \right\} \end{aligned}$$

- Optimal solution

$$\mathcal{U} = -\mathcal{H}^{-1} g = -[\mathcal{S}^{uT} \Gamma^x \mathcal{S}^u + \Gamma^u]^{-1} \mathcal{S}^{uT} \Gamma^x \mathcal{S}^x x(0)$$

Open-Loop Optimal Feedback Control

$$\mathcal{U} = -\left[\mathcal{S}^{uT}\Gamma^x\mathcal{S}^u + \Gamma^u\right]^{-1}\mathcal{S}^{uT}\Gamma^x\mathcal{S}^x x(0) \quad (6)$$

$$J_p(x(0)) = x^T(0)\left[\mathcal{S}^{xT}\Gamma^x\mathcal{S}^x - \mathcal{S}^{xT}\Gamma^x\mathcal{S}^u\left[\mathcal{S}^{uT}\Gamma^x\mathcal{S}^u + \Gamma^u\right]^{-1}\mathcal{S}^{uT}\Gamma^x\mathcal{S}^x\right]x(0) \quad (7)$$

- Open-loop optimal control:
finds a sequence $u(0), u(1), \dots, u(p-1)$ for a given $x(0)$
- Recursively use (6) as in *Receding Horizon Control*
- *Not* efficient computationally
- *Not* generalizable to the stochastic case

Optimal Feedback Control

- One obtains the optimal control move as a function of state $x(0) \in \mathcal{X}$
- Solved using **Dynamic Programming**
- More elegant and closed-loop optimal solution

Definition: Cost-to-go

Cost-to-go is the minimum cost incurred to solve the k -step optimal control problem starting from the state z .

$$J_k(z) \triangleq \min_{u(p-j), \dots, u(p-1)} \sum_{i=p-k}^{p-1} \left[x^T(i)Qx(i) + u^T(i)Ru(i) \right] + x^T(p)Q_f x(p) \quad (8)$$

for the system $x(i+1) = Ax(i) + Bu(i)$ and the initial condition $x(p-j) = z$ is called “ k -step cost-to-go” for z .

Note: $J_p(x(0))$ obtained earlier is the p -step cost-to-go starting with an initial state $x(0)$.

Dynamic Programming

Note from (6), the optimal decision depends only on the current state and not the past states. Starting from the end, consider $J_1(x(p-1))$:

$$\min_{u(p-1)} \left\{ x^T(p-1)Qx(p-1) + u^T(p-1)Ru(p-1) + x^T(p)S(p)x(p) \right\} \quad (9)$$

where $S(p) = Q_t$.

Noting that $x(p) = Ax(p-1) + Bu(p-1)$, we get:

$$J_1(x(p-1)) = \min_{u(p-1)} \left\{ x^T(p-1)(A^T S(p)A + Q_t)x(p-1) + 2x^T(p-1)A^T S(p)B u(p-1) + u^T(p-1)(B^T S(p)B + R)u(p-1) \right\}$$

As before, the optimal solution can be obtained as:

$$u(p-1) = - \underbrace{(B^T S(p)B + R)^{-1} B^T S(p)A}_{L(p-1)} x(p-1)$$

Substituting, the cost-to-go function is:

$$J_1(x(p-1)) = x^T(p-1)S(p-1)x(p-1)$$

where, $S(p-1)$ is given by the following Ricatti Equation:

$$S(p-1) = A^T S(p)A + Q - A^T S(p)B(B^T S(p)B + R)^{-1} B^T S(p)A$$

At the next stage, a two-step-ahead problem is solved to obtain $J_2(x(p-2))$:

$$J_2(x(p-2)) = \min_{u(p-1), u(p-2)} \left\{ x^T(p)S(p)x(p) + \sum_{i=p-2}^{p-1} [x^T(i)Qx(i) + u^T(i)Ru(i)] \right\}$$

Bellman's Optimality Principle states that in order for the above solution to be optimal, the solution from an intermediate state to the final state (i.e., solution to problem (9)) is optimal.

In other words, the above problem is equivalent to one-stage problem with cost-to-go from the previous stage:

$$\begin{aligned} J_2(x(p-2)) &= \min_{u(p-2)} \left\{ J_1(x(p-1)) + \right. \\ &\quad \left. x^T(p-2)Qx(p-2) + u^T(p-2)Ru(p-2) \right\} \\ &= \min_{u(p-2)} \left\{ x^T(p-1)S(p-1)x(p-1) + \right. \\ &\quad \left. x^T(p-2)Qx(p-2) + u^T(p-2)Ru(p-2) \right\} \end{aligned}$$

This equation is in the same form as (9). The optimal solution is

$$u(p-2) = - \underbrace{(B^T S(p-1)B + R)^{-1} B^T S(p-1)A}_{L(p-2)} x(p-2) \quad (10)$$

Dynamic Programming: Generalizing the Solution

Successively solving for $J_{p-k}(x(k))$ and computing cost-to-go, we get:

$$u(k) = -L(k)x(k) \quad \text{for } k = p - 1, \dots, 0$$

where

$$L(k) = -(B^T S(k+1)B + R)^{-1} B^T S(k+1)A \quad (11)$$

$$S(k) = A^T S(k+1)A + Q - A^T S(k+1)B(B^T S(k+1)B + R)^{-1} B^T S(k+1)A \quad (12)$$

Note that (12) is the familiar **Riccati Difference Equation** that we encountered in Kalman Filtering as well.

Comparison of OLOFC and OFC

- For a deterministic case, OLOFC and OFC yield the same solution
- The optimal p -stage cost is: $J_p(x_0) = x^T(0)S(0)x(0)$
- Receding Horizon solution to optimization is computationally demanding
- Dynamic Programming leads to the optimal control solution as an *explicit* linear function, $u(k) = -L(k)x(k)$
- Recursive solution of Riccati equation, required in DP, is straightforward
- Dynamic Programming approach is no longer feasible in presence of input and/or state constraints

Extension to Infinite Horizon Problem

$$J_{\infty}(x_0) = \min_{u(\cdot)} \left\{ \sum_{k=0}^{\infty} [x^T(k)Qx(k) + u^T(k)Ru(k)] \right\} \quad (13)$$

Receding Horizon OLOCP

- Still solve a finite-horizon problem with a receding horizon implementation
- Larger the horizon, greater is the computational effort
- For certain choices of Q_t , the finite horizon problem can be made equivalent to the infinite horizon problem

Asymptotic Solution of DP

- Starting with a specific choice of Q_t , the RDE (12) can be solved recursively until convergence.
- Under certain conditions, RDE converges to an asymptotic value S_{∞} .
- Infinite horizon cost:

$$J_{\infty}(x(0)) = x^T(0)S_{\infty}x(0)$$

Extension of DP to Infinite Horizon

Assuming the RDE solution converges to S_∞ ,

$$u(k) = \underbrace{(B^T S_\infty B + R)^{-1} B^T S_\infty A}_{L_\infty} x(k) \quad (14)$$

$$S_\infty = A^T S_\infty A + Q - A^T S_\infty B (B^T S_\infty B + R)^{-1} B^T S_\infty A \quad (15)$$

- Note that (15) is known as **Algebraic Riccati Equation**
- From duality of estimation and control: The convergence criteria for ARE (15) are same as that considered with Kalman Filter
- The RDE (12) converges if (A, B) is *stabilizable pair*
- The converged solution gives stable controller if $(Q^{1/2}, A)$ is *detectable pair*

Extension of OLOCP to Infinite Horizon

For certain choices of Q_t , the finite horizon problem:

$$\min_{u(0), \dots, u(p-1)} \{V_p(x(0))\}$$

can be made equivalent to the infinite horizon problem:

- 1 Choose Q_t such that:

$$x^T(k+p)Q_t x(k+p) = \min_{u(\cdot)} \left\{ \sum_{i=p}^{\infty} x^T(k+i)Qx(k+i) + u^T(k+i)Ru(k+i) \right\}$$

This will lead to solving the ARE (15) as before. In other words, $Q_t = S_{\infty}$ and the equivalence to infinite horizon control is trivial.

- 2 Choose Q_t such that

$$x^T(k+p)Q_t x(k+p) = \sum_{i=p}^{\infty} x^T(k+i)Q_t x(k+i)$$

We can show that the above Q_t is a solution to **Lyapunov Equation**:

$$Q_t = Q + AQ_t A^T \quad (16)$$

- 3 Solve the finite horizon problem with $x(k+p) = 0$ as a constraint.

Extension to Output Feedback Case

In the discussion so far, we assumed that the state $x(k)$ was known or measured at every time k . In case of **output feedback**, the controller actions are based on state estimates $\hat{x}(k)$:

$$\begin{aligned} \text{Observer} \quad \hat{x}(k) &= A\hat{x}(k-1) + Bu(k-1) + \\ &\quad K \left[y(k) - C(A\hat{x}(k-1) + Bu(k-1)) \right] \\ \text{Controller} \quad u(k) &= -L\hat{x}(k) \end{aligned}$$

If we define $x_e(k) \triangleq x(k) - \hat{x}(k)$, we get:

$$\begin{bmatrix} x(k+1) \\ x_e(k+1) \end{bmatrix} = \begin{bmatrix} A - BL & -BL \\ 0 & A - KCA \end{bmatrix} \begin{bmatrix} x(k) \\ x_e(k) \end{bmatrix} \quad (17)$$

Separation Principle

Since the above equation is one-way coupled, the system is guaranteed to be stable if the controller and the filter are guaranteed to be stable *independently*.

Example

Consider the following linear system:

$$G = \frac{2.69}{(20s+1)(5s+1)} \xrightarrow{\text{c2d}} \begin{cases} x(k+1) = \begin{bmatrix} 1.268 & -0.1839 \\ 2 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} u(k) \\ y(k) = \begin{bmatrix} 0.3128 & 0.1121 \end{bmatrix} x(k) \end{cases}$$

Finite horizon problem with $p = 3$ and $Q = R = I$ and $Q_t = I$:

1 Least Squares Solution

- Solve:

$$U = -\left[S^{uT} \Gamma^x S^u + \Gamma^u\right]^{-1} S^{uT} \Gamma^x S^x x(0)$$

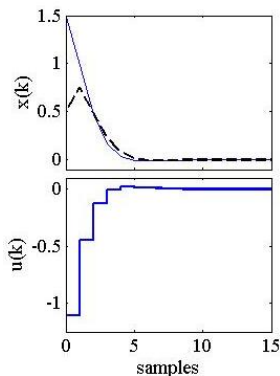
- Result: $U = [-1.1116 \quad -0.4229 \quad -0.0982]^T$
- Optimal value function $J_3(x(0)) = 6.0067$

2 Solution via Dynamic Programming

- $S(3) = Q_t$. Solve the RDE (12) recursively and obtain $u(k) = -L(k)x(k)$ for $k = 0, 1, 2$
- Input moves and value function are exactly the same as before

Extension to Infinite Horizon Case

We solve the same problem with $p = \infty$.



The state and input trajectories starting from $x(0) = [1.5 \ 0.5]^T$ from infinite horizon LQR are shown above

Dynamic Programming Solution

- S_∞ is obtained using dare and

$$u(k) = -L_\infty x(k)$$

is implemented

- $V_\infty(x(0)) = J_\infty = 6.0187$

Least Squares Solution

- The least squares problem is solved

$$U = -[S^{uT} \Gamma^x S^u + \Gamma^u]^{-1} S^{uT} \Gamma^x S^x x(k)$$

and implemented in receding horizon manner.

- $V_\infty(x(0)) = 6.0189$

Finite Horizon \rightarrow Equivalent Infinite Horizon Problem

∞ -horizon LQR: $U = \{ -1.1216, -0.4398, -0.1208, -0.0071, 0.0185, \dots \}$.

- 1 Solving finite horizon problem with $Q_t = S_\infty$
 - Exact same numerical result as above
 - Infinite horizon optimal solution with $J_\infty = 6.0187$
- 2 Solving finite horizon problem with $Q_t = A Q_t A^T + Q$
 - Assumption: system is autonomous beyond $t = p \cdot h$
 - $V_\infty(x(0)) = 6.0386$
 - Input moves: $U = \{ -1.2226, -0.4128, -0.0872, 0.0088, 0.0218, \dots \}$
- 3 Solving finite horizon problem for a large p ($p = 25$) yields same results for all deterministic cases considered here

Linear Quadratic Optimal Control

Part-B: Stochastic Case

Dr. Niket S. Kaisare

Department of Chemical Engineering
Indian Institute of Technology – Madras

January 22, 2009

State Feedback Problem

- Consider the stochastic system

$$x(k+1) = Ax(k) + Bu(k) + \varepsilon_1(k) \quad (18)$$

- $\varepsilon_1(k)$ is a zero-mean Gaussian white noise with covariance R_1 .
- State is assumed to be perfectly measured
- As before, we will consider OLOFC and OFC (using Dynamic Programming) solutions

Open Loop Optimal Solution

$$J_p^{\text{ol}} = \min_{u(\cdot)} E \left\{ \sum_{k=0}^{p-1} [x^T(k)Qx(k) + u^T(k)Ru(k)] + x^T(p)Q_t x(p) \right\} \quad (19)$$

where E is the expectation operator. E and \min do not commute in general. As before,

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(p) \end{bmatrix} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^p \end{bmatrix} x(0) + \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ A^{p-1}B & A^{p-2}B & \cdots & B \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(p-1) \end{bmatrix} + \begin{bmatrix} 0 & 0 & \cdots & 0 \\ I & 0 & \cdots & 0 \\ A & I & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ A^{p-1} & A^{p-2} & \cdots & I \end{bmatrix} \begin{bmatrix} \varepsilon(0) \\ \varepsilon(1) \\ \vdots \\ \varepsilon(p-1) \end{bmatrix} \quad (20)$$

Short-hand notations as before:

$$\mathcal{X} = \mathcal{S}^x x(0) + \mathcal{S}^u \mathcal{U} + \mathcal{S}^\varepsilon \mathcal{E} \quad (21)$$

Using this expression, we get

$$\begin{aligned} V_p &= E \{ \mathcal{X}^T \Gamma^y \mathcal{X} + \mathcal{U}^T \Gamma^u \mathcal{U} \} \\ &= E \left\{ (\mathcal{S}^x x(0) + \mathcal{S}^u \mathcal{U} + \mathcal{S}^\varepsilon \mathcal{E})^T \Gamma^y (\mathcal{S}^x x(0) + \mathcal{S}^u \mathcal{U} + \mathcal{S}^\varepsilon \mathcal{E}) + \mathcal{U}^T \Gamma^u \mathcal{U} \right\} \\ &= (\mathcal{S}^x x(0) + \mathcal{S}^u \mathcal{U})^T \Gamma^y (\mathcal{S}^x x(0) + \mathcal{S}^u \mathcal{U}) + \mathcal{U}^T \Gamma^u \mathcal{U} + E \{ \mathcal{E}^T \mathcal{S}^{\varepsilon T} \Gamma^y \mathcal{S}^\varepsilon \mathcal{E} \} \end{aligned}$$

Two points to note in the above expression:

- This differs from the deterministic case only in the blue colored part
- We can write:

$$\begin{aligned} E \{ \mathcal{E}^T \mathcal{S}^{\varepsilon T} \Gamma^y \mathcal{S}^\varepsilon \mathcal{E} \} &= E \{ \text{trace} \{ \mathcal{S}^{\varepsilon T} \Gamma^y \mathcal{S}^\varepsilon \mathcal{E}^T \mathcal{E} \} \} \\ &= \text{trace} \{ \mathcal{S}^{\varepsilon T} \Gamma^y \mathcal{S}^\varepsilon \bar{R}_1 \} \end{aligned}$$

Since the last term (blue color) in the value function does not involve \mathcal{U} , the solution is the same as the deterministic case:

$$\mathcal{U} = -(\mathcal{S}^{uT} \Gamma^y \mathcal{S}^u + \Gamma^u)^{-1} \mathcal{S}^{uT} \Gamma^y \mathcal{S}^x x_0$$

$$J_p^{\text{ol}}(x_0) = x_0^T \left[\mathcal{S}^{xT} \mathcal{S}^x - \mathcal{S}^{xT} \Gamma^y \mathcal{S}^u (\mathcal{S}^{uT} \Gamma^y \mathcal{S}^u + \Gamma^u)^{-1} \mathcal{S}^{uT} \Gamma^y \mathcal{S}^x \right] x_0 \\ + \text{trace}\{\mathcal{S}^{\varepsilon T} \Gamma^y \mathcal{S}^{\varepsilon} \bar{R}_1\}$$

Linear Quadratic Optimal Control

Constrained Linear Quadratic Regulator

Dr. Niket S. Kaisare

Department of Chemical Engineering
Indian Institute of Technology – Madras

January 22, 2009

Linear Quadratic Optimal Control

Practical Issues and Extensions

Dr. Niket S. Kaisare

Department of Chemical Engineering
Indian Institute of Technology – Madras

January 22, 2009

Topics Covered

- Practical Issues
- Multi-Parametric Programming
- General Stage-Wise Optimization Problem
- Approximate Dynamic Programming Strategies

PRACTICAL ISSUES: Setpoint Tracking

The performance function is modified to:

$$V_p = \sum_{i=0}^{\infty} \left\{ [r(k+i) - y(k+i)]^T Q [r(k+i) - y(k+i)] + u^T(k+i) R u(k+i) \right\} \quad (22)$$

- Proposed solution: Augment the model with the set point

$$\underbrace{\begin{bmatrix} x(k+1) \\ r(k+1) \end{bmatrix}}_{\tilde{x}(k+1)} = \underbrace{\begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} x(k) \\ r(k) \end{bmatrix}}_{\tilde{x}(k)} + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{\tilde{B}} u(k)$$

$$r(k) - y(k) = \begin{bmatrix} -C & I \end{bmatrix} \begin{bmatrix} x(k) \\ r(k) \end{bmatrix}$$

$$\tilde{Q} = \begin{bmatrix} -C & I \end{bmatrix}^T Q \begin{bmatrix} -C & I \end{bmatrix}$$

- At steady state, the input is not zero for offset-free tracking
- $\Delta u = 0$ at steady state for integral action. The above formulation does not guarantee integral action

The following reformulation ensures integral action:

$$V_p = \sum_{i=0}^{\infty} \left\{ [r(k+i) - y(k+i)]^T Q [r(k+i) - y(k+i)] + \Delta u^T(k+i) R \Delta u(k+i) \right\} \quad (23)$$

- Like before, the state needs to be augmented with previous input move

$$\tilde{x}(k) = \begin{bmatrix} x(k) \\ r(k) \\ u(k-1) \end{bmatrix}$$

- With this definition,

$$\tilde{A} = \begin{bmatrix} A & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \tilde{B} = \begin{bmatrix} B \\ 0 \\ I \end{bmatrix}$$

- The problem with the above reformulation is that the state $r(k)$ is **not stabilizable**. The RDE is not guaranteed to converge

Solution: Model Differencing

$$\Delta x(k+1) = A\Delta x(k) + B\Delta u(k)$$

$$\Delta y(k) = C\Delta x(k)$$

$$e(k) \triangleq y(k) - r(k)$$

↓

$$\begin{aligned} e(k+1) &= [y(k) + \Delta y(k+1)] - r(k+1) \\ &= e(k) + \Delta y(k+1) \end{aligned}$$

↓

$$\underbrace{\begin{bmatrix} \Delta x(k+1) \\ e(k+1) \end{bmatrix}}_{\tilde{x}(k+1)} = \underbrace{\begin{bmatrix} A & 0 \\ CA & I \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} \Delta x(k) \\ e(k) \end{bmatrix}}_{\tilde{x}(k)} + \underbrace{\begin{bmatrix} B \\ CB \end{bmatrix}}_{\tilde{B}}$$

With this new definition:

$$\tilde{Q} = \begin{bmatrix} 0 & I \end{bmatrix}^T Q \begin{bmatrix} 0 & I \end{bmatrix}$$

PRACTICAL ISSUES: Disturbance Rejection

The objective function remains the same, though the linear model is:

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

As before, the steady state value of u_∞ is non-zero as long as the disturbance is non-zero.

As a result, $\Delta u(k)$ needs to be penalized as in (23). The following two redefinitions will achieve this:

$$\tilde{x}(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \quad \text{and} \quad \tilde{x}(k) = \begin{bmatrix} \Delta x(k) \\ e(k) \end{bmatrix}$$

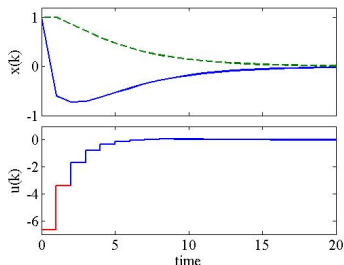
In this case, there are no issues with stabilizability of the new state $\tilde{x}(k)$ and both the reformulations work.

However, for sake of consistency (and to solve servo + regulatory problems), the second definition is the preferred definition.

MULTI-PARAMETRIC PROGRAMMING

- Solution to LQR problem involves solving the ARE (15) and obtaining the input move as $u(k) = -L(k)x(k)$
- In presence of constraints, the control move is no longer linear
- Nonlinearity stems from constraint violation

Consider control of $\frac{2}{s^2+3s+2}$ (sampled at $h = 0.1$)



With constraints $-2 \leq u \leq 2$, the red line will violate constraints. LQR still gives the blue part of the inputs (which satisfy constraints).

The Constrained LQR Problem

Recall that we are interested in computing control moves $u(k), u(k+1), \dots, u(k+p-1)$. The following possibilities exist:

- All constraints are satisfied. Thus LQR result gives the solution
- At least one constraint is violated at the current time k . The solution lies at that active constraint
- Constraints are violated at some time, $k+i$, $i \neq 0$ in the horizon

For each of the cases, the optimal move $u(k)$ is a linear function of $x(k)$, valid over a convex polygon.

The goal of parametric programming is to obtain the optimal control action as an explicit function of the state: $u(k) = U^*(x(k))$. The state x is treated as a parameter and the LQ problem solved to obtain u as a **piece-wise affine function** of the state.

Multi-Parametric Programming: The Set-up

Consider the typical constrained LQ problem

$$\min_{u(\cdot)} \sum_{k=0}^{p-1} [x^T(k)Qx(k) + u^T(k)Ru(k)] + x^T(p)Q_t x(p)$$

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

$$M_x x(k+i) \leq m_x$$

$$M_u u(k+i) \leq m_u$$

While we prefer to have p as large as possible, large p increase complexity of the problem significantly. Hence, we choose a small enough p (eg., $p = 2$) and use Q_t as from the Lyapunov equation $Q_t = A Q_t A^T + Q$.

As done earlier,

$$\mathcal{X} = \mathcal{S}^x x(k) + \mathcal{S}^u \mathcal{U}$$

And the objective function is

$$J_p = x^T(0)Yx(0) + \min \left\{ \mathcal{U}^T \mathcal{H} \mathcal{U} + 2x^T(0)g^T \mathcal{U} \right\}$$

Solution to the unconstrained problem is:

$$\mathcal{U}^{uc} = -\mathcal{H}^{-1}gx(0)$$

Let us define a new variable

$$z \triangleq \mathcal{U} - \mathcal{U}^{uc} = \mathcal{U} - \mathcal{H}^{-1}gx(0)$$

The solution to the unconstrained problem will be $z = \mathbf{0}$ and

$$J_p = \min \left\{ z^T \mathcal{H} z \right\}$$

Question: But what about constraints?

Like in MPC class, we express all constraints in the form:

$$\mathcal{C}\mathcal{U} \leq W + Ex(k)$$

\Downarrow

$$Gz \leq W + Sx(k) \quad \text{where, } S = E + \mathcal{C}\mathcal{H}^{-1}g; \quad G = \mathcal{C}$$

Thus, we now have the reformulated problem

$$J_p = \min \left\{ z^T \mathcal{H} z \right\} \Rightarrow \begin{cases} \mathcal{H} z + G^T \lambda = 0 \\ \lambda_i (G_i z - W_i - S_i x(k)) = 0 \\ \lambda \geq 0 \\ W + Sx(k) \geq Gz \end{cases} \quad (24)$$

The above equation is nothing but Karush-Kuhn-Tucker conditions for optimality. The above can be solved using Matlab command `quadprog`

Practical Implementation in Matlab

From the Matlab help file:

`[X,FVAL,FLAG,OUTPUT,LAMBDA] = QUADPROG(H,f,A,b)`

- H is just \mathcal{H} ; f is $\mathbf{0}$
- A is just G; b is $W + Sx$
- X is just the optimal solution z
- LAMBDA.ineqlin is just λ

The next procedure is fairly straightforward:

- 1 If all $\lambda = 0$:
 - All constraints are satisfied and $z = 0$ is the solution
 - Thus, obtain $u = -Lx$ as the solution
 - Note $z = 0$ implies $-Sx \leq W$. This gives the polygon region over which the above solution is valid
- 2 If $\lambda > 0$ for any of the current constraints
 - The corresponding constraint is active
 - The optimal solution is to satisfy that constraint as equality constraint.
- 3 If for some future constraints $\lambda > 0$
 - Use the procedure given in Pistikopolous et al. (2002) to compute the optimal control move
 - Use this to satisfy constraints $Gz \leq W + Sx$ and $\lambda \geq 0$

Note: While I have given this discussion to promote understanding, item #3 above is used to obtain solutions and convex polygon in all cases

Implementation

- Start with some $x \in \mathcal{X}$
- Use procedure in the previous slide to obtain optimal control move and the constraints that need satisfied
- Remove redundant constraints to obtain a convex polygon CR^i
- Define rest of the region as $\mathcal{X} - CR^i$
- Choose a new x the rest of the region and continue till entire state space is covered

GENERAL STAGE-WISE OPTIMIZATION PROBLEM

- **Stage-wise decision making** is more general form of optimal control
- Instead of looking at regulation, the general problem is that of finding the control actions for a multi-stage-ahead problem subject to various constraints
- Problem: To find control moves $u(k)$ that minimize:

$$\min_{u(\cdot)} \left\{ \sum_{i=0}^{\infty} \mu_i(x(k), u(k)) \right\} \quad (25)$$

Subject to

- Model constraint

$$x(k+1) = f(x(k), u(k)) \quad (26)$$

Note that there are no restrictions on the type of model except that given state $x(k)$ and control actions $u(k)$, we should be able to generate the state $x(k+1)$ at the next stage.

- Inequality constraints

$$h(x(k), u(k)) \leq 0 \quad \text{and} \quad h_e(x(k), u(k)) = 0$$

Relation to Constrained LQ Problem

For an infinite horizon Linear Quadratic control problem:

Single stage cost

$$\mu_i(x(k), u(k)) = x^T(k+i)Qx(k+i) + u^T(k+i)Ru(k+i)$$

Model constraint

$$f(x(k), u(k)) \equiv x(k+1) = Ax(k) + Bu(k)$$

Linear input, output and state constraints

$$h(x(k), u(k)) \leq 0 \equiv \begin{bmatrix} C_1 u(k+i) \\ C_2 C x(k+i) \\ C_3 x(k+i) \end{bmatrix} \leq \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Figure showing philosophy of DP

The optimal solution to a stage-wise optimization problem involves minimizing (or maximizing) the stage-wise cost to reach an intermediate state plus the minimum (or maximum) cost required to go from this state to the terminal state

Dynamic Programming (DP)

As discussed in LQ control case, DP reduces a multi-step problem into an equivalent single step problem through the definition of **cost-to-go**. Since cost-to-go is the optimal cost to go from the next state to the terminal state:

$$J(x(k)) = \min_{u(k), u(k+1), \dots} \left\{ \sum_{i=0}^{\infty} \phi(x(k+i), u(k+i)) \right\} \quad (27)$$

This definition reduces the multi-step problem to the following single-step problem

$$J(x(k)) = \min_{u(k)} \left\{ \phi(x(k), u(k)) + J(x(k+1)) \right\} \quad (28)$$

And the control move that solves the above problem is optimal at the current time

$$u(k) = \mu(x(k)) \triangleq \arg \min_{u(k)} \left\{ \phi(x(k), u(k)) + J(x(k)) \right\} \quad (29)$$

Obtaining Cost-to-go: Value Iteration

The objective of Dynamic Programming is to obtain a solution to cost-to-go function.

- Start with an initial guess of $J(x)$
- For $x(k) \in \mathcal{X}^{\text{feasible}}$, solve

$$J(x(k)) = \min_{u(k)} \{ \phi(x(k), u(k)) + J(x(k+1)) \} \quad (30)$$

where $x(k+1)$ is obtained from the model constraints:

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

- Repeat this for all possible $x \in \mathcal{X}^{\text{feasible}}$

The above procedure is known as **Value Iteration**.

As the state dimension increases, there is an exponential increase in the number of possible $x \in \mathcal{X}$ for which we need to compute cost-to-go.

This exponential increase in computational load is termed as **curse of dimensionality**.

Approximate Dynamic Programming (ADP)

ADP aims to alleviate the curse of dimensionality by obtaining an approximation of the optimal cost-to-go as a function of the state $x \in \mathcal{X}$.

- Iterations of the Bellman equation are performed only on selected “important” states and not the entire state space.
- States selected for iterations are the ones visited during closed-loop simulations using a (suboptimal) control policy. Such states are considered more important than the rest.
- Simulations of the suboptimal control policy generates initial guess for cost-to-go
- The cost-to-go is expressed as a function of state x using a nonlinear approximator, such as neural network, k -nearest neighbor, etc.

Procedure: Initial Cost-to-go Approximation

- 1 Choose a suboptimal control policy (eg., PID, linear MPC etc.)
The important feature of the control policy is that it should cover the desired region of the state space appropriately.
- 2 Perform simulations using the chosen control policy. These simulations result in $u(k) \in \mathcal{U}^{\text{cl}}$, $y(k) \in \mathcal{Y}^{\text{cl}}$ and $x(k) \in \mathcal{X}^{\text{cl}}$ data. Here $(\cdot)^{\text{cl}}$ stands for initial closed-loop data.
- 3 Compute cost-to-go values from simulations using the definition:

$$J(x(k)) \approx V^{\text{cl}} = \sum_{i=0}^{\infty} \phi(x(k+i), u(k+i)) \quad \forall x(k) \in \mathcal{X}^{\text{cl}}$$

- 4 Use an appropriate function approximator that can interpolate the cost-to-go values from the $x(k)$ vs. $J(k)$ data. Let $\tilde{J}^i(x(k))$ represent this function approximation (with $\tilde{J}^0 = J^{\text{cl}}$).

From this stage, we obtain $x(k) \in \mathcal{X}^{\text{cl}}$ and $\tilde{J}^0(x)$ for the iterations stage.

Summary: Value Iteration

- 1 For each state $x(k) \in \mathcal{X}^{\text{cl}}$ solve the Bellman equation:

$$J^{i+1}(x(k)) = \min_{u(k)} \{ \phi(x(k), u(k)) + \tilde{J}^i(x(k+1)) \}$$

$$\text{where } x(k+1) = f(x(k), u(k))$$

subject to all state and input constraints

- 2 Use function approximator to obtain $\tilde{J}^{i+1}(x)$
- 3 Iterate on the Bellman equation until $\tilde{J}(x(k))$ converges
- 4 Once the iterations converge, use the cost-to-go in online control

$$u(k) = \arg \min_{u(k)} \{ \phi(x(k), u(k)) + \tilde{J}(x(k)) \}$$

Policy Iteration

An alternative is to iterate on the optimal policy $\mu(x(k))$ instead of the optimal cost-to-go function.

Policy iteration is a two-step process:

Policy Update:

$$\mu^i(x(k)) = \arg \min_{u(k)} \left\{ \phi(x(k), u(k)) + \tilde{J}^i(x(k+1)) \right\} \quad (31)$$

Value Function Update:

$$\begin{aligned} \tilde{J}^{i+1}(x(k)) &= \phi(x(k), \mu^i(x(k))) + \tilde{J}^{i+1}(x(k+1)) \\ \text{with } x(k+1) &= f(x(k), \mu^i(x(k))) \end{aligned} \quad (32)$$

Note that the value update equation has \tilde{J}^{i+1} on both sides. Thus, value update itself is also an iterative step.

