Part IIB/EIST Part II, Module 4F2

Robust Multivariable Control

Guy-Bart Stan

HANDOUT 1

Optimal control and Dynamic Programming

G.-B. Stan 2009

Course Overview

In the first part of the course you have been shown the theory behind the use of various tools in multivariable robust control; in this part of the course we shall study the algorithms that lie behind these tools. We shall consider:

1. Optimal Control with complete information
   – accurate models and full state feedback.

2. Optimal Control with imperfect observations
   – (but still with accurate models).

3. Feedback system design, with inaccurate models and imperfect observations.

   Salutary warning: "optimisation can expose the weaknesses in thinking which are usually compensated for by soundness of intuition"
   (Whittle)

Contents

1.1 Notation

- $\mathbb{R}^n$ denotes the n-dimensional Euclidean space. This is a **vector space** (also known as a linear space). If $n = 1$, we will drop the subscript and write just $\mathbb{R}$ (the set of real numbers or "the real line").
- $x \in A$ is a shorthand for "x belongs to a set A", e.g. $x \in \mathbb{R}^n$ means that x is an n-dimensional vector.
- If $a, b \in \mathbb{R}$ with $a \leq b$, $[a, b]$ will denote the **closed interval** from a to b, i.e. the set of all real numbers x such that $a \leq x \leq b$.
- I will make no distinction between vectors and real numbers in the notation (no arrows over the letters, bold fond, etc.). Both vectors and real numbers will be denoted by lower case letters, e.g. $x \in \mathbb{R}^n$ or $y \in \mathbb{R}$.
- Matrices will be denoted by upper case letters, e.g. $A \in \mathbb{R}^{n \times m}$ is an matrix with real entries, n rows and m columns.
- $A \in \mathbb{R}^{n \times n}$ is called **symmetric** if $A^T = A$ (it remains the same if we interchange rows and columns. Symmetric matrices have real eigenvalues. A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is called **positive definite** (denoted as $A > 0$) if $x^T A x > 0$ for all $x \neq 0$. This is equivalent to all the eigenvalues of A being greater than zero. In particular it implies that A^{-1} exists. A is called **positive semi-definite** (denoted as $A \geq 0$) if $x^T A x \geq 0$ for all x.
- $f(\cdot) : A \to B$ is a shorthand for a function mapping every element $x \in A$ to an element $f(x) \in B$. Likewise, $f(\cdot, \cdot) : A \times B \to C$ is a function that takes an element of $a \in A$ and an element $b \in B$ and produces an element $f(a, b) \in C$.

1.2 Discrete-time optimal control

(an introduction to the use of Dynamic Programming)

States and Inputs: $x \in X$, $u \in U$ (e.g. $X = \mathbb{R}^n$, $U = \mathbb{R}^m$).

Dynamics:

$$x_{k+1} = f(x_k, u_k)$$
x_0 given

$$\begin{cases} \text{discrete-time state-space system} \\ f(\cdot,\cdot) : X \times U \to X \\ \text{e.g.} \end{cases}$$

Trajectory: Given x_0, each input sequence $u_0, u_1, \ldots, u_{h-1}$ generates a state sequence $x_0, x_1, \ldots, x_h$ starting at x_0 and such that $x_{k+1} = f(x_k, u_k)$ for $k = 0, 1, \ldots, h-1$.

Finite Horizon Cost Function:

$$J(\underbrace{x_0}, \underbrace{u_0, u_1, \ldots, u_{h-1}}) = \sum_{k=0}^{h-1} \underbrace{c(x_k, u_k)} + \overbrace{J_h(x_h)}$$

Objective: Find the "best" input sequence $u_0^*, u_1^*, \ldots, u_{h-1}^*$,

$$\begin{aligned} J^*(x_0) &= J(x_0, u_0^*, u_1^*, \ldots, u_{h-1}^*) \\ &= \min_{u_0, u_1, \ldots, u_{h-1}} J(x_0, u_0, \ldots, u_{h-1}) \end{aligned}$$

Technical Assumptions: Omitted here, but necessary because

- J^* may not be well defined.
- u_0^*, u_1^*, $\ldots$, u_{h-1}^* may not exist, or may be non-unique.

Solution: Define a function $V(\cdot,\cdot) : X \times \{0,\dots,h\} \to \mathbb{R}$ by

$$V(x,k) := \min_{u_k,\dots,u_{h-1}} \left(\sum_{i=k}^{h-1} c(x_i,u_i) + J_h(x_h) \right)$$

where $x_k, x_{k+1}, \dots, x_h$ is the sequence of states generated by $u_k, u_{k+1}, \dots, u_{h-1}$ starting with $x_k = x$. $V(x,k)$ is known as the *value function* or *cost to go*. It is the optimal additional cost from the k^{th} step on, if the state at the k^{th} step is equal to x.

Clearly
$$\begin{cases} V(x,h) = J_h(x), & \text{final cost} \\ V(x,0) = J^*(x), & \text{optimal cost of original problem, } x_0 = x \end{cases}$$

In fact, it is possible to work backwards and compute $V(x,h-1)$ in terms of $V(x,h)$ and then $V(x,h-2)$ in terms of $V(x,h-1)$ etc, until we get back to $V(x,0)$. The usual derivation relies on the *principle of optimality*:

> Assume that the optimal controls u_0^*, u_1^*, ..., u_{h-1}^* lead us from x_0 to x_k at step k. Then the truncated sequence u_k^*, ..., u_{h-1}^* is a solution for the truncated problem

$$\min_{u_k,u_{k+1},\dots,u_{h-1}} \left(\sum_{i=k}^{h-1} c(x_i,u_i) + J_h(x_h) \right)$$

(but we shall use simple algebra for the derivation)

Assume we know $V(x, k+1)$ for all x. Try to compute $V(x, k)$.

$$V(x,k) = \min_{u_k, u_{k+1}, \ldots, u_{h-1}} \Big(\sum_{i=k}^{h-1} c(x_i, u_i) + J_h(x_h) \Big)$$

$$= \min_{u_k, u_{k+1}, \ldots, u_{h-1}} \Big(c(x_k, u_k) + \sum_{i=k+1}^{h-1} c(x_i, u_i) + J_h(x_h) \Big)$$

$$= \min_{u_k} \Big(\min_{u_{k+1}, \ldots, u_{h-1}} \Big(c(x, u_k) + \sum_{i=k+1}^{h-1} c(x_i, u_i) + J_h(x_h) \Big) \Big)$$

$$= \min_{u_k} \Big(c(x, u_k) + \min_{u_{k+1}, \ldots, u_{h-1}} \Big(\sum_{i=k+1}^{h-1} c(x_i, u_i) + J_h(x_h) \Big) \Big)$$

$$= \min_{u_k} \Big(c(x, u_k) + V(x_{k+1}, k+1) \Big)$$

$$= \min_{u_k} \Big(c(x, u_k) + V\big(f(x, u_k), k+1\big) \Big)$$

Recursion, expressing $V(x, k)$ in terms of $V(x, k+1)$.

Summary: We can find the optimal cost and the optimal control simultaneously, by solving the Dynamic Programming equation

$$V(x,k) = \min_u \Big(c(x,u) + V\big(f(x,u),k+1\big)\Big), \qquad k = h-1, h-2, \ldots, 1, 0 \quad (1.1)$$

with the *final* (or *terminal*) condition

$$V(x,h) = J_h(x)$$

The optimal cost is then given by

$$J^*(x_0) = \min_{u_0,u_1,\ldots,u_{h-1}} J(x_0,u_0,u_1,\ldots,u_{h-1}) = V(x_0,0).$$

The optimal input u_k at each step is the value which minimises (1.1) for the current value of the state x_k. If we define

$$g(x,k) = \arg\min_u \Big(c(x,u) + V\big(f(x,u),k+1\big)\Big)$$

then the optimal control is given by

$$u_k^* = g(x_k,k), \quad k = 0,1,\ldots,h-1$$

Notes:

- $\arg\min$ denotes the value which achieves the minimum.
- Backwards recursion in h. To solve it, we first find $V(x,h-1)$ using $V(x,h)$ (by solving the optimisation problem (1.1) for each x), then we find $V(x,h-2)$ using $V(x,h-1)$ etc, all the way back to $V(x,0)$.
- Have converted minimisation over a sequence of h inputs to a sequence of h minimisations over 1 input (but over all states).
- Optimal controls are given by time varying state feedback.
- Solution has been computed for ALL x_0.

Dynamic programming provides a systematic procedure for approaching optimal control problems. Whether or not the procedure can be implemented in practice (say on a computer) depends on whether or not the optimisation problem defined by (1.1) can be solved for all x.

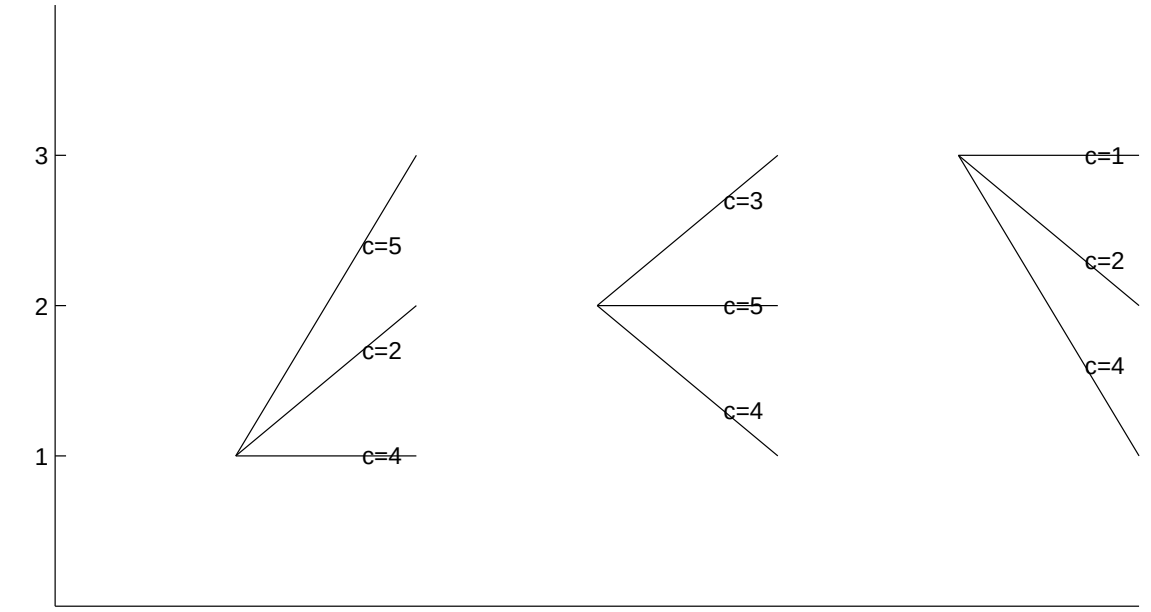
For certain classes of systems the problem can always be solved. For example, if the state and input can only take a finite number of values the optimisation can be performed by enumeration.

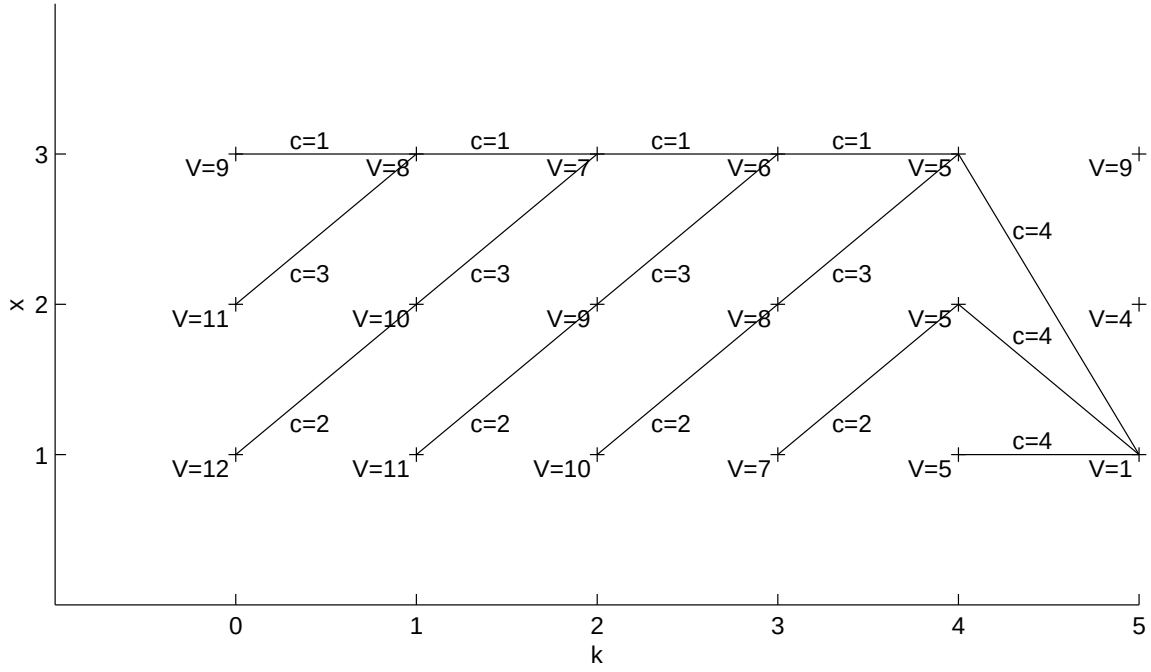
Example: Let $U = \{1, 2, 3\}$, $X = \{1, 2, 3\}$ and

$$\begin{aligned} x_{k+1} &= u_k \\ c(x, u) &= C_{xu} \\ J_h(x) &= x^2 \\ h &= 5; \end{aligned}$$

where

$$C_{xu} = \begin{array}{c|ccc} x \backslash u & 1 & 2 & 3 \\ \hline 1 & 4 & 2 & 5 \\ 2 & 4 & 5 & 3 \\ 3 & 4 & 2 & 1 \end{array}$$


3
2
1
c=5
c=2
c=4
c=3
c=5
c=4
c=1
c=2
c=4


3
x 2
1
0
1
2
3
4
5
k
V=9
V=8
V=7
V=6
V=5
V=9
V=11
V=10
V=9
V=8
V=5
V=4
V=12
V=11
V=10
V=7
V=5
V=1
c=1
c=3
c=2
c=4

1.3 Dynamic programming with disturbances

Dynamics:

$$x_{k+1} = f(x_k, u_k, w_k)$$
$$x_0 \text{ given}$$

Finite Horizon Cost Function:

$$J(x_0, \{u\}, \{w\}) = \sum_{k=0}^{h-1} c(x_k, u_k, w_k) + J_h(x_h)$$

1.3.1 Stochastic Disturbances

If the w_k are random variables, then we can minimise the expected cost:

$$J^*(x_0) = \min_{u_0} E_{w_0} \min_{u_1} E_{w_1} \cdots \min_{u_{h-1}} E_{w_{h-1}} J(x_0, \{u\}, \{w\})$$

Dynamic programming equation becomes

$$V(x,k) = \min_u E_w \Big(c(x,u,w) + V\big(f(x,u,w), k+1\big) \Big)$$

where

$$V(x,k) = \min_{u_k} E_{w_k} \cdots \min_{u_{h-1}} E_{w_{h-1}} \sum_{i=k}^{h-1} c(x_i, u_i, w_i) + J_h(x_h)$$

1.3.2 Worst case disturbances

(2 Player noncooperative dynamic game)
We can also minimise the worst case cost:

$$J^*(x_0) = \min_{u_0} \max_{w_0} \min_{u_1} \max_{w_1} \cdots \min_{u_{h-1}} \max_{w_{h-1}} J(x_0, \{u\}, \{w\})$$

Dynamic programming equation becomes

$$V(x,k) = \min_{u} \max_{w} \Big(c(x,u,w) + V\big(f(x,u,w), k+1\big) \Big)$$

where

$$V(x,k) = \min_{u_k} \max_{w_k} \cdots \min_{u_{h-1}} \max_{w_{h-1}} \sum_{i=k}^{h-1} c(x_i, u_i, w_i) + J_h(x_h)$$

Can solve a wide variety of problems exactly using these techniques (e.g. shortest path problems, optimal scheduling ...)
In addition, most optimal control problems (for example, with x_k and u_k real vectors) can be approximated arbitrarily closely by "discrete" problems, and therefore solved by enumeration. The computation is likely to be horrendous! Fortunately, for certain classes of plants and cost functions we might expect to do rather better (obtain an analytical solution).

1.4 Linear Quadratic Regulator

States and Inputs: $x \in X = \mathbb{R}^n$, $u \in U = \mathbb{R}^m$.

Dynamics:

$$x_{k+1} = Ax_k + Bu_k$$
$$x_0 \text{ given}$$

Cost Function:

$$J(x_0, u_0, u_1, \ldots, u_{h-1}) = \sum_{k=0}^{h-1} \left(x_k^T Q x_k + u_k^T R u_k \right) + x_h^T X_h x_h$$

Q, R, X_h are symmetric matrices, with $Q \geq 0$, $R > 0$ and $X_h \geq 0$ (recall that $R > 0$ means that $u^T R u > 0$ for all $u \neq 0$, and also implies that R^{-1} exists).

Solution: In this case, we can solve the minimisation involved in the Dynamic Programming equation explicitly. We first need a Lemma on the minimisation of quadratic forms.

Lemma 1: [Minimisation of Quadratic Forms.]

Given symmetric matrices Q, R, with $R > 0$, then

$$\min_u \overbrace{\begin{bmatrix} x^T & u^T \end{bmatrix} \begin{bmatrix} Q & S^T \\ S & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}} = x^T (Q - S^T R^{-1} S) x$$

and the minimum is achieved at

$$u = -R^{-1} S x$$

proof:

$$\begin{bmatrix} x^T & u^T \end{bmatrix} \begin{bmatrix} Q & S^T \\ S & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} =$$

The Dynamic Programming equation (1.1) in this case becomes,

$$V(x,k) = \min_u \Big(\underbrace{x^T Q x + u^T R u} + V(\overbrace{Ax + Bu}, k+1) \Big)$$

So,

$$\begin{aligned} & V(x, h-1) \\ & \quad = \min_u \Big(x^T Q x + u^T R u + \overbrace{(Ax+Bu)^T X_h (Ax+Bu)} \Big) \\ & \quad = \min_u \begin{bmatrix} x^T & u^T \end{bmatrix} \begin{bmatrix} Q + A^T X_h A & A^T X_h B \\ B^T X_h A & R + B^T X_h B \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \\ & \quad = x^T \underbrace{\left(Q + A^T X_h A - A^T X_h B (R + B^T X_h B)^{-1} B^T X_h A\right)} x \end{aligned}$$

– another quadratic form in x

i.e.

$$V(x,h) = x^T X_h x$$

$$V(x,h-1) = x^T X_{h-1} x,$$

$$V(x,h-2) = x^T X_{h-2} x,$$

$$\vdots$$

$$V(x,0) = x^T X_0 x,$$

A tedious calculation shows that, in fact, $X_k \geq 0, \quad k = h, h-1, \ldots, 0.$

Summary: To solve the Linear Quadratic Regulator problem solve the backwards difference equation

$$X_{k-1} = Q + A^T X_k A - A^T X_k B(R + B^T X_k B)^{-1} B^T X_k A.$$

The optimal cost is then

$$x_0^T X_0 x_0,$$

and is achieved by the optimal state-feedback control

$$u_k = -(R + B^T X_{k+1} B)^{-1} B^T X_{k+1} A x_k$$

1.5 Continuous-Time Dynamic Programming

States and Inputs: $x \in \mathbb{R}^n$, $u \in U \subseteq \mathbb{R}^m$.

Dynamics:

$$\dot{x} = f(x,u), \quad x(0) = x_0 \text{ given}$$

$$\begin{cases} \text{continuous-time state-space system} \\ f(\cdot,\cdot) : \mathbb{R}^n \times U \to \mathbb{R}^n \\ \text{e.g.} \end{cases}$$

Trajectory: Given $x_0 \in X$ and a horizon $T \geq 0$, each input function $u(\cdot) : [0,T] \to U$ generates a state trajectory $x(\cdot) : [0,T] \to \mathbb{R}^n$ such that $x(0) = x_0$ and $\forall t \in [0,T] : \dot{x}(t) = f(x(t),u(t))$.

Cost Function:

$$J(x_0, u(\cdot)) = \int_0^T c(x(t),u(t))dt + J_T(x(T))$$

Objective: Find the "best" input function $u^*(\cdot) : [0,T] \to U$,

$$J^*(x_0) = J(x_0, u^*(\cdot)) = \min_{u(\cdot)} J(x_0, u(\cdot))$$

Technical Assumptions: on f, U, c, J_h needed.

- Does a unique trajectory $x(\cdot) : [0,T] \to \mathbb{R}^n$ exist?
- Does J^* exist?
- Does $u^*(\cdot) : [0,T] \to U$ exist?

Derivation: Approximate the continuous dynamics by a discrete-time system obtained by sampling every h "time units". The plant equation implies that

$$x(t+h) = x(t) + f(x(t), u(t))h + O(h^2)$$

and the incremental cost accumulated between t and $t+h$ is

$$\int_t^{t+h} c(x(\tau), u(\tau))\, d\tau = c(x(t), u(t))h + O(h^2).$$

So, the Dynamic Programming equation (1.1) becomes

$$V(x,t) = \min_{u \in U} \left(c(x,u)h + V(x + f(x,u)h, t+h) \right) + O(h^2)$$

with $V(x,T) = J_T(x)$.

Consider the Taylor series expansion

$$V(x+\delta x, t+\delta t) \;=\; V(x,t) + \frac{\partial V}{\partial x}\delta x + \frac{\partial V}{\partial t}\delta t + \text{higher order terms}$$

Recall that

$$\frac{\partial V}{\partial x} := \underbrace{\left[\begin{array}{ccccc} \frac{\partial V}{\partial x_1} & \frac{\partial V}{\partial x_2} & \frac{\partial V}{\partial x_3} & \cdots & \frac{\partial V}{\partial x_n} \end{array} \right]}$$

where

$$x = \left[\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \right]$$

Substituting into the Dynamic Programming equation we get

$$V(x,t) = \min_{u \in U} \Bigg(c(x,u)h + V(x,t) + \frac{\partial V(x,t)}{\partial x} f(x,u)h + \frac{\partial V(x,t)}{\partial t} h \Bigg) + O(h^2)$$

Since $V(x,t)$ and $\frac{\partial V(x,t)}{\partial t}$ do not depend on u,

$$-\frac{\partial V(x,t)}{\partial t} = \min_{u \in U} \left(c(x,u) + \frac{\partial V(x,t)}{\partial x} f(x,u) \right) + \frac{O(h^2)}{h}$$

Take limit as $h \to 0$. Recall that $\lim_{h \to 0} \frac{O(h^2)}{h} = 0$.

Summary: To find $V(\cdot,\cdot) : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ solve the PDE

$$-\frac{\partial V(x,t)}{\partial t} = \min_{u \in U} \left(c(x,u) + \frac{\partial V(x,t)}{\partial x} f(x,u) \right) \tag{1.2}$$

with boundary condition

$$V(x,T) = J_T(x).$$

The optimal cost is then given by $V(x_0, 0)$, and the optimal input by

$$u^*(t) = g(x(t), t)$$

where

$$g(x,t) = \arg\min_{u \in U} \left(c(x,u) + \frac{\partial V(x,t)}{\partial x} f(x,u) \right)$$

Notes:

- Equation (1.2) is known as the Hamilton-Jacobi-Bellman PDE. It is the infinitesimal version of (1.1).
- Have turned optimisation over $u(\cdot)$ as a function of time to pointwise optimisation over $u \in U$ (and for all x).
- Optimal control in time varying state feedback form.
- To solve the problem one needs to solve a partial differential equation. Technical difficulties: does a solution exist? In what sense? Can it be computed?
- For certain classes of systems many of these technicalities are easily resolved.

1.6 Continuous-Time Linear Quadratic Regulator

States and Inputs: $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$

Plant:

$$\dot{x} = Ax + Bu,$$
$$x(0) = x_0$$

Cost Function:

$$J\left(x_0, u(\cdot)\right) = \int_0^T c(x,u)\,dt + J_T(x(T))$$

where

$$c(x,u) = x^T Q x + u^T R u, \quad J_T(x) = x^T X_T x$$
$$R = R^T > 0, Q = Q^T \geq 0, X_T = X_T^T \geq 0$$

Solution: (summary) Solve the ODE (Riccati equation)

$$-\dot{X} = Q + XA + A^T X - XBR^{-1}B^T X$$

(backwards in time) with terminal condition

$$X(T) = X_T.$$

The optimal cost is then given by

$$x_0^T X(0) x_0$$

and the optimal input is given by

$$u(t) = -R^{-1}B^T X(t) x(t).$$

Derivation:

LQR Example

$$G(s) = \frac{s-1}{(s-2)(s+2)}$$

with a state-space realization:

$$A = \begin{bmatrix} 0 & 1 \\ 4 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = 0$$

and initial condition

$$x(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Cost Function:

$$\int_0^T \left(y(t)^T y(t) + u(t)^T u(t) \right) dt + x(T)^T X_T x(T)$$

with $T = 4$

So,

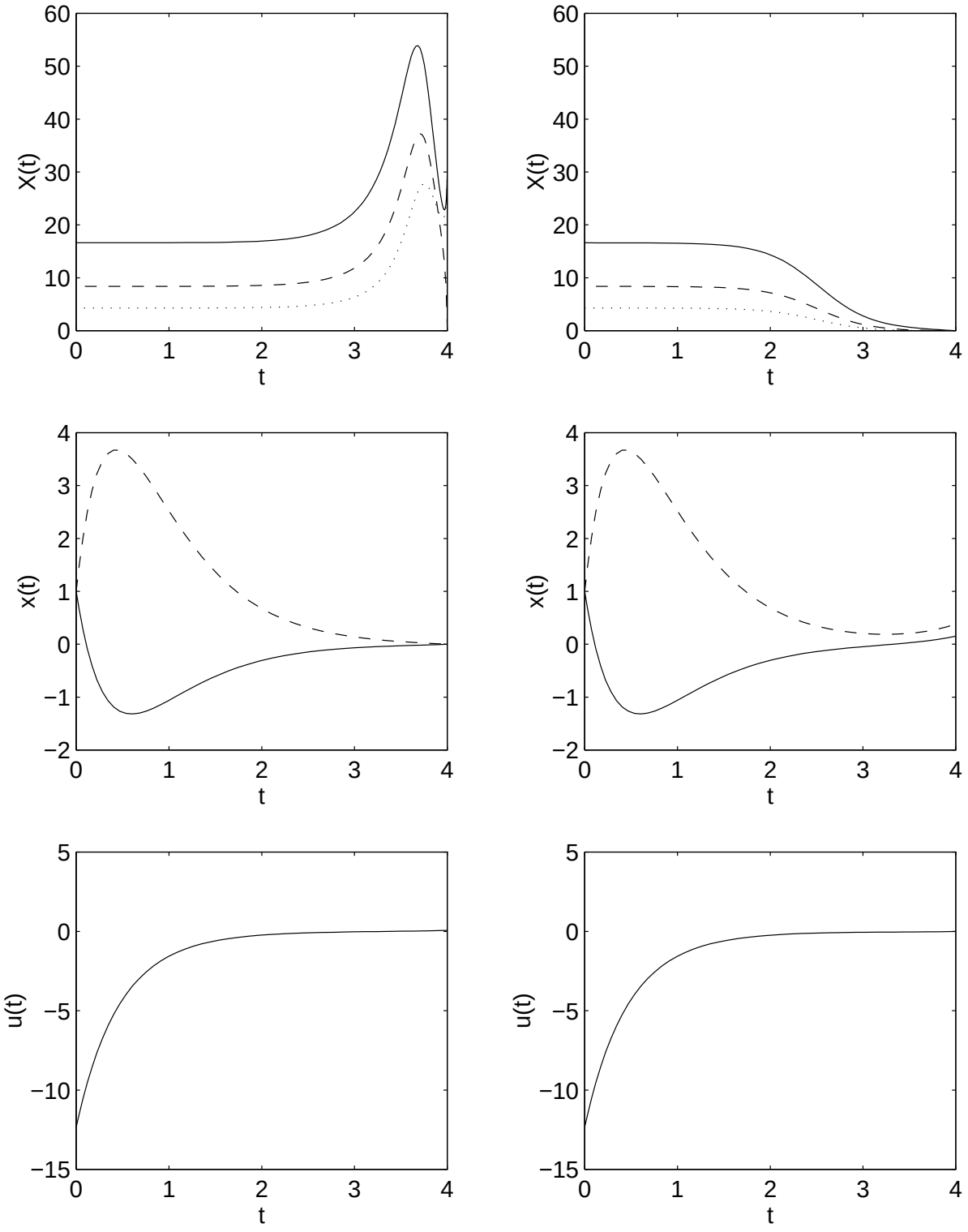
$$Q = C^T C, \quad R = 1$$

1. $X_T = \begin{bmatrix} 30 & 0 \\ 0 & 30 \end{bmatrix} \rightarrow X(0) = \begin{bmatrix} 16.6176 & 8.3725 \\ 8.3725 & 4.2804 \end{bmatrix}$

   Optimal Cost=37.6430

2. $X_T = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \rightarrow X(0) = \begin{bmatrix} 16.6140 & 8.3706 \\ 8.3706 & 4.2793 \end{bmatrix}$

   Optimal Cost=37.6245


X(t)
t
x(t)
u(t)