Lu Sun, and many more.

A Notebook on Control System

To my family, friends and communities members who have been dedicating to the presentation of this notebook, and to all students, researchers and faculty members who might find this notebook helpful.

Contents

Foreword

If software and e-books can be made completely open-source, why not a notebook?

This brings me back to the summer of 2009 when I started my third year as a high school student in Harbin No. 3 High School. In the end of August when the results of Gaokao (National College Entrance Examination of China, annually held in July) were released, people from photocopy shops would start selling notebooks photocopies that they claimed to be from the top scorers of the exam. Much curious as I was about what these notebooks look like, never have I expected myself to actually learn anything from them, mainly for the following three reasons.

First of all, some (in fact many) of these notebooks were more difficult to read than the textbooks. I guess we cannot blame the top scorers for being so smart that they sometimes made things extremely brief or overwhelmingly complicated.

Secondly, why would I want to adapt to notebooks of others when I had my own notebooks which in my opinion should be just as good as theirs.

And lastly, as a student in the top-tier high school myself, I knew that the top scorers were probably my schoolmates. Why would I pay money to a stranger in a photocopy shop for my friends' notebooks, rather than requesting a copy from them directly?

However, my mind changed after becoming an undergraduate student in 2010. There were so many modules and materials to learn for a college student, and as an unfortunate result, students were often distracted from digging deeply into a module (and for those who were still able to do so, you have my highest respect). The situation became worse when I started pursuing my Ph.D. in 2014. As I had to focus on specific research areas entirely, I could hardly split enough time on other irrelevant but still important and interesting contents.

To make a difference, I enforced myself reading articles beyond my comfort zone, which ended up motivating me to take notes to consolidate the knowledge. I used to work with hand-written notebooks. My very first notebook was on Numerical Analysis, an entrance-level module for engineering background graduate students. Till today I still have dozens of these notebooks on my bookshelf. Eventually, it came to me: why not digitizing them, making them accessible online and open-source and letting everyone read and edit it?

—

As most of the open-source software, this notebook does not come with any

"warranty" of any kind, meaning that there is no guarantee that everything in this notebook is correct, and it is not peer reviewed. **Do NOT cite this notebook in your academic research paper or book!** If you find anything helpful here with your research, please trace back to the origin of the knowledge and confirm by yourself.

This notebook is suitable as:

- a quick reference guide;
- a brief introduction for beginners of an area;
- a "cheat sheet" for students to prepare for the exam or for lecturers to prepare the teaching materials.

This notebook is NOT suitable as:

- a direct research reference;
- a replacement of the textbook.

The notebook is NOT peer reviewed, thus is more of a notebook than a book. It is meant to be easy to read, not to be comprehensive and very rigorous.

Although this notebook is open-source, the reference materials of this notebook, including textbooks, journal papers, conference proceedings, etc., may not be open-source. Very likely many of these reference materials are licensed or copyrighted. Please legitimately access these materials and properly use them, should you decided to trace the origin of the knowledge.

Some of the figures in this notebook are plotted using Excalidraw, a very convenient tool to emulate hand drawings. The Excalidraw project can be found on GitHub, *excalidraw/excalidraw*. Other figures may come from MATLAB, R, Python, and other computation engines. The source code to reproduce the results are intended to be included in the same repository of the notebook, but there might be exceptions.

This work might have benefited from the assistance of large language models, which are used exclusively for editing purposes such as correcting grammar and rephrasing sentences, without introducing new content, generating novel information, or changing the original intent of the text.

Preface

Control System

Lewis, F.L., Vrabie, D. and Syrmos, V.L., 2012. Optimal control. John Wiley & Sons.

List of Figures

List of Tables

Part I

Basic Control System

1

Classic Control System

CONTENTS

2

Modern Control System

CONTENTS

Part II

Computer Controlled System

3

Digital Signal Processing

CONTENTS

Digital signal processing is the basis of computer controlled system, and computer controlled system is an important use case of digital signal processing.

This chapter servers as a brief introduction to digital signal processing, including commonly seen discrete signal and digital system features, z-transform and frequency domain analysis of discrete signals and digital systems, signal sampling and reconstruction, discrete Fourier transform and digital filtering.

3.1 Motivation

It has become a common practice nowadays to use digital controllers over analog controllers as the preferable choice in most occasions. A commonly used block diagram of a digital processing system looks like Fig. 3.1. The digital and

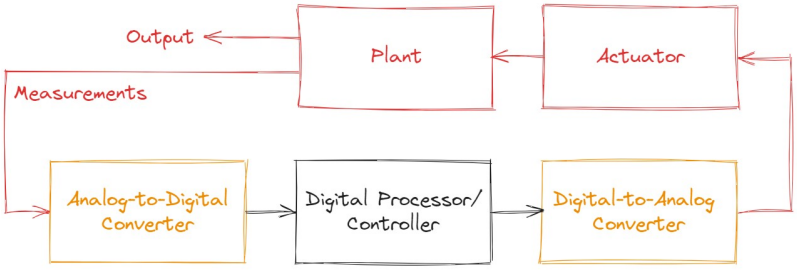

FIGURE 3.1
A block diagram of a digital system.

analog modules are given by the black and the red shapes respectively. The analog-to-digital converter (ADC) and the digital-to-analog converter (DAC) bridges the digital and analog worlds, and they are colored in orange.

A digital processing system has the following advantages over a conventional analog processing system.

- More flexible. Digital processor is usually re-programmable, thus making it cheaper and faster to add and test new features or functions.
- More controllable performance and accuracy. The performance and accuracy of a digital system is more controllable, deterministic and consistent than an analog system.
- Easier shareable. The digitized signals and the control algorithm itself can be easily stored and shared.

However, digital system is not always an superior alternative to the counterpart analog system. There are at least the following drawbacks.

- More expensive when the function is simple and can be easily realized with an analog circuit. The ADCs, DACs and the digital processor can all introduce additional costs. It is possible that sometimes the function is so simple that it is not worth introducing a digital system.
- Slower and has a strict bandwidth requirement. Digital system has been made faster and cheaper in the past years. Nevertheless, there are still tasks where the speed and the bandwidth of a digital system is too slow. In such cases, analog system or optical system might be a better solution.

3.2 Basic Concepts about Digital Signals and Systems

Basic concepts and categories of digital signals and systems are introduced.

3.2.1 Signals Channel and Dimension

Signals, both analog and digital, can be scalars of vectors, and it is often formulated as a (vector) function of some independent variables such as time. The length of the vector and the number of independent variables of a digital signal determine the number of channels and the dimensions of the signal.

A signal with vector size n and number of independent variables m is known as a n-channel-m-dimensional signal. For example, consider a colored image

$$I(x,y) \quad = \quad \begin{bmatrix} I_r(x,y) \\ I_g(x,y) \\ I_b(x,y) \end{bmatrix}$$

where I_r, I_g, I_b and (x, y) are the RGBs and the coordinate of the pixels respectively. This is a 3-channel-2-dimensional signal.

Though multi-channel-multi-dimensional signals are very common, for the convenience of study, the chapter will mostly focus on single-channel-single-dimensional signals.

3.2.2 Continuous, Discrete and Digital Signals

If a signal is continuous in both the time scale (it is defined at any instant of time) and value scale (it has infinite digits of accuracy in its value) is no doubt a continuous signal. Many physical signals, such as the speed of a car, can be modeled by a continuous signal.

Consider sampling a continuous signal by recording its values in countable and discrete timestamps. The signal becomes a discrete-time signal. It is defined only at the sampling timestamps. From value scale vise, it still has infinite digits of accuracy.

Instead of sampling, consider quantizing the signal by mapping its values to a set of discrete values, i.e., usually values with finite digits. The signal becomes a discrete-valued signal. It is defined on a continuous time scale, but the values it can take at any time instant is limited by the digits.

A signal that is both discrete-time and discrete-valued is also known as a digital signal. A demonstrative figure is given in Fig. 3.2.

3.2.3 Deterministic and Random Signals

There is always random noise and stochastic disturbances in a physical system. From this point of view, no system or signal is deterministic. In practice, so long as the signal or system model is to describe the actual signal or system to reasonable accuracy, we would consider the signal or system to be deterministic.

In contrast, if the signal cannot be described by the required reasonable

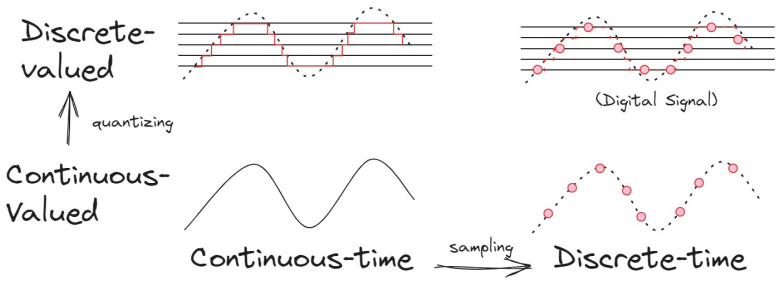

FIGURE 3.2
Continuous and discrete signals.

accuracy, or if it simply too complicated to do so, the signal is considered a random signal.

Deterministic signals and models are relatively easier to analyze. When it comes to random signals and stochastic systems, probability and statistics based analysis are usually adopted.

3.3 Commonly Seen Digital Signals and Features

Commonly seen elementary digital signals and their features are introduced. Notice that similar features can be defined for continuous signals as well such as periodic signal, energy and power of signal, etc., but they are not discussed in this section.

3.3.1 Elementary signals

The unit sample sequence is given by

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$$

The unit step signal is given by

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The unit ramp signal is given by

$$u_r(n) = \begin{cases} n & n \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The exponential signal is given by

$$x(n) = a^n$$

where notice that a can be a complex value denoted by $a = re^{j\theta}$, in which case

$$x(n) = r^n e^{j\theta n} = r^n \left(\cos(\theta n) + j\sin(\theta n)\right)$$

3.3.2 Energy and Power of a Signal

The energy and power of a signal can be calculated by

$$E = \sum_{n=-\infty}^{\infty} |x(n)|^2$$

$$P = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)|^2$$

respectively. The signals whose energy is finite is known as an energy signal. Likewise, the signals with finite power is known as a power signal. From the definition, we know that an energy signal must have a power of 0. Examples are given in Fig. 3.3.

3.3.3 Periodic Signal

A signal $x(n)$ is periodic if

$$x(n) = x(n+N), \text{ for all } n$$

and the smallest $N > 0$ is the period of the signal. Notice that a constant signal $x(n) = c$ is often taken as a periodic signal as well, but of non-definable fundamental period. In the remaining of our discussion, we shall consider the "regular" periodic signal with $N > 0$ unless otherwise emphasized.

It can be proved easily that a period signal has finite power so long as $|x(n)|$ is finite for any timestamp. The energy, on the other hand, is always infinite.

3.3.4 Symmetric and Antisymmetric Signals

A signal $x(n)$ is symmetric (also known as even) if

$$x(n) = x(-n)$$

A signal $x(n)$ is asymmetric (also known as odd) if

$$x(n) = -x(-n)$$

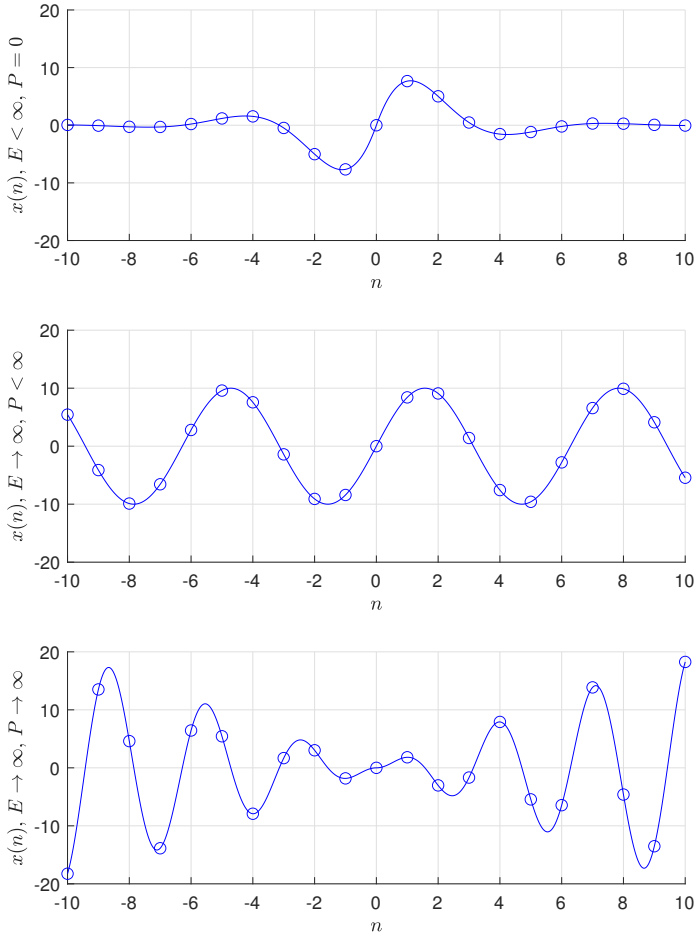

FIGURE 3.3
Example of energy and power signals.

Given any signal $x(n)$, define

$$\begin{aligned} x_e(n) &= \frac{1}{2}\left[x(n) + x(-n)\right] \\ x_o(n) &= \frac{1}{2}\left[x(n) - x(-n)\right] \end{aligned}$$

It can be easily verified that $x(n) = x_e(n) + x_o(n)$. Therefore, we can conclude that any signal can be decomposed into the symmetric and the antisymmetric portions.

3.4 Commonly Seen Operations on a Signal

For a general discrete signal $x(n)$, $x(-n)$ is a fold of the signal at $n = 0$. We can easily prove that $x(-n + 2k)$ is a fold of the signal at $n = k$.

Signal $x(n - k)$ is a shift of the original signal $x(n)$ to the right by k samples, i.e., it is $x(n)$ delayed by k samples. Likewise, $x(n + k)$ is the original signal to shift to the left.

Signal $x(\mu n)$, $\mu \in \mathbb{N}^+$ is a time-scaling or down-sampling of the original signal $x(n)$. Graphical wise, it "squeezes". A demonstrative figure is given in Fig. 3.4. The values captured by the blue circle is the down sampled signal sequence. In this example, $\mu = 2$, i.e., one out of two samples are captured.

3.5 Digital System

A digital system maps discrete signal $x(n)$ to another discrete signal $y(n)$. It can be described by

$$y(n) = \mathcal{T}[x(n)]$$

in general. The system is denoted by $\mathcal{T}$.

3.5.1 Input-Output Model

There are varieties of ways to model the system. A commonly used way is to use an input-output model described by the difference equation. A simple example is given below.

$$y(k) = \sum_{i=-\infty}^{k-1} x(i) = y(k-1) + x(k-1) \tag{3.1}$$

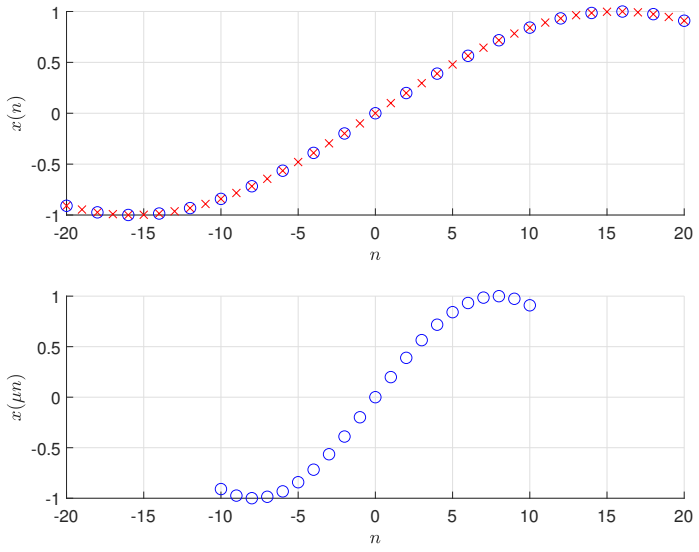

FIGURE 3.4
A demonstration of down-sampling.

Equation (3.1) calculates the aggregated sum of the input signal. For this system, we can see that its output $y(k)$ can be entirely described by its previous value $y(k-1)$ and the current input $x(k)$. This is because all the "historical information" priori to $k-1$ that is useful in calculating $y(k)$ is contained in $y(k-1)$.

In general, if the current status of a system can be formulated as a function of the one-step previous status of the system and the current input and noise, the system model is known as a *Markov process*. Markov process is widely used in modeling the system behavior. The famous state-space realization of a system is a systematic way of using a Markov process to model a system.

By tracing back in time, we can find the "initial state" of a Markov process. In the context of (3.1), initial state can be denoted as $y(0)$. We do not care the values of $y(k)$ and $x(k)$ priori to $k = 0$. From the recursive equation (3.1), it is obvious that $y(k)$ is essentially determined by the initial state $y(0)$ and the historical inputs $x(0), ..., x(k-1)$.

For the convenience of analysis, it is quite common that zero initial state (also known as relaxed initial state) $y(0) = 0$ is assumed, which allows us to focus on the study of input-output relationship. The system response, in which case only affected by the inputs $x(0), ..., x(k-1)$, is known as the *zero state response*. Likewise, there is the *zero input response* of the system where non-zero initial state $y(0) \neq 0$ and zero input $x(k) = 0$ are considered.

3.5.2 Different Types of Digital Systems

“nobreak

3.6 Frequency Domain Analysis

“nobreak

3.7 Sampling and Reconstruction of a Signal

“nobreak

3.8 DFT

“nobreak

3.9 Digital Filtering

4

Computer Controlled System

CONTENTS

Part III

Advanced Control Algorithms

5

Optimal and Robust Control System

CONTENTS

In a typical optimal control problem, a cost function positively correlated with regulation error, control signal energy, etc., is defined. Such control that minimizes the cost function is calculated. In a typical robust control problem, stochastic disturbance is introduced into the model, and such control signal that minimizes the average or worst effect of the disturbance is calculated. More details are introduced in this chapter.

The key to optimal and robust control is to solve the optimization problem using variety of mathematical tools.

5.1 Static Optimization

Assume static model where there is no dynamic response. Find the signal u that minimizes the static cost function $L(u)$ (or L for simplicity). Notice that $u \in \mathbb{R}^m$ and L a scalar.

5.1.1 Optimization without Constraint

Using Taylor series expansion, An increment of L can be formulated as follows.

$$dL = L_u^T du + \frac{1}{2} du^T L_{uu} du + \dots$$

where

$$\begin{aligned} L_u &= \frac{\partial L}{\partial u} \in \mathbb{R}^m \\ L_{uu} &= \frac{\partial^2 L}{\partial u^2} \in \mathbb{R}^{m \times m} \end{aligned} \tag{5.1}$$

about a particular choice of u. A (local) minimum of L is achieved under the following conditions. Firstly,

$$\begin{aligned} L_u &= 0 \\ dL &\geq 0 \end{aligned} \tag{5.2}$$

which indicates that it is a critical point (also known as stationary point) at the choice of u, and L increases around that critical point. Equation (5.2) is equivalent to

$$L_{uu} > 0$$

a positive definite matrix.

Consider the following example.

A Quadratic Surface Example

Let

$$L = \frac{1}{2} u^T Q u + S u \tag{5.3}$$

$$= \frac{1}{2} u^T \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} u + \begin{bmatrix} s_1 & s_2 \end{bmatrix} u \tag{5.4}$$

Find u at the critical point, and discuss whether it is a maximum, minimum or other.

From (5.3) and (5.4),

$$\begin{aligned} L_u &= Qu + S \\ L_{uu} &= Q \end{aligned}$$

At critical point $L_u = 0$,

$$u^* = -Q^{-1}S$$

where u^* denotes the critical point. Whether this critical point is a maximum, minimum, or other cases depends on L_{uu}.

- If $L_{uu} > 0$, i.e. Q is positive definite, the critical point is a minimum.
- If $L_{uu} < 0$, i.e. Q is negative definite, the critical point is a maximum.
- Otherwise,
  - If $|L_{uu}| < 0$, i.e. $q_{11}q_{22} - q_{12}q_{21} < 0$, the critical point is a saddle point.
  - If $|L_{uu}| = 0$, the critical point is a singular point and more information is required to determine the shape at the critical point.

5.1.2 Optimization with Equality Constraint

Consider the optimization with equality constraint as follows.

$$\text{minimize} \quad L(u) \tag{5.5}$$

$$\text{s.t.} \quad f(u) = 0 \tag{5.6}$$

where $u \in \mathbb{R}^m$ and $f(u) \in \mathbb{R}^n$, and without losing generality we assume that $n < m$ (otherwise u can be solved from $f(u) = 0$).

Intuitive Explanation

It is obvious that the solution to (5.5) would not be necessarily at $L_u = 0$. As a matter of fact, at any point of u satisfying $f(u) = 0$, it is not necessary that $L_u = 0$. This is demonstrated by Fig. 5.1, where $L(u)$ is given by the contour (assuming $u \in \mathbb{R}^2$ in this demonstration), and $f(u) = 0$ the red solid line.

The critical point of (5.5) can be found as follows. Consider a necessary condition for the critical point. At the critical point, $L_u = dL/du$ is not necessary zero, and neither is dL for all random chosen du. However, dL should be zero with respect to du when df is zero, i.e. (consider only the first-order Taylor expansion for now),

$$dL = L_u^T du = 0 \tag{5.7}$$

$$s.t. \quad df = f_u du = 0 \tag{5.8}$$

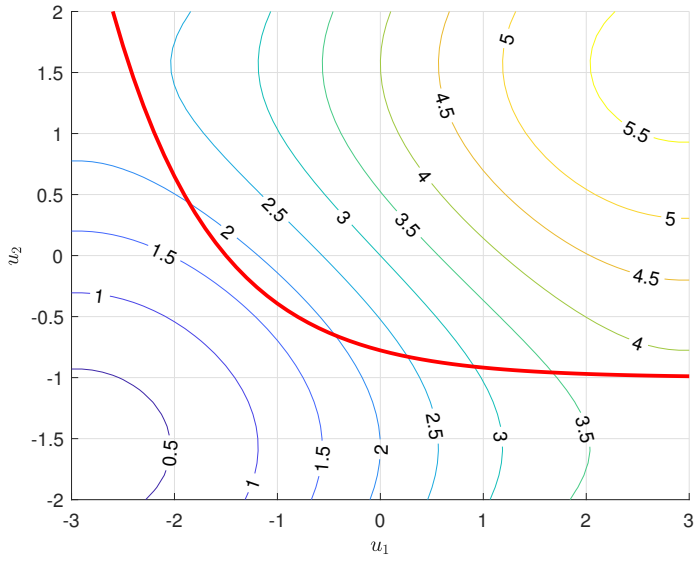

FIGURE 5.1
Static optimization with equality constraint demonstration.

or equivalently

$$\begin{bmatrix} dL_{1\times 1} \\ df_{n\times 1} \end{bmatrix}_{(n+1)\times 1} = \begin{bmatrix} L_u^T{}_{1\times m} \\ f_u{}_{n\times m} \end{bmatrix}_{(n+1)\times m} du_{m\times 1} = 0 \tag{5.9}$$

For reading convenience, the dimensions of the matrices are given in red color.

As a necessary condition, given a critical point, we should be able to solve (5.8) for a linear space of du, and make sure that such du satisfies (5.7). With that been said, the coefficient matrix $[L_u^T; f_u]$ in (5.9) should have rank of at most n so that the last n rows alone can be used to find du then substitute it to the first row for verification. Therefore, (5.9) can be re-written as follows.

$$\begin{aligned} \begin{bmatrix} 1 & \lambda^T \end{bmatrix} \begin{bmatrix} L_u^T \\ f_u \end{bmatrix} &= 0 \\ L_u^T + \lambda^T f_u &= 0 \end{aligned} \tag{5.10}$$

where $\lambda \in \mathbb{R}^n$ is known as the Lagrange multipliers, and (5.10) a set of m equations being a simplified part of Karush–Kuhn–Tucker (KKT) conditions which will be introduced in more details later.

Equation (5.10) can also be interpreted graphically. It suggest that the gradient of $L(u)$ and $f(u)$ should be parallel at the critical point by a factor of λ. This is shown by Fig. 5.2. At the critical point denoted by the red dot, the

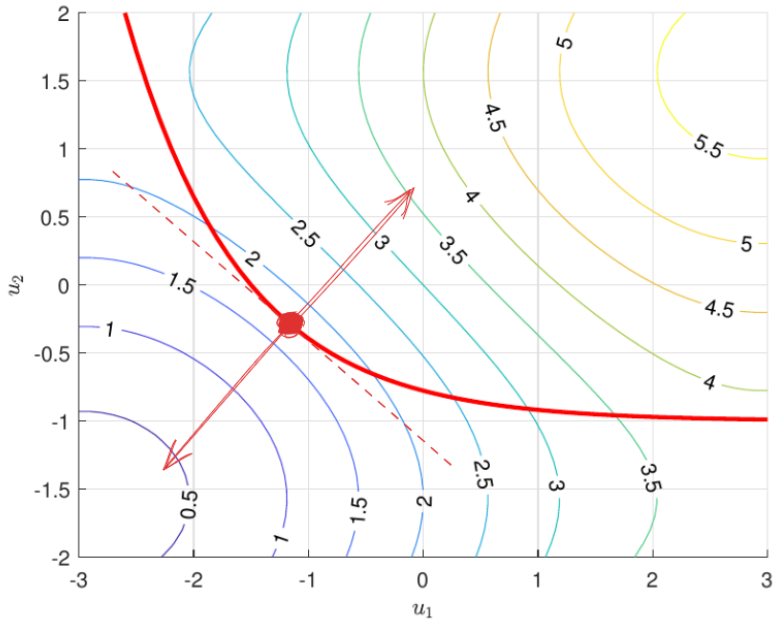

FIGURE 5.2
Graphical explanation to (5.10).

red solid line $f(u) = 0$ tangents the contour $L(u)$ indicated by the red dashed line. The gradient of both $f_u = \nabla f$ and $L_u = \nabla L$ should be perpendicular to the tangent aligning with the red arrows, hence they are parallel.

Equation (5.10) together with (5.6) forms the necessary condition for the solution of the optimization problem.

Lagrangian Function

The necessary condition can also be derived using Lagrangian function as follows. Lagrangian function is a very useful tool in solving optimization problem with different types of constraints. Notice that Lagrangian function in some context may also be called Hamiltonian function. They are very similar mathematically, and mostly differ in the context of usage.

$$\begin{aligned} \mathcal{L}(u, \lambda) &= L(u) + \lambda^T f(u) \\ &= L(u) + \sum_{i=1}^{n} \lambda_i f_i(u) \end{aligned} \tag{5.11}$$

and (5.6), (5.10) can be represented by

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \lambda} &= 0 \\ \frac{\partial \mathcal{L}}{\partial u} &= 0 \end{aligned}$$

respectively.

It is difficult to find the critical point for the original problem with equality constraint. However, by introducing Lagrangian function (5.11), the problem becomes finding critical point for Lagrangian function without constraint (by considering both λ and u as the independent variable of $\mathcal{L}$). The converted problem, which is simpler to solve, is known as the dual problem of the original problem. This is made possible due to the introduction of Lagrange multipliers λ that adds some flexibility to the equation, making it contain the information of equality constraint in a "disguised" manner.

Sufficient Condition

The necessary condition helps to pick up the possible candidates that may be a local minimum of the optimization problem. To verify that the critical point is indeed a local minimum, check the second-order derivative $\partial^2 L/\partial u^2$ of the critical point.

Consider the following example.

A Quadratic Surface Example with Linear Equality Constraint

Let

$$\begin{aligned} & L(x,u) = \frac{1}{2}x^T Q x + \frac{1}{2}u^T R u \qquad (5.12) \\ \text{s.t.} \quad & f(x,u) = x + Bu + c = 0 \end{aligned}$$

where the dimensions are $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $Q \in \mathbb{R}^{n\times n}$, $R \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times m}$, $c \in \mathbb{R}^n$. Find u at the critical point, and discuss whether it is a minimum.

The Lagrangian function of the problem is

$$\mathcal{L} = \frac{1}{2}x^T Q x + \frac{1}{2}u^T R u + \lambda^T (x + Bu + c)$$

where $\lambda \in \mathbb{R}^n$. The necessary condition for a critical point is

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \lambda} &= x + Bu + c = 0 \qquad (5.13) \\ \frac{\partial \mathcal{L}}{\partial x} &= Qx + \lambda = 0 \qquad (5.14) \\ \frac{\partial \mathcal{L}}{\partial u} &= Ru + B^T \lambda = 0 \end{aligned}$$

The solution is

$$\begin{aligned} u &= -\left(R + B^T Q B\right)^{-1} B^T Q c \qquad (5.15) \\ x &= -\left(I - B\left(R + B^T Q B\right)^{-1} B^T Q\right) c \qquad (5.16) \\ \lambda &= \left(Q^{-1} + B R^{-1} B^T\right)^{-1} c \end{aligned}$$

The verify whether it is a minimum w.r.t. u, check the following

$$L_{uu} = R + B^T Q B$$

which is positive definite as long as Q and R are positive definite. Indeed, this is a minimum w.r.t. u. The minimum value can be obtained by substituting (5.15) and (5.16) into (5.12).

$$\begin{aligned} L^* &= \frac{1}{2} c^T \left(Q^{-1} + B R^{-1} B^T\right)^{-1} c \\ &= \frac{1}{2} c^T \lambda \end{aligned}$$

5.1.3 Optimization with Equality and Inequality Constraints

Optimization with inequality constraints is not as widely seen in optimal control as optimization with equality constraint. Nevertheless, it is briefly introduced here. Consider the optimization problem with both equality and inequality constraints as follows.

$$\begin{aligned} \text{minimize} \quad & L(u) \\ \text{s.t.} \quad & f(u) = 0 \\ & g(n) \leq 0 \end{aligned} \qquad (5.17)$$

where $u \in \mathbb{R}^m$, $f(u) \in \mathbb{R}^p$, $g(u) \in \mathbb{R}^q$.

Obviously, the solution of the problem may reside at either $g(u) = 0$ or $g(u) < 0$. Therefore, an intuitive way of solving the problem is to use "trail-and-error" as follows. Ignore the inequality constraint (5.17), and the problem becomes an optimization with equality constraint problem. Test the obtained solution using (5.17). If (5.17) is satisfied, the solution to the new problem is also a solution to the original problem. Otherwise, reformulate the problem and let both $f(u) = 0$, $g(u) = 0$ be equality constraints.

Sort of inspired by this idea, the optimization problem with both equality and inequality constraints described by (**??**), (**??**) and (5.17) can be solved as follows. First, define generalized Lagrangian function

$$\mathcal{L}(u, \alpha, \beta) = L(u) + \alpha^T f(u) + \beta^T g(u)$$

where $\alpha \in \mathbb{R}^p$ and $\beta \in \mathbb{R}^q$ are Lagrange multipliers associated with equality

and inequality constraints, respectively. In some literatures, $\lambda = \begin{bmatrix} \alpha^T & \beta^T \end{bmatrix}^T$ might be used in the notation.

The necessary condition for a solution can be obtained using Karush-Kuhn-Tucker (KKT) conditions as follows. The KKT conditions are essentially first derivative tests for the solution to be optimal. Notice that sometimes KKT can be necessary and sufficient condition for the solution if L, f and g meet certain criteria. This is not discussed here in details.

$$\frac{\partial \mathcal{L}}{\partial u} = 0 \tag{5.18}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha} = 0 \tag{5.19}$$

$$\beta_i g_i(u) = 0 \tag{5.20}$$

$$g_i(u) \leq 0 \tag{5.21}$$

$$\beta_i \geq 0 \tag{5.22}$$

Equations (5.18) and (5.19) are straight forward and they are the same with the previous discussed optimization with equality constraint problem. Equations (5.20), (5.21) and (5.22) discusses how the inequality constraint would affect the critical point:

- In Lagrangian function, let the inequality constraint does not affect its value, hence (5.20).
- Equation (5.20) is achieved as follows:
  - If the critical point is laid on the ith inequality constraint boundary, $g_i(u) = 0$, making (5.20) zero. In this case, the value of β_i does not matter. Without loosing generality, let $\beta_i > 0$.
  - If the critical point is not laid on the ith inequality constraint boundary, $g_i(u) < 0$, in which case enforce $\beta_i = 0$ so that (5.20) is zero.
  - In other words, if $g_i(u) < 0$, $\beta_i = 0$; if $g_i(u) = 0$, $\beta_i > 0$.

Equations (5.18) to (5.22) together form the dual problem of the original problem. Equation (5.20) is known as the complementary slackness, and (5.22) dual feasibility.

5.2 Optimal Control of Discrete System

Optimal control system has a strong connection with the static optimization problem introduced in Section 5.1. In mathematics perspective, they are using similar tools to solve similar problems. It is just that the physical model of the system, which determines how the system evolves giving a control signal, becomes the constraint in the optimization. This is illustrated in the remaining of the chapter.

5.2.1 Problem Formulation

A general optimal control of discrete system problem can be described as follows. Let the system dynamics be described by

$$x(k+1) \quad = \quad f^k\left(x(k), u(k)\right) \tag{5.23}$$

where $k = 0, \ldots$ is the time index, $x(k) \in \mathbb{R}^n$ the state vector ($x(0)$ is the initial state), $u(k) \in \mathbb{R}^m$ the input and f^k the system dynamics. No disturbance or process error is considered. It is assumed that $x(k)$ is known at all time instant.

The objective of the optimization is to find such $u(k)$ that minimizes the following cost function

$$J \quad = \quad \phi\left(N, x(N)\right) + \sum_{k=0}^{N-1} L^k\left(x(k), u(k)\right) \tag{5.24}$$

where N is the time to reach a particular goal, $x(N)$ the final state vector at $k = N$, and L^k the cost relevant to the magnitude of $x(k)$ and $u(k)$ at time instant k. In some cases, N is a parameter to optimize, for example in a "do-it-fast" type of problem. In other cases, N is a fixed value, for example in a "do-the-best-within-given-time" type of problem.

Some commonly seen cost functions are given below.

Minimum Time

Use $u(k)$ to drive the system from its initial state to a given state. The total time consumption is minimized.

$$J \quad = \quad N$$

Minimum Fuel and Energy

Use $u(k)$ to drive the system from its initial state to a given state. The cost of applying the control signal is minimized. To minimize the fuel consumption,

$$J \quad = \quad \sum_{k=0}^{N-1} |u(k)|$$

To minimize energy,

$$J \quad = \quad \sum_{k=0}^{N-1} u(k)^T u(k)$$

Regulation

Use $u(k)$ to regulate the states to zero (usually from a non-zero initial

state). Along the way, both regulation error and the control signal driving energy are considered.

$$J = \frac{1}{2}x(N)^T Sx(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^T Qx(k) + u(k)^T Ru(k)\right)$$

where Q, S and R are weight matrices that trades of the minimization of regulation error (intermediate state error), final state error or control signal energy.

5.2.2 General Solution

A general solution to (5.24) subject to (5.23) is given below.

Define the augmented cost function as the Lagrangian function below, taking into account (5.23) as the equality constraint.

$$J' = \phi\left(N, x(N)\right) + \sum_{k=0}^{N-1}\left(L^k\left(x(k), u(k)\right) + \lambda_{k+1}^T\left(f^k\left(x(k), u(k)\right) - x(k+1)\right)\right)$$

which can be rewritten as

$$\begin{aligned} J' &= \phi\left(N, x(N)\right) - \lambda_N^T x(N) + H^0(x(0), u(0)) \\ &\quad + \sum_{k=0}^{N-1}\left(H^k\left(x(k), u(k)\right) - \lambda_k^T x(k)\right) \end{aligned}$$

where

$$H^k\left(x(k), u(k)\right) = L^k\left(x(k), u(k)\right) + \lambda_{k+1}^T f^k\left(x(k), u(k)\right)$$

is the Hamiltonian defined at each time stamp k. Apply KKT condition and the solution is

$$\begin{aligned} x(k+1) &= \frac{\partial H^k}{\partial \lambda_{k+1}} = f^k\left(x(k), u(k)\right) \\ \lambda_k &= \frac{\partial H^k}{\partial x(k)} = \left(\frac{\partial f^k}{\partial x(k)}\right)^T \lambda_{k+1} + \frac{\partial L^k}{\partial x(k)} \\ 0 &= \frac{\partial H^k}{\partial u(k)} = \left(\frac{\partial f^k}{\partial u(k)}\right)^T \lambda_{k+1} + \frac{\partial L^k}{\partial u(k)} \\ 0 &= \left(\frac{\partial L^0}{\partial x(0)} + \left(\frac{\partial f^0}{\partial x(0)}\right)^T \lambda_1\right)^T dx(0) \\ 0 &= \left(\frac{\partial \phi}{\partial x(N)} - \lambda_N\right)^T dx(N) \end{aligned}$$

5.2.3 Discrete LQR

Linear quadratic regulator (LQR) is a special case of the general problem formulation (5.23) and (5.24). It is one of the most commonly seen optimal control problems.

Consider linear system as follows. It is assumed that the state vector is known exactly.

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

where notice that the process matrix $A(k)$, input matrix $B(k)$ can be time-varying. The cost function is given by

$$J \quad = \quad \frac{1}{2}x(N)^T S_N x(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^T Q_k x(k) + u(k)^T R_k u(k)\right)$$

where $|S_N| \geq 0$, $|Q_k| \geq 0$ and $|R_k| > 0$ are known symmetric matrices. In the LQR problem, the target of the control is to regulate the state vector to zero while given an non-zero initial state, and meantime tries to keep the control signal energy small.

The solution of the problem can be found by constructing the Hamiltonian function as introduced earlier in Section 5.2.2. Result is given below.

$$\begin{aligned} S_k &= A(k)^T\left(S_{k+1} - S_{k+1}B(k)\left(B(k)^T S_{k+1}B(k) + R_k\right)^{-1}B(k)^T S_{k+1}\right)A(k) \\ &\quad + Q_k \qquad (5.25) \\ K(k) &= \left(B(k)^T S_{k+1}B(k) + R_k\right)^{-1}B(k)^T S_{k+1}A(k) \qquad (5.26) \\ u(k) &= -K(k)x(k) \end{aligned}$$

where $k = 0, \ldots, N-1$. Matrix $K(k)$ in (5.26) is known as the Kalman gain (it might be more intuitive to simply call it the "control gain") and it can be calculated offline.

5.2.4 Suboptimal Discrete LQR

Notice that $K(k)$ is time varying in general. Only if A, B, Q and R are constant and at the same time $N \to \infty$, S_k becomes a constant and so does $K(k)$. In such case, the cost function becomes

$$J \quad = \quad \frac{1}{2}\sum_{k=0}^{\infty}\left(x(k)^T Q x(k) + u(k)^T R u(k)\right)$$

The problem can be solved similarly with S the unique positive definite solution to the following discrete time algebraic Riccati equation

$$S \quad = \quad A^T\left(S - SB\left(B^T SB + R\right)^{-1}B^T S\right)A \qquad (5.27)$$

and the Kalman gain

$$K_\infty = \left(B^T S B + R\right)^{-1} B^T S A \tag{5.28}$$

which is also known as the static Kalman gain.

In a LTI system where A, B, Q and R are constant matrices and N a finite number, dealing with time varying $K(k)$ can be problematic in practice due to the large memory and offline computation load requirement. It is sometimes preferable to rather use suboptimal control $u = -K_\infty x(k)$ instead of $u = -K(k)x(k)$, where K_∞ is calculated using (5.28), assuming $N \to \infty$ and ignoring $x(N)^T S_N x(N)$ in the cost function.

It is important to verify whether the system remains asymptotically stable when suboptimal control law is applied. Only the conclusion is given as follows: the system (A, B) is stabilizable if and only if:

- Equation (5.27) has unique positive definite solution, and (5.25) converges to that solution.
- The closed-loop plant $(A - BK_\infty)$ is asymptotically stable.

The proof is ignored.

This implies that so long as the original system is stabilizable, the suboptimal control $u = -K_\infty x(k)$ guarantees the asymptotic stability of the system.

5.3 Optimal Control of Continuous System

The dynamics of a continuous time domain system is described using differential equations, and the solution can be obtained using Lagrangian function similar to previous sections. However, do notice that calculus of variations is used when solving the problem. Calculus of variations is briefly reviewed in the appendix. For more details of calculus of variations, check *A Notebook on Calculus* by Lu Sun in the same series.

5.3.1 Problem Formulation

Let the system dynamics be described by the following state-space model

$$\dot{x} = f(x, u, t)$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$. Let the cost function be

$$J = \phi\left(x(T), T\right) + \int_0^T L\left(x(t), u(t), t\right) dt \tag{5.29}$$

where $[0, T]$ is the time interval of interest. The final time T may vary when different controls are applied. In addition, define $\psi(x(T), T)$. The objective of the optimization is to find optimal $u(t)$ on the time interval $[0, T]$, so that J in (5.29) is minimized, and $\psi(x(T), T) = 0$.

Notice that ψ is different from ϕ in (5.29). Both of them are about the final state, but ϕ is made small while ψ needs to be made exactly zero. This restriction introduced by *psi* is useful. For example, in a position regulation problem, ψ can be defined as

$$\psi = \begin{bmatrix} x(T) - X \\ \dot{x}(T) \end{bmatrix}$$

where $x(T) - X = 0$ enforces zero regulation error, and $\dot{x}(T) = 0$ enforces a complete stop of the system motion.

5.3.2 General Solution

Define the augmented cost function as the Lagrangian function below.

$$\begin{aligned} J' &= \phi(x(T), T) + \nu^T \psi(x(T), T) \\ &\quad + \int_0^T \left(L(x, u, t) + \lambda^T \left(f(x, u, t) - \dot{x} \right) \right) dt \end{aligned}$$

where ν, λ are Lagrange multipliers. Define Hamiltonian function as follows.

$$H(x, u, t) = L(x, u, t) + \lambda^T f(x, u, t)$$

The solution of the optimization problem can be obtained by letting

State: $$\dot{x} = \frac{\partial H}{\partial \lambda} = f$$

Costate: $$-\dot{\lambda} = \frac{\partial H}{\partial x} = \frac{\partial f^T}{\partial x}\lambda + \frac{\partial L}{\partial x}$$

Stationarity: $$0 = \frac{\partial H}{\partial u} = \frac{\partial L}{\partial u} + \frac{\partial f^T}{\partial u}\lambda$$

Boundary: $x(0)$ is given

$$\left(\phi_x + \psi_x^T \nu - \lambda \right)^T \Big|_T dx(T) + \left(\phi_t + \psi_t^T \nu + H \right)^T \Big|_T dT = 0$$

It is often not easy to solve analytical solution of $u(t)$ from the above equations.

5.3.3 Continuous LQR

LQR in discrete time domain has been introduced earlier. Its application in continuous time domain is introduced in this section as follows. Let the system dynamics be described by

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$. The state vector $x(t)$ is assumed known exactly. The cost function is given by

$$J = \frac{1}{2}x(T)^T S_T x(T) + \frac{1}{2}\int_0^T \left(x(t)^T Q_t x(t) + u(t)^T R_t u(t)\right) dt$$

where $|S_T| \geq 0$, $|Q_t| \geq 0$ and $|R_t| > 0$ are known symmetric matrices.

The solution can be obtained by formulating the Hamiltonian. Result is given below.

$$\begin{aligned} -\dot{S}(t) &= A(t)^T S(t) + S(t)A(t) - S(t)B(t)R_t^{-1}B(t)^T S(t) + Q_t \quad (5.30) \\ K(t) &= R_t^{-1}B(t)^T S(t) \quad (5.31) \\ u(t) &= -K(t)x(t) \end{aligned}$$

5.3.4 Suboptimal Continuous LQR

Just like Section 5.2.4, the continuous LQR also has suboptimal solution. The motivation remains the same: to tackle the problem where Kalman gain $K(t)$ in (5.31) is time-varying and can be difficult or tedious to calculate, and we want an alternative solution with constant Kalman gain.

Assume time-invariant A, B in the process model and constant Q, R. Let time-invariant S be the solution of the following ARE

$$0 = A^T S + SA - SBR^{-1}B^T S + Q \quad (5.32)$$

which is obtained from (5.30) by letting $\dot{S} = 0$ assuming convergence. Notice that the solution to (5.32) is not necessarily unique. One of the solutions of S can be obtained by calculating (5.30) until it converges to $S(\infty)$, and $S = S(\infty)$ is obviously a solution to (5.32). This would be the solution that we would use through out the remaining of the section. Alternatively, S can be calculated using the analytical solution to the ARE which is not covered in this notebook.

With the obtained constant $S = S(\infty)$, Kalman gain in (5.31) becomes

$$K = R^{-1}B^T S(\infty)$$

which is time-invariant. This produces a suboptimal solution but with a constant Kalman gain that is easier for implementation.

5.4 LQR Properties

The problem formulation and the solution of LQR have been introduced in earlier sections. The properties, extensions, practical implementations, etc., have not yet been covered. In this section and the few sections following, these more advanced topics are discussed.

In this section, we will start with the discussion of LQR properties, such as its robustness.

5.5 LQR for Tracking Problem

LQR by itself is a regulator that regulates the state vector x to a constant value r, and at the same time balancing the control signal energy. LQR can also be used in tracking, in which case the reference signal is no longer a constant, but time-variant. Tracking is an important extension of LQR.

5.6 Other LQR Extensions

“nobreak

5.7 LQG

“nobreak

5.8 H_2 and H_∞ Control

6

Model Predictive Control System

CONTENTS

Model predictive control system (MPC) belongs to the broader family of optimal control system. It defines a cost function and a bunch of constraints about the states and control signals. The objective is to find such control signal trajectory that fulfills the constraints while minimizing the cost. The optimization problem is solved recurrently at each timestamp where only the first control input from the optimal sequence is actually applied to the system.

Comparing with other optimal controllers such as LQR, MPC is more flexible and robust, and has a better adaption to model dynamics and different kinds of constraints. This makes MPC extremely popular in industry. The drawback of MPC is that it is more computationally demanding than LQR and other commonly seen optimal controllers.

The references of this chapter include [3].

6.1 Introduction

The MPC uses dynamic models to forecast the system trajectory, and substitute the it into the cost function. It looks for the optimal control sequence that minimizes the cost function. The first signal in the sequence is taken as the best decision at the current moment, and it is implemented to the actuator.

The following information is required to calculate the optimal control sequence:

- The plant, which is used to forecast the system trajectory
  - Model

  - Current state
  - Constraints for both state and control signal

- Cost function

In practice, the model of the plant is usually calibrated by parameter estimation and the current state by state estimation. The constraints can be obtained from the equipment handbook. The cost function, usually in the form of a functional, is defined by the user.

6.1.1 Model and Constraint

The model plays an absolutely critical role in MPC. A model can be either linear or nonlinear, continuous or discrete, in state-space or input-output realizations, centralized or discrete (not spatially uniform), deterministic or stochastic. Different problem formulations and mathematical tools are used to describe the different types of the models and solve the different types of optimization problems.

MPC has a good adaption to different types of constraints. The constraints can be largely divided into two categories, namely physical constraints and performance constraints. Input signal $u(k)$ related constraints are often physical constraints reflecting the physical limits of the system. State vector $x(k)$ and output $y(k)$ related constraints, on the other hand, are often performance constraints. The performance constraints describes the lowest performance goals that the user expects from the system.

Input signal $u(k)$ related constraints are often given in the following form.

$$\begin{bmatrix} -I \\ I \end{bmatrix} u(k) \quad \leq \quad \begin{bmatrix} -u_{\min} \\ u_{\max} \end{bmatrix} \tag{6.1}$$

State vector $x(k)$ related constraints are often given in the following form.

$$Fx(k) \quad \leq \quad f \tag{6.2}$$

It is possible to use augmented state vector that combines states with inputs. Let

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

and both constraints in (6.1) and (6.2) can be expressed by the form

$$F\tilde{x}(k) + Eu(k) \quad \leq \quad e \tag{6.3}$$

Equation (6.3) can also be used to limit the difference signal of the control signal. Consider

$$\begin{bmatrix} 0 & -I \\ 0 & I \end{bmatrix} \tilde{x}(k) + \begin{bmatrix} I \\ -I \end{bmatrix} u(k) \quad \leq \quad \begin{bmatrix} \Delta_{\max} \\ -\Delta_{\min} \end{bmatrix}$$

in (6.3), which essentially translates to

$$\Delta_{\min} \leq u(k) - u(k-1) \leq \Delta_{\max}$$

Other constraints include limiting the state vector and/or the control signal to be discrete values such as integers. These types of constraints are not as commonly seen as the aforementioned ones.

6.1.2 Parameter Estimation and State Estimation

The most commonly used parameter estimation method is the least squares estimation. The most commonly used state estimation method is the Kalman filter. Both least squares estimator and Kalman filter are linear filters and they are designed optimal for Gaussian noise. Least squares estimator can be implemented in a recursive manner, in which case it becomes a special case of Kalman filter mathematically.

It is possible to add "forget factor" to least squares estimator or Kalman filter. A window with fixed length is specified. When a new measurement set becomes available, the earliest measurement in the window will be removed, thus its impact on the state estimate eliminated. This leads to the moving horizon estimation (in contrast, the conventional Kalman filter implementation is also known as growing-memory estimation), which is less efficient but more robust to time-varying in the system.

6.1.3 Optimal Control

The optimal control to the plant vary with the definition of the cost function and constraints. In this section, for simplicity, a linear quadratic regulator problem is used to demonstrate MPC. Notice that although LQR is often regarded as a stand alone optimal control problem, there is no harm to introduce LQR as a simplified MPC problem, just for demonstration purpose.

Finite LQR Formulation

Consider the following dynamic model in discrete time domain. Current timestamp is $k = 0$. For simplicity, it is assumed that the model is known and the state is measurable precisely, thus there is no need to build an estimator for the model or the state.

$$x(k+1) = Ax(k) + Bu(k) \tag{6.4}$$
$$y(k) = x(k) \tag{6.5}$$

With the above model, it is possible to predict how the state evolves given a sequence of inputs as below.

$$u = \begin{bmatrix} u(0) & u(1) & \dots & u(N-1) \end{bmatrix}$$

where N is the largest time scale considered in this problem.

The purpose of the control is to regulate non-zero initial state $x(0)$ to zero while keeping the magnitude of $u(k)$ small. Therefore, define the cost function as follows.

$$V\left(x(0),u\right) \quad = \quad \frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^T Q x(k)+u(k)^T R u(k)\right)+\frac{1}{2}x(N)^T P_f x(N) \tag{6.6}$$

where $Q \geq 0$, $P_f \geq 0$ and $R > 0$ are symmetric matrices of user's choice. It can be proved that the above problem formulation must have a unique solution.

A Brief Introduction to Dynamic Programming

The aforementioned optimization problem in (6.6) can be solved via variety of ways. Consider using dynamic programming (DP).

A brief introduction to DP is given as follows. For illustration of DP, consider the following simple optimization problem.

$$x^*, y^* \quad = \quad \arg\min_{x,y} f(x,y) \tag{6.7}$$

Since x, y are coupled in $f(x, y)$, an intuitive way of solving this optimization problem is to let

$$\begin{cases} \dfrac{\partial}{\partial x} f(x,y) & = & 0 \\ \dfrac{\partial}{\partial y} f(x,y) & = & 0 \end{cases}$$

and if $f(x, y)$ is quadratics of x, y, the above should generate a unique solution.

Alternatively, consider solving (6.7) using the following approach. Think of (6.7) as a two-stage optimization problem. Given any constant value of y, there is an associated optimized $x^*(y)$ which minimizes (6.7). Using this method, we can transform $f(x, y)$ into $f\left(x^*(y), y\right)$, a function of y alone. From there, the optimal y can be found easily. Subsequently, $x^*(y)$ can be obtained as well. To summarize, (6.7) is equivalent of

$$\begin{aligned} y^* &= \arg\min_{y}\left(\min_{x} f(x,y)\right) \\ x^* &= \arg\min_{x} f(x,y^*) \end{aligned}$$

Solving using DP as above can simplifies the problem, especially When only part of the optimal variable values is required (for example, only y^* is required).

Solution to the Finite LQR Problem

The LQR problem in (6.6) is essentially something similar with (6.7). The

initial state $x(0)$ is a constant in the optimization problem, the control signals $u(0), ..., u(N-1)$ to be optimized, and the coupling introduced by the system dynamics (6.4). The ultimate goal of the optimization problem is to determine $u(0)$.

Similarly, the optimization is divided into multiple stages, where in each stage only one instant $u(k)$ is considered, and other variables leading to that stage is considered as constant. We start with the last control input $u(N-1)$. The optimization problem is formulated as follows.

$$\begin{aligned} V^{\text{sub}}(N-1) &= \frac{1}{2}\left(x(N-1)^T Q x(N-1) + u(N-1)^T R u(N-1)\right) \\ &\quad + \frac{1}{2} x(N)^T P_f x(N) \\ \text{s.t.} &\quad x(N) = Ax(N-1) + Bu(N-1) \end{aligned}$$

where $x(N-1)$ is taken as a constant from another stage.

Solving $u(N-1)$ as a function of $x(N-1)$ gives

$$\begin{aligned} u^*(N-1) &= K_{N-1} x(N-1) \\ x(N) &= (A + BK_{N-1}) x(N-1) \\ V^{\text{sub}}(N-1) &= \frac{1}{2} x(N-1)^T \Pi_{N-1} x(N-1) \end{aligned}$$

where K_{N-1} and Π_{N-1} are two intermediate parameters calculated as follows.

$$\begin{aligned} K_{N-1} &= -(B^T P_f B + R)^{-1} B^T P_f A \\ \Pi_{N-1} &= Q + A^T P_f A - A^T P_f B (B^T P_f B + R)^{-1} B^T P_f A \end{aligned}$$

Consider the next stage optimization problem as follows.

$$\begin{aligned} V^{\text{sub}}(N-2) &= \frac{1}{2}\left(x(N-2)^T Q x(N-2) + u(N-2)^T R u(N-2)\right) \\ \text{s.t.} &\quad x(N-1) = Ax(N-2) + Bu(N-2) \end{aligned}$$

where $x(N-2)$ is taken as constant and $u^*(N-2)$ is to be found. The solution is given by

$$\begin{aligned} u^*(N-2) &= K_{N-2} x(N-2) \\ x(N-1) &= (A + BK_{N-2}) x(N-2) \\ V^{\text{sub}}(N-2) &= \frac{1}{2} x(N-2)^T \Pi_{N-2} x(N-2) \end{aligned}$$

where

$$\begin{aligned} K_{N-2} &= -(B^T \Pi_{N-1} B + R)^{-1} B^T \Pi_{N-1} A \\ \Pi_{N-1} &= Q + A^T \Pi_{N-1} A - A^T \Pi_{N-1} B (B^T \Pi_{N-1} B + R)^{-1} B^T \Pi_{N-1} A \end{aligned}$$

It is clear by now that the above calculations can be done recursively to obtain $u^*(0)$. Notice that all the control signals have the following form

$$u^*(k) = K_k x(k)$$

which implies that the optimal control can be taken as a classic state-space feedback controller with time-varying control gains.

Infinite LQR Problem

Since the recursive calculation is done N times to calculate $K_{N-1}, ..., K_0$, apparently different value of N results in different K_0. It is worth mentioning that optimal control does not suggest stable control. In other words, it is possible for a specific system plant, specific choice of Q, R, P_f, and specific N, to lead to an unstable closed loop system, i.e., $A + BK_0$ with eigenvalues larger than 1. However, it is guaranteed that when $N \to \infty$, K_0 converges, and $A + BK_0$ is always stable. This introduces the infinite LQR problem, where the cost function corresponding with (6.6) is changed to

$$V(x(0), u) = \frac{1}{2} \sum_{k=0}^{\infty} \left(x(k)^T Q x(k) + u(k)^T R u(k) \right)$$

Infinite LQR can be taken as a special case of finite LQR where $N \to \infty$.

We have been making an underlying assumption that the original plant is controllable. This means that the system can be driven from any initial state to $x = 0$ in finite time. Apply this concept to infinite LQR problem. This implies that in the beginning finite steps of this infinite control sequence, the state should have already been regulated to zero.

Since the control gain $K(k), k = N-1, N-2, ...$ converges to K, the first infinite control gains would be K. Therefore, it can be concluded that the infinite LQR suggests the control gain to remain K until all states becomes zero. Therefore, the infinite LQR problem results in constant control gain rather than time-varying control gains as in the finite LQR problem. The control gain can be obtained by solving the Riccati equation of Π as follows.

$$\begin{aligned} K &= -(B^T \Pi B + R)^{-1} B^T \Pi A \\ \Pi &= Q + A^T \Pi A - A^T \Pi B (B^T \Pi B + R)^{-1} B^T \Pi A \end{aligned}$$

6.1.4 Commonly Seen MPC Objectives

The most commonly seen MPC objectives are tracking, state regulation, and disturbance rejection.

- Tracking: the output of the plant shall follow the given reference point (not necessarily to be a constant) as close as possible.

- State regulation: the state vector of the plant shall converge to a given steady-state as soon as possible, while not violating system restrictions.
- Disturbance rejection: the plant shall maintain its operating point under unknown stochastic noise and disturbances.

7

Adaptive Control System

CONTENTS

Adaptive controller is generally referred to a controller with adjustable parameters and associated mechanism to adjust such parameters, to conform to new circumstances.

Comparing with most of the other control systems, an adaptive control system does not require a very precise modeling of the plant. More precisely, an adaptive control system can adapt the control gains to the unknown or time-varying plant and maintain the stability of the closed-loop system.

The references of this chapter include:

- Åström, K.J. and Wittenmark, B., 2013. Adaptive control. Courier Corporation [1].

7.1 Introduction

Adaptive controller is generally referred to a controller with adjustable parameters and associated mechanism to adjust such parameters, to conform to new circumstances.

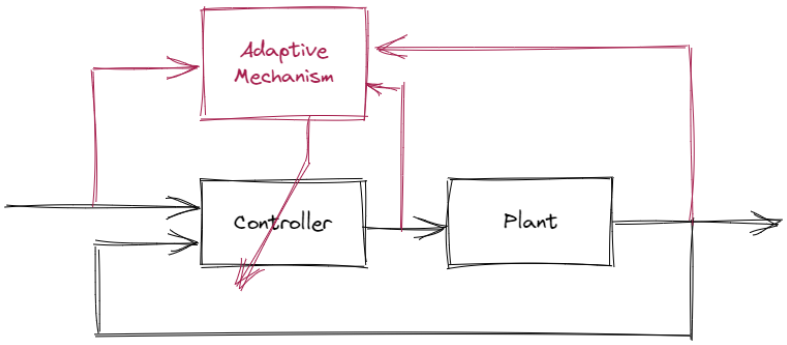

FIGURE 7.1
Adaptive control system general schema.

Due to the parameter adjustment mechanism, the controller becomes nonlinear. For example, while a conventional PID controller is linear, an adaptive PID controller is nonlinear since the $P(t)$, $I(t)$ and $D(t)$ are variables instead of constant gains.

In practice, an adaptive controller is often a modified version of a conventional controller just like the adaptive PID controller example. There are often 2 types of feedback loops in an adaptive control system: the original loops that comes with the conventional controller, and the additional loops to adjust the parameters of the conventional controller. A demonstrative figure is given in Fig. 7.1.

7.1.1 A Brief History of Adaptive Control System

The first adaptive control system application was designed for autopilots of a high-performance aircraft, where the conventional control system was found to work for one condition but not for the other. Gain scheduling was adopted to solve this problem.

Later on more sophisticated adaptive control systems that utilizes state-space model and stability theory were introduced. Stochastic control theory and system identification techniques were developed side-by-side.

The proof of stability of the adaptive control system, especially universal stability of the system, started to draw attention. The proof can be done often only with strong restrictions. People started to investigate the connection and difference of robust control and system identification with adaptive control and system identification. Computer controlled system and artificial neural network started to emerge and people realized the connection between adaptive control and computer learning.

Adaptive control system commercialization started in 1980s, and are used

in handling process dynamics and disturbances, and to provide automatic tuning of controller parameters.

7.1.2 Conventional Control System

The most conventional way of designing a control system is as follows.

1. Assume a linear model of the plant at the operating point.
2. Calculate or estimate the parameters of the plant model.
3. Design a closed-loop control system for the plant.

The control system designed following the above approach is usually intrinsically insensitive to modeling error and disturbance to some extend (due to the closed-loop implementation). But there are difficulties that could cause variation and troubles.

- **Nonlinear actuator** is one of the sources of modeling uncertainty, making the control system work only on/near the pre-determined operating point, but not universally.
- **Flow and speed variations** in the pipes of a system, if exist, can change from time to time. The different flow and speed can cause changes in the process dynamics, thus change the plant model.
- **Environmental dynamics and uncertainties** often draw significant concerns in control. For example, the aircraft flying at different height and speed suffers from different environmental conditions. The same applies to ship steering at different speed.

7.1.3 Adaptive Control Schema

Different adaptive control schemes have been designed to tackle the above issues. The commonly used ones are briefly introduced here.

Gain Scheduling

Measurement feedback may correlate well with changes in the process dynamics. The idea of gain scheduling is to map the process dynamic parameters with the control parameters, by either a function manner or a table lookup manner, i.e., schedule the control parameters to compensate the the process dynamics.

A demonstrative figure is given in Fig. 7.2.

Gain scheduling is one of the most intuitive adaptive control schema, and has the earliest use case among all.

Model-Reference Adaptive System (MRAS)

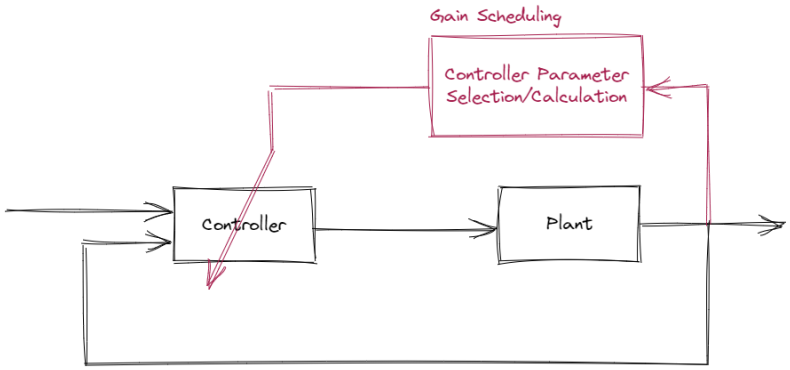

FIGURE 7.2
Gain scheduling schema.

In this scenario, a reference model is built that generates the "ideal" output of the closed-loop system given the reference point. This is often a virtual model that does not suffer from disturbances and error.

The MRAS then uses the output of both the actual plant and the reference model to calculate the controller parameter adjustment mechanism to minimize the difference between the outputs of the two models.

A demonstrative figure is given in Fig. 7.3.

The MRAS is somewhat like a learning system. Usually, there is a "adaptation rate" (just like the learning rate) that controls the adapting speed.

Self-Tuning Regulator (STR)

The idea of STR is to estimate the system process and design and change the whole control system in real-time. Unlike the gain scheduling approach and MRAS where parameter adjustment mechanism is calculated in real-time, in STR approach, the controller designing problem is solved in real-time. From this sense, STR can be taken as the "high-end" gain scheduling problem.

A demonstrative figure is given in Fig. 7.4.

STR can be made very flexible, because it is essentially a new control system design from scratch each time the environment changes.

Dual Control

The aforementioned adaptive control schemas do not taken into account the stochastics and statistics of the measurement and estimation uncertainty.

To analyze the performance of the system under estimation uncertainty and to develop control algorithm that can optimize the system performance under such uncertainty, nonlinear stochastic control theory needs to be used, which leads to the notion of dual control.

A demonstrative figure is given in Fig. 7.5.

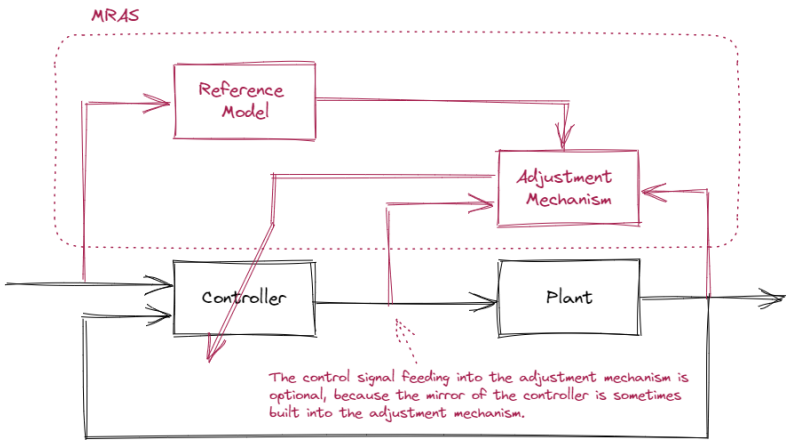

FIGURE 7.3
MARS schema.

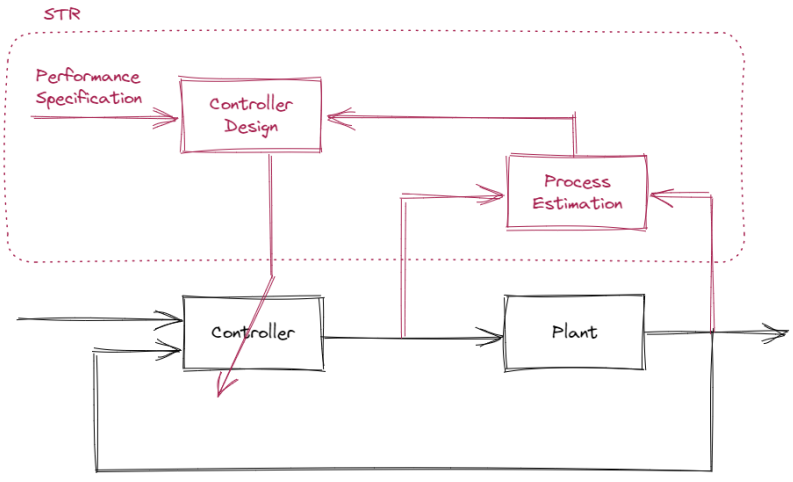

FIGURE 7.4
STR schema.

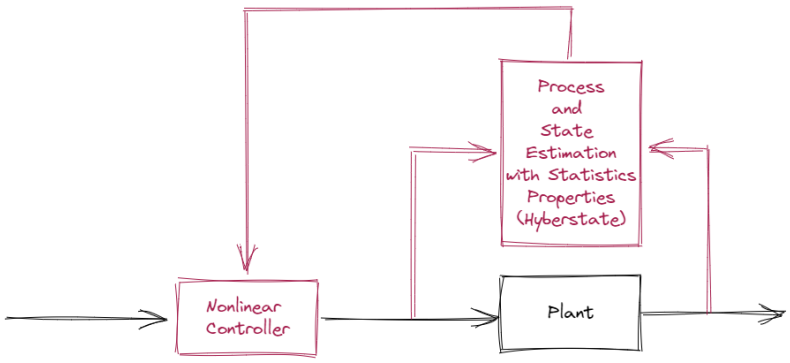

FIGURE 7.5
Dual control schema.

This control system is essentially a nonlinear controller with a robust/adaptive filter that estimates the process dynamics and the states of the plant, together with statistics information of the estimates.

7.1.4 A Typical Adaptive Control Problem Formulation

For simplicity, consider the following LTI plant realizations.

Continuous time domain state space model:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned}$$

Continuous time domain transfer function model:

$$G(s) = \frac{B(s)}{A(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_m}{s^n + a_1 s^{n-1} + \ldots + a_n}$$

Continuous time domain input-output model (with p the differential operator, $p = \frac{d}{dt}$):

$$y(t) = G(p)u(t)$$

Discrete time domain state-space model:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned}$$

Discrete time domain transfer function:

$$G(z) = \frac{B(z)}{A(z)} = \frac{b_0 z^m + b_q z^{m-1} + \ldots + b_m}{z^n + a_1 z^{n-1} + \ldots + a_n}$$

Discrete time domain input-output model (with q the forward shift operator, $qy(k) = y(k+1)$):

$$y(k) \quad = \quad H(q)u(k)$$

The above models may be describing the same plant, with different level of details. For example, when dealing with transfer function and input-output models, the initial condition of the system is often ignored. When dealing with discrete time domain plant model, it is open a sampled system derived from the original system.

If the parameters in the models are known, it is straight forward to design a closed-loop controller for them. This is called the underlying design problem. The adaptive control problem, on the other hand, is to design a controller without knowing the parameters. In response, the controller comes with adjustable control gains and associated adjustment mechanism.

A typical adaptive controller design shall include the following procedures:

1. Characterize the desired closed-loop behavior.
2. Determine a suitable control law with adjustable parameters.
3. Design adjustment mechanism.
4. Implement the control law with the parameters adjusted using the adjustment mechanism.

Some successful use cases for adaptive control systems include

- Automatic tuning controller, such as the PID automatic tuning controller. In practice, all conventional control systems can benefit from automatic parameter tuning, if done correctly.
- Gain scheduling, which is the standard technique used in flight control system for high-performance aircraft, and is becoming increasingly popular in industrial process control.
- Continuous adaptation control for continuously varying system.
- "Human-in-the-loop" control system.

Since adaptive control system is often more costly and complicated than a conventional constant-gain controller, it should be used wisely and not abused. If the dynamics of the system are manageable, consider using parametric model based control algorithm rather than the adaptive control system, the later of which is usually more complicated.

7.2 Parameter Estimation

Parameter estimation (or more generally, system identification) is widely used, either directly or indirectly, in many adaptive control schemas. An obvious example is STR, where the process dynamics estimation is built into the controller. Parameter estimation also happens implicitly in MRAS.

The key factors in system identification include the following.

- Selection of model structure.
- Experiment design.
- Parameter and state estimation.
- Validation.

One thing to notice is that in adaptive control scope, the plant is assumed to change continuously, thus the parameter estimation needs to be done in a recursive manner.

7.2.1 Least Squares Estimation

Karl Friedrich Gauss defines LS estimation problem as follows.

LS estimation:

The parameters in the model should be chose such that the sum of the squares of the differences between the observed and the computed values, multiplied by numbers that measure the degree of precision, is a minimum.

Notice that from the above definition, if all observations share the same degree of precision, the multipliers to all observations shall be the same, otherwise they should be different. Conventionally, the later case is referred as "Weighted LS (WLS) estimation" where each multiplier is called the "weight" associated with the observation. The more precise an observation, the higher its associated weight. On the other hand, the former case where all observations share the same precision is referred as LS estimation.

To reduce confusion, from now on LS estimation and WLS estimation are used to refer to the former and later cases, respectively.

Formulate LS estimation problem on a LTI system as follows. Let $\theta_i, i = 1, ..., n$ be n unknown parameters in the model and $y(j), j = 1, ..., m$ be m

measurements collected from the mode. It is required that $m \geq n$. Variable $\phi_i(j)$ is called a regression variable (or a regressor) that links the measurements with the model parameters, so that

$$y(j) = \phi_1(j)\theta_1 + \phi_2(j)\theta_2 + \ldots + \phi_n(j)\theta_n + \epsilon(j) \tag{7.1}$$

where $\epsilon(j)$ is assumed to be independent measurement noise associated with $y(j)$. Equation (7.1) is called a regression model. Putting (7.1) into matrix form gives

$$y = \Phi\theta + \epsilon \tag{7.2}$$

where

$$\begin{aligned} y &= \begin{bmatrix} y(1) & \ldots & y(m) \end{bmatrix}^T \\ \Phi &= \begin{bmatrix} \phi_1(1) & \ldots & \phi_n(1) \\ \vdots & \ddots & \vdots \\ \phi_1(m) & \ldots & \phi_n(m) \end{bmatrix} \\ \theta &= \begin{bmatrix} \theta_1 & \ldots & \theta_n \end{bmatrix}^T \\ \epsilon &= \begin{bmatrix} \epsilon(1) & \ldots & \epsilon(m) \end{bmatrix}^T \end{aligned}$$

with y the measurement vector, Φ the regression model matrix, and θ the unknown system parameter vector.

LS estimation of θ in (7.2) is formulated as follows. Select estimation $\hat{\theta} = [\hat{\theta}_1, ..., \hat{\theta}_n]^T$ so that the following cost function $V(\hat{\theta})$ is minimized.

$$V(\hat{\theta}) = \frac{1}{2}\sum_{i=1}^{m} e(i)^2 \tag{7.3}$$

where

$$\begin{aligned} e(i) &= y(i) - \left(\phi_1(i)\hat{\theta}_1 + \phi_2(i)\hat{\theta}_2 + \ldots + \phi_n(i)\hat{\theta}_n\right) \\ &= y(i) - \phi(i)^T\hat{\theta} \end{aligned} \tag{7.4}$$

with

$$\phi(i)^T = \begin{bmatrix} \phi_1(i) & \ldots & \phi_n(i) \end{bmatrix} \tag{7.5}$$

which is the ith row of Φ. Variable $e(i)$ is called the residual.

Put everything into matrix format. Equation (7.3) becomes

$$\begin{aligned} V(\hat{\theta}) &= \frac{1}{2}\sum_{i=1}^{m}\left(y(i) - \phi(i)^T\hat{\theta}\right)^2 \\ &= \frac{1}{2}(y - \Phi\hat{\theta})^T(y - \Phi\hat{\theta}) \end{aligned} \tag{7.6}$$

Denote $e = [e(1), ..., e(m)]^T$. From (7.4),

$$e = y - \Phi\hat{\theta} \tag{7.7}$$

and (7.6) becomes

$$V(\hat{\theta}) = \frac{1}{2}e^T e \tag{7.8}$$

The quadratic optimization problem given in (7.8) has the following unique analytical solution. Differentiating (7.8) w.r.t. θ gives (using numerator-layout notation)

$$\frac{dV(\hat{\theta})}{d\theta} = \frac{dV(\hat{\theta})}{de}\frac{de}{d\theta} \tag{7.9}$$

$$= -e^T\Phi \tag{7.10}$$

Equating (7.10) to zero gives the minimum of (7.8) as follows.

$$-e^T\Phi = 0 \tag{7.11}$$

Substituting (7.7) into (7.11) gives

$$\begin{aligned} -\left(y - \Phi\hat{\theta}\right)^T\Phi &= 0 \\ -\Phi^T\left(y - \Phi\hat{\theta}\right) &= 0 \\ \Phi^T\Phi\hat{\theta} &= \Phi^T y \\ \hat{\theta} &= \left(\Phi^T\Phi\right)^{-1}\Phi^T y \end{aligned} \tag{7.12}$$

provided that $\Phi^T\Phi$ is nonsingular, which is known as the excitation condition. Equation (7.12) is the solution to the LS estimation problem in (7.3).

A Quick Note on Matrix Differentiation

In numerator-layout notion, the scalar $V(\hat{\theta})$ differentiated w.r.t. the vector e in (7.9) is given by

$$\begin{aligned} \frac{dV(\hat{\theta})}{de} &= \begin{bmatrix} \frac{dV(\hat{\theta})}{de(1)} & \cdots & \frac{dV(\hat{\theta})}{de(m)} \end{bmatrix} \\ &= \begin{bmatrix} e(1) & \ldots & e(m) \end{bmatrix} \\ &= e^T \end{aligned}$$

The vector e differentiated w.r.t. the vector θ is given by

$$\begin{aligned}\frac{de}{d\theta} &= \begin{bmatrix} \frac{de(1)}{d\theta_1} & \cdots & \frac{de(m)}{d\theta_n} \\ \vdots & \ddots & \vdots \\ \frac{de(m)}{d\theta_1} & \cdots & \frac{de(m)}{d\theta_n} \end{bmatrix} \\ &= \begin{bmatrix} -\phi_1(1) & \cdots & -\phi_n(1) \\ \vdots & \ddots & \vdots \\ -\phi_1(m) & \cdots & -\phi_n(m) \end{bmatrix} \\ &= -\Phi\end{aligned}$$

where

$$\frac{de(i)}{d\theta_j} = -\phi_j(i)$$

because of (7.4).

Therefore,

$$\frac{dV(\hat{\theta})}{d\theta} = \frac{dV(\hat{\theta})}{de}\frac{de}{d\theta} = -e^T\Phi$$

which is (7.10).

A WLS estimation problem can be formulated similarly, by changing (7.3) with

$$V(\hat{\theta}) = \frac{1}{2}\sum_{i=1}^{m} w(i)e(i)^2$$

The solution to a WLS estimation can be similarly derived and the result is

$$\hat{\theta} = \left(\Phi^T W\Phi\right)^{-1}\Phi^T W y$$

where $W = \text{diag}(w(1), ..., w(m))$.

7.2.2 Statistics Properties of LS Estimation

In the discussions below, We will assume that all measurement noise $\epsilon(i), i = 1, ..., m$ are i.i.d. noise, whose mean and variance are zero and σ^2 respectively.

Substituting (7.2) into (7.12) gives

$$\begin{aligned}\hat{\theta} &= \left(\Phi^T\Phi\right)^{-1}\Phi^T\left(\Phi\theta + \epsilon\right) \\ &= \theta + \left(\Phi^T\Phi\right)^{-1}\Phi^T\epsilon \qquad (7.13)\end{aligned}$$

Using (7.13), the following statistics properties of LS estimation can be drawn.

$$\begin{aligned} E\left[\hat{\theta}\right] &= \theta \\ \text{Cov}\left[\hat{\theta}\right] &= E\left[\hat{\theta}\hat{\theta}^T\right] \\ &= \sigma^2\left(\Phi^T\Phi\right)^{-1} \end{aligned}$$

which suggests that the estimator is zero mean. With the fixed number n of parameters to be estimated, the estimation variance decreases as the number m of observations increases, and the variance converges to zero eventually. The variance of the noise σ^2 can potentially be estimated from the state estimate $\hat{\theta}$. Details are given as follows. Substituting (7.2), (7.12) into (7.7) gives

$$e = \left(I - \Phi\left(\Phi^T\Phi\right)^{-1}\Phi^T\right)\epsilon \tag{7.14}$$

Using (7.14),

$$E\left[ee^T\right] = \sigma^2\left(I - \Phi\left(\Phi^T\Phi\right)^{-1}\Phi^T\right)\left(I - \Phi\left(\Phi^T\Phi\right)^{-1}\Phi^T\right)^T \tag{7.15}$$

Taking the trace of matrix from both sides of (7.15) gives

$$E\left[e^Te\right] = (m-n)\sigma^2 \tag{7.16}$$

From (7.8) and (7.16),

$$\sigma^2 = \frac{2}{m-n}E\left[V(\hat{\theta})\right]$$

can be used to estimate the variance of the noise.

7.2.3 Recursive LS Estimation

Recursive state estimation allows "updating" the state estimate each time a new measurement set becomes available. It takes advantages of previously calculated state estimate with less measurements, instead of calculating everything from scratch, which is also known as the batch estimation. Recursive estimation saves computational and memory burden than the batch estimation. Given that observations are often naturally obtained sequentially in real time, recursive estimation has a large demand in practice.

Let each measurement be obtained associated with a incremental time stamp $k = 1, 2, ...$ While the number of parameters to be estimated remains constant n, the number of measurement m is dynamic and replaced by the notion k.

It is worth mentioning that the size of Φ in (7.2) grows with k, in the sense that each time a new measurement becomes available, Φ grows by one

row. For convenience, $\Phi(k)$ is used to describe Φ in (7.2) when there are k measurements.

As more and more measurements come in, the state estimate $\hat{\theta}$ also evolves and become more and more accurate. Denote $\hat{\theta}(k)$ the state estimate when up to k measurements are available, $k \geq n$.

In recursive estimation schema, the estimator is formulated as a Markov process, and the state estimate $\hat{\theta}(k+1)$ is formulated as a function of $\hat{\theta}(k)$ and $y(k+1)$, and does not directly involve the earlier state estimates $\hat{\theta}(k-1), \hat{\theta}(k-2), ...$ or measurements $y(k), y(k-1), ...$ This is possible because the information of $\hat{\theta}(k-1), \hat{\theta}(k-2), ...$ and $y(k), y(k-1), ...$ are integrated into $\hat{\theta}(k)$.

The derivation of the recursive estimation is as follows. Notice that $\phi(i)^T$ in (7.5) is the ith row of $\Phi(k)$, hence

$$\Phi(k)^T \Phi(k) = \sum_{i=1}^{k} \phi(i)\phi(i)^T \tag{7.17}$$

Assume that the excitation condition holds. Denote $P(k) = \left(\Phi(k)^T \Phi(k)\right)^{-1}$, or equivalently $P(k)^{-1} = \Phi(k)^T \Phi(k)$. Use (7.17),

$$P(k+1)^{-1} = P(k)^{-1} + \phi(k+1)\phi(k+1)^T \tag{7.18}$$

which can be calculated recursively.

Rewrite (7.12) as follows.

$$\begin{aligned}
\hat{\theta}(k) &= \left(\sum_{i=1}^{k} \phi(i)\phi(i)^T\right)^{-1} \sum_{i=1}^{k} \phi(i)y(i) \\
&= P(k) \sum_{i=1}^{k} \phi(i)y(i) \\
\hat{\theta}(k+1) &= \left(\sum_{i=1}^{k+1} \phi(i)\phi(i)^T\right)^{-1} \sum_{i=1}^{k+1} \phi(i)y(i) \\
&= P(k+1) \left(\sum_{i=1}^{k} \phi(i)y(i) + \phi(k+1)y(k+1)\right) \\
&= P(k+1) \left(P(k)^{-1}\hat{\theta}(k) + \phi(k+1)y(k+1)\right) \\
&= P(k+1) \left(\left(P(k+1)^{-1} - \phi(k+1)\phi(k+1)^T\right)\hat{\theta}(k) \right. \\
&\quad \left. +\phi(k+1)y(k+1)\right) \\
&= \hat{\theta}(k) \\
&\quad +P(k+1)\phi(k+1)\left(y(k+1) - \phi(k+1)^T\hat{\theta}(k)\right)
\end{aligned} \tag{7.19}$$

Equation (7.18) and (7.19) give the recursive LS estimation. Notice that the (7.18) gives the recursive calculation of $P(k)^{-1}$. The recursive calculation of $P(k)$ can be easily derived from (7.18) using matrix inversion lemma as follows.

$$\begin{aligned}
P(k+1) &= P(k) - P(k)\phi(k+1) \\
&\quad \times \left(I + \phi(k+1)^T P(k)\phi(k+1)\right)^{-1} \phi(k+1)^T P(k) \\
P(k+1)\phi(k+1) &= P(k)\phi(k+1) \\
&\quad \times \Big(I - \left(I + \phi(k+1)^T P(k)\phi(k+1)\right)^{-1} \\
&\quad \times \phi(k+1)^T P(k)\phi(k+1)\Big) \\
&= P(k)\phi(k+1)\left(I + \phi(k+1)^T P(k)\phi(k+1)\right)^{-1} \\
&\quad \times \left(\left(I + \phi(k+1)^T P(k)\phi(k+1)\right)\right. \\
&\quad \left. -\phi(k+1)^T P(k)\phi(k+1)\right) \\
&= P(k)\phi(k+1)\left(I + \phi(k+1)^T P(k)\phi(k+1)\right)^{-1}
\end{aligned}$$

where notice that in the above equations the identity matrix I under the scope of our discussion should be 1, as the measurement at any instant, $y(k)$, is a scalar. We followed the convention using I because in practice, the measurement at any instant can be a vector. Hopefully this explanation helps to clear some confusion.

Denote $K(k) = P(k)\phi(k)$ the Kalman gain, and residual $\varepsilon(k+1) = y(k+1) - \phi(k+1)^T\hat{\theta}(x)$ the innovation vector, where $\phi(k+1)^T\hat{\theta}(x)$ can be interpreted as a "guess" of $y(k+1)$ using the state estimate $\hat{\theta}(x)$. Recursive LS estimation can be taken as a special case of Kalman filter where (a) measurement $y(k)$ is a scalar; (b) process matrix is unit matrix; (c) input and process noise are both zero.

7.2.4 LS Estimation with Time-Varying Parameters

If the parameter is time varying and has a solid dynamic model, considering using Kalman filter and other parametric model based estimators. If not, consider two scenarios: (a) parameters changes abruptly and infrequently; (b) parameters changes slowly and continuously.

Consider case (a). In this case, the abrupt changes can be detected from the magnitude of the innovation vector. The innovation vector covariance is jointly decided by the measurement noise and the state estimate precision. When the parameter is constant, the covariance matrix converges to a certain value that can be calculated analytically. A simple test can be used to detect when the innovation vector violates the statistics assumption, indicating that there is an abrupt parameters change.

In such case, just "reset" the system by assuming the new measurement is the first set of measurement collected by the system. In practice, this can

be achieved equivalently by reset $P(k-1)$ and/or $P(k)$ to a diagonal matrix with large elements.

Consider case (b). One way to handle this parameter change is to use moving horizon estimation, or introducing a forgetting factor into the cost function. An example of introducing forgetting factor is given as follows. In (7.6), set

$$V(\hat{\theta}) = \frac{1}{2}\sum_{i=1}^{k}\lambda^{k-i}\left(y(i)-\phi(i)^T\hat{\theta}\right)^2 \tag{7.20}$$

where $0<\lambda<1$ is a forgetting factor, and the method given in (7.20) is called exponential forgetting. By introducing the forgetting factor, the contribution of innovation introduced by early measurements and state estimates to the cost function is reduced, thus reducing their impact on the state estimates.

The batch and recursive algorithms to calculate the state estimates for (7.20) can be derived similarly. Some key derivations are given below.

Equation (7.12) becomes

$$\begin{aligned}\hat{\theta} &= \left(\Phi^T\Lambda\Phi\right)^{-1}\Phi^T\Lambda y\\ \Lambda &= \text{diag}\begin{pmatrix}\lambda^{m-1} & \dots & \lambda & 1\end{pmatrix}\end{aligned}$$

Equation (7.18) becomes

$$\begin{aligned}P(k)^{-1} &= \Phi(k)^T\Lambda\Phi(k)\\ P(k+1)^{-1} &= \lambda P(k)^{-1}+\phi(k+1)\phi(k)^T\end{aligned}$$

Applying matrix inversion lemma to the above equation gives

$$\begin{aligned}P(k+1) &= \frac{1}{\lambda}P(k)-\frac{1}{\lambda}P(k)\phi(k+1)\\ &\quad\times\left(\lambda I+\phi(k+1)^TP(k)\phi(k+1)\right)^{-1}\phi(k+1)^TP(k)\\ K(k+1) &= P(k+1)\phi(k+1)\\ &= P(k)\phi(k+1)\left(\lambda I+\phi(k+1)^TP(k)\phi(k+1)\right)^{-1}\end{aligned}$$

Equation (7.19) has the same eventual form of

$$\hat{\theta}(k+1) = \hat{\theta}(k)+P(k+1)\phi(k+1)\left(y(k+1)-\phi(k+1)^T\hat{\theta}(x)\right)$$

7.2.5 Simplified LS Estimation Methods

The recursive LS algorithm (7.18) and (7.19) requires the updates of $P(k)$ which involves matrix inversion, where $P(k)$ has a size of $n\times n$, and n r2 the number of parameters to be estimated. When n is large, this calculation can be time consuming.

Consider interpreting the cost function (7.6) as follows.

$$\hat{\theta}(k) = \arg\min_{\hat{\theta}} V_k(\hat{\theta}) = \arg\min_{\hat{\theta}} \frac{1}{2}\sum_{i=1}^{k}\left(y(i) - \phi(i)^T\hat{\theta}\right)^2 \quad (7.21)$$

$$\hat{\theta}(k+1) = \arg\min_{\hat{\theta}} V_{k+1}(\hat{\theta}) = \arg\min_{\hat{\theta}} \frac{1}{2}\sum_{i=1}^{k+1}\left(y(i) - \phi(i)^T\hat{\theta}\right)^2 \quad (7.22)$$

where

$$V_i(\hat{\theta}) = \frac{1}{2}\sum_{j=1}^{i}\left(y(j) - \phi(j)^T\hat{\theta}\right)^2$$

is the cost function formulated using the first i measurements $y(1), ..., y(i)$.

Formulating (7.22) as a function of $V_k(\hat{\theta})$ gives

$$\hat{\theta}(k+1) = \arg\min_{\hat{\theta}}\left[V_k(\hat{\theta}) + \frac{1}{2}\left(y(k+1) - \phi(k+1)^T\hat{\theta}\right)^2\right] \quad (7.23)$$

We already know that $\hat{\theta}(k)$ minimizes $V_k(\hat{\theta})$ from (7.21), and $V_k(\hat{\theta})$ happens to be the first term in (7.23). Therefore, (7.23) is essentially an optimization problem that concerns the following terms

$$\|\hat{\theta} - \hat{\theta}(k)\| \quad (7.24)$$

and

$$\|y(k+1) - \phi(k+1)^T\hat{\theta}\| \quad (7.25)$$

where $\|\cdot\|$ denotes the Euclidean norm of a vector, and the weights associated with the two terms in the optimization problem are determined by the covariance of the state estimate and measurement noise.

Consider a modification to the optimization problem in (7.23) as follows. Let (7.24) be the only term in the optimization, and (7.25) be a equality constraint, i.e.,

$$\min \quad \frac{1}{2}\left(\hat{\theta} - \hat{\theta}(k)\right)^T\left(\hat{\theta} - \hat{\theta}(k)\right) \quad (7.26)$$

$$\text{s.t.} \quad y(k+1) - \phi(k+1)^T\hat{\theta} = 0 \quad (7.27)$$

Notice that this modified optimization problem given by (7.26) does not necessarily give the same result as the original problem in (7.23). This is because $\hat{\theta}(k+1)$ in (7.23) may not meet the equality constraint. Nevertheless, (7.26) is an approximation to $\hat{\theta}(k+1)$ in (7.23).

Optimization (7.26) can be solved using Lagrange multiplier method. Details are neglected here, and the result is given by

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \frac{\phi(k+1)}{\phi(k+1)^T\phi(k+1)}\left(y(k+1) - \phi(k+1)^T\hat{\theta}(k)\right) \quad (7.28)$$

Parameters $0 < \gamma < 2$ and $\alpha > 0$ are added to (7.28) as shown below. The motivations of adding γ and α are to make it possible to change the step length of the parameter adjustment and to avoid a potential problem what would occur when $\phi(k+1) = 0$.

$$\hat{\theta}(k+1) \quad = \quad \hat{\theta}(k) + \frac{\gamma\phi(k+1)}{\alpha + \phi(k+1)^T\phi(k+1)} \Big(y(k+1) - \phi(k+1)^T\hat{\theta}(k)\Big) \tag{7.29}$$

which is known as the projection algorithm. The choice $0 < \gamma < 2$ guarantees the convergency of (7.29).

One may think the quality constraint (7.27) is too strong. When measurement error is considered, a modified version of (7.28) is given below.

$$\hat{\theta}(k+1) \quad = \quad \hat{\theta}(k) + \frac{\phi(k+1)}{\sum_{i=1}^{k+1} \phi(i)^T\phi(i)} \Big(y(k+1) - \phi(k+1)^T\hat{\theta}(k)\Big) \tag{7.30}$$

which is known as the stochastic approximation (SA) algorithm.

A further simplification to (7.29) and (7.30) is given below.

$$\hat{\theta}(k+1) \quad = \quad \hat{\theta}(k) + \gamma\phi(k+1) \Big(y(k+1) - \phi(k+1)^T\hat{\theta}(k)\Big)$$

which is known as the least mean square (LMS) algorithm.

7.2.6 Plant Models for LS Estimation

The LS estimation algorithm can be applied to the models introduced in this section. By assuming a model, experiments can be designed and empirical data can be collected which serves as the input to the LS estimation algorithm. The parameters of the model can then be decided.

The initial states of the plant is ignored for now. In practice, with the model and parameters known, state estimation is carried out to determine the state of the system.

Finite Impulse Response (FIR) Model

An LTI system can be uniquely characterized by its impulse response. A demonstrative figure is given in Fig. 7.6. For an asymptotically stable system, the transient will converge to zero exponentially fast, depending on the position of the poles in the transfer function.

Consider sampling the outputs of Fig. 7.6 in discrete time domain. The input at k, i.e., $u(k)$, would affect the consequent outputs $y(k+1), y(k+2), ...$ This sequence can go infinitely long in theory, but in practice we only consider the first finite number of outputs before y settles to zero. The length of the sequence is determined by the sampling period and the time constant of the system.

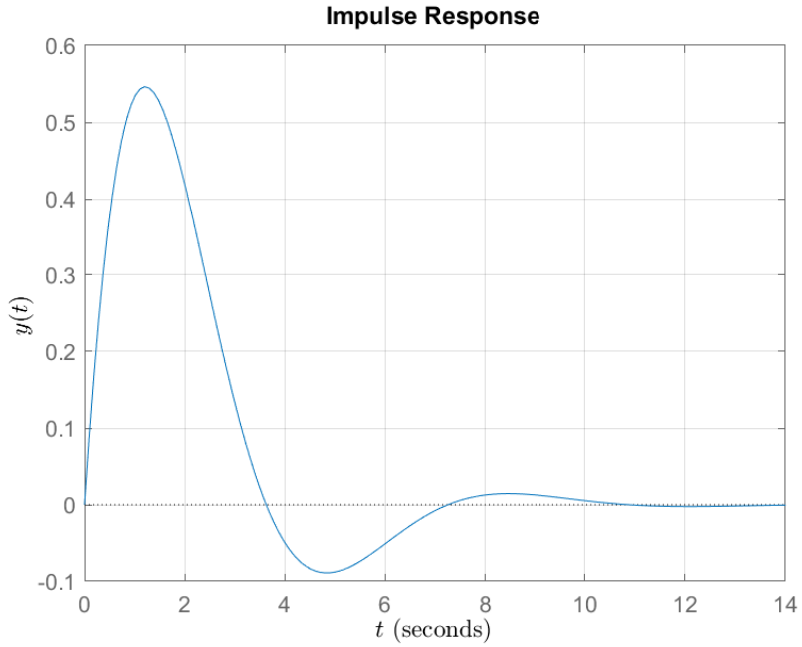

FIGURE 7.6
LTI system impulse response demonstration.

Due to the superposition of the LTI system, the above also indicates that output of the system at k, i.e., $y(k)$, is affected by the past input signals $u(k-1), u(k-2), ...$, as given by the following equation. Likewise, only finite number of inputs are included in the equation.

$$y(k) = b_1 u(k-1) + b_2 u(k-2) + \ldots + b_n u(k-n) + \epsilon(k) \quad (7.31)$$

Equation (7.31) is known as the FIR model of a system. It has n parameters to be estimated, and it fits well into the LS estimation schema described by (7.1).

General Transfer Function Model

Consider a more general LTI model described by the following input-output relations.

$$\begin{aligned} A(q)y(k) &= B(q)u(k) \\ A(q) &= q^n + a_1 q^{n-1} + \ldots + a_n \\ B(q) &= b_1 q^{m-1} + b_2 q^{m-2} + \ldots + b_m \end{aligned}$$

where q is the forward shift operator $qy(k) = y(k+1)$. The goal of the parameter estimation is to estimate parameters $a_i, i = 1, ..., n$ and $b_i, i = 1, ..., m$.

Rewrite (7.32) as follows.

$$\begin{aligned} y(k) = & -a_1 y(k-1) - a_2 y(k-2) - \ldots - a_n y(k-n) \\ & + b_1 u(k-1) + b_2 u(k-2) + \ldots + b_m u(k-m) \end{aligned} \quad (7.32)$$

In (7.32), treat $y(k-1), ..., y(k-n)$ and $u(k-1), ..., u(k-m)$ as regressors and a LS estimation problem can be formulated.

Nonlinear Model

LS estimation can be applied to nonlinear models as long as the output to the parameters to be estimated are of linear relationship. An example is given as follows. Consider

$$y(k) + ay(k-1) = b_1 u(k-1) + b_2 \sin(u(k-2))$$

This is a nonlinear model due to the sine function applied to $u(k-2)$. However, it has no impact on parameter estimation, as $y(k)$ is linear to a, b_1 and b_2. The nonlinear portion is wrapped into the regressor.

Stochastic Model

LS estimation is designed to handle independent and identically distributed Gaussian noise. This indicates that the noise should be white noise, i.e., $\epsilon(i)$ and $\epsilon(j)$, $i \neq j$ shall be independent. In addition, the noise shall not correlate with the regressor by any means.

In the case where the noise are not while, i.e., $\epsilon(i)$ and $\epsilon(j)$ correlates for

some $i \neq j$, it is possible to get the parameters estimate as follows. Formulate the system dynamic model as

$$\begin{aligned} A(q)y(k) &= B(q)u(k) + C(q)\varepsilon(k) \\ y(k) &= -a_1 y(k-1) - \ldots - a_n y(k-n) \\ &\quad + b_1 u(k-1) + \ldots b_m u(k-m) \\ &\quad + c_1 \varepsilon(k-1) + \ldots + c_n \varepsilon(k-n) \end{aligned} \tag{7.33}$$

where $\varepsilon(k)$ are re-formulated i.i.d. noise and $C(q)$ is used to describe the correlation of the noise. The parameters in $C(q)$ will be estimated together with $A(q)$ and $B(q)$ to get a sense of the bandwidth of the original noise $\epsilon(k)$.

Notice that (7.33) is different from (7.32) because ε on the right side of (7.33) is unknown, while everything on the right side of (7.32) is known. To tackle this issue, measurement residual is used to approximate ε, i.e.,

$$\begin{aligned} e(k) &= y(k) - \psi(k-1)^T \hat{\theta}(k-1) \\ \psi(k-1) &= \begin{bmatrix} -y(k-1) & \ldots & -y(k-n) \\ u(k-1) & \ldots & u(k-m) \\ e(k-1) & \ldots & e(k-n) \end{bmatrix}^T \\ \theta &= \begin{bmatrix} a_1 & \ldots & a_n & b_1 & \ldots & b_m & c_1 & \ldots & c_n \end{bmatrix}^T \end{aligned} \tag{7.34}$$

is used as a replacement of $\varepsilon(k)$ when formulating the parameter estimation. Notice that $\hat{\theta}(k-1)$ is the most recent parameter estimation update before receiving $y(k)$. This method is known as the extended least square estimation.

There is a similar alternative where the residual $e(k)$ is estimated using another regression model

$$\hat{C}(q)e(k) = \hat{A}(q)y(k) - \hat{B}(q)u(k)$$

instead of (7.34). This is known as the recursive maximum likelihood method.

7.2.7 Practical Issues in Parameter Estimation

The precision of the parameter estimate is largely affected by practical issues such as experimental conditions. In performing system identification automatically as in adaptive control, the precision can be crucial to the performance of the control system. This section studies the influence of the practical issues to the parameter estimate precision.

Excitation Condition in FIR Model

As introduced earlier, to carry out parameter estimation for an FIR model given by (7.31), a common way is to supply a series of control signal to the model, and use the output of the model to do LS estimation. The control

signal needs to be designed strategically so that the excitation condition is satisfied.

Substitute (7.31) into (7.1) and (7.2) to get the excitation condition for the FIR model as follows.

Part IV

System Identification

8

Parameter and State Estimation

CONTENTS

9

Optimal and Robust State Estimation

CONTENTS

10

Adaptive State Estimation

CONTENTS

Part V

Expanded Research Areas

11

Fuzzy Control System

CONTENTS

Fuzzy logics provide a way of quantitatively interpreting and calculating attributes or reasonings described by vague languages. In contrast to classical control methods where variables are numbered precisely and control laws described clearly by mathematical equations, a fuzzy controller uses objective terms such as "good", "bad", "high", "low", "very", "a little" to describe variables, and use rules "if something then something" to determine the control signals. This makes a fuzzy controller naturally good at realizing human knowledge base into practice.

Different from ANN which is highly data driven and requires a lot of samples and computations to train a network, a fuzzy controller is an expert-based system that requires only the membership functions used for fuzzification of variables and a set of rules for inference. ANN may reveal the building bricks of a human brain, while fuzzy control system reflects how a human interprets the knowledge that he learns from past experience and uses in his daily life. This feature of a fuzzy control system is a bless and a curse at the same time. Fuzzy control system does not rely on training, thus is handy when there are few training samples. On the other hand, this also means that it cannot benefit from the accumulation of the samples and it is lack of evolving capability comparing with ANN and other evolutionary algorithms.

11.1 Fuzzy Set and Fuzzy Relation

Fuzzy set is not a set from mathematical perspective, but a projection derived from a set. Fuzzy set and fuzzy relation are the fundamentals of fuzzy logics. Details are given in the remaining of this section.

11.1.1 Fuzzy Set

The fundamentals of fuzzy logics and fuzzy control systems are built on fuzzy set and membership function. Fuzzy set is not a set from mathematical perspective, but a projection from a set to a real number between $[0, 1]$. The function used in the projection is called the member function of the fuzzy set.

Consider set U with finite or infinite cardinality. Define a projection $\mu_{\underset{\sim}{A}}$ from $x \in U$ to $[0, 1]$ as follows.

$$\mu_{\underset{\sim}{A}} \quad : \quad x \in U \rightarrow [0, 1]$$

This projection, together with its domain (also known as the universe of a fuzzy set) defines a fuzzy set. The projection $\mu_{\underset{\sim}{A}}$ is called the membership function of the fuzzy set $\underset{\sim}{A}$. The value of $\mu_{\underset{\sim}{A}}(x)$ for $x \in U$ describes in what extent x belongs to $\underset{\sim}{A}$.

It is clear from the definition the difference between a set and a fuzzy set. For a set, an element either belongs to the set or does not belong to the set, which can be described by

$$\mu_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

where μ_A is a binary function. In contrast, fuzzy set defines a continuous functions $\mu_{\underset{\sim}{A}}$ which can take any value between $[0, 1]$. From this stand, a fuzzy set is an extension to a set.

An example of a fuzzy set is given below. Let $U \geq 0$ be the age of a person. Define 2 fuzzy sets, "young adult" as $\underset{\sim}{A}$ and "middle-aged adult" as $\underset{\sim}{B}$, with membership function defined in Fig. 11.1.

From Fig. 11.1, we can see how a person is categorized according to the age. For example, a 5-year-old child is by no means a young adult or an adult, as $\mu_A = \mu_B(5) = 0$ from the figure. A 27-year-old can be considered "quite" a young adult, with $\mu_A = 0.8$, and "somehow" adult, with $\mu_A = 0.3$.

The maximum value of a membership function is defined as the height of the fuzzy set. In many occasions the height is 1, but it is not always the case. An element whose associated membership function equals to half of the height is called the crossover point.

It is possible to set up a threshold for the membership function, and derive

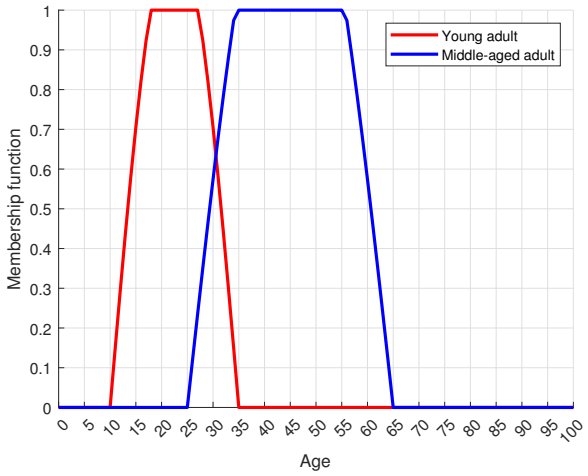

FIGURE 11.1
Membership function of fuzzy sets "young adult" and "middle-aged adult".

a set from a fuzzy set as follows.

$$\underset{\sim}{A}_{\alpha} = \left\{ x \middle| \mu_{\underset{\sim}{A}}(e) \geq \alpha \right\}$$

Notice that $\underset{\sim}{A}_{\alpha}$ is a set (not a fuzzy set) known as the cut set of $\underset{\sim}{A}$. A fuzzy set can have infinite number of cut sets by choosing different α. A fuzzy set can be derived from all its cut sets as follows.

$$\underset{\sim}{A} = \bigcup_{\alpha \in [0,1]} \alpha A_{\alpha}$$

where αA_{α} is a fuzzy set defined on A_{α} with, universe A_{α} and membership function value α. The $\cup$ calculates the union of fuzzy sets, more of which is introduced later.

Define the support and core of a fuzzy set as follows, respectively.

$$\begin{aligned} \text{supp}(\underset{\sim}{A}) &= \left\{ x \middle| \mu_{\underset{\sim}{A}}(x) > 0 \right\} \\ C(\underset{\sim}{A}) &= \left\{ x \middle| \mu_{\underset{\sim}{A}}(x) = 1 \right\} \end{aligned}$$

A fuzzy set $\underset{\sim}{A}$ can be represented by its base set A together with associated membership functions $\mu_{\underset{\sim}{A}}$. Alternatively, use the following

$$\begin{aligned} \underset{\sim}{A} &= \sum_i \frac{\mu_{\underset{\sim}{A}}(x_i)}{x_i} \\ \underset{\sim}{A} &= \int_E \frac{\mu_{\underset{\sim}{A}}(x)}{x} \end{aligned}$$

to represent a fuzzy set with discrete and continuous universe, respectively. Notice that in the above representation, "/", "$\sum$", and "$\int$" are not division, summation and integration, but just conventional notations used in the fuzzy set theory.

11.1.2 Fuzzy Set Operations

Operations such as union and intersection are defined on fuzzy sets. They are summarized in Table 11.1.

Operation	Symbol	Membership Function
Union	$\underset{\sim}{A} \cup \underset{\sim}{B}$	$\mu_{\underset{\sim}{A}\cup\underset{\sim}{B}}(e) = \max(\mu_{\underset{\sim}{A}}(e), \mu_{\underset{\sim}{B}}(e))$
Intersection	$\underset{\sim}{A} \cap \underset{\sim}{B}$	$\mu_{\underset{\sim}{A}\cap\underset{\sim}{B}}(e) = \min(\mu_{\underset{\sim}{A}}(e), \mu_{\underset{\sim}{B}}(e))$
Complement	$\underset{\sim}{A}'$	$\mu_{\underset{\sim}{A}'}(e) = 1 - \mu_{\underset{\sim}{A}}(e)$

TABLE 11.1
Commonly used fuzzy set operations.

Notice that Table 11.1 gives only one out of many ways to define these operations. There are other ways to define them. For example, for complement, there is Sugeno's complement and Yager's complement as follows. The ones given in Table 11.1 is the most commonly used definition.

$$\begin{aligned} N_s(e) &= \frac{1-e}{1+\lambda e} \\ N_w(e) &= (1-a^w)^{\frac{1}{w}} \end{aligned}$$

More operations can be found in [2].

11.1.3 Commonly Used Membership Functions

Membership functions can be chosen flexibly as long as it is bounded by $[0, 1]$ for each element. There are some commonly used ones that has been proved useful in general problems as shown in Fig. 11.2. They, together with their combinations, are widely used in fuzzy control system design.

MATLAB provides built-in functions to generate the functions in 11.2, such as `trimf` for triangular membership function, `trapmf` for trapezoidal function, `gaussmf` for Gaussian, and `gbellmf` for generalized-bell function, respectively. There are other membership functions built-in to MATLAB as well.

11.1.4 Fuzzy Relation

In set theory, a relation of two elements is defined using sets. For example, let $a, b \in \mathbb{R}$. Let a customized relation "$\oplus$" be defined on $\mathbb{R} \times \mathbb{R} = \mathbb{R}^2$. We can say

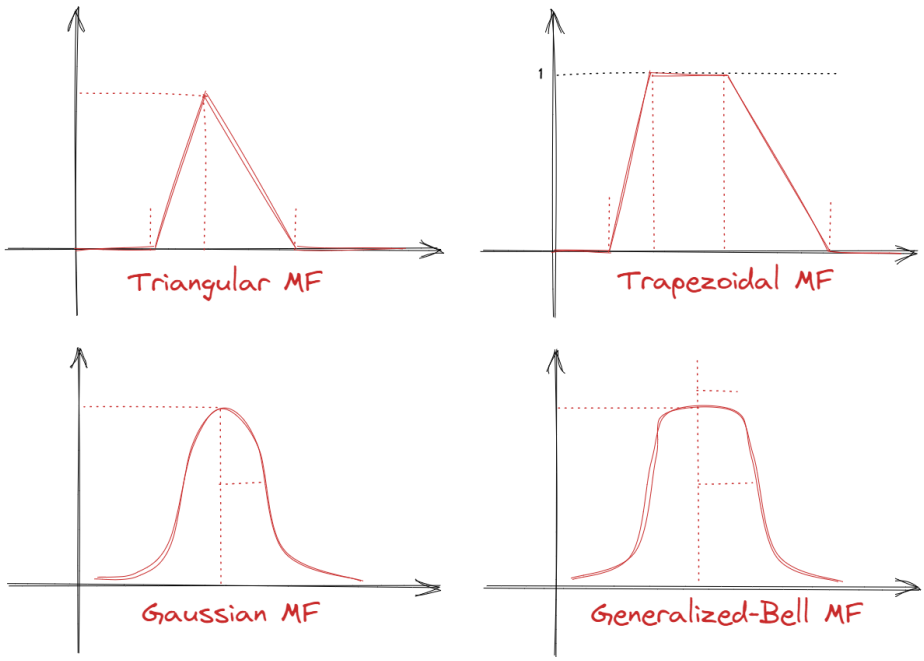

FIGURE 11.2
Commonly used membership functions.

$a \oplus b$ if and only if there is a set defined on $\mathbb{R}^2$ associated with "$\oplus$" relation, and (a, b) is in that set. Here "$\times$" denotes the Cartesian product.

Fuzzy relation is defined likewise, except that the set is replaced with fuzzy set as follows. Let $u \in U$, $v \in V$ be two variables with the same or different domains, and $D = U \times V$. The fuzzy relation of "$\oplus$" of two elements u and v is defined by the following fuzzy set

$$\mu_{\underset{\sim}{D}}(u, v) \quad \in \quad [0, 1]$$

associated with "$\oplus$".

It is possible to use a matrix to record the fuzzy relations, in which case the matrix is called a fuzzy matrix as it is made up of membership functions. Other choice include fuzzy graph, which takes advantage of graph theory to represent the fuzzy relation of different nodes in a universe.

The relation can be derived from chains. For example, consider relation P defined one $X \times Y$, and Q on $Y \times Z$. A relation on X, Z can be derived. One way (this is not the only way) is to use

$$\mu_{X \times Z}(x, z) \quad = \quad \sup \left[\min \left(\mu_{X \times Y}(x, y), \mu_{Y \times Z}(y, z) \right) \right]$$

11.2 Fuzzy Control System

Fuzzy control system is a practice of utilizing fuzzy logics in engineering. It mainly includes fuzzification of the physical measurement signals to fuzzy variables, fuzzy inference, and defuzzification of the fuzzy variables to physical control signals. Membership functions are used in the fuzzification and defuzzification of signals, and rules are used in the fuzzy inference.

A simple schema of a fuzzy control system is given in Fig. 11.3. Details are covered in the remaining of this section.

11.2.1 Fuzzification

xxx

11.2.2 Fuzzy Interface

xxx

11.2.3 Fuzzy Controller Design

xxx

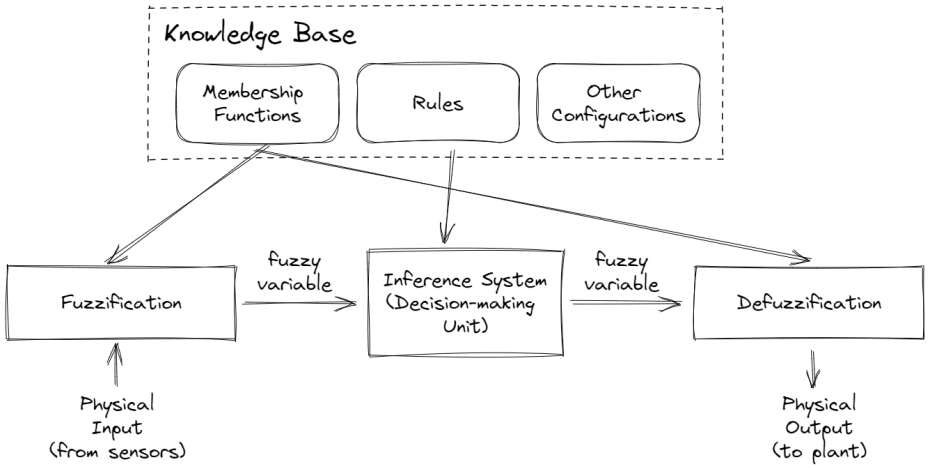

FIGURE 11.3
A simple schema for a fuzzy control system.

11.2.4 Fuzzy Controller Performance Analysis

xxx

11.3 Fuzzy Modeling and Fuzzy System Identification

xxx

11.4 Fuzzy System Auto-tuning

The performance of a fuzzy control system is mainly determined by the choice of membership functions, inference rules and defuzzification methods. In the case where there is no such an expert that can carefully design everything, a data-driven approach can be implemented that automatically tunes the parameters setup in a fuzzy control system for the optimal performance.

Notice that the auto-tuning does not change the fact that a fuzzy control system is a rule-based expert system. Therefore, the advantage of fuzzy system being intuitive to interpret persists.

Nowadays, the most popular data-driven approach for data analysis is arti-

ficial neural network. It is possible to use ANN to find the optimal parameters of a fuzzy system, hence, fuzzy neural networks.

11.5 Fuzzy Control System Design Using MATLAB

MATLAB fuzzy logic toolbox provides tools to define membership functions and fuzzy inference systems. They are introduced in this section via examples.

11.5.1 Example: Vibration Detection

Consider the example given below. Design and implement the system in MATLAB as follows.

Vibration Detection from Measurement Signal

Oscillation has been detected from a continuously sampled signal. The oscillation might be caused by system process dynamics, system vibration, or measurement noise. The oscillation frequency and amplitude together help to reveal the cause of the oscillation, specifically the likelihood that the oscillation is caused by system vibration.

The input to the system is the oscillating measurement signal sampled at $1Hz$. An FFT is applied to the signal to get the oscillation frequency and amplitude, which lay in the range of $[0, 0.5](Hz)$ and $[0, 10]$, respectively. The oscillation frequency and amplitude is then sent to the fuzzy controller.

In the fuzzy controller, the two signals are fuzzified to form fuzzy variables. The fuzzy variables are passed to the fuzzy inference system, where rules are defined to determine the likelihood that the oscillation is caused by vibration, which is also a fuzzy variable. Finally, the likelihood fuzzy variable is defuzzified to obtain the vibration likelihood, which should be a variable in $[0, 1]$.

Membership Functions for Input and Outputs

The first step in solving this problem is to understand how an human expert would interpret oscillation frequency and amplitude, and derive the likelihood from them. Take oscillation frequency as an example. It is agreed on that a frequency that is too low or too high may indicate something other than vibration. For example, frequency lower than $0.05Hz$ will be categorized as "low". Similarly, frequency higher than $0.4Hz$ is probably too "high" for vibration. Frequency around $0.2Hz$ is the "fine" frequency, which increases the likelihood of vibration.

Imagine asking many experts on how they categorize each frequency

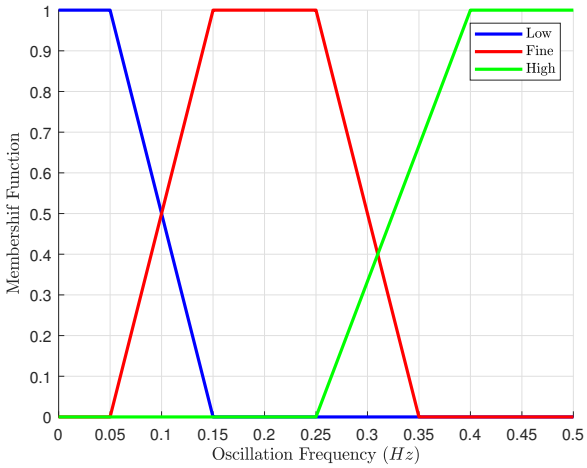

FIGURE 11.4
Membership functions for 3 fuzzy sets "low", "fine" and "high" defined on input signal oscillation frequency.

$0.01Hz, 0.02Hz, ..., 0.5Hz$ into one of the 3 categories "low", "fine" or "high". Consider using the interview results to form the membership for the 3 categories as given in Fig. 11.4. The curves in Fig. 11.4 is obtained by fitting the "percentage of agreement" using trapezoidal functions. Figure 11.4 can be interpreted as in what extend one would agree that the specified frequency shall belong to a category.

The same applies to the oscillation amplitude, where 2 categories are defined, namely "low" and "fine", as shown by Fig. 11.5. Instead of using trapezoidal functions, generalized bell functions are used to fit the interview results.

With the above defined membership functions, each frequency and amplitude can be translated into a fuzzy variable using fuzzification. For example, frequency $0.12Hz$ is $[0.3, 0.7, 0]$ where the three values in the vector represents the membership functions values of the given frequency mapping to each fuzzy set.

The output of the system is the likelihood of vibration of the system, scaled by a parameter in $[0, 1]$. Define 4 fuzzy sets, namely "very likely", "likely", "unlikely", "very unlikely", as shown in Fig. 11.6.

With the above defined membership functions, each likelihood fuzzy variable can be translated into the likelihood of the occurrence of vibration via defuzzification. For example, likelihood fuzzy variable $[0, 0.1, 0.3, 0.9]$ means that the the likelihood is not at all "very unlikely", a tiny little bit "unlikely", somewhat "likely", and quite much "very likely". Intuitively, this should translate into a high likelihood around $0.7 \sim 0.9$ using Fig. 11.6.

There are different ways of translating the output fuzzy variables into a

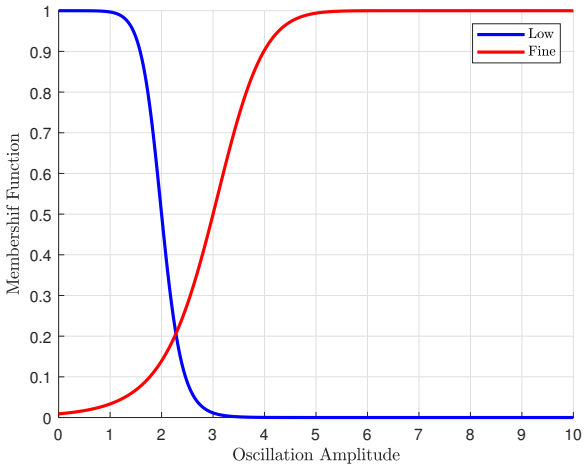

FIGURE 11.5
Membership functions for 2 fuzzy sets "low" and "fine" defined on input signal oscillation amplitude.

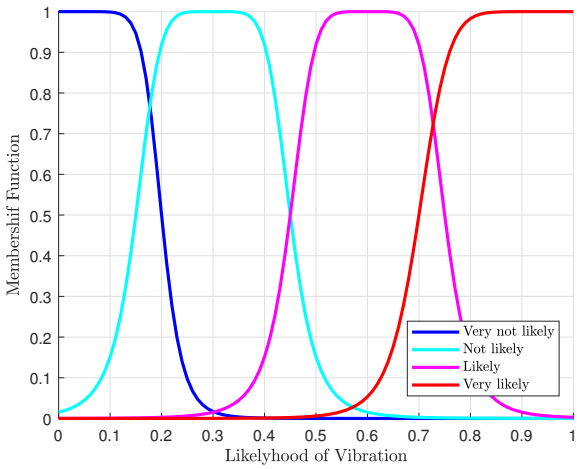

FIGURE 11.6
Membership functions for 4 fuzzy sets "very unlikely", "unlikely", "likely" and "very likely", defined on input signal oscillation amplitude.

physical signal, such as "centroid", "bisector", "middle of maximum", etc. More details can be found in MATLAB help document under "defuzzification methods". For example, $[0, 0.1, 0.3, 0.9]$ would translate into 0.73 using membership function given in Fig. 11.6 using centroid method, and 0.79 if using bisector method instead.

Inference

Fuzzy inference system is rule-based. Examples of the rules may include the following. Notice that in this example, all combinations of oscillation frequency and amplitude are listed. This is not always necessary in practice. Some of the rules may seem conflict with each other. This does not matter as the fuzzy controller will balance the weight of each output fuzzy set.

- If oscillation frequency is fine and oscillation amplitude is fine, likelihood of vibration is very likely.
- If oscillation frequency is fine and oscillation amplitude is low, likelihood of vibration is likely.
- If oscillation frequency is high and oscillation amplitude is fine, likelihood of vibration is likely.
- If oscillation frequency is high and oscillation amplitude is low, likelihood of vibration is unlikely.
- If oscillation frequency is low and oscillation amplitude is fine, likelihood of vibration is likely.
- If oscillation frequency is low and oscillation amplitude is low, likelihood of vibration is very unlikely.
- If oscillation amplitude is fine, likelihood of vibration is likely.

It is worth mentioning that "AND", "OR", "NOT" can be used in the propositions, as shown above, for example "frequency is fine AND amplitude is fine". The membership function of "frequency is fine AND amplitude is fine" can be derived from the two individual membership functions following rules defined in Table 11.1 or its alternatives.

The output fuzzy sets are the "pools" that collect scores. The input is checked against each rule, which add scores to its associated pools. The scores form the value output fuzzy variable.

Realization

With the above, it is possible to program the fizzy control system from scratch. Alternatively, fuzzy inference systems tools provided by MATLAB can also be used to quickly deploy a fuzzy control system. Both methods are investigated as follows.

Consider programming from scratch. The following MATLAB program

can be used to detect vibration. Notice that for simplicity, the membership functions and rules are hard-coded into the function.

```
fs = 1;                              % Sampling frequency
T = 1/fs;                            % Sampling period
L = 600;                             % Number of samples
t = (0:L-1)*T;                       % Time vector
measured_torque = 5*sin(2*pi*0.2*t);
measured_torque = measured_torque + 2*sin(2*pi*0.02*t);
measured_torque = measured_torque + normrnd(0, 1, [1, length(
    measured_torque)]);
detect_vibration(fs, measured_torque)

function vibration_likelihood = detect_vibration(fs, measured_torque)
       % fft analysis to get the oscillation frequency and amplitude
       T = 1/fs;
       L = length(measured_torque);
       t = (0:L-1)*T;
       figure
       plot(t, measured_torque, 'b-')
       grid on
       xlabel("Time", "Interpreter", "latex")
       ylabel("Torque", "Interpreter", "latex")
       measured_torque_fft = fft(measured_torque);
       P2 = abs(measured_torque_fft/L);
       P1 = P2(1:L/2+1);
       P1(2:end-1) = 2*P1(2:end-1);
       f = fs*(0:(L/2))/L;
       figure
       plot(f, P1)
       grid on
       title("Single-Sided Amplitude Spectrum of Measured Torque")
       xlabel("$f$ ($Hz$)", "Interpreter", "latex")
       ylabel("$|P_1(f)|$", "Interpreter", "latex")
       maj_f = f(P1 == max(P1));
       maj_P1 = max(P1);
       % fuzzification
       x_freq = 0:0.01:0.5;
       mf_x_freq = [];
       mf_x_freq.low = trapmf(x_freq, [-1, 0, 0.05, 0.15]);
       mf_x_freq.fine = trapmf(x_freq, [0.05, 0.15, 0.25, 0.35]);
       mf_x_freq.high = trapmf(x_freq, [0.25, 0.40, 0.5, 1]);
       [~, x_freq_ind] = min(abs(maj_f-x_freq));
       x_freq_fuzzy = [mf_x_freq.low(x_freq_ind), mf_x_freq.fine(
           x_freq_ind), mf_x_freq.high(x_freq_ind)];
       x_amp = 0:0.01:10;
       mf_x_amp = [];
       mf_x_amp.low = gbellmf(x_amp, [4, 10, -2]);
       mf_x_amp.fine = gbellmf(x_amp, [5, 5, 8]);
       [~, x_amp_ind] = min(abs(maj_P1-x_amp));
```

```
        x_amp_fuzzy = [mf_x_amp.low(x_amp_ind), mf_x_amp.fine(x_amp_ind)
            ];
        % inference
        y_likelihood_fuzzy = [0,0,0,0]; % very unlikely, unlikely,
            likely, very likely
        y_likelihood_fuzzy(4) = max(y_likelihood_fuzzy(4), min(
            x_freq_fuzzy(2), x_amp_fuzzy(2)));
        y_likelihood_fuzzy(3) = max(y_likelihood_fuzzy(3), min(
            x_freq_fuzzy(2), x_amp_fuzzy(1)));
        y_likelihood_fuzzy(3) = max(y_likelihood_fuzzy(3), min(
            x_freq_fuzzy(3), x_amp_fuzzy(2)));
        y_likelihood_fuzzy(3) = max(y_likelihood_fuzzy(3), min(
            x_freq_fuzzy(1), x_amp_fuzzy(2)));
        y_likelihood_fuzzy(2) = max(y_likelihood_fuzzy(2), min(
            x_freq_fuzzy(3), x_amp_fuzzy(1)));
        y_likelihood_fuzzy(1) = max(y_likelihood_fuzzy(2), min(
            x_freq_fuzzy(1), x_amp_fuzzy(1)));
        % defizzification
        x_likelihood = 0:0.01:1;
        mf_x_likelihood = [];
        mf_x_likelihood.verynotlikely = gbellmf(x_likelihood, [0.2, 5,
            0]);
        mf_x_likelihood.notlikely = gbellmf(x_likelihood, [0.15, 3,
            0.3]);
        mf_x_likelihood.likely = gbellmf(x_likelihood, [0.15, 3, 0.6]);
        mf_x_likelihood.verylikely = gbellmf(x_likelihood, [0.3, 5, 1]);
        mf = max(min(y_likelihood_fuzzy(1), mf_x_likelihood.
            verynotlikely), ...
               max(min(y_likelihood_fuzzy(2), mf_x_likelihood.notlikely
                   ), ...
               max(min(y_likelihood_fuzzy(3), mf_x_likelihood.likely),
                   ...
               min(y_likelihood_fuzzy(4), mf_x_likelihood.verylikely)))
                   );
        vibration_likelihood = defuzz(x_likelihood , mf, 'mom');
end
```

Running the above code gives the following results in Figs. 11.7 and 11.8. The function returns a likelihood of 0.84.

If increase the frequency from $0.2Hz$ to $0.45Hz$, i.e., let

```
measured_torque = 5*sin(2*pi*0.4*t);
measured_torque = measured_torque + 2*sin(2*pi*0.02*t);
measured_torque = measured_torque + normrnd(0, 1, [1, length(
    measured_torque)]);
```

the likelihood becomes 0.60. If further drop the amplitude and let

```
measured_torque = 2*sin(2*pi*0.4*t);
measured_torque = measured_torque + 1*sin(2*pi*0.02*t);
```

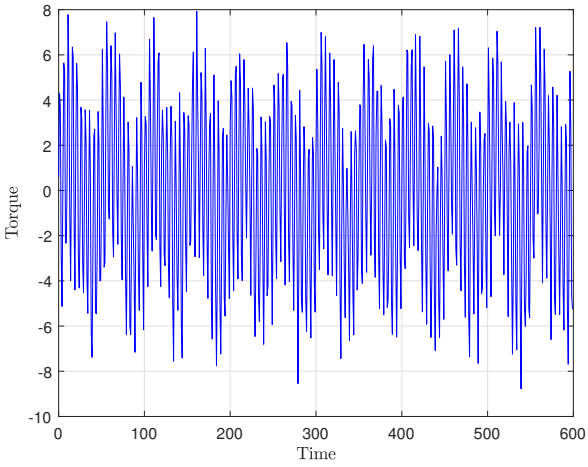

FIGURE 11.7
Measured torque.

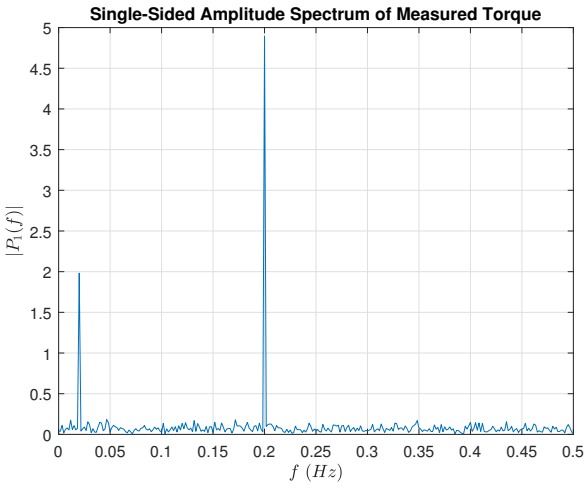

FIGURE 11.8
Measured torque single-sided amplitude spectrum.

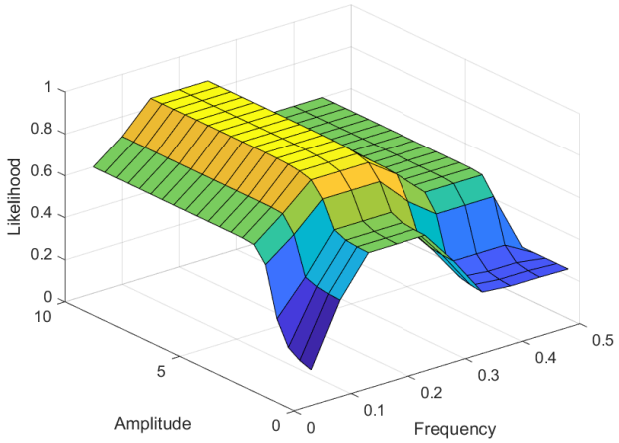

FIGURE 11.9
Likelihood of vibration as a function of oscillation frequency and amplitude.

```
measured_torque = measured_torque + normrnd(0, 1, [1, length(
    measured_torque)]);
```

the likelihood further drops to 0.30. By an exhaustive search, we can plot likelihood as a function of oscillation frequency and amplitude as given in Fig. XXX.

Consider programming using MATLAB built-in tools to reproduce the above results. MATLAB provides variety choice of fuzzy inference systems. In this example, `mamfis` is used as follows.

```
vd_fis = mamfis( ...
       'NumInputs', 2, ...
       'NumInputMFs', [3, 2], ...
       'NumOutputs', 1, ...
       'NumOutputMFs', 4, ...
       'AddRule', 'none' ...
       );
% ios
vd_fis.Inputs(1).Name = 'torque_frequency';
vd_fis.Inputs(1).Range = [0, 0.5];
vd_fis.Inputs(1).MembershipFunctions(1).Name = 'low';
vd_fis.Inputs(1).MembershipFunctions(1).Type = 'trapmf';
vd_fis.Inputs(1).MembershipFunctions(1).Parameters = [-1, 0, 0.05,
    0.15];
vd_fis.Inputs(1).MembershipFunctions(2).Name = 'fine';
vd_fis.Inputs(1).MembershipFunctions(2).Type = 'trapmf';
vd_fis.Inputs(1).MembershipFunctions(2).Parameters = [0.05, 0.15, 0.25,
     0.35];
```

```
vd_fis.Inputs(1).MembershipFunctions(3).Name = 'high';
vd_fis.Inputs(1).MembershipFunctions(3).Type = 'trapmf';
vd_fis.Inputs(1).MembershipFunctions(3).Parameters = [0.25, 0.40, 0.5,
    1];
vd_fis.Inputs(2).Name = 'torque_amplitude';
vd_fis.Inputs(2).Range = [0, 10];
vd_fis.Inputs(2).MembershipFunctions(1).Name = 'low';
vd_fis.Inputs(2).MembershipFunctions(1).Type = 'gbellmf';
vd_fis.Inputs(2).MembershipFunctions(1).Parameters = [4, 10, -2];
vd_fis.Inputs(2).MembershipFunctions(2).Name = 'fine';
vd_fis.Inputs(2).MembershipFunctions(2).Type = 'gbellmf';
vd_fis.Inputs(2).MembershipFunctions(2).Parameters = [5, 5, 8];
plotmf(vd_fis, 'input', 1, 1000)
vd_fis.Outputs(1).Name = 'vibration_likelihood';
vd_fis.Outputs(1).Range = [0, 1];
vd_fis.Outputs(1).MembershipFunctions(1).Name = 'very_unlikely';
vd_fis.Outputs(1).MembershipFunctions(1).Type = 'gbellmf';
vd_fis.Outputs(1).MembershipFunctions(1).Parameters = [0.2, 5, 0];
vd_fis.Outputs(1).MembershipFunctions(2).Name = 'unlikely';
vd_fis.Outputs(1).MembershipFunctions(2).Type = 'gbellmf';
vd_fis.Outputs(1).MembershipFunctions(2).Parameters = [0.15, 3, 0.3];
vd_fis.Outputs(1).MembershipFunctions(3).Name = 'likely';
vd_fis.Outputs(1).MembershipFunctions(3).Type = 'gbellmf';
vd_fis.Outputs(1).MembershipFunctions(3).Parameters = [0.15, 3, 0.6];
vd_fis.Outputs(1).MembershipFunctions(4).Name = 'very_likely';
vd_fis.Outputs(1).MembershipFunctions(4).Type = 'gbellmf';
vd_fis.Outputs(1).MembershipFunctions(4).Parameters = [0.3, 5, 1];
plotmf(vd_fis, 'output', 1, 1000)
% rules
rules = ["if torque_frequency is fine and torque_amplitude is fine then
     vibration_likelihood is very_likely"; ...
       "if torque_frequency is fine and torque_amplitude is low then
           vibration_likelihood is likely"; ...
       "if torque_frequency is high and torque_amplitude is fine then
           vibration_likelihood is likely"; ...
       "if torque_frequency is high and torque_amplitude is low then
           vibration_likelihood is unlikely"; ...
       "if torque_frequency is low and torque_amplitude is fine then
           vibration_likelihood is likely"; ...
       "if torque_frequency is low and torque_amplitude is low then
           vibration_likelihood is very_unlikely"; ...
       "if torque_amplitude is fine then vibration_likelihood is likely
           " ...
       ];
vd_fis = addRule(vd_fis, rules);
```

Use

```
evalfis(vd_fis, [<frequency>, <amplitude>])
```

to get the output of the fuzzy controller. Use

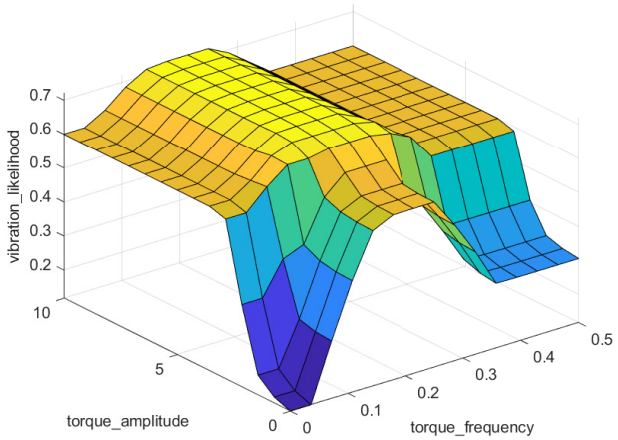

FIGURE 11.10
Likelihood of vibration as a function of oscillation frequency and amplitude using `mamfis`.

```
gensurf(vd_fis)
```

to obtain the plot of likelihood as a function of the oscillation frequency and amplitude as given in Fig. 11.10, which is similar with Fig. 11.9.

12

Multi-Agent Control System

CONTENTS

A

Matrix Algebra

CONTENTS

This appendix chapter reviews matrix algebra, one of the most fundamental and widely used mathematical tools in this notebook. Notice that this chapter only serves as a brief summary, and cannot replace the linear algebra textbook or notebook.

A.1 Basic Operators

Let A, B be two square matrices of the same size. The following holds true

$$\begin{aligned} |A| &= |A^T| \\ |AB| &= |A||B| \end{aligned}$$

where $|\cdot|$ calculates the determinant of a square matrix. Furthermore, if both A, B are nonsingular,

$$(AB)^{-1} = B^{-1}A^{-1}$$

Let $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{n\times q}$. The following holds true

$$(AB)^T = B^T A^T$$

Furthermore, if $p = q$,

$$\text{trace}(AB) = \text{trace}(BA)$$

The *Kronecker product* of two matrix $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$ is defined as follows.

$$A \otimes B = [a_{ij}B] \in \mathbb{R}^{mp\times nq}$$

where a_{ij} is an element in A. Notice that some literature defines it as $[Ab_{ij}]$ instead, which is different from the earlier definition.

The stacking operator is defined as follows. Let $A \in \mathbb{R}^{m \times n}$,

$$s(A) = \begin{bmatrix} a_{1,1} \\ \vdots \\ a_{m,1} \\ \vdots \\ a_{1,n} \\ \vdots \\ a_{m,n} \end{bmatrix}$$

i.e., it stacks the columns of A into a column vector.

The following holds true

$$s(ABC) = s(C^T \otimes A)s(B)$$

Let $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times q}$,

$$|A \otimes B| = |A|^p |B|^m$$

If λ_i is an eigenvalue of A with eigenvector v_i, then λ_i^{-1} is an eigenvalue of A^{-1} with the same eigenvector. This is because

$$Av_i = \lambda_i v_i$$

implies that

$$\lambda^{-1} v_i = A^{-1} v_i$$

For another matrix B with eigenvalue μ_i with respect to eigenvector w_i, $\lambda_i \mu_i$ is an eigenvalue of $A \otimes B$ w.r.t eigenvector $v_i \otimes w_i$.

The well-known matrix inversion lemma is

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1}$$

whose proof can be obtained using blockwise matrix inversion, which is neglected here.

A.2 Partitioned Matrix

The following matrix A is *block diagonal* if

$$A = \begin{bmatrix} \ddots & & 0 \\ & A_{ii} & \\ 0 & & \ddots \end{bmatrix}$$

In this case, A can be denoted by $A = \text{diag}(\ldots, A_{ii}, \ldots)$. Furthermore, if A_{ii} is square matrix, A becomes square matrix, and

$$|A| = \Pi_i |A_{ii}|$$

If A is nonsingular,

$$A^{-1} = \text{diag}(\ldots, A_{ii}^{-1}, \ldots)$$

A.3 Quadratic Forms

Let $x \in \mathbb{R}^n$ a (column) vector. The square of the Euclidean norm is

$$||x||^2 = x^T x$$

Let S be a nonsingular transformation, and $P = S^T S$ a symmetric matrix,

$$\begin{aligned} ||x||_P^2 &= ||Sx||^2 \\ &= x^T S^T S x \\ &= x P x^T \end{aligned}$$

The quadratic form of x looks like the following

$$x^T Q x \tag{A.1}$$

where Q is real. Notice that Q can be divided into symmetric and asymmetric parts

$$Q = Q_s + Q_a$$

where

$$\begin{aligned} Q_s &= (Q + Q^T)/2 \\ Q_a &= (Q - Q^T)/2 \end{aligned} \tag{A.2}$$

and (A.1) becomes

$$\begin{aligned} x^T Q x &= x^T (Q_s + Q_a) x \\ &= x^T Q_s x \end{aligned}$$

where notice that $x^T Q_a x = 0$. This is because $x^T Q_a x = x^T Q_a^T x = -x^T Q_a x$ since $x^T Q_a x$ is a scalar and from (A.2) $Q_a^T = -Q_a$.

In (A.1), we say $Q > 0$ is positive definite if for all nonzero x, $x^T Q x > 0$. This is equivalent of saying that for all eigenvalues λ_i of Q, $\lambda_i > 0$. Semi-positive definite, semi-negative definite, and negative definite can be defined similarly.

A.4 Matrix Calculus

Matrix calculus defines how the differential of a vector/matrix w.r.t another vector/matrix is defined.

Consider s being a scalar and $x \in \mathbb{R}^n$ a factor. Let x be a vector function of s. The differential of x w.r.t. s is defined as follows.

$$\frac{dx}{ds} = \begin{bmatrix} \frac{dx_1}{ds} \\ \vdots \\ \frac{dx_n}{ds} \end{bmatrix}$$

If s is a function of x, the gradient of s w.r.t. x has two forms, the gradient form where the result is a column vector, and the Jacobian form where the result is a row vector. They are transpose of each other. For example, the gradient form is

$$\frac{\partial s}{\partial x} = \nabla s = \begin{bmatrix} \frac{\partial s}{\partial x_1} \\ \vdots \\ \frac{\partial s}{\partial x_n} \end{bmatrix} \tag{A.3}$$

and

$$ds = (\nabla s)^T dx$$

where $dx = [dx_1, \ldots, dx_n]^T$. The Jacobian form denoted by $J(s)$ is defined as the transpose of (A.3). Furthermore, if s is a function of multiple vectors, for example both x and y,

$$ds = \left(\frac{\partial s}{\partial x}\right)^T dx + \left(\frac{\partial s}{\partial y}\right)^T dy$$

The *Hessian* of a scalar s w.r.t. a vector x is the second derivative

$$\frac{\partial^2 s}{\partial x^2} = \left[\frac{\partial^2 s}{\partial x_i \partial x_j}\right]$$

Let $f \in \mathbb{R}^m$ be a vector function of vector $x \in \mathbb{R}^n$. The jacobian of f w.r.t. x is a $m \times n$ matrix given by

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Let A be a non-singular square matrix of function of time t.

$$\frac{d}{dt}\left(A^{-1}\right) = -A^{-1}\frac{d}{dt}\left(A\right)A^{-1}$$

Notice that in references, the differential of a variable x or a matrix A w.r.t. time can also be denoted as $\dot{x}$ and $\dot{A}$, respectively.

Let y, A be constant vector and matrix, respectively, with dimensions so that the following expressions make sense. Then we have

$$\begin{aligned}
&\frac{\partial}{\partial x}\left(y^T x\right) = \frac{\partial}{\partial x}\left(x^T y\right) = y \\
&\frac{\partial}{\partial x}\left(Ax\right) = A \\
&\frac{\partial}{\partial x}\left(y^T Ax\right) = \frac{\partial}{\partial x}\left(x^T A^T y\right) = A^T y \\
&\frac{\partial}{\partial x}\left(y^T f(x)\right) = \frac{\partial}{\partial x}\left(f(x)^T y\right) = \left(\frac{\partial f}{\partial x}\right)^T y \\
&\frac{\partial}{\partial x}\left(x^T Ax\right) = Ax + A^T x \\
&\frac{\partial^2}{\partial x^2}\left(x^T Ax\right) = A + A^T
\end{aligned}$$

In a special case where Q is symmetric,

$$\begin{aligned}
\frac{\partial}{\partial x}\left(x^T Qx\right) &= 2Qx \\
\frac{\partial^2}{\partial x^2}\left(x^T Qx\right) &= 2Q \\
\frac{\partial}{\partial x}\left((x-y)^T Q(x-y)\right) &= 2Q(x-y) \\
\frac{\partial^2}{\partial x^2}\left((x-y)^T Q(x-y)\right) &= 2Q
\end{aligned}$$

Furthermore, if f, g are two vector functions of vector x,

$$\frac{\partial}{\partial x}\left(f^T g\right) = \left(\frac{\partial}{\partial x} f\right)^T g + \left(\frac{\partial}{\partial x} g\right)^T f$$

B

Brief Introduction to Calculus of Variations

CONTENTS

Calculus of variations studies the change of the value of a functional w.r.t the choice of function. A commonly seen objective of calculus of variations is to find the optimal function among a set of functions (usually defined by restrictions) that maximizes or minimizes a functional.

Notice that under the scope of this notebook, only a few ideas from calculus of variations are needed. Therefore, only a brief introduction is introduced. More details about calculus of variations can be found in another notebook *A Notebook on Calculus*.

B.1 Problem Formulation

Consider a cost function defined by

$$J = \int_0^T h\left(x(t), t\right) dt \tag{B.1}$$

where $x(t)$ is a function of t. The objective is to find $x(t)$ in the searching space that minimizes the above cost function.

B.2 Brief Discussion of the Solution

Similar with static optimization, the solution to (B.1) can be obtained by studying how the change of each element in the cost function would affect the change of the value of the cost function, i.e., how $dx(t)$ and dt affect dJ.

There are some obvious differences of (B.1) comparing with discrete time

optimization such as the following

$$J = \sum_{k=0}^{N-1} h\left(x(k), u(k)\right) \tag{B.2}$$

The major differences are

1. In (B.1) integration is used, while in (B.2) sum is used.
2. In (B.1), the two variables $x(t)$ and t are correlated, while in (B.2), $x(t)$ and $u(t)$ can be treated as independent variables, with the system dynamics $x(k+1) = f\left(x(k) + u(k)\right)$ the equality constraint.

Integration and sum are in the essence describing the same thing, where integration a limiting case of sum. Therefore, this first difference does not affect much how we would proceed with the calculation. It is the second difference that forms the main challenge.

Consider $x(t)$ being a function of t defined from $t = 0$ to $t = T$. Consider a neighborhood function $x(t) + dx(t)$. Notice that the neighborhood function $x(t) + dx(t)$ is not necessarily cut-off at $t = T$. Without loosing generality, assume that the neighborhood function is stretched from $t = T$ to $t = T + dT$. The final value of the original and the neighborhood functions can be bridged as follows.

$$\left[x(t) + dx(t)\right]\big|_{t=T+dT} - x(T) = \delta x(T) + \dot{x}(T)dT$$

where $\delta x(T)$ describes a portion of the final value increment of the neighborhood function irrelevant to change in time, and $\dot{x}(T)dT$ the portion caused due to the change in time. Notice that since $x(t)$ and $x(t) + dx(t)$ are close enough, their slope at $t = T$ shall be approximately the same. This is demonstrated by Fig. B.1.

Another relation that describes dJ in (B.1) is given below. This is known as the Leibniz's rule for functionals.

$$dJ = h\left(x(T), T\right) dT - h\left(x(0), 0\right) dt_0 + \int_{t=0}^{T} \left(h_x^T\left(x(t), t\right) \delta x(t)\right) dt$$

where dT and dt_0 describe the stretch of the neighborhood function at the end and the beginning w.r.t. t, respectively, and

$$h_x = \frac{\partial h}{\partial x}$$

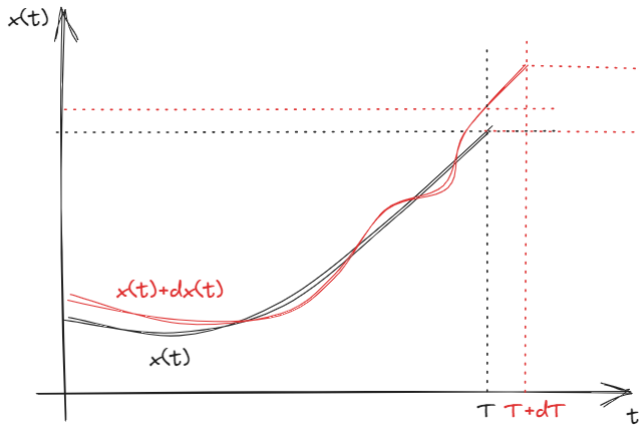

FIGURE B.1
A demonstration of the calculation of $dx(T)$ from calculus of variations perspective.

Bibliography

[1] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[2] Masaharu Mizumoto and Kokichi Tanaka. Fuzzy sets and their operations. *Information and Control*, 48(1):30–48, 1981.

[3] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.