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 or 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 around the end of August when the results of Gaokao (National College Entrance Examination of China, annually held in July) are released, people from photocopy shops would start selling notebooks photocopies that they claim 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 understand than the textbooks. I guess we cannot blame the top scorers for being so smart that they sometimes make 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 of the coming year would probably be a schoolmate or a classmate. Why would I want to pay that much money to a complete stranger in a photocopy shop for my friend's notebook, rather than requesting a copy from him or her directly?

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

This motivated me to start reading and taking notebooks for selected books and articles, just to force myself to spent time learning new subjects out of my comfort zone. I used to take 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 on my hand dozens of these notebooks. Eventually, one day it suddenly came to me: why not digitalize them, and make them accessible online and open-source, and let everyone read and edit it?

—

As most of the open-source software, this notebook (and it applies to the other notebooks in this series as well) does not come with any "warranty" of any kind, meaning that there is no guarantee for the statement and knowledge in this notebook to be absolutely correct as it is not peer reviewed. **Do NOT cite this notebook in your academic research paper or book!** Of course, if you find anything helpful with your research, please trace back to the origin of the citation and double confirm it yourself, then on top of that determine whether or not to use it in your research.

This notebook is suitable as:

- a quick reference guide;
- a brief introduction for beginners of the module;
- a "cheat sheet" for students to prepare for the exam (Don't bring it to the exam unless it is allowed by your lecturer!) or for lecturers to prepare the teaching materials.

This notebook is NOT suitable as:

- a direct research reference;
- a replacement to the textbook;

because as explained the notebook is NOT peer reviewed and it is meant to be simple and easy to read. It is not necessary brief, but all the tedious explanation and derivation, if any, shall be "fold into appendix" and a reader can easily skip those things without any interruption to the reading experience.

—

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.

Some of the figures in this notebook is drawn using Excalidraw, a very interesting tool for machine to emulate hand-writing. The Excalidraw project can be found in GitHub, *excalidraw/excalidraw*.

Preface

Control System

List of Figures

List of Tables

Part I

Classic Control System

1

Classic Control System

CONTENTS

2

Multivariable Control System

CONTENTS

Part II

Modern Control System

3

Modern Control System

CONTENTS

4

Computer Controlled System

CONTENTS

Part III

Advanced Control System

5

Optimal and Robust Control System

CONTENTS

6

Model Predictive Control System

CONTENTS

Model predictive controller belongs to the family of optimal controllers. Given the plant model and a control signal, the behavior of the system can be foreseen. MPC finds the control signal with the optimal forecast.

Comparing with other optimal controllers such as LQG, MPC has a better adaption to different kinds of restrictions. This makes MPC extremely popular in industry. From this perspective, many optimal controllers can be taken as special cases of MPC.

The references of this chapter include:

- Rawlings, J.B., Mayne, D.Q. and Diehl, M., 2017. Model predictive control: theory, computation, and design (Vol. 2). Madison, WI: Nob Hill Publishing. [3].

6.1 Introduction

The MPC uses system dynamic model to forecast system behavior, and optimize the forecast to produce the best decision at the current moment.

From the above description, we can see that the there are at least 3 key factors of MPC:

- Plant model and constraints, to be derived from the plant.
- Plant model parameters, to be obtained via parameter estimation.
- Plant current state, to be obtained via state estimation.

- Optimal control, to be obtained by defining the performance index and solving the functional.

where the model derived from assumptions, its parameters obtained by parameter estimation. Plant current state is obtained by state estimation. Finally, an optimization problem with varieties of restrictions is proposed to generate the control signal.

6.1.1 Models and Constraints

The model in an MPC plays the most important role. Many types of models are used to describe the plant in the MPC scope. Depending on the plant, the model can be either in continuous time domain or in discrete time domain; either state-space or input-output; either centralized or discrete (discrete models are used when the behavior of the system is not spatially uniform); and either deterministic or stochastic. Different mathematics tools are used to describe and solve MPC problems of different types of models.

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. It describes physical limits of the system. If the controller does not respect these constraints, the physics enforce them. An example of a physical constraint is the maximum input power to a motor.

State vector $x(k)$ and output $y(k)$ related constraints, on the other hand, are often performance constraints. They describe the desirable performance of the system, and they may or may not be achievable in practice. MPC is able to find out whether the performance constraints are achievable or not. Should the performance constraints not be achievable, they need to be modified and downgraded.

Input signal related constraints are often given in the following format.

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

State vector related constraints are often given in the following format.

$$Fx(k) \quad \leq \quad f$$

When augmented state vector is used, it is possible to combine the input and the state vector together to form constraints with more flexibility. For example, let

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

Then the following constraint

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

essentially translates to

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

which can become handy sometimes.

It is also possible to limit the state vector and/or the input to be integers or discrete values, which is not commonly seen in other optimal controllers.

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.

$$x(k+1) = Ax(k) + Bu(k) \tag{6.1}$$
$$y(k) = x(k) \tag{6.2}$$

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.

$$V\left(x(0),u\right) \quad = \quad \frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^T Q x(k)+u(k)^T R u(k)\right)+\frac{1}{2}x(N)^T P_f x(N) \tag{6.3}$$

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 (6.3) 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.

$$x^*, y^* \quad = \quad \arg\min_{x,y} f(x,y) \tag{6.4}$$

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 (6.4) using the following approach. Think of (6.4) as a two-stage optimization problem. Given any constant value of y, there is an associated optimized $x^*(y)$ which minimizes (6.4). 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, (6.4) 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 (6.3) is essentially something similar with (6.4). 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 (6.1). 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 (6.3) 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.

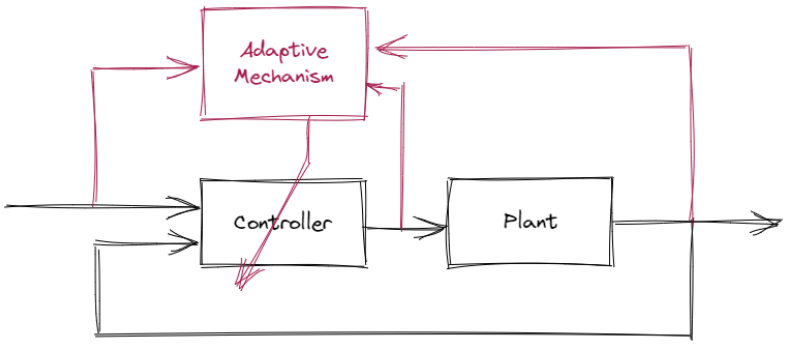

FIGURE 7.1
Adaptive control system general schema.

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)

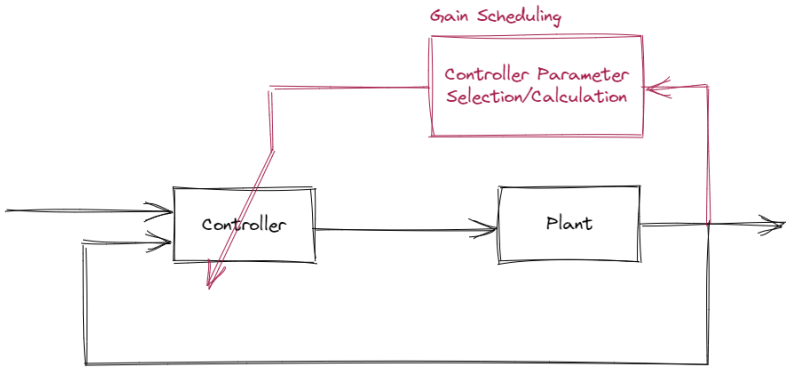

FIGURE 7.2
Gain scheduling schema.

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.

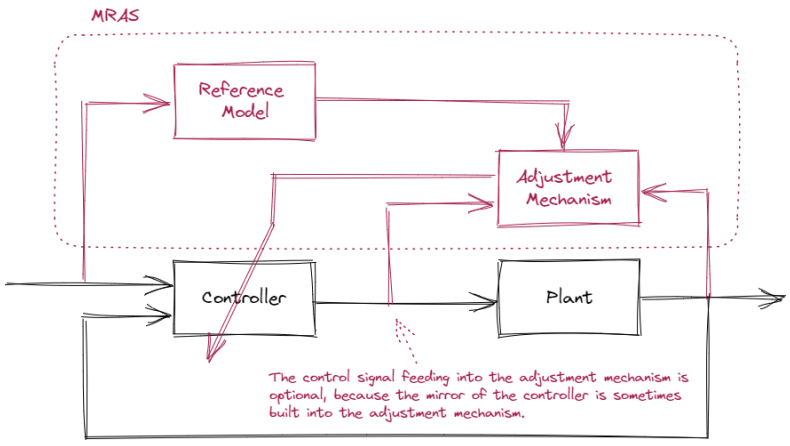

FIGURE 7.3
MARS schema.

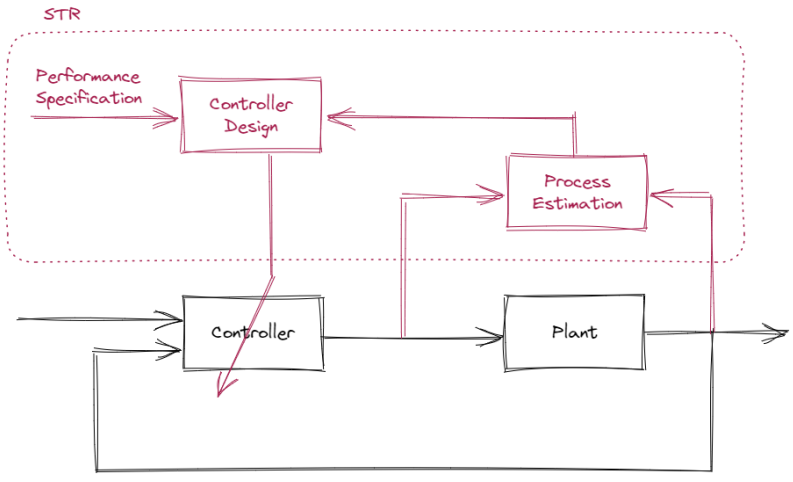

FIGURE 7.4
STR schema.

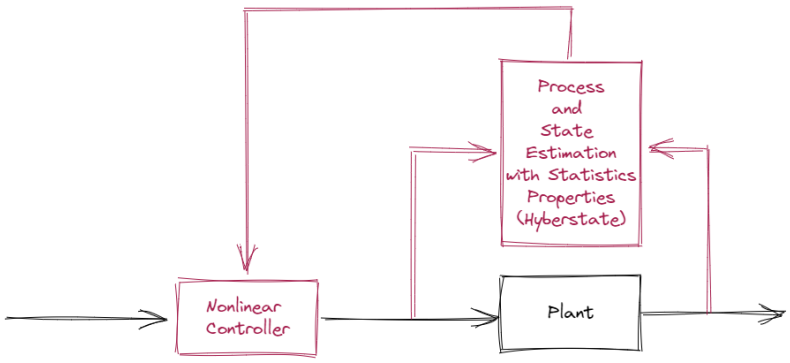

FIGURE 7.5
Dual control schema.

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.

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

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.

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

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.

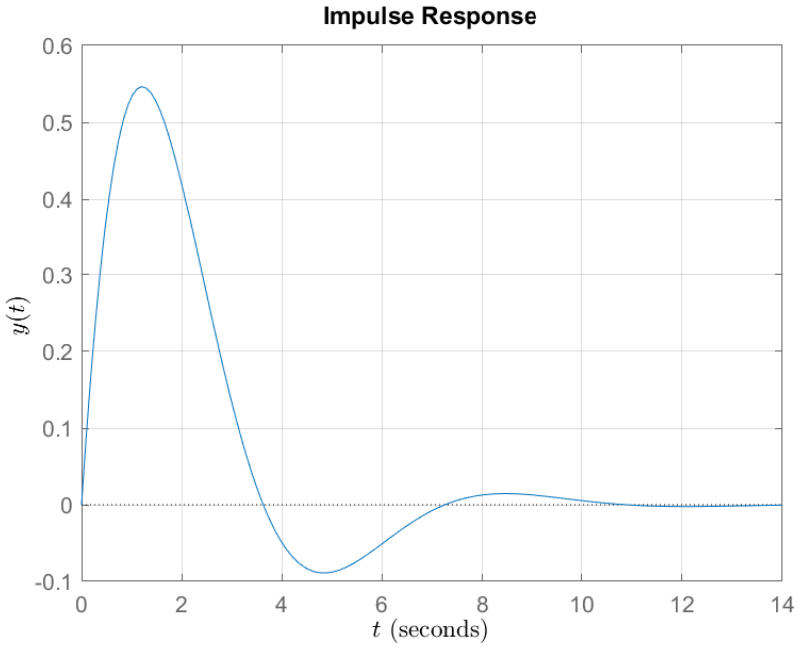

FIGURE 7.6
LTI system impulse response demonstration.

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

Classic System Identification and State Estimation

CONTENTS

9

Optimal and Robust State Estimation

CONTENTS

10

Adaptive State Estimation

CONTENTS

Part V

Artificial Intelligence

11

Artificial Neural Network

CONTENTS

Part VI

Advanced Topics

12

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.

12.1 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.

12.1.1 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. 12.1.

From Fig. 12.1, 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

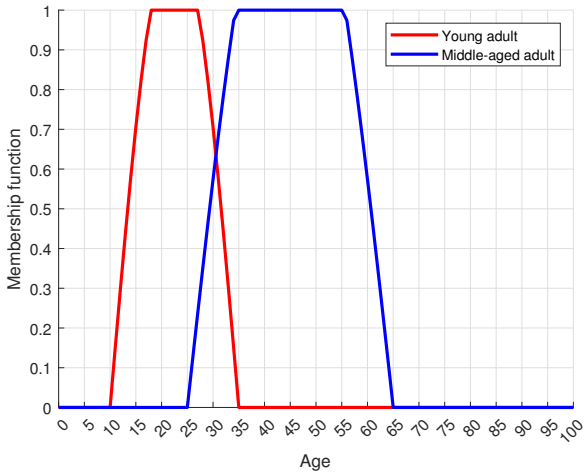

FIGURE 12.1
Membership function of fuzzy sets "young adult" and "middle-aged adult".

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.

12.1.2 Fuzzy Set Operations

Operations such as union and intersection are defined on fuzzy sets. They are summarized in Table 12.1.

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 12.1
Commonly used fuzzy set operations.

Notice that Table 12.1 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 12.1 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].

12.1.3 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. 12.2. They, together with their combinations, are widely used in fuzzy control system design.

MATLAB provides built-in functions to generate the functions in 12.2, 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.

12.1.4 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

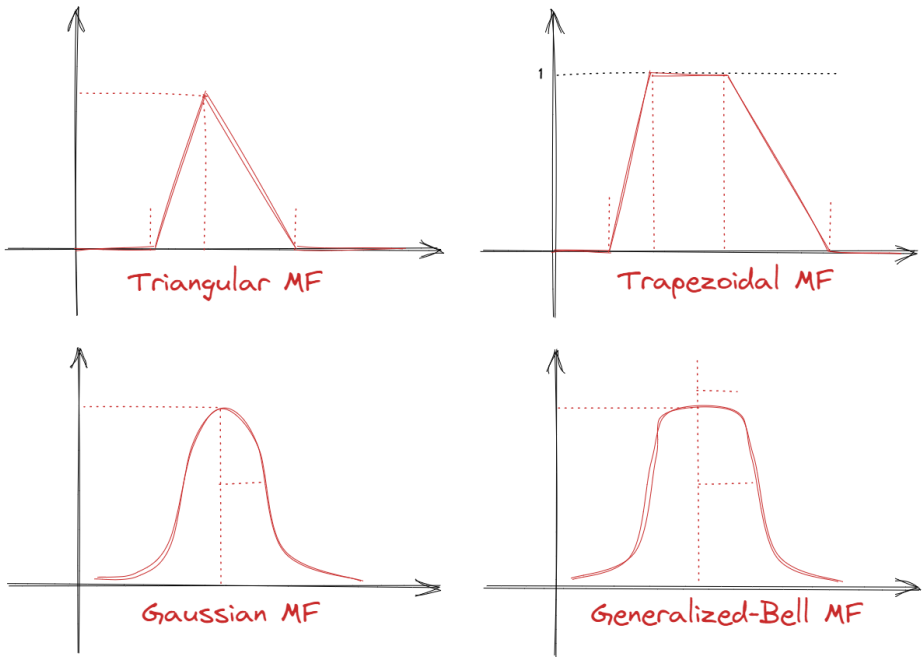

FIGURE 12.2
Commonly used membership functions.

$a \oplus b$ 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

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

12.2 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. 12.3. Details are covered in the remaining of this section.

12.2.1 Fuzzification

xxx

12.2.2 Fuzzy Interface

xxx

12.2.3 Fuzzy Controller Design

xxx

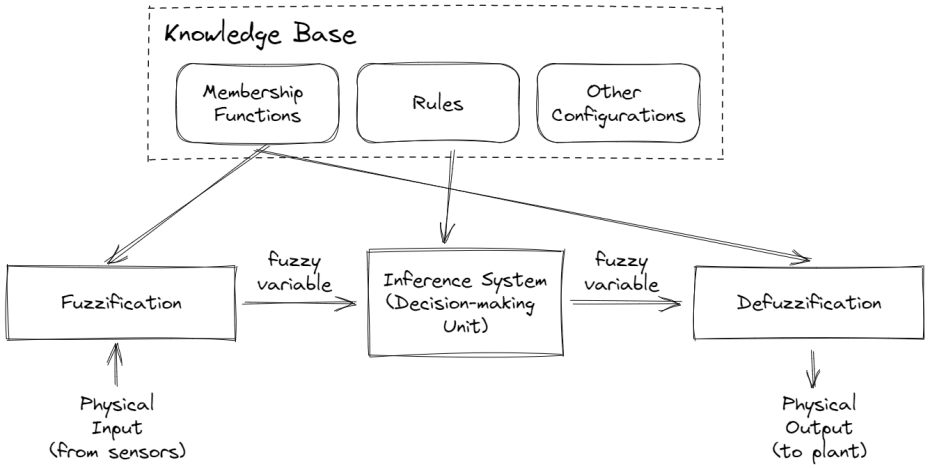

FIGURE 12.3
A simple schema for a fuzzy control system.

12.2.4 Fuzzy Controller Performance Analysis

xxx

12.3 Fuzzy Modeling and Fuzzy System Identification

xxx

12.4 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.

12.5 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.

12.5.1 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

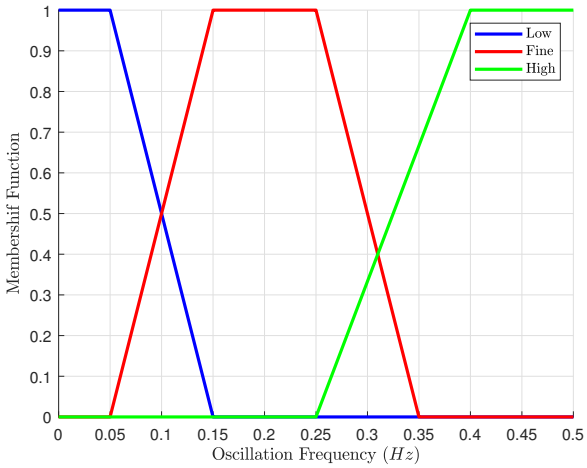

FIGURE 12.4
Membership functions for 3 fuzzy sets "low", "fine" and "high" defined on input signal oscillation frequency.

$0.01Hz, 0.02Hz, ..., 0.5Hz$ 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. 12.4. The curves in Fig. 12.4 is obtained by fitting the "percentage of agreement" using trapezoidal functions. Figure 12.4 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. 12.5. 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. 12.6.

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. 12.6.

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

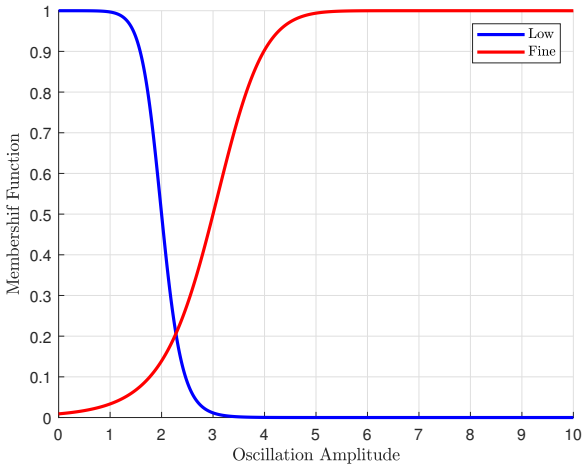

FIGURE 12.5
Membership functions for 2 fuzzy sets "low" and "fine" defined on input signal oscillation amplitude.

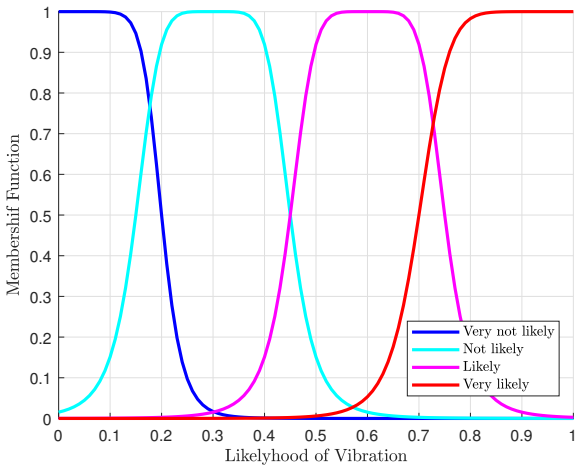

FIGURE 12.6
Membership functions for 4 fuzzy sets "very unlikely", "unlikely", "likely" and "very likely", defined on input signal oscillation amplitude.

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. 12.6 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 12.1 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. 12.7 and 12.8. 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);
```

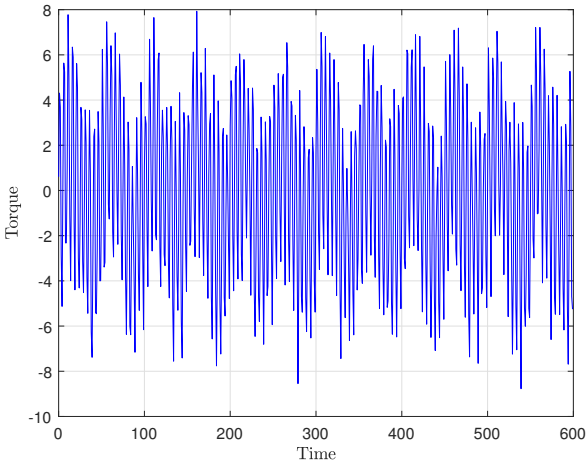

FIGURE 12.7
Measured torque.

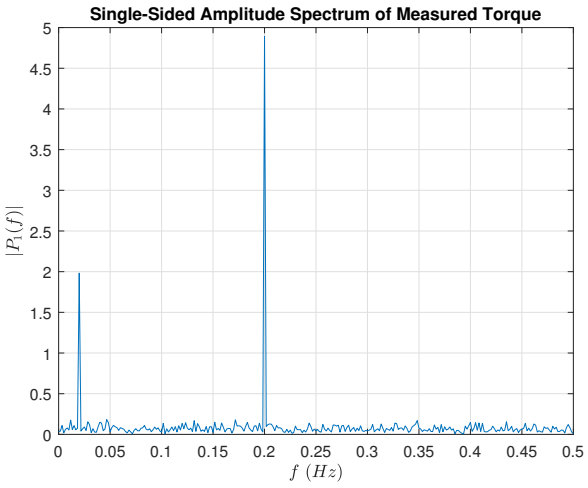

FIGURE 12.8
Measured torque single-sided amplitude spectrum.

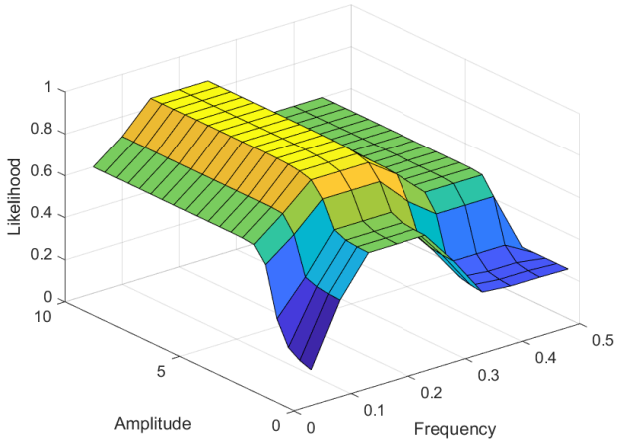

FIGURE 12.9
Likelihood of vibration as a function of oscillation frequency and amplitude.

```
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

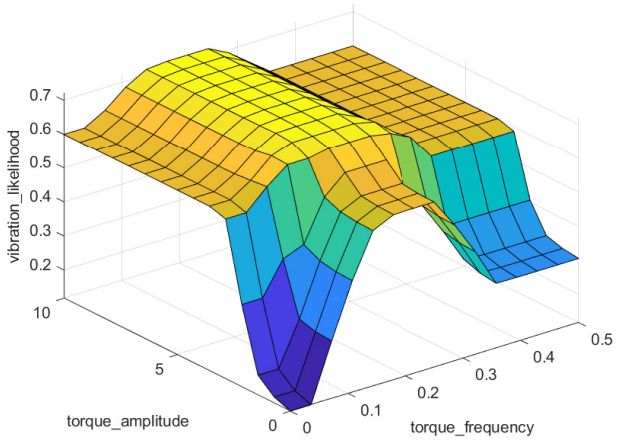

FIGURE 12.10
Likelihood of vibration as a function of oscillation frequency and amplitude using `mamfis`.

```
gensurf(vd_fis)
```

to obtain the plot of likelihood as a function of the oscillation frequency and amplitude as given in Fig. 12.10, which is similar with Fig. 12.9.

13

Multi-Agent Control System

CONTENTS

14

Transformer

CONTENTS

Transformer is a new schematic design of deep learning model which, compared with other designs such as CNN and RNN, is better at parallel training. It has become a handy tool for natural language processing and human interfacing.

In this Chapter, selected research papers on transformer are discussed. *ChatGPT*, an AI chatbot that uses transformer, is introduced as an example.

14.1 Transformer Initial Proposal

TBA

14.2 ChatGPT

TBA

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.