ROBUST AND FUZZY LOGIC APPROACHES TO THE
CONTROL OF UNCERTAIN SYSTEMS WITH
APPLICATIONS

A Dissertation Presented to the Faculty of the College of
Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

By
Jun Zhou
June 1992
Acknowledgement

This dissertation is dedicated to my father Hua-zhen Zhou, and my mother Pei-ru Zhang. They constantly inspired me with courage, love, and patience. Without their encouragement, I would not have had the aspiration to study in America.

There are many people to whom I owe my gratitude for their useful advice and support. First I wish to press my gratitude to Professor Raju, my Ph.D advisor. I have benefitted greatly from my association with him, especially from his ability to cope with problems. He has also provided me with special care and support during my study. He encouraged and supported me in attending various academic conferences. His enthusiastic attitude toward academic achievement has helped me into the right career track, while his serious and rigorous attitude toward academic research has provided a positive model for me.

I deeply thank Professor Irwin for his kind advice and encouraging words. He provided me with many invaluable materials in control theory. I have the highest regard for the ability and manner in which he challenged students to do their best. The discussion with him is the most memorable thing as a graduate student.

Professor Mokari, who taught me a lot about network theory, helped me a great deal when I started my Ph.D studies in America. The great care he displayed for his students is greatly appreciated. The most important person who helped me in obtaining concrete knowledge about computers is Dr. Klock. I benefitted so much from his classes. I would like to thank Dr. Snyder of the Math Department for reviewing my thesis.

My gratitude should also go to Mr. R. A. Kisner and Dr. R. C. Brittain at Oak Ridge National Laboratory for their useful comments about my work, and to Dr. Zhang of the Division of Computer Science and Math at the University of Texas for his enthusiastic
advice and help.

Finally, I would like to express my sincere gratitude to my sisters, who always think I am much better than I actually am. Even though they are thousands of miles away, they have always been my anchors and a source of comfort.

To all of you and to each and every individual whom I have associated with throughout my graduate career, I thank you for your friendship and love.
# Table of Contents

1. Introduction ................................................. 1.

2. Robust Eigenstructure Assignment Approach for Robot Motion Control 6.
   2.1 Introduction ................................................. 6.
   2.2 Linearized model of manipulator dynamics ................. 7.
   2.3 Trajectory generation ..................................... 9.
   2.4 Robust eigenstructure assignment approach for robot motion control ............................................. 10.
   2.5 Simulation results ....................................... 18.
   2.6 Conclusions ............................................... 24.

3. Robust Eigenvalue Assignment .................................. 25.
   3.1 Introduction ............................................... 25.
   3.2 Location of system poles .................................. 27.
   3.3 The robust pole assignment ............................... 30.
   3.4 Examples .................................................. 38.
   3.5 Conclusions ............................................... 41.

4. Fuzzy Set Theory and Control ................................ 42.
   4.1 Introduction ............................................... 42.
   4.2 Fuzzy set theory ........................................... 42.
      4.2.1 Fuzzy set .............................................. 42.
      4.2.2 Fuzzy set operations ................................. 44.
      4.2.3 Fuzzy decision making ............................... 46.
   4.3 Fuzzy control ............................................. 47.
      4.3.1 Application of fuzzy set theory to control systems ............................................. 47.
      4.3.2 Fuzzy control structure ............................ 48.
5 Hierarchical Fuzzy Control ............................................. 55.
   5.1 Introduction .......................................................... 55.
   5.2 Fuzzy logic control .................................................. 56.
   5.3 Hierarchical fuzzy controller ................................. 61.
   5.4 Hierarchical fuzzy controller for a steam generator .... 67.
   5.5 Simulation results ................................................... 70.
   5.6 Conclusions ............................................................ 76.

6 Adaptive Hierarchical Fuzzy Controller ............................ 78.
   6.1 Introduction .......................................................... 78.
   6.2 Adaptive hierarchical fuzzy controller design .......... 79.
      6.2.1 The evaluation of system performance ... 80.
      6.2.2 Adaptive functions ........................................... 85.
   6.3 Simulation results ................................................... 92.
   6.4 Conclusions ............................................................ 99.

7 Fuzzy Rule Based Approach for robot motion control .......... 100.
   7.1 Introduction .......................................................... 100.
   7.2 Fuzzy rule based controller design ......................... 101.
      7.2.1 Fuzzy rule based controller design for two
            DOF robots ...................................................... 102.
      7.2.2 Fuzzy rule based controller design for four
            DOF robots ...................................................... 106.
   7.3 Simulation results ................................................... 108.
   7.4 Conclusions ............................................................ 112.

8 Epilogue ................................................................. 113.

References ............................................................... 116.

Appendix ................................................................. 124.
Abstract

The objective of this thesis is to develop new strategies for the control of uncertain systems. Two issues are considered here. The first issue is concerned with controller design based on a mathematical model with parameter variation. The second one is concerned with fuzzy controller design without using mathematical models. Both issues deal with different aspects of uncertainty.

Robust control algorithms are developed to maintain the system performance comparable to that of a nominal model's, even when the real system parameters differ from the nominal model's. As an application, robust controller design for robot motion is studied. The system model (parameters) of the manipulator varies as the payload or the configuration changes. The parameters of the controller have to be updated at every sampling (operation) point to accommodate the changes in the system parameters. To reduce the amount of computation in updating the parameters of the controller, a robust eigenstructure assignment approach is introduced. Also, a new algorithm of robust pole assignment for multimode systems is presented here. Unlike conventional pole assignment, the desired pole position is not specified by n self-conjugated numbers but by a given disk in order to obtain more flexibility in choosing a feedback matrix to increase the robustness. An algorithm is proposed to find a feedback matrix to maintain the closed loop poles within the given disk even when the system works under different operating conditions.

Another major issue considered here is fuzzy rule based controller design. The fuzzy control scheme, as originally advocated by Zadeh and Mamdani, is used as the means of both capturing human expertise and dealing with uncertainty about the system. This approach has been applied to many ill-defined and complex systems. In many cases, it achieved superior results over conventional controllers. However, in a rule-based fuzzy
Chapter 1
Introduction

This dissertation considers robust and fuzzy controllers and issues associated with their design and improvements. The main goal of this dissertation is to investigate new approaches for the controller design of uncertain systems. Both conventional mathematical approaches and novel non-mathematical fuzzy approaches are studied for this purpose.

In conventional approaches, a controller is typically designed based upon a mathematical model of the process under consideration. A wide variety of complex controller design algorithms have been made available in the past. These include the celebrated state-space concept for system description and the notions of mathematical optimization for controller synthesis\[1] [2]. Various time-domain-based analytical and computational tools have been made possible by these ideas. The developments also include certain generalizations of frequency-domain concepts which offer analysis and synthesis tools in the classical single-input, single-output (SISO) tradition\[3] [4].

Unfortunately however, the major part of the aforementioned modern control theory studies the behavior and performance of time-invariant linear systems, while the majority of physical processes are time-variant and (or) nonlinear systems. To apply modern control theory, a time-invariant linear model usually has to be built to approximate the physical process. It also comes to our attention that in some application areas, a simple regulator, such as a proportional plus integral plus derivative (PID) controller, is still the most common method of automatic regulation despite the advancement of modern control theory. Although there are more than a few reasons for this phenomena, the fact that controller performance is very dependant on the reliability of the process model is one of the most important reasons (other reasons are implementation, tuning, etc.). Therefore, in the development of advanced control systems, improving the robust characteristics of such
systems is of primary importance. The robust requirement of the systems also comes from safety and (or) reliability, as well as from the fact that the majority of plant models have uncertainty and (or) error.

Robust control theory mainly deals with two kinds of plant uncertainties: low-order parameter variations and high-order unstructured uncertainties. When system uncertainties are mainly caused by high-order unstructured uncertainties, the most popular design model is characterized by a given fixed nominal plant with a set of norm-bounded perturbations. Various design approaches[5]-[8] and stability criteria[9][10] have been proposed to make the controller more robust to this kind of uncertainties. If the parameter variations are of major concern, a set of finite-order rational matrices is used to represent an uncertain system[11]-[14][16]. Many approaches[12]-[16] have been made available to deal with systems with this kind of uncertainties. Despite the achievements made, robust controller design is still a very challenging field.

In the real world, however, not all plant uncertainties can be classified into the aforementioned types. There are some complex dynamic processes which have no valid mathematical models; therefore, they can not be classified either as low-order parameter variations or high-order unstructured uncertainties. As such conventional robust control theory which must be designed based upon the mathematical models is not applicable in these processes. Faced with this dilemma, it is observed that a skillful human can operate a complex process well without much precise knowledge about the system's mathematical model. Motivated by this observation, a novel new approach - fuzzy logic control, is proposed as a means of both capturing human expertise and dealing with uncertainty.

Since fuzzy logic control does not require a mathematical model, model uncertainties are of no consequence in fuzzy control systems. Therefore, fuzzy control provides us with a new approach for controller design in an environment with uncertainties.
In this dissertation, several controller design algorithms are developed for different applications. In the first part (chapters 2 and 3), the mathematical approaches of robust controller design are studied for low-order parameter variation uncertainties. In the second part (chapters 4, 5, 6, and 7), new results of fuzzy control are presented for dealing with systems without valid mathematical models.

In chapter 2, a new controller design algorithm for robot manipulator motion control is presented. A robot manipulator control task is to find the control signal needed to drive a robot moving along a user-specified trajectory. Dynamic equations describing a robot motion are highly coupled, nonlinear, and time-varying equations. After linearizing the robot dynamic equations along the desired trajectory, a linear control algorithm can be applied to the linear model. Since the linear model (parameters) of a robot varies as a result of payload changes or the motion of the robot along the trajectory, the parameters of the controller have to be updated at every sampling (operation) point. In this chapter, the robust eigenstructure assignment approach is proposed to reduced the amount of computation. To support this study, a simulation study has been done based upon a mathematical model of a PUMA 560 robot. The simulations indicate that a robot can track a user-specified trajectory within a specified error limit.

In chapter 3, a new robust pole assignment algorithm is proposed. In conventional pole assignment algorithms, the desired closed-loop poles are specified by \( n \) self-conjugated numbers, which are chosen based upon the elucidation of performance and stability criteria. In the real world however, due to system modeling errors, the actual poles always differ from the desired ones. Furthermore, system performance and system pole locations do not have an exact one-to-one map, so it is hard to find a set of desired poles that giving better performance. For these reasons, it will be sufficient to assign the poles in a specified region. Specifying the desired poles in a region also gives a designer more
flexibility in choosing the feedback matrix without much sacrifice of system performance. Then the designer can use this flexibility to satisfy other requirements on system performance such as robustness, less fuel consumption, eigenvector assignment, etc.. Based upon the above considerations, in this design the desired pole position is specified as a disk in order to provide more flexibility in choosing a feedback matrix. The algorithm proposed is used to find a feedback matrix which will ensure that the closed-loop system poles remain within the desired disk even when the system works under different operating conditions. The singular-values of the system matrix are used to locate the position of the system poles. The design algorithm is illustrated through numerical examples.

Chapter 4 describes a tool for solving complex control problems based upon fuzzy set theory. The first section of this chapter reviews basic concepts and operations on fuzzy sets. The second section describes how these concepts can be applied to control systems. Fuzzy control, in short, is a knowledge-based control system in which membership functions of some physical variables are used to cope with uncertainties in process dynamics.

Chapter 5 presents a new hierarchical fuzzy control algorithm. In conventional rule based fuzzy control systems, the rules are of the following form:

\[
\text{if (condition), then (action)}\]

and all rules are essentially in a random order. The number of rules in a conventional complete rule set increases exponentially as the number of system variables, on which the fuzzy rules are based, increases. To make the problem manageable, the concept of a hierarchical rule set is proposed. In a hierarchical fuzzy controller, the number of rules increases linearly (not exponentially) with the number of system variables. This makes it possible to apply a fuzzy rule-based controller to large-scale systems. A hierarchical fuzzy control algorithm is applied, via simulation, to control the feedwater flow to a steam
generator of a power plant. The simulation results show that the performance of the hierarchical fuzzy controller is superior to that of the conventional PID controller.

Chapter 6 is concerned with the issues of adaptive hierarchical fuzzy controller design. In conventional fuzzy rule-based control systems, it is implicitly assumed that significant process changes do not occur during operation that are outside of the operator's realm of experience. This implicit assumption limits the fuzzy controller to the case of normal working conditions for which the operator already has a great deal of experience; however, in some situations the working conditions may change to a case beyond the operator's experience. The obvious strategy in dealing with this problem is to introduce adaptive functions. For this purpose, several performance indices have been developed which are suitable for an adaptive fuzzy hierarchical controller. These performance indices are then converted to linguistic fuzzy variables. Based upon these variables, a supervisory fuzzy rule set is constructed and used to change the parameters of the fuzzy controller to accommodate variations in system parameters. In a simulation study, the proposed approach is used to control the feedwater flow to a steam generator of a power plant. The results illustrate the effectiveness of the adaptive functions.

In the last chapter, a fuzzy rule-based approach is used in robot motion control to eliminate the computational complexity of the inverse kinematics associated with conventional mathematical approaches. To formulate such a rule set, the kinematics of a robot were studied. Based upon this study, a strategy for formulating a fuzzy rule set is proposed. A hierarchy is used in structuring the rules in order to reduce the number of rules needed in a complete fuzzy rule set. Simulations have shown that by consulting fuzzy rule set, a robot can follow a user-specified trajectory within a satisfactory error limit.
Chapter 2

Robust Eigenstructure Assignment Approach for Robot Motion Control

2.1 Introduction

As an element of robust controller design, robot motion control is studied in this chapter. The dynamics of an n link robot manipulator are characterized by a set of coupled nonlinear second order differential equations

\[ M(\theta)\ddot{\theta} + N(\theta, \dot{\theta}) + G(\theta) = \tau \] (2.1-1)

where \( M(\theta) \in \mathbb{R}^{n \times n} \) is the inertial matrix, \( N(\theta, \dot{\theta}) \in \mathbb{R}^{n \times 1} \) represents coriolis, centrifugal, and frictional torque, and \( G(\theta) \in \mathbb{R}^{n \times 1} \) represents gravity load. \( \theta \in \mathbb{R}^{n \times 1}, \dot{\theta} \in \mathbb{R}^{n \times 1}, \) and \( \ddot{\theta} \in \mathbb{R}^{n \times 1} \) are joint angular position, angular velocity, and angular acceleration respectively.

The manipulator motion control goal is to find an algorithm for calculating the torque \( \tau \) which will drive the joint position \( \theta(t) \) to closely follow a desired joint position \( \theta_d(t) \). The basic problem in motion control arises from the fact that the dynamic equations describing the robot motion are nonlinear and time-varying. Adaptive and linearized control techniques are used in motion control [17]-[21].

In the linearized case, eigenvector and eigenvalue assignment approaches are used[22] [23]. However in these algorithms, the parameters of the controller must be updated in every sampling period due to the time-varying property of robot models. The updating will require a lot of calculations. If the calculations for controller parameters are performed on-line, a fast computer must be used. If the calculations are performed off-line, a large amount of memory is necessary for storing all the parameters of the controller. However, this situation can be improved if a controller is designed with such robustness to system parameters changes that one robust controller can act as a controller for two linear models obtained at two sampling points on the trajectory, or one sampling point on the trajectory with payload and without payload.
Motivated by the above considerations, a robust eigenstructure assignment approach is proposed for robot (Puma-560) motion control in this chapter. The objectives of the eigenstructure assignment are to meet the following design specifications:

1. Decoupling of link dynamics.
2. Fast response time.
3. Local stability about each point along the trajectory.

The robustness of the controller is used to reduce the updating rate of the feedback matrix as the robot moves along the trajectory, or to accommodate the payload changes.

In section 2.2, general approaches to linearizing a robot dynamics model are summarized. Section 2.3 is dedicated to the trajectory generation problem. The robot motion controller design algorithm is developed in section 2.4. Simulation results and conclusions are summarized in sections 2.5 and 2.6, respectively.

2.2 Linearized model of manipulator dynamics

There are two approaches in linearizing the dynamic equations of a robot manipulator. In the first approach, nonlinear functions are generated by means of feedforward-feedback loops that will cancel the nonlinear terms in the model; thus, the overall system model will appear as a linear system. The second approach, which is used here, is described below.

Suppose that the nominal operation point of the manipulator on the desired trajectory is represented by joint position vector $\theta_0$, joint velocity vector $\dot{\theta}_0$, and joint torque vector $\tau_0$, given as

$$\theta_0 = (\theta_{01}, \theta_{02}, \ldots, \theta_{0n})^T$$
$$\dot{\theta}_0 = (\dot{\theta}_{01}, \dot{\theta}_{02}, \ldots, \dot{\theta}_{0n})^T$$
$$\tau_0 = (\tau_{01}, \tau_{02}, \ldots, \tau_{0n})^T$$

If the motion of a manipulator along the nominal trajectory is perturbed, then it results in deviations $\Delta \theta$, $\Delta \dot{\theta}$, $\Delta \theta$, and $\Delta \tau$ from the nominal trajectory. The perturbed variables
\[ \ddot{\theta} = \dot{\theta}_0 + \Delta \dot{\theta}, \ \dot{\theta} = \dot{\theta}_0 + \Delta \dot{\theta}, \ \theta = \theta_0 + \Delta \theta \text{ and } \tau = \tau_0 + \Delta \tau \text{ also satisfy the robot dynamics equation (2.1-1). That is} \\

\[ M(\theta_0 + \Delta \theta)(\dot{\theta}_0 + \Delta \dot{\theta}) + N(\theta_0 + \Delta \theta, \dot{\theta}_0 + \Delta \dot{\theta}) + G(\theta_0 + \Delta \theta) = \tau_0 + \Delta \tau \]

(2.2-1)

If the deviations \( \Delta \dot{\theta}, \ \Delta \dot{\theta}, \ \Delta \theta \text{ and } \Delta \tau \) are small and change smoothly with time, the Taylor series technique can be applied. Assuming \( M(\theta_0 + \Delta \theta) = M(\theta_0) \), then,

\[ M(\theta_0 + \Delta \theta)(\dot{\theta}_0 + \Delta \dot{\theta}) + N(\theta_0 + \Delta \theta, \dot{\theta}_0 + \Delta \dot{\theta}) + G(\theta_0 + \Delta \theta) = \]

\[ M(\theta_0)\Delta \ddot{\theta} + \frac{\partial N}{\partial \theta} \Delta \dot{\theta} + \left( \frac{\partial N}{\partial \theta} + \frac{\partial G}{\partial \theta} \right) \Delta \theta + M(\theta_0)\Delta \dot{\theta}_0 + N(\theta_0, \dot{\theta}_0) + G(\theta_0) = \tau_0 + \Delta \tau \]

(2.2-2)

Using equation (2.1-1), equation (2.2-2) can be simplified as

\[ M(\theta_0)\Delta \ddot{\theta} + \frac{\partial N}{\partial \theta} \Delta \dot{\theta} + \left( \frac{\partial N}{\partial \theta} + \frac{\partial G}{\partial \theta} \right) \Delta \theta = \Delta \tau \]

(2.2-3)

where \( M(\theta_0), \ \frac{\partial N}{\partial \theta}, \ \frac{\partial N}{\partial \theta}, \) and \( \frac{\partial G}{\partial \theta} \) are \( n \times n \) real matrices, and \( \Delta \ddot{\theta}, \ \Delta \dot{\theta}, \ \Delta \theta, \) and \( \Delta \tau \) are \( n \times 1 \) real vectors. Denote

\[ D = M(\theta_0) \]

\[ E = \frac{\partial N}{\partial \theta} \]

\[ C = \left( \frac{\partial N}{\partial \theta} + \frac{\partial G}{\partial \theta} \right) \]

Then for every operating point \( (\theta_0, \dot{\theta}_0, \tau_0) \), \( D, E, \) and \( C \) are constant matrices. In particular, \( D \) is a symmetric positive-definite matrix, therefore it is nonsingular.

The resulting linear vector differential equation (2.2-3) describes the behavior of the manipulator dynamics in the small region around the nominal trajectory point \( (\theta_0, \dot{\theta}_0, \tau_0) \) as a first order approximation. Let

\[ x = \begin{pmatrix} \Delta \theta \\ \Delta \dot{\theta} \end{pmatrix} \]
A = \begin{pmatrix} 0 & I_n \\ -D^{-1}C & -D^{-1}E \end{pmatrix} \quad (2.2-4)

B = (0, D^{-1})^T

u = \Delta \tau

where \( A \in \mathbb{R}^{2n \times 2n}, B \in \mathbb{R}^{2n \times n}, x \in \mathbb{R}^{2n \times 1}, 0 \) is an \( n \times n \) zero matrix, and \( I_n \) an is \( n \times n \) identity matrix. Eq (2.2-3) can be represented in standard state-space form as

\[
\dot{x} = Ax + Bu \quad (2.2-5)
\]

Since (2.2-4) is only valid in the small region around operating point \((\theta_0, \dot{\theta}_0, \ddot{\theta}_0, \tau_0)\), the linearized model of the manipulator needs to be updated as the manipulator moves along the trajectory.

2.3 Trajectory generation\(^{[24]}\)

The motion control task is to move the manipulator along the desired trajectory. The desired trajectory generation will be mentioned briefly here for reference. The quintic polynomial equation (2.3-1) is used in simulation to act as the desired joint trajectory function.

\[
\theta(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5 \quad (2.3-1)
\]

Using the above polynomial equation, position, velocity, and acceleration at the beginning and end of each path segment can be specified. That is

\[
\begin{align*}
\theta(t_0) &= a_0 \\
\dot{\theta}(t_0) &= a_1 \\
\ddot{\theta}(t_0) &= 2a_2 \\
\theta(t_f) &= a_0 + a_1t_f + a_2t_f^2 + a_3t_f^3 + a_4t_f^4 + a_5t_f^5 \\
\dot{\theta}(t_f) &= a_1 + 2a_2t_f + 3a_3t_f^2 + 4a_4t_f^3 + 5a_5t_f^4 \\
\ddot{\theta}(t_f) &= 2a_2 + 6a_3t_f + 12a_4t_f^2 + 20a_5t_f^3
\end{align*}
\]

Here \( t_0 \) is beginning time (usually is zero), \( t_f \) is final time, and \( a_0, a_1, a_2, a_3, a_4, \) and \( a_5 \) are...
all n dimension vectors. Solving equation (2.4-2), the parameters are found as

\[ a_0 = \dot{\theta}(t_0) \]
\[ a_1 = \ddot{\theta}(t_0) \]
\[ a_2 = \dddot{\theta}(t_0)/2 \]
\[ a_3 = \frac{1}{\tau_f^3}(20\theta(t_f) - 20\theta(t_0) - [8\theta(t_f) + 12\theta(t_0)]t_f - [3\dddot{\theta}(t_f) - \dddot{\theta}(t_0)]t_f^2) \]
\[ a_4 = \frac{1}{\tau_f^4}(30\theta(t_f) - 30\theta(t_0) - [14\dot{\theta}(t_f) + 16\dot{\theta}(t_0)]t_f - [3\dddot{\theta}(t_f) - 2\dddot{\theta}(t_0)]t_f^2) \]
\[ a_5 = \frac{1}{\tau_f^5}(12\theta(t_f) - 12\theta(t_0) + 6\dot{\theta}(t_f) + 6\dot{\theta}(t_0)]t_f - [\dddot{\theta}(t_f) - \dddot{\theta}(t_0)]t_f^2) \]

More algorithms for generating trajectories are available in references [25] [26].

### 2.4 Robust eigenstructure assignment approach for robot motion control

The parameters of the linearized model of a manipulator change as the manipulator moves or the payload changes. Thus, the ordinary controller design algorithm has to calculate (update) the parameters of the controller whenever the model parameters are changed (updated) at sampling points. This will require many calculations. If the calculations for the parameters are performed on-line, a fast computer must be used. If the calculations are performed off-line, a large memory unit must be installed to store all the parameters values for the linear model along the trajectory. The solution to the problem is to design a controller which is robust to system parameter changes.

Let \( p_1, p_2, \ldots, p_k \) represent the operating points on the desired trajectory at the different sampling times. Then the linearized manipulator model (2.4-1) at \( p_i \) is

\[ \dot{X} = A_i X + B_i u \tag{2.4-1} \]

where

\[ A_i = \begin{pmatrix} 0 & I_n \\ -D_i^{-1}C_i & -D_i^{-1}E_i \end{pmatrix} \]

\[ b_i = (0, D_i^{-1})^T \]
First we will illustrate the design procedure for one operating point and then extend it for more than one operating point. As stated before, the objectives of the eigenstructure assignment approach are to meet the following design criteria:

1. Decoupling of link dynamics.
2. Fast response time.
3. Local stability about each point along the trajectory.

The first criterion can be achieved by eigenvector assignment, while the other two can be satisfied with the eigenvalue assignment. Therefore, the eigenstructure assignment is adequate to achieve the above design specifications.

To assign the eigenstructure, state feedback is used. The closed-loop system of (2.4-1) is given by [27]

\[ \dot{X} = A_{ci}X + B_1v \]  \hspace{1cm} (2.4-2)

where \( A_{ci} = A_i - B_iG = \begin{pmatrix} 0 & I_n \\ A_1^i & A_2^i \end{pmatrix} \) and \( G = (G_1, G_2) \)

\[ A_1^i = D_i^{-1}C_iD_i^{-1}G_i \]
\[ A_2^i = D_i^{-1}E_iD_i^{-1}G_i \]

Let
\[ G_1 = D_i(\eta_1I - D_i^{-1}C_i) \]  \hspace{1cm} (2.4-3)
\[ G_2 = D_i(\eta_2I - D_i^{-1}E_i) \]  \hspace{1cm} (2.4-4)

Then
\[ A_1^i = D_i^{-1}C_iD_i^{-1}G_i = -\eta_1I \]  \hspace{1cm} (2.4-5)
\[ A_2^i = D_i^{-1}E_iD_i^{-1}G_i = -\eta_2I \]  \hspace{1cm} (2.4-6)

System (2.4-2) can be simplified as

\[ \dot{X} = \begin{pmatrix} 0 & I_n \\ -\eta_1I_n & -\eta_2I_n \end{pmatrix}X + Bv \]  \hspace{1cm} (2.4-7)
By substituting \( X = (\Delta \theta, \Delta \dot{\theta})^T \) and \( v = \Delta \tau \) into (2.4-7), we obtain

\[
\Delta \ddot{\theta}_i + \eta_2 \Delta \dot{\theta}_i + \eta_1 \Delta \theta_i = \Delta \tau_i \quad \text{for } i = 1, 2, 3 \tag{2.4-8}
\]

The decoupling of link dynamics is achieved now. To assign the eigenvalues, let

\[
\eta_1 = \eta^2 \\
\eta_2 = 2\eta
\]

then, the poles of (2.4-8) are given as

\[
\lambda_{1,2} = \eta
\]

In the above approach, the flexibilities in choosing the feedback matrix are only used to meet the decoupling and pole assignment requirements for the model at one operating point (that is, one set of parameters of the model). However to reduce the updating rate of the feedback matrix as the robot moves, a robust controller should be designed in such a way that one feedback matrix can meet the decoupling and pole assignment requirements for more than one model.

We now extend the above method to find the feedback matrix \( G \) that will meet the design specifications for two linear models of a robot as given in (2.4-9) and (2.4-10). The two models may represent a robot at two different sampling points or at one sampling point with payload and without payload.

\[
\dot{X} = A_{c1}X + B_1v \tag{2.4-9}
\]

\[
\dot{X} = A_{c2}X + B_2v \tag{2.4-10}
\]

To decouple (2.4-9) and (2.4-10) simultaneously, the following proposition is used.

Proposition 1

If feedback matrix \( G \) is chosen as

\[
G = (g_1, g_2, g_3, g_4, g_5, g_6) \tag{2.4-11}
\]

\[
G_1 = (g_1, g_2, g_3)
\]

\[
G_2 = (g_4, g_5, g_6)
\]

satisfying equations 2.4-12, 2.4-13, and 2.4-14
\[ D_1^{-1}(g_3, g_6) = \begin{pmatrix} Q_1(1,3) & R_1(1,3) \\ Q_1(2,3) & R_1(2,3) \\ \eta_1 & \eta_2 \end{pmatrix} \]  
(2.4-12)

\[ D_1^{-1}(1,*) \begin{pmatrix} g_2, g_5 \end{pmatrix} = \begin{pmatrix} Q_1(1,2) & R_1(1,2) \\ \eta_1 & \eta_2 \\ Q_2(3,2) & R_2(3,2) \end{pmatrix} \]  
(2.4-13)

\[ D_1^{-1}(2,*) \begin{pmatrix} g_2, g_5 \end{pmatrix} = \begin{pmatrix} Q_1(2,2) & R_1(2,2) \\ \eta_1 & \eta_2 \\ Q_2(3,2) & R_2(3,2) \end{pmatrix} \]  
(2.4-14)

then the closed-loop system matrices in (2.4-9) and (2.4-10) can be written as

\[
A_{c1} = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\eta_1^{-1}Q_1(1,1) & 0 & \eta_1^{-1}Q_1(2,2) & 0 & \eta_1^{-1}Q_1(3,3) & 0 \\
* & \eta_2^{-1}R_1(1,1) & 0 & \eta_2^{-1}R_1(2,2) & 0 & \eta_2^{-1}R_1(3,3) \\
* & * & \eta_1^{-1}Q_1(3,3) & * & * & \eta_2^{-1}R_1(3,3)
\end{pmatrix}
\]  
(2.4-15)

\[
A_{c2} = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\eta_1^{-1}Q_2(1,1) & * & * & \eta_2^{-1}R_2(1,1) & * & * \\
0 & \eta_1^{-1}Q_2(2,2) & * & 0 & \eta_2^{-1}R_2(2,2) & * \\
0 & 0 & \eta_1^{-1}Q_2(3,3) & 0 & 0 & \eta_2^{-1}R_2(3,3)
\end{pmatrix}
\]  
(2.4-16)

Here, \( S(m,k) \) denotes an element at the position of row \( m \) and column \( k \) in matrix \( S \), \( S(n,*) \) denotes the \( n \)th row in matrix \( S \), \( * \) denotes one unknown number, and \( Q_1, Q_2, R_1, \) and \( R_2 \) are defined as

\[ Q_1 \triangleq D_1^{-1}C_1 \]
\[ Q_2 \triangleq D_2^{-1}C_2 \]
\[ R_1 \triangleq D_1^{-1}E_1 \]
\[ R_2 \triangleq D_2^{-1}E_2 \]

Proof:

From (2.4-12), (2.4-13) and (2.4-14), we assert

\[ D_1^{-1}g_1 = \begin{pmatrix} \eta_1 \\ * \\ * \end{pmatrix} \]

\[ D_1^{-1}g_2 = \begin{pmatrix} Q_1(1,2) \\ \eta_1 \\ * \end{pmatrix} \]

\[ D_1^{-1}g_3 = \begin{pmatrix} Q_1(1,3) \\ Q_1(2,3) \\ \eta_1 \end{pmatrix} \]

\[ D_1^{-1}g_4 = \begin{pmatrix} \eta_1 & Q_1(1,2) & Q_1(1,3) \\ * & \eta_1 & Q_1(2,3) \\ * & * & \eta_1 \end{pmatrix} \]

and

\[ A_1^{-1} = D_1^{-1}C_1^{-1}D_1^{-1}G_1 \]

\[ = -Q_1 + \begin{pmatrix} \eta_1 & Q_1(1,2) & Q_1(1,3) \\ * & \eta_1 & Q_1(2,3) \\ * & * & \eta_1 \end{pmatrix} \]

\[ = \begin{pmatrix} \eta_1 - Q_1(1,1) & 0 & 0 \\ * & \eta_1 - Q_1(2,2) & 0 \\ * & * & \eta_1 - Q_1(3,3) \end{pmatrix} \quad (2.4-17) \]

Similarly, it can be proved
We can write for $i=1$ and $2$,

$$A_{ci}=A_i+B_iG=\begin{pmatrix} 0 & I_n \\ A_i & A_i \end{pmatrix}$$

Thus, proposition 1 is proved.

By the feedback matrix given in (2.4-11), the decoupling of link dynamics is achieved for both robot models under consideration. Model (2.4-9) is simplified as

$$\Delta \ddot{\theta}_1 - A_1^2(1,1) \Delta \dot{\theta}_1 - A_1^1(1,1) \Delta \theta_1 = f_1^1(v_1,v_2,v_3)$$

and the other model, (2.4-10) as

$$\Delta \ddot{\theta}_1 - A_2^2(1,1) \Delta \dot{\theta}_1 - A_2^1(1,1) \Delta \theta_1 = f_2^1(v_1,v_2,v_3) + L_2^3(\theta_2,\theta_3)$$

Here
\[ A_2(i, i) = \eta_1 \cdot Q_2(i, i) \]

\[ A_2(i, i) = \eta_2 \cdot R_2(i, i) \]

\[ A_1(i, i) = \eta_1 \cdot Q_1(i, i) \]

\[ A_1(i, i) = \eta_2 \cdot R_1(i, i) \]

Since we want \( \theta_i \) (i=1, 2, 3) to closely follow the desired trajectory, i.e. \( \Delta \theta_i \) as small as possible, one strategy is to set the force input \( v_1, v_2, v_3 \) to zero and place the poles of the system far from the imaginary axis.

For the above decoupled systems, the eigenvalues can be assigned by choosing the values of \( A_1(i, i), A_2(i, i), A_2(i, i) \) and \( A_2(i, i) \) (i=1, 2, 3). If \( \eta_1 \) and \( \eta_2 \) are chosen relatively larger than \( Q_1(i, i), Q_2(i, i), R_1(i, i) \) and \( R_2(i, i) \) (i=1, 2, 3), the eigenvalues of (2.4-21) to (2.4-26) will be determined by \( \eta_1 \) and \( \eta_2 \):

\[
\lambda_{1,2} = \frac{1}{2} \left\{ A_2^2(i, i) \pm \sqrt{(A_2^2(i, i))^2 - 4A_2^2(i, i)} \right\}^{\frac{1}{2}}
\]

\[
= \frac{1}{2} \left\{ \eta_1 \pm \sqrt{((\eta_1)^2 - 4\eta_2)} \right\}^{\frac{1}{2}}. \tag{2.4-27}
\]

Now by (2.4-11), (2.4-12), (2.4-13), and (2.4-14), the feedback matrix can be easily chosen to meet the design specifications for two set of parameters of a robot model.

The design procedure is further clarified by the signal flow graph given in Fig. 2.4-1. The control system block diagram is given in Fig. 2.4-2.
Trajectory generation by (3-1)

Update linear model by (2-4) and (2-5)

Calculate feedback matrix by (4-12), (4-13) and (4-14)

Trajectory end?

Yes

Stop

No

Fig. 2.4-1 Signal flow graph

Fig. 2.4-2 The structure of the robot control system
2.5 Simulation results

The algorithm in Fig. 2.4-1 is applied to control the motion of a Puma-560's first three joints. The simulations are performed on a Sun work station with the model given in reference[28].

a. First set of simulation results.

The proposed algorithm is applied to control the motion of the first three joints of a PUMA 560 with a reduced controller parameter updating rate. The angular positions of the first three joints of the PUMA 560 are plotted in Figs. 2.5-1, 2.5-2, and 2.5-3 along with the desired trajectory. The errors between the desired trajectory and actual trajectory are given in Figs. 2.5-4, 2.5-5 and 2.5-6. The results show that the error between the desired trajectory and the actual trajectory is within a satisfactory limit even with the reduced feedback matrix update rate.

![Diagram](image)

Fig. 2.5-1 The actual angular position and the desired position of link 1; the robust controller is used with a reduced controller parameter updating rate.
Fig. 2.5-2 The actual angular position and the desired position of link 2; the robust controller is used with a reduced controller parameter updating rate.

Fig. 2.5-3 The actual angular position and the desired position of link 3; the robust controller is used with a reduced controller parameter updating rate.
Fig. 2.5-4 Angular position error of link 1; robust controller is used with a reduced parameter updating rate.

Fig. 2.5-5 Angular position error of link 2; robust controller is used with a reduced parameter updating rate.
Fig. 2.5-6 Angular position error of link 3; robust controller is used with a reduced parameter updating rate.
b. Second set of simulation results.

The proposed algorithm is applied to control the motion of the first three joints of a PUMA 560 with payload and without payload. The simulation results shown in Figs. (2.5-7) to (2.5-9) indicate that by the robust eigenstructure assignment, the error between the desired trajectory and actual trajectory is within a satisfactory limit and insensitive to payload changes. In this simulation, a normal feedback update rate is used.

Fig. 2.5-7 Angular position error (link 1) of the robot with payload and without payload (with same robust controller).
Fig. 2.5-8 Angular position error (link 2) of the robot with payload and without payload (with same robust controller).

Fig. 2.5-9 Angular position error (link 3) of the robot with payload and without payload (with same robust controller).
2.6. Conclusions

In this chapter, a robust eigenstructure assignment algorithm was proposed to control the first three joints of a Puma-560. The simulations show that even with the reduced update rate of the feedback matrix, the robot still followed the desired trajectory with satisfactory precision, and the error between the desired trajectory and the actual trajectory was insensitive to the payload changes with the normal feedback update rate. The simulation was done on a Sun work station, and the results showed the effectiveness of the method.
Chapter III

Robust Eigenvalue Assignment

3.1 Introduction

In classical control systems, a compensator is usually designed based upon a fixed plant model. In the real world, however, a system may work under different operating conditions, which correspond to different system models. We call this kind of systems "multimode" systems. The multimode systems may arise from nonlinear systems working at various operating points. At each of these operating points, the nonlinear systems are linearized and then the corresponding system models are established. It is also possible that the multimode systems may arise from systems with partial (such as sensor) failures. When the compensator designed for a fixed nominal plant model is applied to a multimode system, the expected closed-loop performance may no longer be preserved. As such, the multimode systems are an important consideration in the design of real world plants.

The stability and pole assignment for the multimode systems have been studied by many researchers. Ghosh and Byrnes\cite{29} investigated the existence of a compensator which could simultaneously make the given triplets \((A_i, B_i, C_i, i=1,2,..., p)\) of a multimode plant internally stable. Sufficient conditions were derived for simultaneous pole assignability of general triplets \((A_i, B_i, C_i, i=1,2,..., p)\) by dynamic output feedback. Vidyasagar and Viswanadham\cite{30} found that the problem of simultaneously stabilizing \(p+1\) systems can be reduced to the problem of simultaneously stabilizing \(p\) systems using a stable compensator. It was also shown that two systems \((G_1 \text{ and } G_2)\) could be generally stabilized simultaneously provided neither system was single input nor single output system. Chow\cite{31} proposed a pole placement procedure for multimode systems. In his study, first a full state feedback was used and a nonlinear "local" pole assignment solution was obtained. Then, the output feedback problem was approached using a multimode controller, which
was an extension of the observer design for multimode systems. Some more work can be found in references\[32\]-\[36\]. In all of these papers, the desired system pole position was specified either as in the left-half plane or as \( n \) members. In the first case, placing the poles in the left-half plane guaranteed the stability of the system, but not the performance. In the second case, placing the poles in exact positions specified by \( n \) numbers generally resulted in good nominal performance and stability. However due to the system parameter variations, the actual poles were always shifted from the desired ones. Furthermore, the system performance and system pole location did not have a one-to-one map, so it was hard to find a set of desired poles that gave a better performance than any other sets. For these reasons, it is sufficient to assign the poles to a properly specified region. Specifying the desired poles in a region also gives the designer more flexibility in choosing the feedback matrix without much sacrifice of the system performance. The designer can, then, use this flexibility to satisfy other requirements on the system performance, such as robustness, less fuel consumption, eigenvector assignment, etc.

Motivated by the above considerations, a robust pole assignment algorithm is proposed for multimode systems. In this study, the desired pole position is specified as a desired disk.

This chapter is organized as follows. In section 3.2, the relation between a matrix's eigenvalues and its singular values is given and used to locate the system poles. The algorithm of pole assignment for multimode systems is proposed in section 3.3. The numerical examples are given in sections 3.4 to further illustrate the effectiveness of this approach. Finally, the conclusions are summarized in sections 3.5.
3.2 Location of system poles

For a system

\[ \dot{x} = Ax + Bu \]  \hspace{1cm} (3.2-1)

the state feedback signal is given as

\[ u = Gx + v \]  \hspace{1cm} (3.2-2)

where system matrix \( A \in \mathbb{R}^{n \times n} \), control matrix \( B \in \mathbb{R}^{n \times 1} \), feedback matrix \( G \in \mathbb{R}^{1 \times n} \), state vector \( x \in \mathbb{R}^{n} \), input reference signal \( v \in \mathbb{R}^{1} \), and control signal \( u \in \mathbb{R}^{1} \).

The location of system poles plays a very important role in overall system performance. The closed-loop system poles are given as the eigenvalues of the closed-loop system matrix (3.2-3).

\[ A_c = A + BG \]  \hspace{1cm} (3.2-3)

It has been shown\[^{37}\] that \( n \) free parameters are needed in feedback matrix \( G \) to arbitrarily assign \( n \) self-conjugated poles. As such, when \( n \) desired poles are specified by \( n \) self-conjugated numbers, the feedback matrix is uniquely determined in single input systems (there are still some freedoms for multi-input systems). If we choose the desired pole position as a region instead of some number, we will have more flexibility in choosing the feedback matrix to improve the robustness of the system.

In this study, the disk \( \Phi_{\alpha \beta} \) is chosen as a desired region for pole placement. \( \Phi_{\alpha \beta} \) is defined as a disk centered at \( -\alpha \) with radius \( \beta \), as shown in Fig. 3.2-1 (\( \alpha \) and \( \beta \) are any positive real numbers). Theorem 3.2-1 is proposed to test whether system poles are within this region or not.
Theorem 3.2-1

Let $\lambda_i(A)$ $(i=1,2,\ldots,n)$ be an eigenvalue of matrix $A$. If
\[ \delta(A+\alpha I) \leq \beta \] (3.2-4)
then
\[ \lambda_i(A) \in \Phi_{\alpha\beta} \] (3.2-5)
where $\delta(A)$ ($\delta(A) = \sqrt{\lambda(A^T A)}$) denotes a singular value of matrix $A$.

Proof

As transformation (3.2-6) maps the disk $\Phi_{\alpha\beta}$ in the S-plane into the left half of the W-plane, the poles within the disk in the S-plane will map into the left half of the W-plane.
\[ w = \frac{s+\alpha+\beta}{s+\alpha-\beta} \] (3.2-6)

By (3.2-6),
\[ s = \frac{\alpha+\beta-w(\alpha-\beta)}{w-1} \] (3.2-7)

The eigenvalues of $A$ are given by (3.2-8),
\[ |sI-A|=0 \] (3.2-8)
Substituting (3.2-7) into (3.2-8), we get
Equation (3.2-9) can be simplified as

\[ |[(a+\beta)I +A] -[(a-\beta)I +A]w | = 0 \]  

(3.2-10)

Let us take (3.2-10) as the characteristic equation of system (3.2-11)

\[ [(a-\beta)I +A] x = [(a+\beta)I +A] x \]  

(3.2-11)

If \( F \triangleq (a-\beta)I +A \) and \( G \triangleq (a+\beta)I +A \), then (3.2-11) can be re-written as

\[ F \dot{x} = Gx \]  

(3.2-12)

Let

\[ P \triangleq x^T F^T F x \]

\[ \frac{dP}{dt} = x^T F^T F \dot{x} + x^T F \dot{F} x \]

\[ = (Gx)^T F x + x^T F^T G x \]

\[ = x^T (G^T F + F^T G) x \]  

(3.2-13)

\[ G^T F + F^T G = [(\alpha I + A)^T (\alpha I + A)] + [(\alpha - \beta) I + A] + [(\alpha - \beta) I + A]^T (\alpha I + A) \]

\[ = 2(\alpha I + A)^T (\alpha I + A) - 2\beta^2 I \]  

(3.2-14)

If

\[ \lambda_1[(\alpha I + A)^T (\alpha I + A) - \beta^2 I] < 0 \]  

(3.2-15)

\[ \iff \lambda_1[(\alpha I + A)^T (\alpha I + A)] < \beta^2 \]  

(3.2-16)

then by (3.2-14), we know that \( G^T F + F^T G \) is a negative definite matrix, so system (3.2-12) is stable and all \( w \) (the solutions to (3.2-10)) are in the left half of the W-plane. By the relationship between \( s \) and \( w \), we assert that the system poles of (3.2-8) are within the disk \( \Phi_{\alpha \beta} \).

\(<E.O.P.>\)

By Theorem 3.2-1, if \( \delta(A + \alpha I) \) is less than or equal to \( \beta \), then the eigenvalues of \( A \) will lie in the disk \( \Phi_{\alpha \beta} \). For a special case when \( \beta = 0 \) and condition (3.2-4) still holds, then it is asserted that the all eigenvalues of \( A \) are the same and equal to \( \alpha \).
3.3 The robust pole assignment

In conventional pole assignment, we want to find a feedback matrix $G$ to place the eigenvalues of the closed-loop system matrix $A_c$ to a set of desired values. In this section, we will show how to modify the pole assignment technique to accommodate the multimode systems.

Let us suppose that the multimode system is working under $p$ operating conditions (points), which correspond to $p$ triplets of system models $(A_i, B_i, C_i, i=1,2,...,p)$. The design problem can now be stated as follows:

**Definition of problem:**

For given $p$ triplets of a multimode system, how can we find a feedback matrix $G$ that will place all of the eigenvalues of the closed-loop system matrices (3.3-1) into the desired disk specified by $\Phi_{ab}$?

$$A_c = A_i + B_i G \quad (i=1,2,...,p) \quad (3.3-1)$$

To facilitate the problem, the single input multimode system (3.3-2) will be studied first.

$$\dot{x} = A_i x + b_i u$$
$$u = g^T x + v \quad (3.3-2)$$

Here $g, b_i \in \mathbb{R}^n \quad (i=1,2,...,p)$.

To find an $n \times 1$ matrix $g$ for the pole assignment problem, the following theorem will be needed.

**Theorem 3.3-1**

If $A$ and $b$ are a controllable pair, then there exists a non singular transformation
matrix $T$ such that

$$\frac{2}{\beta} T^{-1}(A+bg^T+\alpha I)T = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_0 & a_1 & a_2 & \ldots & a_{n-1} \end{pmatrix} \quad (3.3-3)$$

$$T^{-1}b = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \quad (3.3-4)$$

Theorem 3.3-2

The eigenvalues of the closed-loop system matrix in (3.3-2) will lie in the disk $\Phi_{\alpha\beta}$ if the roots of the equation (3.3-5) are less than or equal to 4.

$$s^2 - (\xi+1)s+\alpha_0^2 = 0 \quad (3.3-5)$$

where $\xi = \sum_{i=0}^{i=n-1} \alpha_i^2$, and $\alpha_i$ $(i=0,1,\ldots,n-1)$ is given in (3.3-3).

Proof

Denote:

$$\tilde{A} = \frac{2}{\beta} (A+bg^T),$$

If

$$\lambda(A+bg^T) \in \Phi_{\alpha\beta}$$

$$\Leftrightarrow \lambda(\tilde{A}) \in \Phi_{\alpha_2}$$

$$\Leftrightarrow \lambda(T^{-1}\tilde{A}T) \in \Phi_{\alpha_2}$$

$$\Leftrightarrow \delta(T^{-1}\tilde{A}T+\alpha I) \leq 2$$

then

$$\delta(T^{-1}\tilde{A}T+\alpha I) = \delta(T^{-1}(\tilde{A}+\alpha I)T)$$
\[ \frac{1}{\lambda^2} \left[ (T^{-1}(\tilde{A} + \alpha I)T)(T^{-1}(\tilde{A} + \alpha I)T)^T \right], \]

(3.3-6)

\[
\det(sI-[T^{-1}(\tilde{A} + \alpha I)T][T^{-1}(\tilde{A} + \alpha I)T]^T]
\]

\begin{align*}
&= \det \left\{ sI - \begin{pmatrix}
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
a_0 & a_1 & a_2 & \ldots & a_{n-1}
\end{pmatrix}
\begin{pmatrix}
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
a_0 & a_1 & a_2 & \ldots & a_{n-1}
\end{pmatrix}^T \right\} \\
&= \det \left\{ sI - \begin{pmatrix}
1 & 0 & 0 & \ldots & a_1 \\
0 & 1 & 0 & \ldots & a_2 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
a_1 & a_2 & a_3 & \ldots & \xi
\end{pmatrix} \right\} \\
&= -a_1(-1)^{n+1} \det \left( \begin{pmatrix}
s-1 & 0 & 0 & \ldots & -a_1 \\
0 & s-1 & 0 & \ldots & -a_2 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & -a_{n-1}
\end{pmatrix} \right) \cdot
\begin{pmatrix}
0 & 0 & 0 & \ldots & -a_1 \\
0 & 0 & 0 & \ldots & -a_2 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & -a_{n-1}
\end{pmatrix}
\]

\begin{align*}
&= -a_1(-1)^{n+2} \det \begin{pmatrix}
s-1 & 0 & 0 & \ldots & 0 \\
0 & s-1 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & s-1
\end{pmatrix}
\]
\end{align*}

\begin{align*}
&= -a_1^2(s-1)^{n-2} - a_2^2(s-1)^{n-2} - \ldots - a_{n-1}^2(s-1)^{n-2} + (s-\xi)(s-1)^{n-1} \\
&= - \sum_{i=1}^{n-1} a_i^2(s-1)^{n-2} + (s-\xi)(s-1)^{n-1} \\
&= (s^2(\xi+1)s+\xi^2-2\xi a_0^2)(s-1)^{n-2}=0 \tag{3.3-7}
\end{align*}

If we denote the roots of (3.3-7) as \( \eta_i \), then by (3.3-6) and (3.3-7), we know that

\[ \eta_i^2 = \delta(T^{-1} \tilde{A}T + \alpha I) \]

Therefore, if

\[ \eta_i \leq 4 \quad (i=1, 2, \ldots, n) \]

then

\[ \delta(T^{-1} \tilde{A}T + \alpha I) \leq 2 \]
Knowing that (3.3-7) has \( n-2 \) roots equal to 1 and the roots of (3.3-5) are less than or equal to 4, we assert that
\[
\lambda(A+bg^T) \in \Phi_{a\beta}.
\]

By Theorem 3.3-2, we can tell whether or not \( \lambda(A) \in \Phi_{a\beta} \) by checking the roots of the second order polynomial equation (3.3-5).

The following theorem further simplifies the problem.

Theorem 3.3-3

If (3.3-8) holds, then the system of (3.3-2) will have the poles within the disk \( \Phi_{a\beta} \).

\[
\xi = \sum_{i=0}^{i=n-1} (a_i)^2 \leq 3 \tag{3.3-8}
\]

Proof

Since the roots of (3.3-7) can be written as
\[
\lambda_{1,2} = \frac{1}{2} \left\{ ((\xi+1)+[((\xi+1)^2-4a_0]^\frac{1}{2}) \right\}
\]
\[
\leq \frac{1}{2} \left\{ ((\xi+1)+[(\xi+1)^2-4a_0]^\frac{1}{2}) \right\} \tag{3.3-9}
\]
\[
\leq \frac{1}{2} \left\{ ((\xi+1)+[(\xi+1)^2]^\frac{1}{2}) \right\} \tag{3.3-10}
\]
\[
\leq \frac{1}{2} \left\{ ((\xi+1)+((\xi+1)) \right\} \tag{3.3-11}
\]
\[
= (\xi+1)
\]
\[
\leq 4
\]

Then, by Theorem 3.3-2, it is concluded that the poles of system (3.3-4) lie within the disk \( \Phi_{a\beta} \). < E.O.P >

If the multimode system (3.3-2) is controllable, then by Theorem 3.2-1 there is a transform matrix \( T_i \) such that
Furthermore, by Theorem 3.3-3, the closed-loop poles of (3.3-2) will be within the disk \( \Phi_{a,b} \) if we find a feedback matrix \( g \) in (3.3-12) so that \( a^i_j \) \((j=0, 1, \ldots, n-1)\) satisfies (3.3-13).\[
\xi_i = \sum_{j=0}^{n-1}(a^i_j)^2 \leq 3
\] (3.3-13)

Now, the pole assignment problem can be re-stated as: for given triplets \((A_i, b_i, C_i, i=1, 2, \ldots, p)\), how can we find a feedback matrix \( g \) so that \( \xi_i \) satisfies (3.3-13) for \( i=1, 2, \ldots, p \)? The easy way to make \( \xi_i \) satisfy (3.3-13) is to find a \( g \) matrix such that \( \sum_{i=1}^{p} \xi_i \) reaches its minimum value. To find such a \( g \) matrix, the following two theorems are proposed.

**Theorem 3.3-4**

The pole placement criterion function (3.3-14) will be minimized if feedback matrix is given as (3.3-15)

\[
J = \sum_{i=1}^{p} \xi_i
\] (3.3-14)

\[
g^T = \left( \sum_{i=1}^{p} \hat{A}_i T_i^T \right)^{-1} \left( \sum_{i=1}^{p} T_i T_i^T \right)^{-1}
\] (3.3-15)

where \( \xi_i \) is given by (3.3-13), and \( T_i \) and \( \hat{A}_i \) are specified in (3.3-16) and (3.3-17).\[
\frac{2}{\beta} T_i^{-1}(A_i + b_i g T_i + \alpha I) T_i = \begin{pmatrix}
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
\hat{A}_0 & \hat{A}_1 & \hat{A}_2 & \ldots & \hat{A}_{n-1}
\end{pmatrix}
\] (3.3-16)
\[ \hat{a}^i = (\hat{a}^i_0, \hat{a}^i_1, \hat{a}^i_2, \ldots, \hat{a}^i_{n-1}) \] (3.3-17)

Proof

Since
\[ T_i^{-1}b_i = (0, 0, \ldots, 0, 1)^T \]
\[ \Rightarrow \quad T_i^{-1}(A_i + \alpha I + b_i g^T)T_i = \begin{pmatrix} 0 & 0 & \vdots & 0 \\ 0 & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ g^T T_i + \hat{a}_i \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \] (3.3-18)
\[ \Rightarrow \quad \xi_i = (g^T T_i + \hat{a}_i)^T (g^T T_i + \hat{a}_i)^T \] (3.3-19)
\[ \Rightarrow \quad \sum_{i=1}^{i=p} \xi_i = \sum_{i=1}^{i=p} [g^T T_i + \hat{a}_i]^T [g^T T_i + \hat{a}_i]^T \] (3.3-20)
\[ \Rightarrow \quad \frac{\partial}{\partial g^T} \sum_{i=1}^{i=p} \xi_i = \sum_{i=1}^{i=p} (g^T T_i + \hat{a}_i)^T T_i = 0 \] (3.3-21)

and
\[ \frac{\partial^2}{(\partial g^T)^2} \sum_{i=1}^{i=p} \xi_i \geq \sum_{i=1}^{i=p} T_i T_i^T > 0 \] (3.3-22)

Therefore, by choosing \( g \) as the solution to (3.3-21), that is (3.3-15), the pole placement criterion function (3.3-14) will be minimized. <E.O.P.>

By Theorem 3.3-4, the feedback matrix which will minimize the sum of \( \xi_i \) can be obtained. It may be argued that it does not necessarily minimize every \( \xi_i \) and will not guarantee that (3.3-13) is satisfied. The fact is that, with the flexibility in selecting the parameters of \( g \), it will not be possible to minimize every \( \xi_i \). However, the following theorem does provide an opportunity to choose the feedback matrix to satisfy the condition (3.3-13).
Theorem 3.3-5

The weight-pole placement criterion function (3.3-23) will be minimized if the feedback matrix is given as (3.3-24).

\[
\psi = \sum_{i=1}^{i=p} \eta_i \xi^i \quad (3.3-23)
\]

\[
g^T = - \left( \sum_{i=1}^{i=p} \eta_i \xi^i \right)^{-1} \left( \sum_{i=1}^{i=p} \xi^i \right) \quad (3.3-24)
\]

Proof

The proof is very similar to that of Theorem 3.3-5. <E.O.P.>

In (3.3-23) and (3.3-24), changing weights \( \eta_i \) will result in changes of \( \xi^i \). It is understandable that if \( \eta_i \) increases, a decrease in \( \xi^i \) is expected, and vice versa. For small \( p \) values, we use (3.3-15) to compute \( g \). For large \( p \) value, we use (3.3-24) with suitable \( \eta_i \) values obtained by trial and error.

For the given \( p \) triplets \((A_i, b_i, C_i)\) and the desired pole disk \( \Phi_{\alpha\beta} \), the feedback matrix can be calculated by (3.3-15) or (3.3-24). This solves the pole assignment problem for the single-input multimode systems.

Now we will show how the above algorithm can be applied to the multi-input multimode system (3.3-25).

\[
\dot{x} = A_x x + B_x u \quad (3.3-25a)
\]

\[
u = G^T x + v \quad (3.3-25b)
\]

First, let us choose the feedback matrix \( G \) as

\[
G = f x g^T \quad (3.3-26)
\]

Then closed-loop system matrix can be written as

\[
A_{\text{lc}} = A_l + B_l f x g^T = A_l + b_l g^T \quad (3.3-27)
\]
where $b_i = Bf \in \mathbb{R}^n$. From (3.3-27), it is easy to see that $b_i$ acts as a pseudo-control matrix.

The system can be re-written as

$$\dot{x} = (A_i + b_i g^T)x + B_i v$$  \hspace{1cm} (3.3-28)

System (3.3-28) is in the same form as the single input system (3.3-2); therefore $g$ can be determined as if (3.3-28) is a single input system (under the assumption that $(A_i, b_i)$ is a controllable pair). Based upon this idea, the pole placement algorithm for multi-input multimode systems is given in Fig. 3.3-1.

---

**Fig. 3.3-1** The algorithm for multi-input multimode systems pole assignment.

In Fig. 3.3-1, first the maximum number of independent vectors $B_i f_1, B_i f_2, \ldots, B_i f_r$
(i=1,2,...,p) is calculated to remove redundancy. Then for a given $b_k = B_if_k$, the corresponding partial feedback matrix $g_k$ is calculated as if the system is a single input system. The eigenvalues of the resulting system are tested to determine whether the design goal has been achieved or not. If not, an additional partial feedback matrix is calculated by choosing $b = b^{k+1} = B_if_{k+1}$. Finally, the total feedback matrix is given as $G = \sum_{k=1}^{m} f_kg_k$ (m is the number of iterations in calculating the feedback matrix).

It is well understood that it is generally impossible to assign the poles simultaneously by a static feedback matrix for a single input system with multiple models. However, if we choose the desired pole as a disk, the simultaneous pole assignment for a single input system may be realized by a static feedback matrix, as shown by the following examples.

### 3.4 Examples

**Example 1**

The control of the longitudinal short period mode for an F5E fighter at two operating points is considered here\(^{38}\). The two models $\{A_1, b_1\}$ and $\{A_2, b_2\}$ are given as (3.4-1) and (3.4-2).

$$
A_1 = \begin{pmatrix}
-0.9896E+00 & 0.1741E+02 & 0.9625E+02 \\
-0.2648E+00 & -0.8512E+00 & -0.1139E+02 \\
0.0000E+00 & 0.0000E+00 & -0.2500E+03
\end{pmatrix} \tag{3.4-1a}
$$

$$
b_1 = \begin{pmatrix}
-0.9739E+00 \\
0.0000E+00 \\
0.2500E+02
\end{pmatrix} \tag{3.4-1b}
$$

$$
A_2 = \begin{pmatrix}
-0.6506E+00 & 0.1801E+02 & 0.8436E+02 \\
-0.8200E+00 & -0.6585E+00 & -0.1081E+02 \\
0.0000E+00 & 0.0000E+00 & -0.2500E+03
\end{pmatrix} \tag{3.4-2a}
$$

$$
b_2 = \begin{pmatrix}
-0.8810E+00 \\
0.0000E+00 \\
0.2500E+02
\end{pmatrix} \tag{3.4-2b}
$$

If we specify the desired disk as $\alpha = 12$ and $\beta = 6$, then by (3.3-15), the feedback
The matrix is given in (3.4-3).

\[ g = \begin{pmatrix} -0.170E-01 \\ -0.103E+00 \\ -0.883E+00 \end{pmatrix} \] (3.4-3)

The closed-loop system matrices are given in (3.4-4) and (3.4-5).

\[ A_2 = \begin{pmatrix} -0.265E+01 & 0.741E+01 & 0.103E+02 \\ -0.265E+00 & -0.851E+00 & -0.114E+02 \\ 0.426E+00 & 0.257E+00 & -0.294E+02 \end{pmatrix} \] (3.4-4)

\[ A_2 = \begin{pmatrix} -0.215E+01 & 0.897E+01 & 0.658E+01 \\ -0.820E-01 & -0.659E+00 & -0.108E+02 \\ 0.426E+01 & 0.257E+01 & -0.294E+02 \end{pmatrix} \] (3.4-5)

The closed-loop poles are:

\[ \{ \lambda(A_{c1}) \} = \{-14.4020, -9.2255 \pm 2.1228j\} \]

\[ \{ \lambda(A_{c2}) \} = \{-6.1245, -13.01917 \pm 2.2327j\}. \]

It is easy to verify that the resulting closed-loop system poles are within the specified disk.

Example 2

The dynamics of a helicopter in a vertical plane for an air speed of 60 to 170 knots is considered here\cite{39}. The two models \( \{A_1, b_1\} \) and \( \{A_2, b_2\} \) are given as (3.4-6) and (3.4-7).

\[ A_1 = \begin{pmatrix} -0.3660E-01 & 0.2710E-01 & 0.1880E-01 & -0.4555E+00 \\ 0.4820E-01 & -0.1010E+01 & 0.2400E-02 & -0.4021E+01 \\ 0.1002E+00 & 0.2855E+00 & -0.7070E+01 & 0.1323E+01 \\ -0.0000E+00 & -0.0000E+00 & 0.1000E+01 & -0.0000E+00 \end{pmatrix} \] (3.4-6a)

\[ b_1 = \begin{pmatrix} 0.4422E+00 & -0.5520E+01 \\ 0.1761E+00 & 0.4990E+01 \\ 0.3045E+01 & 0.0000E+00 \\ -0.7592E+01 & 0.0000E+00 \end{pmatrix} \] (3.4-6b)

\[ A_2 = \begin{pmatrix} -0.3660E-01 & 0.2710E-01 & 0.1880E-01 & -0.4555E+00 \\ 0.4820E-01 & -0.1010E+01 & 0.2400E-02 & -0.4021E+01 \\ 0.1002E+00 & 0.6630E-01 & -0.7070E+01 & 0.1198E+00 \\ -0.0000E+00 & -0.0000E+00 & 0.1000E+01 & -0.0000E+00 \end{pmatrix} \] (3.4-7a)
Let us specify the desired disk as $\alpha=12$ and $\beta=11$, and choose $f_1=(1,0)^T$ and $f_2=(1,0)^T$. Then by (3.3-15), the feedback matrices are given in (3.4-8) and (3.4-9) respectively.

$$g_1 = \begin{bmatrix} 0.3492E+03 \\ 0.1458E+01 \\ 0.2286E+02 \\ -0.3826E+02 \end{bmatrix}$$ (3.4-8)

$$g_2 = \begin{bmatrix} 0.1484E-02 \\ 0.8064E-01 \\ -0.1749E-01 \\ -0.4735E-02 \end{bmatrix}$$ (3.4-9)

The resulting closed-loop system poles are:

$$\lambda \{ A_{c_1} \} = \lambda \{ A_1 - B_1 f_1 g_1^T \} = \{-19.4117, -10.9211, -1.9706 \pm 0.2689j\}$$

$$\lambda \{ A_{c_2} \} = \lambda \{ A_2 - B_2 f_2 g_2^T \} = \{-17.9850, -6.0146 \mp 3.1208j, -1.2462\}$$

The total feedback matrix is

$$G = (f_1 g_1^T + f_2 g_2^T)^T$$

$$= \begin{bmatrix} 0.3492E+03 & 0.1484E-02 \\ 0.1458E+01 & -0.8064E-02 \\ 0.2286E+02 & -0.1749E-01 \\ -0.3826E+02 & -0.4735E-02 \end{bmatrix}$$

The resulting closed-loop poles are within the specified disk. These examples illustrate how to use this pole assignment algorithm in single input and multi-input multimode system.
3.5 Conclusions

In this chapter, we studied a robust pole assignment algorithm for multimode systems. In order to have more flexibility to improve the robustness of the systems, the desired pole position was specified as a region. Using the relationship between the eigenvalues and singular-values of a matrix, the sufficient conditions for pole assignability by a static feedback matrix is given. However, the necessary conditions for pole assignability are still under investigation.
Chapter IV

Fuzzy Set Theory and Control

4.1 Introduction

Fuzzy set theory enables us to handle inexact, vague data and yet to work in a mathematically strict and rigorous way. It was first advocated by Zadeh in 1965⁴⁰, and since then the concept has been studied⁴¹ and applied to many complex problems by a number of researchers. In this chapter, some basic concepts of fuzzy sets, fuzzy logic, and fuzzy control are presented. These concepts and interpretations were originally presented in references⁴²⁴³. The next section of this chapter reviews the concepts of fuzzy sets and fuzzy logic. The third section is dedicated to how fuzzy logic is applied to control systems.

4.2 Fuzzy set theory

4.2.1 Fuzzy sets

In ordinary set theory, any finite or infinite number of elements with some common properties is a set by definition. A membership function of a set is defined as

\[ f_{A}(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases} \] ⁴.2-1

In fuzzy set theory, the membership function is generalized to take any value in the closed interval [0, 1]. Elements of a fuzzy set will have different grades of membership. Fuzzy set theory, therefore, can be said to be a generalization of conventional set theory. A more vigorous definition of fuzzy set is given next.

Definition 4.2-1

Let \( X \) be the universe of discourse. Fuzzy set \( A \) on \( X \) is defined by its membership function \( \mu_{A}(x) \), which assigns to each element \( x \) in \( X \) a real number in the interval [0, 1].
The value of $\mu_A(x)$ represents the grade of membership of $x$ in $A$. Fuzzy set $A$ is a map from the universe of discourse $X$ into the closed interval $[0, 1]$ by its membership function $\mu_A(x)$; i.e.,

$$\mu_A(x): X \rightarrow [0, 1]$$ \hfill (4.2-2)

To clarify the difference between an ordinary set and a fuzzy set, let us consider the following example. Let $X$ be the universe of discourse whose elements are humans. Let $A$ be a set (in the conventional sense) of "Females".

As depicted in Fig. 4.2-1, elements belonging to the set "Females" are clearly differentiable from the non-elements (male). Let us consider another set "Young women". If the set "Young women" is constructed in a similar way as the set "Females" does, the membership function of the set "Young women" will be as shown in Fig. 4.2-2a. However, there is a question about the membership of the set since it is no longer clear.
who belongs to the set and who does not. This ambiguity of the membership is caused by the word language, which is blindly defined to be the ages between 20 and 30 in Fig. 4.2a. In natural language, which we all use in daily life, such simple words as "Young women" may be interpreted in many ways, and yet understood perfectly by almost everyone. Here, to understand means to know the intention of the usage by a speaker. Hence, describing the membership of elements by either 0 or 1 is not only an inadequate means of interpreting the true meaning, but it may also distort the information which ought to be carried by the words. A membership function such as that shown in Fig. 4.2-2b implies a more relaxed definition of the words "Young women" and appears more adequate in representing the meaning of the words. If a set has such a membership function, the set becomes a fuzzy set.

![Fig. 4-2-2 Membership functions for the set "Young women"](image)

The words "Young women" is blindly defined to be the ages between 20 and 30 in (a); however, a relaxed definition such as that depicted in (b) more adequately represents the meaning of the words. If a set is defined by a membership function in (b), the set is called a fuzzy set.

4.2.2 Fuzzy set operations

Having defined fuzzy sets, we can now define relationships between two or more fuzzy sets. As a set is uniquely determined by its membership function (and vise versa), the
relationships between and the operations on any sets are uniquely given by that of the corresponding membership functions.

Definition 4.2-2

Let \( x \) be elements of the universe of discourse \( X \). The union of two fuzzy sets \( A \) and \( B \) on \( X \), denoted \( A \cup B \), is defined by (4.2-3),

\[
\mu_{A \cup B}(x) = \mu_A(x) \cup \mu_B(x) = \max \{ \mu_A(x), \mu_B(x) \}.
\] (4.2-3)

Definition 4.2-3

Let \( x \) be elements of the universe of discourse \( X \). The intersection of two fuzzy sets \( A \) and \( B \) on \( X \), denoted \( A \cap B \), is defined by (4.2-4),

\[
\mu_{A \cap B}(x) = \mu_A(x) \cap \mu_B(x) = \min \{ \mu_A(x), \mu_B(x) \}.
\] (4.2-4)

Definition 4.2-4

Let \( x \) be elements of the universe of discourse \( X \). The complement \( A' \) of fuzzy set \( A \) is defined by (4.2-5),

\[
\mu_{A'}(x) = 1 - \mu_A(x).
\] (4.2-5)

Based upon the above definitions, the following properties hold for fuzzy sets \( A, B, \) and \( C \),

1. commutativity: \( A \cup B = B \cup A, A \cap B = B \cap A \).
2. associativity: \( (A \cup B) \cup C = A \cup (B \cup C), (A \cap B) \cap C = A \cap (B \cap C) \).
3. idempotency: \( A \cup A = A, A \cap A = A \).
4. absorption: \( A \cup (A \cap B) = A, A \cap (A \cup B) = A \).
5. distributivity: \( A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \),
\[ A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \)
4.2.3 Fuzzy decision making

In this section, we will discuss a method of decision making, called direct fuzzy logic, which is based upon fuzzy information. Direct fuzzy logic is based upon the assumption that each membership value for all the fuzzy sets is the true value. Consider the following conditions and inference:

Condition 1: If \( x \) is \( A \), then \( y \) is \( B \),
Condition 2: \( x' \) is \( A' \),
inference: \( y' \) is \( B' \).

Let \( R \) be a fuzzy relationship between \( A \) and \( B \), which can be written as

\[
R = A \rightarrow B = A \times B \tag{4.2-6}
\]

which is denoted as the cross product of two fuzzy sets. The membership function of \( R \) is obtained by a minimization operator \( \land \),

\[
\mu_R(x, y) = \mu_A(x) \land \mu_B(y) \tag{4.2-7}
\]

Then if a fuzzy set \( A' \) and a fuzzy relationship \( R \) are given, a fuzzy set \( B' \) can be obtained by a fuzzy composition, denoted by 'o',

\[
B' = A' \circ R \tag{4.2-8}
\]

and the membership function is

\[
\mu_{B'}(y) = \vee \{ \mu_{A'}(x) \land \mu_R(x, y) \}
= \vee \{ \mu_{A'}(x) \land \mu_A(x) \land \mu_B(y) \} \tag{4.2-9}
\]

where \( \vee \) is a maximization operator. The process is depicted in Fig. 4.2-3.
Fig. 4.2-3 Fuzzy input-output relation. The diagram indicates the methodology for obtaining a fuzzy output, B', when a fuzzy input, A', is given and a fuzzy relationship, R, between the input and output of the system is known.

In a physical system, a non-fuzzy value $x_0$ is most often used as an input in place of fuzzy input $A'$. This non-fuzzy input can be represented as a fuzzy input forming a membership function

$$
\mu_{A'}(x) = \delta(x-x_0)
$$

(4.2-10)

where

$$
\delta(x)=\begin{cases} 
0 & \text{otherwise} \\
1 & \text{if } x=x_0 
\end{cases}
$$

(4.2-11)

In this case, (4.2-9) can be simplified to

$$
\mu_{B'}(y)=\cup\{ \mu_{A'}(x) \land \mu_A(x) \land \mu_B(y) \}
$$

$$
=\cup\{ \delta(x-x_0) \land \mu_A(x) \land \mu_B(y) \}
$$

$$
=\mu_A(x_0) \land \mu_B(y)
$$

$$
= R(x_0, y)
$$

(4.2-12)

4.3. Fuzzy Control

4.3.1 Application of fuzzy sets to control systems

The theory of fuzzy sets has been applied to a wide variety of areas, and fuzzy control is one of them. Fuzzy control is a strategy in which knowledge and experience of human control operators is expressed in IF-THEN linguistic rules.

The idea of fuzzy control has gained wide popularity since a conceptual framework
was introduced by Zadeh in 1973\cite{39} and Mamdani pioneering work on the first application report in 1974\cite{40}. In Mamdani's work, a fuzzy controller was designed for a small laboratory steam engine to regulate the speed and the boiler pressure by adjusting the throttle and the boiler heat. The performance of the controller was found to be better than a well tuned PI controller. A number of other applications have been reported in references\cite{44}-\cite{50}.

4.3.2 Fuzzy control structure

A typical closed-loop control system consists of basically four elements: a plant to be controlled, sensing devices which are used to monitor the state of the plant, actuators, and a controller. In manual control systems, a human operator acts as a feedback controller. In the case of single-input-single-output (SISO) systems, the controller (operator) often uses the difference between a sensor output and a target value (call this difference an "ERROR") and change of the error (call this value a "SLOPE") which are recognized by an operator through monitoring devices. Given these two values, ERROR and SLOPE, appropriate control actions are taken by changing control variable(s) (call this action a "CHANGE") based upon the operator's knowledge and experience. The control decision depends upon appropriate interpretation of inputs (ERROR and SLOPE), which are assumed to represent the state of the plant. The process of decision making is a highly complex process. Knowledge and experience about a particular plant enable the operator to perform control tasks successfully. However, reconstruction of such a process in mathematical equations seems difficult, if not impossible. Nevertheless, this human control protocol is the basis of the fuzzy controller structure. A typical fuzzy controller, as shown in Fig. 4.3-1, consists of the following three elements: 1) a fuzzifier which converts the physical inputs into the fuzzy variables by the corresponding membership functions, 2) an inference engine with a decision rule base, and 3) a de-fuzzifier which converts the fuzzy control decisions into the crisp non-fuzzy (i.e. physical) control signals.
1. Fuzzifier

Let "ERROR" be a fuzzy variable which is a transformed variable from a non-fuzzy variable $e(t)$ by fuzzy sets, LP, MP, SP, ZP, Z, ZN, SN, MN, and LN, where

- LP: large positive
- MP: medium positive
- SP: small positive
- ZP: zero positive
- Z: zero
- ZN: zero negative
- SN: small negative
- MN: medium negative
- LN: large negative

and

$$e(t) = y_d - y(t)$$ \hspace{1cm} (4.3-1)

where $y(t)$ is an output of the system at time t, and $y_d$ is the corresponding desired output value. Similarly, other variables, such as SLOPE, can be fuzzy variables which are transformed from non-fuzzy variables, such as $s(t)$, by the aforementioned fuzzy sets, and $s(t)$ is defined by
\[ s(t) = (y(t) - y(t-T))/T \] (4.3-2)

where \( y(t-T) \) is the output value which is measured at one sampling interval before \( t \), and \( T \) is the sampling interval in seconds.

Let us further define a new fuzzy variable "CHANGE", which is transformed from a non-fuzzy variable \( c(t) \), where \( c(t) \) is a control actuation value change, i.e.

\[ u(t) = u(t-T) + c(t) \] (4.3-3)

where \( u(t) \) is a controller output, \( u(t-T) \) is a controller output at one sampling interval before, and \( c(t) \) is a change in control output. The fuzzy variable CHANGE belongs to fuzzy sets, LPC, MPC, SPC, ZPC, ZC, ZNC, SNC, MNC, and LNC, where

- LPC: large positive change
- MPC: medium positive change
- SPC: small positive change
- ZPC: zero positive change
- ZC: zero change
- ZNC: zero negative change
- SNC: small negative change
- MNC: medium negative change
- LNC: large negative change

The variables ERROR, SLOPE, and CHANGE have their membership values for each of the fuzzy sets. The membership functions can have many forms as shown in Fig. 4.3-2.
In this study, we use a triangular function of the form,

\[
\mu(x) = \begin{cases} 
\frac{x-a}{b-a} & \text{if } a<x<b \\
\frac{x-c}{b-c} & \text{if } b<x<c \\
0 & \text{if } x<a \text{ or } x>c
\end{cases}
\]

where \( x \) is an input, and \( a, b, \) and \( c \) are some real numbers satisfying the condition \( a<b<c \).

By appropriately varying the values of \( a, b, \) and \( c \), fuzziness of the physical variable, \( x \), can be described in different ways. The choice of \( a, b, \) and \( c \)'s values depend upon the practical situation.
2. Control decision rules and fuzzy logic

Control decision rules in a fuzzy controller are represented by a group of IF-THEN type statements, IF<condition>, THEN <action>. A typical heuristic control rule may be written as follows:

If e(t) is large negative (LN) and s(t) is large positive (LP), then control output change, c(t), is large positive (LPC).

The whole rule set can be represented in the following general form:

Rule 1: IF e(t) is E_1 and s(t) is S_1, THEN c(t) is C_1
Rule 2: IF e(t) is E_2 and s(t) is S_2, THEN c(t) is C_2

Rule n: IF e(t) is E_n and s(t) is S_n, THEN c(t) is C_n

Here, e(t) is an error as defined in (4.3-1), s(t) is a slope as defined in (4.3-2), and c(t) is a control actuation value change as defined in (4.3-3). E_i, S_i, and C_i (i=1,2,...n) are fuzzy sets, specified by corresponding membership functions. Applying the inputs (ERROR and SLOPE) as conditions to the decision rules yields a set of actions (CHANGE) with various degree of membership in the fuzzy sets. These membership values are obtained by an inference engine: fuzzy logic.

Let the relationship among fuzzy sets of the i-th rule, R_i, be expressed as

\[ R_i = (E_i \times S_i) \times C_i \]  \hspace{1cm} (4.3-5)

where \( \times \) is a cross product. Then its membership function can be re-written as

\[ \mu_{R_i}(e,s,c) = \left\{ \mu_{E_i}(e) \land \mu_{S_i}(s) \land \mu_{C_i}(c) \right\} \]  \hspace{1cm} (4.3-6)

where \( \land \) is defined as a minimization operation (or as a multiplication operation). R is called a fuzzy relationship, and the fuzzy relationship of the rules, R, can be written as

\[ R = R_1 \cup R_2 \cup ... \cup R_N \]  \hspace{1cm} (4.3-7)

where \( \cup \) is a union of the fuzzy sets.
Now, suppose there are fuzzy sets $E_0$ and $S_0$ as inputs to the system, then an output of the inference, $C_0$, is given as

$$C_0 = (E_0 \times S_0) \circ R$$

(4.3-8)

where $\circ$ means a fuzzy composition, $(E_0 \times S_0)$ is a cross production of $E_0$ and $S_0$, and its membership function is defined as

$$\mu_{E_0 \times S_0}(e, s) = \mu_{E_0}(e) \land \mu_{S_0}(s).$$

If non-fuzzy values, $e_0$ and $s_0$, are given as inputs, then the membership function of an output $c_0$ is given as

$$\mu_{C_0}(c) = \mu_{C_1}(e_0, s_0, c) \lor \mu_{C_2}(e_0, s_0, c) \lor \ldots \lor \mu_{C_n}(e_0, s_0, c)$$

$$= [\mu_{E_1}(e_0) \land \mu_{S_1}(s_0) \land \mu_{C_1}(c)] \lor [\mu_{E_2}(e_0) \land \mu_{S_2}(s_0) \land \mu_{C_2}(c)] \lor \ldots$$

$$\lor [\mu_{E_n}(e_0) \land \mu_{S_n}(s_0) \land \mu_{C_n}(c)]$$

$$= [w_1 \land \mu_{C_1}(c)] \lor [w_2 \land \mu_{C_2}(c)] \lor \ldots \lor [w_n \land \mu_{C_n}(c)]$$

(4.3-9)

where

$$w_i = \mu_{E_i}(e_0) \land \mu_{S_i}(s_0), \quad i = 1, 2, \ldots, n$$

(4.3-10)

and $\lor$ is a maximization operator.

Since the operation is based upon maximization and minimization operations, this fuzzy composition is often called a "max-min" composition. Equation 4.3-10 can be interpreted as an indication of a matching level between inputs $e$ and $s$ and the IF part of the $i$-th rule. The inference output, $C$, expressed in terms of fuzzy sets, is obtained from the above process. It is not possible, however, to directly feed $C$ into a system as a control actuation value without first transforming it into a non-fuzzy value (de-fuzzification). The process of de-fuzzification is discussed next.

3. De-fuzzifier

The outcomes from the decision rules must now be transformed into non-fuzzy variables. This process is called de-fuzzification. An easy way to do so is to find a membership function for CHANGE that has a maximum membership value. Taking an
inverse of the function gives a non-fuzzy value \( c_0(t) \), i.e.,

\[
c_0(t) = \mu_{C_k}^{-1}(w_k)
\]

(4.3-11)

where \( k \) is such that

\[
w_k = \max_{i=1 \ldots n} \{w_i\}
\]

(4.3-12)

and \( c_k \) is the \( k \)-th rule.

The other way to de-fuzzify is to compute an average value of non-zero entries as follows:

\[
c_0(t) = \sum_{i=k}^{i=k} \frac{c_i}{n}
\]

(4.3-13)

where

\[
c_i = \mu_{C_i}^{-1}(w_i)
\]

(4.3-14)

\[
n = \sum_{i=1}^{i=k} j
\]

(4.3-15)

\[
j = \begin{cases} 1 & \text{if } w_i = 1 \\ 0 & \text{if } w_i = 0 \end{cases}
\]

Since the latter de-fuzzification approach is much simpler in manipulation than the first one, it is very popular in practice.
Chapter V

Hierarchical Fuzzy Control

5.1 Introduction

Since the pioneering works of Zadeh [40][50] and Mamdani and Assilian [41][51], fuzzy logic control has been successfully implemented in many industrial applications [52]-[55].

In most rule-based fuzzy controllers, the output of the controller is obtained by consulting the rule set, which is based upon system variables such as error signals and their rate. In this chapter, a complete rule set is defined - which means that the rule set will be valid for all conditions (contingencies). As the number of system variables increases, the number of rules in a conventional complete rule set increases exponentially. This requires the computer to process a huge data base, and is often accompanied by memory overload and longer computational time.

To make the problem manageable, the concept of a 'hierarchical rule set' is proposed. In a hierarchical structured rule basis, the number of rules increases linearly (not exponentially) with the number of system variables. This makes it possible to apply a fuzzy rule-based controller to large-scale systems.

The remainder of the chapter is organized as follows: in section 5.2, different kinds of membership functions of fuzzy sets, \( \alpha \)-cut sets of fuzzy sets, and a basic fuzzy logic controller are illustrated. A hierarchical structure is developed in section 5.3. In section 5.4, the hierarchical structure is applied to control the feedwater flow of a steam generator of a power plant. The simulation results are shown in section 5.5, and a summary of the contributions of this chapter is given in section 5.6.
5.2 Fuzzy logic controller

Unlike a regular analytic controller, a fuzzy rule-based controller is made of rules that certain actions be initiated should a specific set of conditions occur.

As mentioned in chapter IV, in conventional rule set R, the rule j is of the form (j=1,2,.....n):

\[
\text{if}(x_1 \text{ is } a_{1j}, x_2 \text{ is } a_{2j}, \ldots, x_n \text{ is } a_{nj}), \text{ then } (\text{output is } b_j) \tag{5.2-1}
\]

where \( x_1, x_2, x_3, \ldots, x_n \) are some of the system variables, \( a_{ij} \in A \) (i=1, n, j=1, k), \( b_j \in B \) (j=1, k), and A and B are the sets whose elements are fuzzy sets and could be the same. For example A=B={lp, mp, sp, zp, z, zn, sn, mn, ln}, where lp, mp, sp, zp, z, zn, sn, mn and ln are fuzzy sets, which mean respectively large positive, medium positive, small positive, zero positive, zero, zero negative, small negative, medium negative and large negative sets.

The membership functions of the fuzzy sets \( a_{ij} \) and \( b_j \) can be defined in many ways. Fig. (4.3-2) shows two kinds of membership functions which are often used in practice. The choice of membership functions depends on the practical situation.

Let the membership function of fuzzy set \( a_{ij} \) be denoted as \( u_{a_{ij}}(x) \). Then by the fuzzy rule in (5.2-1), the fuzzy controller's output can be calculated as

\[
u_j = u_{a_j}(X) * y_j \tag{5.2-2}
\]

where \( y_j \) satisfies (5.2-3)

\[
u_{b_j}(y_j) = 1. \tag{5.2-3}
\]

and
\[ u_{a_j}(X) = \min u_{a_{ij}}(x_i) \quad \text{for} \quad i = 1, \ldots, n \quad (5.2-4) \]

The output yielded by the whole set of rules is given by

\[ u_0 = \frac{\sum_{i=1}^{n} u_{a_i} \cdot y_i}{\sum_{i=1}^{n} u_{b_i}} \quad (5.2-5) \]

In (5.2-5), all of the rules are used in calculating the output \( u_0 \). This makes the calculation complex as well as time consuming. In fact, not all of the rules are of the same importance. Some rules, which yield larger weights, are most applicable for a given input (contingency) and the remaining, which yield smaller weights, only bring noise to the calculation of output \( u_0 \). The concept of \( \alpha \)-cut set is used in the study to select suitable rules in the calculation of output \( u_0 \).

Definition (5.2-1) \[ ^{56} \]

The \( \alpha \)-cut set of a fuzzy set, \( a \), is defined as

\[ \hat{a} = \{ x \mid u_a(x) > \alpha \} \quad (5.2-6) \]

where \( \alpha \) is a real number.

For some kinds of membership functions (like the exponential type), system variables belong to all of the fuzzy sets in the sense that \( u_{a_{ij}}(x) > 0 \) for all \( i, j = 1, 2, \ldots, n \). But they only relate to a limited number of their \( \alpha \)-cut sets, if \( \alpha > 0 \), as shown in Fig. 5.2-1. By substituting \( a_{ij} \) in (5.2-4) with its \( \alpha \)-cut set, the total number of rules used in calculating the output will be reduced. This is because the system variables (inputs \( x_i, i = 1, 2, \ldots, n \)) belong only to a few \( \alpha \)-cut sets and many fuzzy rule conditions will no longer be satisfied.
In non-fuzzy (analytic) controller design, using the plant (system) model and system performance criteria, a control algorithm is developed to produce the desired output for all inputs (contingencies). In fuzzy controller design, use is made of rules that designate that certain actions be taken should a specific set of conditions occur. To take care of all of the conditions (contingencies), the rule set must be complete. This issue is addressed below with the use of cover set concept.

Let \( F = \{ln, mn, sn, zn, z, zp, sp, mp, lp\} \). Then the concept of cover set of \( F \) can be defined as
Definition 5.2-2

The cover set of $F$ is a fuzzy set denoted by $\Psi$, with the membership function,

$$u_\Psi(x) = \text{Max } u_\alpha(x), \quad \forall \alpha \in F$$ (5.2-7)

Fig. (5.2-2) shows the membership functions of the cover set of $F$ for two kinds of membership functions.

Fig. 5.2-2 The membership functions of the cover sets;

a. derived from fuzzy sets of triangular type membership functions,

b. derived from fuzzy sets of exponential type membership functions.

With the definition of a cover set, the system variables and the fuzzy sets can be related. Theorem 5.2-1 clarifies this relationship.

Theorem 5.2-1

For each $X=(x_1, x_2, x_3, \ldots, x_n)$, there exists a set $\Phi=(s_1, s_2, s_3, \ldots, s_n)$, which
satisfies
\[ u_{S_j}(x_i) \geq u_{a_j}(x_i) \quad j, i=1, 2, 3, \ldots, n \]  \hspace{1cm} (5.2-8)

where \( a_j \in F, s_i \in \Phi \ j, i=1, 2, 3, \ldots, n \)

Proof

The proof is obvious if we choose \( u_{S_j}(x_i) = u_{\phi}(x_i) \)

To specify a complete rule set, let us denote

\[ FA = \{ \Phi \mid \Phi = (s_1, s_2, s_3, \ldots, s_n), s_i \in F, i=1, \ldots, n \} \]  \hspace{1cm} (5.2-9)

then we have the following definition.

Definition 5.2-3.

The set of rules, \( R \), is said to be complete, if there is an onto and into map between \( FA \) and \( R \).

This map can be easily found by corresponding \( s_i \) in (5.2-8) with \( a_{kj} \) in (5.2-1).

A complete set of rules has the following property.

Theorem 5.2-2

A complete set of rules has \( m^n \) different rules, where \( m \) is the number of fuzzy sets in \( F \) and \( n \) is the number of system variables in a rule.

Proof

For \( n=1 \), it is obvious that the rule set has \( m \) different rules, so the Theorem is true. Suppose that the Theorem is true for \( n=n_1 \).

Let \( n=n_1+1 \). Then for every rule with \( n_1 \) variables, there corresponds \( m \) different rules if we add one variable. In other words, each rule with \( n_1 \) variables (there are \( m^{n_1} \) such rules) will change into \( m \) rules with \( n_1+1 \) variables. Therefore, the total number of rules \( k \) is
Example 5.2-1

Let

\[ F = \{ l_1, s_n, z, s_p, l_p \} \]
\[ X = (x_1, x_2, x_3, x_4) \]

So \( n = 4 \), \( m = 5 \), and the number of the rules for the complete set is given by

\[ k = m^4 = 6^4 = 1296 \]

If \( n = 5 \) and \( m = 5 \), then \( k = 5^5 = 5^5 = 8125 \). We see that even for some small values of \( n \) and \( m \), the number of rules in a complete rule set is very large for a computer to handle in a real time control situation.

From (5.2-10), it is clear that the total number of rules is an exponential function of the number of system variables. It means that the number of rules increases rapidly as the number of system variables increases. For a multidimensional system (i.e. \( n \) is large), it may become unproductive to realize a rule-based fuzzy controller. To overcome this problem, a hierarchical fuzzy control structure is proposed in the next section.

5.3 Hierarchical fuzzy controller

Hierarchical structure is used most effectively in the control of large-scale systems. In fuzzy control, the hierarchy is also effective in structuring the rules to make the fuzzy controller suitable for relatively large systems.

In this study, we form the rule sets in a hierarchical way, such that the most influential parameters are chosen as the system variables in the first level, the next most important parameters are chosen as the system variables in the second level, and so on.

The rules in the first level are of following form:
The rules in the ith (i>1) are of the form:

if \( (x_{N+1} \text{ is } a_{N+1,1}, \ldots, x_{N+n} \text{ is } a_{N+n,1}, \text{ and } y_{i-1} \text{ is } b_{i-1}) \), then (output \( y_i \) is \( b_i \))

(5.3-1)

where \( N_i = \sum_{j=1}^{i-1} n_j \leq n \), \( n_j \) = the number of system variables used in the jth level, \( x_j \) (j=1, 2, \ldots, n) is the system variable, and \( y_i \) (i=1,2,\ldots,l-1) is the ith level output used in level \( (i+1) \) as the input variable.

In the hierarchy, the first level gives an approximate output \( y_1 \), which is then modified by the second level rule set. The second level variables include the approximate output \( y_1 \) of the first level and system variables as shown in (5.3-2). This process is repeated in succeeding levels of the hierarchy.

The flow diagram of the hierarchy is shown in Fig. 5.3-1.

At each ith level (i>1) of Fig. 5.3-1, one or more system variables may be considered in addition to the output of the previous level in the development of rule set. But it will be shown later that the total number of rules will be minimized if only one system variable is added in each succeeding level. If we put all variables in the first level, the structure is the same as the conventional one. This means the conventional rule-based fuzzy controller is a special case of the hierarchical one.
Theorem 5.3-1

For the hierarchical structure containing \( n \) system variables, if \( L \) is the number of hierarchical levels, and \( n_i \) is the number of input variables used in the \( i \)th level including output variable of the \((i-1)\)th level (if \( i > 1 \)), the total number of rules (for complete set) \( k \) is:

\[
k = \sum_{i=1}^{L} m^{n_i}
\]

(5.3-3)

where \( m \) is the number of fuzzy sets in \( F \), and
\[ n_1 + \sum_{i=2}^{L} (n_i - 1) = n \]  \hspace{1cm} (5.3-4)

Proof

In level \( i \), since there are \( n_i \) variables, by Theorem 5.2-2, the total number of rules in this level is \( m^{n_i} \). This is true for all other levels. The total number of rules is equal to the sum of the rules in each level, so

\[ k = \sum_{i=1}^{L} m^{n_i} \]

which proves Theorem 5.3-1.

If \( n_i = t = \) a constant integer \( (i=1, 2, \ldots, L) \), then from (5.3-4),

\[ L = \frac{n-t}{t-1} + 1 \]

So

\[ k = \left( \frac{n-t}{t-1} + 1 \right) \cdot m^t \]

The above equation means that by the hierarchical structure, the number of rules in a complete set is reduced to a linear function of the number of system variables \( n \), instead of an exponential function of \( n \) as in the conventional case.

Theorem 5.3-2

In the hierarchical structure containing \( n \) system variables, if \( m \geq 2 \) and \( n_i \geq 2 \) (\( m, n_i \) are defined as in Theorem 5.3-1), the total number of rules in a complete rule set will reach its minimum value when \( n_i = t = 2 \), and its maximum value when \( n_i = n_1 = n \).

Proof

Suppose level \( i \) has \( n_i \) input variables, and let \( n_i \geq 3 \). By Theorem 5.2-2, the number of rules in that level is \( m^{n_i} \).
If we split that level into two levels which use 2 and \( n_{i-1} \) input variables respectively, then the total number of rules \( k_{i2} \) in those two levels is

\[
k_{i2} = m^2 + m^{n_{i-1}}
\]

(5.3-5)

since \( n_i \geq 3 \) and \( n_{i-1} \geq 2 \).

So

\[
m^2 \leq m^{n_{i-1}}
\]

and

\[
m^2 + m^{n_{i-1}} \leq m^{n_{i-1}} + m^{n_{i-1}} = 2m^{n_{i-1}}
\]

(5.3-6)

Since we also assumed \( m \geq 2 \), (5.3-7) holds true

\[
2m^{n_{i-1}} \leq m^{n_i}
\]

(5.3-7)

From (5.3-5), (5.3-6), and (5.3-7), we have

\[
m^2 + m^{n_{i-1}} \leq m^{n_i}
\]

It follows that the total number of rules in the hierarchical structure will be reduced if we split any level having three or more input variables into two levels, one of which has two input variables. Repeating the process for all levels, we assert that the total number of rules reaches its minimum value if every level has only two input variables.

Now let us combine any two arbitrary levels with system variables \( n_i \) and \( n_j \) into one level. Then the total number of rules \( k_{ij} \) in this level is:

\[
k_{ij} = m^{n_i + n_{j-1}}
\]

(5.3-8)

Without loss of generality, let

\[
n_i \geq n_j
\]

Then

\[
m^{n_i + n_j} \leq m^{n_i} + m^{n_i} = 2m^{n_i}
\]

(5.3-9)

Again, since

\[
n_j \geq 2, \quad n_{j-1} \geq 1, \quad \text{and} \quad m \geq 2,
\]
Then
\[ 2^*m_i^n \leq m^*m_i^n \leq m^{n_i+n_j-1} \]  
\[ (5.3-10) \]
From (5.3-8), (5.3-9), and (5.3-10), it is clear that
\[ m^{n_i+m_j^n} \leq m^{n_i+n_j-1} = k_{ij} \]
So we conclude that the total number of rules will be increased if we combine any two levels into one level. Using this result repeatedly, we can assert that the total number of rules will reach its maximum value if \( n_i = n_1 = n \). The above proves Theorem 5.3-2.

To illustrate the above results further, let us consider example 5.3-1.

Example 5.3-1

For comparison, let us consider the same problem as in example 5.2-1, i.e., \( n=4 \) and \( m=5 \), and choosing \( n_1 = t=2 \), we assert by Theorem 5.3-1 that
\[ L=3 \quad \text{and} \quad k=3*m^1=3*5=15 \]
Comparing this with the result of example 5.2-1, we see that 625 rules are needed in the non-hierarchical case, and only 75 rules are needed for the hierarchical case.

It can be seen that by using a hierarchical structure, the number of rules is greatly reduced. Furthermore, Theorem 5.3-2 states that to minimize the total number of rules, only two variables should be chosen at each level. By virtue of hierarchical structure, it is also very easy to add or subtract one variable without altering other rules in the set. Finally, it should be pointed out that the process followed in hierarchical structure is very similar to that used by an operator. The operator makes the tentative decision based upon some important parameters, and then later he or she modifies it when more information (parameters) becomes available. This is the same procedure that we have followed in this hierarchical structure, which is convenient for formulating a fuzzy rule set using the operator's experience.
5.4 Hierarchical fuzzy controller for a steam generator

In this section, the hierarchical fuzzy control method developed earlier is used to control the feedwater flow of a steam drum. The configuration of the steam drum is shown in Fig. 5.4-1.

![Diagram of steam drum configuration](image)

The main goal of the controller is to maintain the water level in the steam drum at a desired value. The dynamic model for the steam drum has 18 state variables. Based upon four of these variables, a hierarchical fuzzy rule set is constructed. The closed-loop model of the system with the fuzzy controller is shown in Fig. 5.4-2.

The dynamic model of the drum is given by (5.4-1)

$$\frac{dx}{dt} = A \times x + B_d \times u_d + B_0 \times u_0$$  \hspace{1cm} (5.4-1)

where $x$ is the system state vector, $A$ is the system matrix, $B_d$ is the disturbance input matrix, $u_d$ is the disturbance input (unit step function), $B_0$ is the input matrix, and $u_0$ is the input obtained from the fuzzy controller. The details of the model are given in reference [8].
In this study, all system variables and output are normalized so that they can be represented in the interval of [-1,1], which is called a universe of discourse. By doing so, all the variables can be compared relatively with some universal concepts such as small, large, medium, etc. The fuzzy sets lp, mp, sp, zp, z, zn, sn, mn and ln, whose membership functions are given in Fig. 4.3-2a, are also based upon this universe of discourse.

The hierarchical structure used here, as shown in Fig. 5.4-2, has two levels. In the first level, the level of the drum and its derivative are chosen as the system variables. In the second level, the derivative of a signal which is a linear function of the evaporator steam exit quality, downcomer flow, evaporator rising mixture flow, and \( u_m \) (the output of the first level rule set in Fig. 5.4-2) are chosen as the system variables.

In the hierarchical fuzzy rule set there are two kinds of rules, one is called the 'integral kind' and the other is the 'proportional kind.'

The integral kind of rule is of the following form:

\[ \text{If } (x_1 \text{ is } a_{1,k} , x_2 \text{ is } a_{2,k}) \text{, then increase the output by } b_k \]
The proportional kind of rule has the different form:

\[
\text{If } (x_1 \text{ is } a_{1j}, x_2 \text{ is } a_{2j}), \text{ then the output is } b_j
\]

The main difference between these two kinds of rules is that for the integration kind of rule, the output is not only dependent upon the present condition but also on the past, which means that it has memory. For the proportional kind of rule, the output is only dependent upon the present condition (or input).

In the first level, only the proportional kind of rules is used, while in the second level both the proportional and integral kind of fuzzy rules are used to form the complete rule set which is further divided into two subsets, subset one containing only the integral kind of rules, and subset two containing the other type. Therefore, output \( u_o \) is the contribution of both subsets.

\[
u_o = k_i * u_i + k_p * u_p
\]

where \( u_i \) and \( u_p \) are the sub-outputs due to the integral kind and the proportional type rules respectively, and \( k_i \) and \( k_p \) are the corresponding weight parameters.

By adjusting the values of \( k_i \) and \( k_p \), we can balance the performance of the system in terms such as rise time and overshoot. It is observed that an increase in \( k_i \) results in an increase in overshoot and a decrease in rise time; similarly an increase in \( k_p \) decreases rise time. The overall guidance for choosing \( k_i \) and \( k_p \) is very similar to the case of choosing the parameters of a PID controller. It should be noticed that the absolute values of \( k_p \) and \( k_i \) also depend upon the normalized values of \( u_i \) and \( u_p \).

The function of the first level rule set is to calculate the \( u_m \) which is the desired value of the rate of change of level \( dl/dt \) that makes level 1 reach or maintain the desired level.
function of the second level is to determine the actual output \( u_o \) that makes \( dl/dt \) reach or maintain its desired value given by the first level output \( u_m \), in the shortest possible time with the least overshoot.

The complete rule set that is actually used as the fuzzy rule-based controller is listed in the Appendix. The total number of rules is 129, which is less than the theoretical minimum (273), because some rules which yield the same output are merged.

### 5.5 Simulation results

The dynamic model of the drum given in (5.4-1), with the initial condition \( x(0)=0 \), is used in the simulation. The simulation is performed on a Vax-11/750 using Fortran language under a Unix operating system. To compare the effectiveness of the fuzzy controller with a well tuned PID controller\[^{[57]}\], three kinds of disturbance have been introduced into the system. The results are summarized below:

a. Ten percent steam valve opening disturbance

The simulation results of the drum level and feedwater flow to the input disturbance are shown in Figs. 5.5-1 and 5.5-2.

With the use of the fuzzy controller, the drum level is kept at the desired level practically all of the time, making the drum level insensitive to the disturbance. From Fig. 5.5-2, it is clear that both rise time and overshoot, in the case of the fuzzy controller, are less than those which are due to the PID controller. In general, a shorter rise time yields a larger overshoot. We need to reduce both of them for better system performance, as we have done here. Since the drum level and feedwater flow are related, the overall control system performance is satisfactory only if both of them are controlled properly. The reason is that if the feedwater flow has too much overshoot, which means that the drum has been fed with more water than desired, the water level (drum level) will be higher than desired.
If the feedwater rise time is too long, which means that drum has been fed less water than desired, the drum level will be less than desired too. From Figs. 5.5-1 and 5.5-2, it is clear that for the PID control system, the drum level is lower than desired at the beginning due to longer feedwater flow rise time, and then it is higher than the desired due to the overshoot of the feedwater flow.

Fig. 5.5-1 Drum level's response to a 10% steam valve opening perturbation. With the hierarchical fuzzy controller, the drum level is practically at the desired level for all the time.
Fig. 5.5-2 Feedwater flow's response to a 10% valve opening disturbance.

With the hierarchical fuzzy controller, the feedwater flow has a shorter rise time and lower overshoot than that with the PID controller.
b. 10°F feedwater flow temperature increase perturbation.

Fig. 5.5-3 shows that when using the fuzzy controller, the drum level can be maintained at its desired level even with the disturbance, while using the PID controller, the level changes as the perturbation occurs and has to wait about 200 seconds to reach its desired value. Fig. 5.5-4 shows that when using the PID controller, the feedwater flow is not increased as quickly as it should be, so the drum level drops first, as shown in Fig. 5.5-3. Then the feedwater flow does not stop when it reaches the desired value, thereby causing too much overshoot in the feedwater flow, as well as in the drum level.

![Graph showing drum level response to 10°F feedwater temperature increase perturbation.](image)

**Fig. 5.5-3** Drum level's response to 10°F feedwater temperature increase perturbation. With the hierarchical fuzzy controller, the drum level is practically at the desired level all of the time.
Fig. 5.5-4 Feedwater flow's response to 10°F feedwater temperature increase disturbance. With the hierarchical fuzzy controller, the feedwater flow has a shorter rise time and lower overshoot than it does with the PID controller.
c. Case of $+10^\circ$F inlet sodium temperature increase perturbation

From Figs. 5.5-5 and 5.5-6, one can conclude that the fuzzy controller yields a better performance than the PID controller does for the case of the inlet sodium temperature perturbation.

Fig. 5.5-5 Drum level's response to $10^\circ$F inlet sodium temperature increase perturbation. With the hierarchical fuzzy controller, the drum level is practically at the desired level all of the time.
Fig. 5.5-6 Feedwater flow's response to 10°F inlet sodium temperature increase disturbance. With the hierarchical fuzzy controller, the feedwater flow has a shorter rise time and lower overshoot than that with the PID controller.

The simulation results for the three different kinds of disturbances suggest that the fuzzy controller has a better performance than the PID controller does.

5.6 Conclusions

An examination of a conventional fuzzy controller revealed that the total number of rules in a complete rule set of a fuzzy controller is an exponential function of system variables. To overcome this difficulty, a hierarchical structure was proposed, in which the total number of rules was only a linear function of the system variables.

The hierarchical fuzzy control method was applied to control the feedwater flow to a steam generator of a power plant. Then the effectiveness of the fuzzy controller was studied
through simulations of several disturbances. The results showed that the fuzzy controller yielded a better performance than the PID controller in terms of less rise time and less overshoot for feedwater flow and maintenance of the drum level at the desired value.
Chapter VI

Adaptive Fuzzy Hierarchical Controller

6.1 Introduction

It has recently been established that fuzzy control to industrial processes has often produced results superior to classical control[58]-[71]. However, in the application of such rules to control a system, it is implicitly assumed that significant process change does not occur that is outside the operator's experience. This implicit assumption limits the application of the fuzzy controller to the case of normal working conditions for which the operator already has considerable experience. To accommodate abnormal working conditions however, adaptive functions should be introduced to adjust the parameters of a fuzzy controller to meet the unexpected cases that may exist in the real world. Walter H. Bare[72] studied a self-tuning rule-based controller for a gasoline refinery catalytic reformer. The performance of the self-tuning controller was tested against perturbations of a simulation model of the catalytic reformer. T.J. Procyk and E. H. Mamdani[73] proposed a linguistic self-organizing fuzzy controller and concluded that an adaptive controller using a linguistic description of the control strategy was able to learn how to successfully control a wide variety of processes in a relatively short time. S. Daley and K. F. Gill[74] further studied the self-organizing fuzzy logic controller and applied it to a more complex multivariable process, and the results show that the self-organizing fuzzy logic controller can perform reasonably well in a complex process with limited process knowledge. Motivated by these works, in this chapter an adaptive hierarchical fuzzy controller is studied, with system performance being measured and expressed by some fuzzy variables (performance indices). Based upon these fuzzy variables, a supervisory fuzzy rule set is constructed. The supervisory fuzzy rule set is used to adjust the parameters of the hierarchical rule-based fuzzy controller to achieve better performance, even in the case of
unexpected changes in system parameters.

Section 6.2 is concerned with the adaptive hierarchical fuzzy controller. The system performance evaluation and its description by linguistic syntax variables are given in section 6.2.1. In section 6.2.2, the adaptive functions are introduced into the system by constructing a supervisory fuzzy rule set. The supervisory fuzzy rule set, which is based upon linguistic syntax variables of system performance, adjusts the parameters of the hierarchical fuzzy controller to achieve a better performance in case of changes in system parameters or operating conditions. The simulation results of the proposed work are given in section 6.3 to further illustrate the effectiveness of the adaptive functions. Conclusions are given in section 6.4.

6.2 Adaptive hierarchical fuzzy controller design

A block diagram of the proposed adaptive hierarchical fuzzy control system is given in Fig. 6.2-1.

![Fig. 6.2-1 The configuration of the proposed adaptive fuzzy control system](image)

The block diagram has four main parts: hierarchical fuzzy controller, plant, performance evaluation, and adaptive functions. A detailed design procedure of a
hierarchical fuzzy controller is given in reference\cite{75}. It is also briefly described in section 2, and through an example (Fig. 6.2-7) in section 3 of this chapter. The main thrust of this chapter is about performance evaluation and adaptive functions. A detailed description of the results is given below.

6.2.1 The evaluation of system performance

The purpose of introducing adaptive functions into a fuzzy control system is to improve system performance when parameter changes occur. In order to analyze and design an adaptive fuzzy controller, we must define and measure the performance of the system. Based upon the performance of the system, the parameters of the controller can be adjusted in order to provide the desired response. Because most systems are inherently dynamic systems, the design specifications for control systems should normally include several time-response indices, as well as criteria for steady-state accuracy. The conventionally defined performance indices are evaluated after observing both transient and steady-state responses. In a real environment however, unexpected changes in system parameters or operating conditions can occur, which may cause the system to produce an undesirable output. An adaptive controller should change the parameters of the control system to prevent the undesired output from occurring. Because the adaptive action is based upon the performance indices of the system, these have to be calculated on line. For this purpose, some new performance indices will be formulated in this section.

Let us consider a typical control problem shown in Fig. 6.2-2. Denote \( y_d(t) \) and \( y \) as the desired and actual system responses (output) respectively. The error signal \( e(t) \) is defined by
The control objective is to keep $e(t)=0$ at all times. The system performance indices should therefore indicate the changes of $e(t)$. Hence, the following performance indices are adopted.

1. Pseudo-damping rate:

A pseudo-damping rate, $d_r$, is given as

$$d_r = \frac{r(t_2)}{r(t_1)}$$

where

$$r(t) \triangleq e^2(t) + p*(\frac{de}{dt})^2$$

$p>0$ is a weight constant, and $t_1$ and $t_2$ are two consecutive sampling points with $t_2 > t_1$. For $d_r=1$, that is $r(t)=\text{constant}$, it describes the ellipsoids in the phase plane of $e(t)$ and $\frac{de}{dt}$.

Fig. 6.2-3 shows ellipsoids with different major and minor axes. As $r(t)$ is a function of both $e(t)$ and $\frac{de}{dt}$, it describes both the present and the future state of the system. A small value of $r(t)$, both $e(t)$ and $\frac{de}{dt}$ are small in absolute value, means that the system state is near its desired position, and will not undergo large changes in the near future. A large value of $r(t)$ means that either the present state of the system differs greatly from the desired state, or it will change greatly in the near future. It is expected that when $d_r(t) < 1$ for all $t$, $r(t)$ will decrease monotonically, which indicates that the system exhibits good performance. When
\( d_i(t) > 1 \), \( r(t) \) will increase monotonically, and the system will be unstable or, at the least, the controller will need to be adjusted. In short, \( r(t) \) is a good indicator of the system.

![Fig. 6.2-3 Ellipsoids in a phase plane is described by \( r(t) = \text{constant} \)](image)

2. Degree of oscillation:

In control systems, converging (or damping) oscillations exist because of complex poles. Divergent oscillation indicates instability of the system. A performance index is therefore, needed to describe the degree of oscillation. For this purpose, let us observe the phase plane shown in Fig. 6.2-4. Line \( \text{o}e_i \) \((i=1,2, \ldots m)\) begins at point \( o \) at an angle \( i \times a \). Suppose that we begin to observe a trajectory at time \( t_0 \) (point \( b \) in Fig. 6.2-4). The trajectory meets line \( \text{o}e_i \) at a point with distance \( r_{i1} \), and then meets \( \text{o}e_i \) again at another point with distance \( r_{i2} \). We define the degree of oscillation as

\[
\zeta = \frac{r_{i2}}{r_{i1}} \quad (6.2-3)
\]

In general, whenever a trajectory meets a line \( \text{o}e_i \) \((i=1,2, \ldots m)\), the value \( r_{i,j} \) is computed and compared with the most recent value of \( r_{i,j-1} \). The degree of oscillation is then calculated as

\[
\zeta = \frac{r_{i,j}}{r_{i,j-1}} \quad (6.2-4)
\]
3. Offset:

\[ \zeta \triangleq \frac{r_{i2}}{r_{i1}} \]

where \( i \) is the sampling point, \( k \) is the length of the observing window, and current time is \( t = t_n \).

The offset \( \zeta \) is an index to determine whether \( y(t) \) is in the vicinity of the desired output value, above the desired output value, or below the desired output value.

4. Overshoot:

\[ \zeta_s \triangleq \frac{\text{Max}}{\text{in } (n-1)T_p < t < nT_p} \left| e(t) \right| \]

The overshoot is actually the largest deviation value (in an absolute sense) from the desired output value in the observation period \( T_p \). In Fig. 6.2-5, it is clearly shown that the overshoot equals 1 and 0.4 in the first and second periods respectively.
The four performance indices developed above are used in the next section concerning adaptive functions. Both the pseudo-damping rate and the degree of oscillation measure system current or near future behavior. However, the pseudo-damping rate is more like a "local" performance index, while the degree of oscillation is more like a "regional" performance index. Both overshoot and offset are the measurements of the previous system state. Hence, they represent the system state in the past.

As a specific example, let us formulate the above performance indices for a steam generator with the configuration shown in Fig. 6.2-6. The main goal of the steam generator controller is to maintain water level $l$ in the steam drum at a desired value by controlling the feedwater flow. The desired drum level is calibrated as $l=0$. The dynamic model for the steam drum has 18 state variables. Based upon 4 of these variables, a hierarchical fuzzy rule-based controller was constructed\cite{23}. The closed-loop model of the system with a fuzzy controller is shown in Fig. 6.2-7.

The dynamic model of the drum is given by:

$$\frac{dx}{dt} = Ax + B_d u_d + B_0 u_0$$  \hspace{1cm} (6.2-7)

where $x$ is the system state vector, $A$ is the system matrix, $B_d$ is the disturbance input matrix, $u_d$ is the disturbance input, $B_0$ is the input matrix, and $u_0$ is the input obtained from the fuzzy controller. The details of the model are given in reference\cite{57}. 

![Figure 6.2-5](image)  

Fig. 6.2-5 A typical system response, in which the overshoot is 1.0 and 0.5 in the first and second period respectively.
Fig. 6.2-6 Configuration of a steam drum. The control goal is to keep the drum water level in a desired position by adjusting the feedwater valves.

Fig. 6.2-7 The configuration of the existing hierarchical fuzzy control system.

The calculation of the proposed performance indices is straightforward. For a steam drum, the desired response is desired drum level $l_d$, which is set to zero at all times. Therefore, by substituting $e$ with -1, the corresponding performance indices can be calculated for the steam generator.

6.2.2. Adaptive functions

In this section, adaptive functions will be introduced to improve system performance when changes in system parameters or operating conditions occur.

The main adaptive functions include the following:

1. Tuning of normalization.
2. Tuning of weight parameters.

3. Tuning of parameters $k_p$ and $k_i$ in the rule-based controller.

a. Tuning of normalization:

In fuzzy logic control, it is necessary to determine the universe of discourse to give the semantics of a fuzzy variable, i.e. its membership function. The universe of discourse in fuzzy control is normalized into the interval $[-1,1]$. This normalization is done by careful calibration under normal operating conditions. However, when parameters change, the calibration should be repeated. The algorithm shown in Fig. 6.2-8 is followed to tune normalization of the fuzzy variables. The tuning will provide a stronger control action when an undesirable response occurs.

![Diagram](image)

**Fig. 6.2-8** The algorithm for re-calibration when system parameters have changed.
b. Tuning of weight parameters:

Hierarchical fuzzy control contains several level rule sets. The first level rule set gives a basic control action, while the higher level rule sets initiate fine tuning control action based upon the base (gross) control action. In general, the first level rule set depends upon only a few important system variables, while the higher level rule sets rely on more system variables as explained in section 2. In our example of feedwater control, the first level rule set is based upon drum level $l$ and $dl/dt$, and the second level rule set on evaporator steam exit quality, downcomer flow, and evaporator rising mixture flow. These variables have influence on the change of $dl/dt$. The relationship between $dl/dt$ and evaporator steam exit quality, downcomer flow, and evaporator rising mixture flow depends upon the system parameters. This relationship will change when the system parameters change, and may result in an undesirable fine tuning control action. To avoid initiating an undesirable control action, the final control action should mainly depend upon the first level rule set when system parameter perturbation occurs.

Unlike the conventional hierarchical fuzzy controller, the total (final) control action of an adaptive controller is composed of the control actions due to different level rule sets; that is:

$$u_{tot} = \sum_{i=1}^{L} k_i u_i$$

where $L$ is the total number of levels in the hierarchy and $u_{tot}$ is the final control action. $u_i$ is the control action obtained by consulting the $i$th level rule set. $k_i$ is the corresponding weight parameters and satisfies $k_i > 0$, and $\sum_{i=1}^{L} k_i = 1$.

The algorithm (given in Fig. 6.2-9) shows how to calculate $k_i$ for a two-level hierarchy when system parameters are changed.
In Fig. 6.2-9, \( t_1 \) and \( t_2 \) are the update times for \( h \) and \( k_1 \), where \( h \) is a vector holding the sampled values of \( r(t) \). Initially, \( t_1, t_2, \) and \( h \) are set to be zero. \( \text{ept}_1 \) and \( \text{ept}_2 \) are the time intervals for updating \( h \) and \( k_1 \) respectively. The value of \( h(i) \) is transferred to \( h(i+1) \) for \( i=n-1, n-2, \ldots, 1 \), and the last element of \( h \), \( h(n) \), is abandoned. A new value of \( r(t) = l(t)^2 + p^* (dl/dt)^2 \) is generated and stored in \( h(1) \). If \( t-t_2 > \text{ept}_2 \), time interval \( \text{ept}_2 \) has passed since the last update of \( k_1 \) and \( k_2 \) values. The 7th and 8th blocks of Fig. 6.2-9 show how \( k_1 \) and \( k_2 \) are calculated. \( k_1 \) and \( k_2 \) reflect the changes of \( r(t) \) over a long time period. A large value of \( k_1 \) means that \( r(t) \) increased most of the time, and a large value of \( k_2 \) means that \( r(t) \) decreased most of the time. For a large \( k_1 \), the final control action will mainly
depend on the first level rule set's output.

c. Tuning of parameters $k_p$ an $k_i$ in a fuzzy controller:

In the fuzzy rule set, two kinds of rules are used: "integral kind" and "proportional kind". The integral kind rule is of the following form:

$$\text{if } (x_1 \text{ is } a_{1,k}, x_2 \text{ is } a_{2,k}), \text{ then } (\text{increase the output by } b_k).$$

The proportional kind rule has a different form:

$$\text{if } (x_1 \text{ is } a_{1,j}, x_2 \text{ is } a_{2,j}), \text{ then } (\text{the output is } b_j).$$

The main difference between these two kinds of rules is that for the integration kind of rule, the output depends upon the present condition as well as on the past. For the proportional kind of rule, the output only depends upon the present condition (or input). In this study, both proportional and integral kinds of fuzzy rules are used at each level. At kth level ($k=1,2$), the control action is

$$u_k = k_p u_{k,p} + k_i u_{k,i} \quad 6.2-8$$

where $u_{k,p}$ and $u_{k,i}$ are the control actions obtained by consulting proportional and integral kinds of fuzzy rules respectively. $k_p$ and $k_i$ are weighting parameters. The final control action, therefore, is

$$u_{tot} = k_1 u_1 + k_2 u_2 + \ldots$$

$$= k_p ((k_1 u_{1,p} + k_2 u_{2,p} + \ldots) + k_i ((k_1 u_{1,i} + k_2 u_{2,i} + \ldots)) \quad 6.2-9$$

As in a conventional PI controller, the parameters $k_p$ and $k_i$ play an important role in overall system performance. To achieve a desirable performance, $k_p$ and $k_i$ should be readjusted as system parameters change. In this study, a supervisory rule set consisting of two subsets is constructed to tune $k_p$ and $k_i$. Subset 1 is based upon the performance indices for the pseudo-damping rate and the degree of oscillation, and subset 2 is based upon performance indices for offset and overshoot. As mentioned earlier, the pseudo-damping rate and the degree of oscillation are current system descriptions, while the offset
and overshoot are only indicators of past performance. Based upon this fact, rule set 1 will be initiated more frequently than rule set 2. Thus, the tuning of $k_p$ and $k_i$ will be related more to the system's current status.

In building the supervisory rule set, the knowledge on tuning $k_p$ and $k_i$, gained from the experience of tuning PI controllers, is used and stated as a set of linguistic statements. Such statements (rules) will necessarily contain fuzzy quantities which are the pseudo-damping rate, oscillation rate, offset and overshoot. Next, membership functions are established for fuzzy quantities presented in the fuzzy rules. The resulting fuzzy rule sets are shown in tabular form in Figs. 6.2-10 and 6.2-11. As an example, the first rule in Fig. 6.2-10(a) can be stated linguistically as: if the pseudo-damping rate is very fast and the degree of oscillation is very small, then the control gain $k_p$ should be changed by zero percent. Similarly, all of the other rules can be stated linguistically.

The fuzzy quantities such as very fast, very small, small, medium, etc. are used in the statements and the corresponding membership functions are, therefore, needed. The choice of membership function is situation dependent. For our studies, the triangular type membership function (Fig. 4.1-5) was used. The same kind of fuzzy logic, as stated in section 2, may also be applied for the supervisory fuzzy tuning rule set to initiate a particular tuning action.

<table>
<thead>
<tr>
<th>DO</th>
<th>PDR</th>
<th>Very fast</th>
<th>Little fast</th>
<th>Normal</th>
<th>Little slow</th>
<th>Too slow</th>
<th>Unstable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very small</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>SP</td>
<td>MP</td>
<td>LP</td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
<td></td>
</tr>
<tr>
<td>Big</td>
<td>SN</td>
<td>SN</td>
<td>Z</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td></td>
</tr>
<tr>
<td>Too big</td>
<td>MN</td>
<td>SN</td>
<td>Z</td>
<td>SP</td>
<td>SP</td>
<td>SN</td>
<td></td>
</tr>
<tr>
<td>PDR</td>
<td>DO</td>
<td>Very fast</td>
<td>Little fast</td>
<td>Normal</td>
<td>Little slow</td>
<td>Too slow</td>
<td>Unstable</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-----------</td>
<td>-------------</td>
<td>--------</td>
<td>-------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>Very small</td>
<td>Z</td>
<td>Z</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
<td>MP</td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>SP</td>
<td>SP</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>SN</td>
<td>Z</td>
<td>Z</td>
<td>SP</td>
<td>SN</td>
<td>MN</td>
<td></td>
</tr>
<tr>
<td>Big</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
<td>MN</td>
<td></td>
</tr>
<tr>
<td>Too big</td>
<td>MN</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
<td>MN</td>
<td>LN</td>
<td></td>
</tr>
</tbody>
</table>

(b)

Fig. 6.2-10 Supervisory rule subset 1

(a) Supervisory rule set for adjusting $k_p$

(b) Supervisory rule set for adjusting $k_i$

PDR=Pseudo-damping rate, DO=Degree of oscillation.

<table>
<thead>
<tr>
<th>OV</th>
<th>OS</th>
<th>LN</th>
<th>SN</th>
<th>Normal</th>
<th>SP</th>
<th>LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>SP</td>
<td>SP</td>
<td>Z</td>
<td>Z</td>
<td>SN</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>Big</td>
<td>SP</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>SN</td>
<td></td>
</tr>
<tr>
<td>Too Big</td>
<td>SP</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>SN</td>
<td></td>
</tr>
</tbody>
</table>

(a)

Fig. 6.2-11 Supervisory rule subset 2 (OS=Off set, OV=Overshoot)

(a) Supervisory rule set for adjusting $k_p$

(b) Supervisory rule set for adjusting $k_i$
6.3 Simulation results

The adaptive fuzzy hierarchical controller was applied via simulation to control the feedwater flow to the steam generator. The simulations showed that with the adaptive functions, the fuzzy controller is even more robust to system parameter changes. The simulations were performed on a SUN work station using the system model given in reference[57]. The four most important parameters were changed to show the robustness of the controller.

In simulation 1, parameter A(18,19) of system matrix A was changed from its original value of 0.262*10^{-3} to the value of 0.112*10^{-3}. When a 10% steam valve opening was applied, as shown in Fig. 6.3-1, the fuzzy controller with adaptive functions maintained the drum level almost constantly at zero, while the fuzzy controller without adaptive functions failed. Figs. 6.3-2 and 6.3-3 further illustrate the difference between the two controllers. With adaptive functions, feedwater flow responded properly to the valve opening; without adaptive functions, the feedwater flow oscillated and made the drum level unstable.
Fig. 6.3-1 Drum level response to a 10% valve opening disturbance with system parameter $A_{(18,19)}$ having a 63% decrease.

Fig. 6.3-2 Feedwater response to a 10% valve opening disturbance with system parameter $A_{(18,19)}$ having a 63% decrease.
Fig. 6.3-3 Feedwater response to a 10% valve opening disturbance with system parameter A(18,19) having a 63% decrease (to detail the adaptive control system response).

In simulation 2, we changed the values of A(8,8) from -0.089 to -0.099, A(18,8) from \(-0.26 \times 10^{-3}\) to \(-0.29 \times 10^{-3}\), and A(18,9) from \(0.22 \times 10^{-3}\) to \(0.16 \times 10^{-3}\). When a 10% steam valve opening was applied, responses, as shown in Figs. 6.3-4, 6.3-5 and 6.3-6, were very similar to that of simulation 1, confirming that a controller with adaptive functions provides a robust behavior.
Fig. 6.3-4 Drum level response to 10% valve opening disturbance with system parameters $A(8,8)$ and $A(18,8)$ having an 11.5% increase and a 17% decrease respectively.

Fig. 6.3-5 Feedwater response to a 10% valve opening disturbance with system parameters $A(8,8)$ and $A(18,8)$ having an 11.5% increase and a 17% decrease respectively.
Fig. 6.3-6 Feedwater response to a 10% valve opening disturbance with system parameters $A(8,8)$ and $A(18,8)$ having an 11.5% increase and a 17% decrease respectively (to detail the adaptive control system response).

The same kind of behavior was observed in simulation 3, in which we changed the values of $A(18,3)$ from -0.13 to -0.43. The plots of simulation 3 are given in Figs. 6.3-7, 6.3-8 and 6.3-9.
Fig. 6.3-7 Drum level response to a 10% valve opening disturbance with system parameter $A(18,8)$ having a 231% increase.
Fig. 6.3-8 Feedwater response to a 10% valve opening disturbance with system parameter A(18,3) having a 231% increase.

Fig. 6.3-9 Feedwater response to a 10% valve opening disturbance with system parameter A(18,3) having a 231% increase (to detail the adaptive control system response).
6.4 Conclusions

In this chapter, the adaptive fuzzy control algorithm was studied and then applied via simulation to a steam generator's drum level control. The simulations indicated that the fuzzy controller with the adaptive functions exhibited more robustness as compared to the fuzzy controller without adaptive functions. This method can be applied to other control problems without much difficulty. However, the stability criterion of the adaptive fuzzy control still needs more investigation. In investigating the stability criterion, the system model of the process will need to be used. We also noticed that to further improve the system performance, the fuzzy rules should represent not only human experience, but also the results of rigorous mathematical analysis of the system. By this theoretical analysis, a systematic strategy may be formulated as how to change the fuzzy rules and the associated membership functions to improve the system performance. This effort may narrow down the gap between fuzzy controllers and conventional controllers, and as a result, conventional controller design and fuzzy controller design can benefit (or compensate) each other.
Chapter VII
Fuzzy Rule-based Approach for Robot Motion Control

7.1 Introduction

In robot motion control, a typical task is to move the end-effector of a robot along a specified (desired) trajectory. The relationship between the end-effector position $X(t)$ and joint angle $\Theta(t)$ can be represented by (7.1-1) or (7.1-2).

\[
X(t) = F[\Theta(t)] \quad (7.1-1)
\]
\[\Theta(t) = F^{-1}[X(t)] \quad (7.1-2)
\]

Equations (7.1-1) and (7.1-2) are known as the forward and inverse kinematics of a robot respectively. For existing robots, the forward kinematic model (7.1-1) is easy to formulate and evaluate; however, the inverse kinematic model (7.1-2) represents a very complex mathematical mapping between $X(t)$ and $\Theta(t)$. In some cases, the map (7.1-2) is practically impossible to formulate, particularly for redundant robots\[76][77].

Several approaches have been proposed to solve the redundancy and inverse kinematic problems. Among them are the optimization constraint techniques\[76][78] which obtain unique inverse kinematic solutions under additional constraints. However, this approach also brings an additional computational burden to the host processor. In addition to the computational penalty, the traditional mathematical approaches of deriving inverse kinematic equations have led to another undesirable by-product: kinematic singularities, which arise from the inversion of an ill-conditioned matrix.

We have observed that many complex and ill-defined systems perform reasonably well by consulting fuzzy rules\[51][52], which are a set of linguistic statements reflecting human's knowledge and experience with the systems. This fuzzy rule-based control
scheme has been successfully applied in the control of steam generators\cite{58}, automatic train operation systems\cite{60}, elevator control\cite{61}, nuclear reactor control\cite{53}, and automobile transmission control\cite{62}. For many applications, the fuzzy logic control has produced results superior to those yielded by classical control\cite{70-74}.

In this study, we propose a fuzzy rule-based approach for robot motion control. The fuzzy rules have been arranged hierarchically to reduce the number of fuzzy rules needed for the control of a robot.

The chapter is organized as follows. In section 7.2, the fuzzy rules are formulated for a two degree of freedom (DOF) robot using the hierarchical structure. Based upon the study of two DOF robots, the fuzzy rule sets are formulated for four DOF redundant planar robot motion control. The simulation results are given in section 7.3, and the final conclusions are given in section 7.4.

7.2 Fuzzy rule-based controller design

Unlike a conventional mathematical approach, in fuzzy control, use is made of fuzzy rules that designate that certain actions be taken should a specific set of conditions occur. These fuzzy rules are necessarily the reflection of human knowledge and experience. The central point of this study is how to formulate the appropriate fuzzy rules. It is amazing to observe that a human can control his arm elegantly without much of sharp mathematics, whereas robot control requires complex mathematical formulas. This may be a hint for us to use non-mathematical fuzzy logic for the control of robot motion.

To facilitate the problem of fuzzy controller design, two DOF planar robots are first considered, and then the scheme is extended to that of four DOF planar robots with kinematic redundancy.
7.2.1 Fuzzy rule-based controller design for two DOF robots.

Suppose we want to move the robot from an initial position \((x_0, y_0)\) to a desired position \((x_d, y_d)\), as in figure 7.2-1, where \(\alpha_{10}\) and \(\alpha_{20}\) (\(\alpha_{1d}\) and \(\alpha_{2d}\)) are the initial (desired) angular position of joints 1 and 2 respectively. Initial robot arm length \(r_0\) and desired length \(r_d\) are the corresponding distances between the end-effector and the origin point. Let us denote \(l_1\) and \(l_2\) as the lengths of links 1 and 2 respectively. The current robot arm length \(r\) is given by (7.2-2).

\[
r = \sqrt{x^2(t) + y^2(t)} \quad \text{(7.2-2)}
\]

\[
r^2 = l_1^2 + l_2^2 + 2l_1l_2\cos\alpha_2 \quad \text{(7.2-3)}
\]

![Fig. 7.2-1 A two DOF robot at initial and desired configurations](image)

Since \(r\) is only a function of \(\alpha_2\), we can achieve the desired arm length \(r_d\) by changing \(\alpha_2\) alone. The mathematical relationship between the change needed in arm length and the change in \(\alpha_2\) to achieve \(r_d\) can be derived as

\[
\Delta^2 r = r_d^2 - r_0^2
\]
\[ \Delta^2 r = l_1^2 + l_2^2 + 2l_1l_2\cos \alpha_{2d} - (l_1^2 + l_2^2 + 2l_1l_2\cos \alpha_{20}) \] (7.2-4)

\[ = 2l_1l_2(\cos \alpha_{2d} - \cos \alpha_{20}) \]

\[ = 2l_1l_2[ \cos(\alpha_{20} + \Delta \alpha_2) - \cos \alpha_{20}] \] (7.2-5)

\[ = 2l_1l_2(\cos \alpha_{20}\cos \Delta \alpha_2 - \sin \alpha_{20}\sin \Delta \alpha_2 - \cos \alpha_{20}) \] (7.2-6)

If \( \Delta \alpha_2 \) is small, (7.2-6) becomes

\[ \Delta^2 r = 2l_1l_2(\cos \alpha_{20} - \sin \alpha_{20}\times \Delta \alpha_2 - \cos \alpha_{20}) \]

\[ = -2l_1l_2 \sin \alpha_{20}\times \Delta \alpha_2 \] (7.2-7)

In this study, instead of \( \Delta r \), \( \Delta^2 r \) is used as input signal to the fuzzy rules. Using \( \Delta^2 r \) will actually reduce the calculation, since \( r \) is usually calculated by adding \( x^2 \) and \( y^2 \). After achieving the desired \( r_d \) through the change in \( \alpha_2 \), \( \alpha_1 \) is changed to rotate the robot arm to reach the desired position. A pictorial description of this process is given in Fig. 7.2-2, where \( \theta^A_{e_o} \leq \angle EOX \), \( E \), \( O \) and \( X \) are represent the point at the end-effector, origin, and positive \( X \) axis, and \( \Delta \theta^A_{e_o} = \theta_{e_o} - \theta_d \), \( \theta_d \) and \( \theta_o \) represent desired and initial angles, \( \theta \), respectively.

Based upon the above consideration, the fuzzy rules may be arranged into two levels, as shown in Figure 7.2-3, to form the fuzzy controller. The first and second level rule sets are used to change \( \alpha_2 \) and \( \alpha_1 \) respectively. By using this hierarchical fuzzy control techniques\(^{[24]}\), the total number of rules needed to develop the fuzzy controller is reduced. This idea will be further illustrated through the following comparative studies.
An intuitive rule is of the following form:

If (\(\alpha_{10}\) is \(x_1\)) and (\(\alpha_{20}\) is \(x_2\)) and (\(x_d\) is \(x_3\)) and (\(y_d\) is \(x_4\)), then

(\(\alpha_{10}\) is changed by \(\Delta\alpha_1\)), and (\(\alpha_{20}\) is changed by \(\Delta\alpha_2\)).

(7.2-8)

In this sample fuzzy rule, \(x_1, x_2, x_3, x_4, \Delta\alpha_1,\) and \(\Delta\alpha_2 \in A\) are some linguistic values. Here \(A=\{LN, MN, SN, ZN, Z, ZP, SP, MP, LP\}\).

If the rules of form 7.2-8 are used, then by Theorem 5.2-2, the total number of rules needed will be \(9^4=6561\). To reduce this huge number of rules, a two level hierarchy is
The proposed hierarchical fuzzy rule sets for control of robot motion are actually used. In the first level of the hierarchy, rules of following form are used,

\[
\text{If (robot arm length need to be changed by } x_1 \text{) and (current joint angle } \alpha_2 \text{ is } x_2\text{), then (change } \alpha_2 \text{ by } \Delta \alpha_2. \tag{7.2-9}\]

The fuzzy rules in level two are of the form:

\[
\text{If (change in angle } \alpha_2 \text{ is } \Delta \alpha_2 \text{) and (desired angle change } \Delta \theta \text{ is } x_2\text{), then (change } \alpha_1 \text{ by } \Delta \alpha_1. \tag{7.2-10}\]

The rules in levels one and two, with a total number of \(2^9q^2=162\), form the basis for the robot motion control. By comparing the number of rules needed in hierarchical and non-hierarchical controllers, the advantage of the hierarchy is clearly illustrated.

Fig. 7.2-4 shows a block diagram of the proposed fuzzy robot control concept. The developed fuzzy rule sets reside within the fuzzy controller, which outputs incremental joint commands to the individual joints of the robot based upon the robot configuration and the deviations of the actual end point to the desired end point. The actual Cartesian end point is determined by applying forward kinematic equations on joint angles. The desired trajectory is user-specified.
7.2.2 Fuzzy rule-based controller design for four DOF robots.

The configuration of a four DOF robot with kinematic redundancy is shown in Fig. 7.2-5. The conventional notations, shown in Fig. 7.2-5, are as shown below:

\[
\begin{align*}
\theta^\Delta &\leq \text{EOX, E, O, and X represent the point at the end-effector, origin, and positive X axis,} \\
\alpha_i &\text{ the joint angular position for joint } i \ (i=1,2,3,4), \\
r &\text{the distance between the end-effector and the origin point,} \\
x_i \text{ and } y_i &\text{ the Cartesian coordinates of joint } i \ (i=1,2,3,4), \text{ and} \\
x \text{ and } y &\text{ the Cartesian coordinates of the end-effector,}
\end{align*}
\]

As a convention, \((x_1, y_1)\)=(0, 0) is the origin point.
In this four DOF robot study, we want to control the robot moving along the desired trajectory by changing its joint angles at every sampling point by consulting the fuzzy rule sets. Based upon the discussion in section 7.2-1, we can decompose the movement of a robot into two parts:

- **action one**: stretching or folding of the robot arm,
- **action two**: rotation of the robot arm.

Action one involves the changes of joint angles $\alpha_2$, $\alpha_3$, and $\alpha_4$, while action two involves the change of joint angles $\alpha_1$. Further study reveals that as far as the end-effector position is concerned, action one can be completed by changing one value of $\{\alpha_2, \alpha_3, \alpha_4\}$. Therefore, at this moment, the robot has only two joints that need to be changed, one is joint $i$ for action one, and the other is joint 1 for action two, with corresponding pseudo link lengths $L_1$ and $L_2$ given in (7.2-11) and (7.2-12). Here integer $i \in \{2, 3, 4\}$.

\[
L_1 = (x_i^2 + y_i^2)^{\frac{1}{2}} \quad \text{(7.2-11)}
\]

\[
L_2 = [(x_i-x)^2 + (y_i-y)^2]^{\frac{1}{2}} \quad \text{(7.2-12)}
\]

Hence, the control of a four DOF robot can be treated as that of a two DOF robot. The fuzzy rule sets, therefore, are formulated in a similar fashion as in section 7.2-1.

By consulting the fuzzy rule sets, we change only two of the four joint angles at every sampling point. We may change other joint angles at other sampling times, so it may look as if all joints are moving to follow the desired trajectory. We introduce the following strategy to determine the joint angle to be changed in a redundant robot. If we want to stretch the robot arm, the joint with the largest absolute angle will be moved and vice versa. By doing so, we can avoid forcing one joint to move to its singular position, and achieve efficiency in terms of the least changes in angle to stretch(fold) the robot arm.

As this fuzzy rule-based approach does not require an inverse kinematic model, the
fact that a kinematic redundant robot does not have a unique solution is of no consequence.
As mentioned before, this non-uniqueness of the kinematic solution has inhibited real-time
motion control of kinematically redundant robots.

The block diagram of the fuzzy rule-based controller is the same as that shown in Fig.
7.2-4. The simulation results of the four DOF planar robot are given in the next section.

7.3 Simulation results

The simulations of the proposed algorithm are done on a Sun-SPARC work station.

Fig. 7.3-1 shows the trajectory of a four DOF robot which tracks a user-specified
circle and some corresponding configurations of the robot. The configurations of the robot
are all in reasonably good positions in the sense that these positions keep all joints away
from their singular points. It also shows that the robot has passed one of its singular
points, which usually causes over-flow in conventional mathematical algorithms. Fig. 7.3-
2 shows that the error between the desired trajectory and the actual trajectory is within a
specified limit.

Fig. 7.3-1 A four DOF robot tracking a specified circle and
some of the corresponding configurations.
Fig. 7.3-2  The absolute error between the end-effector's actual position and the desired position while the robot is tracking a circle.

Fig. 7.3-3 shows the trajectory of a robot tracking a user-specified straight line and partial corresponding configurations. Fig. 7.3-4 shows the error between the desired trajectory and the actual trajectory. The previous comments on Figs. 7.3-1 and 7.3-2 are applicable for Figs. 7.3-3 and 7.3-4 respectively.

Fig. 7.3-3  A four DOF robot tracking a specified straight line and some of the corresponding configurations.
Fig. 7.3-4 The absolute error between the end-effector's actual position and the desired position while the robot is tracking a straight line.

In the last simulation, we want the robot to reach its desired end position without specifying the trajectory to be followed by the robot. The comments on Figs. 7.3-1 and 7.3-2 are applicable for Figs. 7.3-5 and 7.3-6 respectively. Figs. 7.3-5 and 7.3-6 also show that no matter what the initial error is, the steady-state error will fall within the specified limit. This also illustrates the robustness of the fuzzy controller.
Fig. 7.3-5 A robot moving from its initial position to its desired position without a specified trajectory to be followed by the robot. The complete trajectory and partial configurations are shown in the figure.

Fig. 7.3-6 The absolute error between the end-effector’s actual position and the desired position while the robot is moving from its initial position to its desired position without a specified trajectory.
7.4 Conclusions

In this chapter, the fuzzy rule-based robot motion control algorithm was proposed. It was shown that the robot can track user-specified trajectories within the specified limitations. We also observed that this fuzzy rule-based controller is robust to disturbances.
Chapter 8

Epilogue

In this thesis, new strategies for the control of uncertain systems were developed. The two main topics, studied here, were robust controller and fuzzy controller design with applications. These two topics deal with different aspects of uncertainty.

In the study of robust controller design, two algorithms were developed for different applications. One was for robot motion control via robust eigenstructure assignment. With the robust controller, the updating rate of the controller parameters was reduced to accommodate real time implementation (on-line calculation). The other approach was for control of multimode systems via robust pole assignment. In this approach, unlike conventional pole assignment, the desired pole position was not specified by n self-conjugated numbers, but by a given disk to obtain more flexibility in choosing the feedback matrix to increase robustness. An algorithm was proposed to find a feedback matrix to maintain the closed-loop poles within the given disk even when the system worked under different operating conditions. The numerical examples were given to illustrate the effectiveness of the proposed approach.

The main contributions of the above two approaches were on how to use freedom in choosing the controller parameters to increase the robustness of such a controller while still maintaining the required nominal system performance. In the first approach, the freedom was gained by partially decoupling (not total decoupling) the interaction between links of a robot. In the second approach, the freedom was gained by specifying the desired poles within a disk, not as n exact numbers. Both approaches achieved the required nominal system performance while increasing the robustness. It was noticed that the best way to obtain the nominal performance in a real system was by robust design.
Although the proposed approaches improved the robustness of the system to low-order parameter variations, the robustness to high-order unstructured uncertainties was not considered here. Future work, therefore, should focus on how to improve system robustness to both low-order parameter variations and high-order unstructured uncertainties. Another interesting research topic would be to find the necessary and sufficient conditions for robust pole assignment.

In the study of fuzzy controller design, a hierarchical fuzzy control algorithm was developed to accommodate fuzzy controller design for large-scale systems. It was proved that the number of rules of a fuzzy controller increased linearly with the number of system variables using the proposed hierarchical approach, while the number of rules increased exponentially using the conventional approaches. Based upon the proposed hierarchical fuzzy controller, an adaptive hierarchical fuzzy control structure was proposed. The performance indices were formulated and converted as linguistic fuzzy variables. Based upon these linguistic fuzzy variables, adaptive functions were introduced to accommodate unexpected working condition changes. The proposed algorithm was applied via simulation to control the feedwater flow to a steam generator of a power plant. The simulation results show the effectiveness of the proposed algorithm.

The main contribution of this part of the work was on how to reduce the number of rules in a complete rule set of a fuzzy controller and introduce adaptive functions so that the fuzzy control system would be more versatile and robust. In the last part of the dissertation, the fuzzy controller was successfully applied to control the motion of a robot. The central importance of this work was to eliminate the computational complexity of the inverse kinematics associated with conventional mathematical approaches.

As the fuzzy control was originally proposed as a means of both capturing human
expertise and dealing with uncertainty, rigorous mathematical analysis was not adopted in this study. However, to further improve system performance, rigorous mathematical analysis will be necessary. Future research will include how to use Popov criterion (or Circle criterion) in fuzzy controller stability analysis and how to improve system performance using rigorous mathematical approaches.

The ultimate goal for future research is to integrate conventional rigorous mathematical approaches with fuzzy approaches. We believe that by integrating conventional approaches with fuzzy control approaches we will be able to provide effective approaches for control of uncertain systems, and other engineering problems (like image processing, etc.).
References


[23] D. M. Liu and F. Z. Liu, " A pole placement approach to multivariable control of


[34] J. H. Chow, "A pole-placement design approach for systems with multiple


[66] M. Yonekara, "The application of fuzzy set theory to the temperature of box annealing furnace using simulation techniques," in Proc. 8th IFAC World
Congress, Tokyo, 1981.


[78] C. W. de Silva, C. L. Chung and C. Lawrence, "Base reaction optimization of
Let us denote: \( g \) as greater than or equal to, and \( l^* \) as less than or equal to \( * \), where \( * \) represent the name of a fuzzy set.

We use the statement

\[
\text{when}(x_1, gz, x_2, gsp, uln)
\]

to mean:

if \( x_1 \) is greater than or equal to zero and \( x_2 \) is greater than or equal to small positive, then output is large negative.

The general form for the rules in the computer is

\[
\text{when}(x_1, x_1's \text{ condition, } x_2, x_2's \text{ condition, output})
\]

The complete rule set is given below:

### 1. The first level rule set

<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>( x_1, \text{ zm}, x_2, \text{ gmp, unm} )</td>
<td>( x_1, \text{ gm} )</td>
<td>( x_2, \text{ gmp, unm} )</td>
</tr>
<tr>
<td>2.</td>
<td>( x_1, \text{ ln, zm}, x_2, \text{ any, ulp} )</td>
<td>( x_1, \text{ zn} )</td>
<td>( x_2, \text{ zm, uzn} )</td>
</tr>
<tr>
<td>3.</td>
<td>( x_1, \text{ zn}, x_2, \text{ sp, usn} )</td>
<td>( x_1, \text{ sp} )</td>
<td>( x_2, \text{ sp, usn} )</td>
</tr>
<tr>
<td>4.</td>
<td>( x_1, \text{ sp}, x_2, \text{ zm, unm} )</td>
<td>( x_1, \text{ sp} )</td>
<td>( x_2, \text{ zm, unm} )</td>
</tr>
<tr>
<td>5.</td>
<td>( x_1, \text{ zm}, x_2, \text{ zm, uzn} )</td>
<td>( x_1, \text{ zm} )</td>
<td>( x_2, \text{ zm, uzn} )</td>
</tr>
<tr>
<td>6.</td>
<td>( x_1, \text{ sp}, x_2, \text{ zm, usn} )</td>
<td>( x_1, \text{ sp} )</td>
<td>( x_2, \text{ zm, usn} )</td>
</tr>
<tr>
<td>7.</td>
<td>( x_1, \text{ zm}, x_2, \text{ zm, uzn} )</td>
<td>( x_1, \text{ zm} )</td>
<td>( x_2, \text{ zm, uzn} )</td>
</tr>
</tbody>
</table>

### 2. The second level rule set

<table>
<thead>
<tr>
<th>Subset 1 (integral kind)</th>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>( x_1, \text{ zm}, x_2, \text{ gmp, unm} )</td>
<td>( x_1, \text{ zm} )</td>
<td>( x_2, \text{ zm, uzn} )</td>
</tr>
<tr>
<td>1.2</td>
<td>( x_1, \text{ zm}, x_2, \text{ zm, uzn} )</td>
<td>( x_1, \text{ zm} )</td>
<td>( x_2, \text{ zm, uzn} )</td>
</tr>
<tr>
<td>1.3</td>
<td>( x_1, \text{ zm}, x_2, \text{ zm, uzn} )</td>
<td>( x_1, \text{ zm} )</td>
<td>( x_2, \text{ zm, uzn} )</td>
</tr>
<tr>
<td>Condition 1</td>
<td>Condition 2</td>
<td>Condition 3</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>when(x1, zn, x2, z, uz)</td>
<td>when(x1, mn, x2, zn, usp)</td>
<td>when(x1, sn, x2, mn, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, z, x2, z, uz)</td>
<td>when(x1, sn, x2, zn, uz)</td>
<td>when(x1, lzn, x2, ln, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, lp, x2, z, uln)</td>
<td>when(x1, zn, x2, z, uz)</td>
<td>when(x1, gmp, x2, lzn, uln)</td>
<td></td>
</tr>
<tr>
<td>when(x1, zn, x2, mn, usn)</td>
<td>when(x1, gz, x2, zm, uln)</td>
<td>Subset 2 (proportional kind)</td>
<td></td>
</tr>
<tr>
<td>when(x1, lz, x2, gmp, ulp)</td>
<td>when(x1, lp, x2, zp, ump)</td>
<td>when(x1, sp, x2, sp, usp)</td>
<td></td>
</tr>
<tr>
<td>when(x1, gzp, x2, lp, ump)</td>
<td>when(x1, mz, x2, mp, usp)</td>
<td>when(x1, sp, x2, sp, uzp)</td>
<td></td>
</tr>
<tr>
<td>when(x1, sp, x2, mp, usp)</td>
<td>when(x1, sm, x2, zp, usp)</td>
<td>when(x1, sp, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, gmp, x2, mp, usp)</td>
<td>when(x1, sn, x2, zp, usp)</td>
<td>when(x1, lp, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, lsn, x2, sp, ulp)</td>
<td>when(x1, z, x2, zp, usp)</td>
<td>when(x1, zn, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, zn, x2, sp, ump)</td>
<td>when(x1, sp, x2, zp, usn)</td>
<td>when(x1, sn, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, z, x2, sp, usp)</td>
<td>when(x1, zm, x2, mp, usn)</td>
<td>when(x1, zn, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, sp, x2, sp, usn)</td>
<td>when(x1, z, x2, lm, usn)</td>
<td>when(x1, sn, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, lp, x2, sp, usn)</td>
<td>when(x1, mp, x2, zm, usn)</td>
<td>when(x1, sn, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, mp, x2, sp, usn)</td>
<td>when(x1, zm, x2, mp, usn)</td>
<td>when(x1, zn, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, sm, x2, zp, ulp)</td>
<td>when(x1, sp, x2, sp, usn)</td>
<td>when(x1, mn, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, zn, x2, sp, ump)</td>
<td>when(x1, zm, x2, mp, usn)</td>
<td>when(x1, zn, x2, sp, usn)</td>
<td></td>
</tr>
<tr>
<td>when(x1, z, x2, zp, usp)</td>
<td>when(x1, sn, x2, sp, usn)</td>
<td>when(x1, zn, x2, sp, usn)</td>
<td></td>
</tr>
</tbody>
</table>