A neural-net and fuzzy-inference based method for fault detection and diagnosis in modern process systems

Wang, Gaige

2018

Wang, G. (2018). A neural-net and fuzzy-inference based method for fault detection and diagnosis in modern process systems. Master's thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/75527>

<https://doi.org/10.32657/10356/75527>

**A Neural-Net and Fuzzy-Inference Based Method for
Fault Detection and Diagnosis
in Modern Process Systems**

Wang Gaige

School of Electrical & Electronic Engineering

2018

**A Neural-Net and Fuzzy-Inference Based Method for
Fault Detection and Diagnosis
in Modern Process Systems**

Wang Gaige

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Master of Engineering

2018

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

Date

Wang Gaige

Acknowledgements

First, I would like to thank my supervisors, Prof. Cai Wenjian and my previous supervisor Prof. Wang Gang, for guiding me through my graduate studies. They have given me both patient guidance, constructive suggestions and thoughtful discussions in the path of pursuing my research goals. Their specialty standard and rigorous attitudes towards research will affect and benefit the rest of my life.

In addition, I am also grateful to the following lab colleagues: Dr. Luo Yunhui, Dr. Chen Jianda and Dr. Hong Wei for their unfailing support and assistance. Their views and ideas have inspired me a lot. They are not only excellent cooperators in research but also good friends in life.

Finally, I would like to express my special gratitude to my family for their love and support.

Table of Contents

Table of Contents	IV
List of Figures	VI
List of Tables	VII
Abstract	VIII
Chapter 1 Introduction	1
1.1 Background and Motivation	1
1.2 Literature Review	2
1.2.1 FDD Methods Overview	2
1.2.2 Review of Data Driven Based FDD	4
1.2.3 Review of Integrated FDD Schemes	4
1.2.4 Problems of Existing FDD Schemes	5
1.3 Contributions	7
1.4 Outline of Contents	7
Chapter 2 Methodology	8
2.1 Artificial Neural Networks	8
2.1.1 Multilayer Neural Networks	9
2.1.2 Learning Process	10
2.2 Fuzzy Inference System	14
2.2.1 Fuzzy Logic	16
2.2.2 Fuzzy Reasoning	19
2.3 Conclusion	20
Chapter 3 Framework Design	21
3.1 Ideas Behind the Scheme	21
3.2 Brief Description of the Framework	22
3.3 Collection of Training Data	23
3.4 Selection of Fault Indices	23
3.5 Design of Fault Detector	24
3.5.1 Data Preprocessor	25
3.5.2 System Modeling	27
3.5.3 Residual Control Chart	28
3.6 Design of Fault Classifier	29
3.6.1 The Comparative Explanation of How FIS Works	31
3.6.2 The Extraction of Fuzzy Rules	32
3.6.3 The Selection of Linguistic Values	33
3.6.4 The Construction of Membership Functions	33
3.7 Conclusion	34
Chapter 4 Case Study	36
4.1 Chiller Plant	36
4.1.1 Introduction of Chiller System	36
4.1.2 Faults and Effects	37
4.1.3 Fault Index Selection	38
4.1.4 Experimental Data	39

4.2	Implementation of the Proposed FDD Scheme	39
4.2.1	Input Feature Selection	40
4.2.2	Neural Network Configuration	40
4.2.3	Data Preprocessing	41
4.2.4	Training of Reference Models	43
4.2.5	Codification of Fuzzy Rules	43
4.2.6	Construction of Membership Functions.....	45
4.2.7	Architecture of Constructed Fuzzy Inference System	47
4.3	Experimental Results	47
4.3.1	Experimental Datasets	47
4.3.2	Results of Fault Detection and Identification	48
4.4	Comparison with Existing Methods.....	54
4.4.1	Configurations for Competing Schemes	55
4.4.2	Comparison I	56
4.4.3	Comparison II	59
4.5	Conclusion	62
Chapter 5	Conclusion and Future Work.....	63
5.1	Conclusion	63
5.2	Future work	63
References		65

List of Figures

Fig. 2. 1 The general architectures of two typical neural nets.....	9
Fig. 2. 2 The architecture of multilayer neural network.....	9
Fig. 2. 3 The surface shape of an ideal cost function	11
Fig. 2. 4 The trajectories of moving toward the surface minima.....	12
Fig. 2.5 The architecture of a fuzzy inference system.....	16
Fig. 2. 6 Commonly used membership functions.....	17
Fig. 2. 7 Linguistic values and their membership functions	17
Fig. 2.8 The illustration of the process of fuzzy reasoning	19
Fig. 2. 9 The example illustration of Mamdani inference process	20
Fig. 3. 1 The schematic diagram of proposed FDD scheme	22
Fig. 3.2 The schematic diagram of fault detector.....	25
Fig. 3.3 The distribution of two-dimensional data with probability contour lines	26
Fig. 3.4 Steady state analyses by tracking set points and present values	27
Fig. 3.5 The architecture of a single hidden layer neural network	28
Fig. 3.6 The schematic diagram of control chart.....	29
Fig. 3. 7 The structure of a FIS-based fault classifier	30
Fig. 3. 8 The typical architecture of a RBF Network	30
Fig. 3. 9 The space distribution of class specified data.....	33
Fig. 3. 10 The MF of linguistic value “ZERO” under normal operating conditions	34
Fig. 4.1 The Diagram of evaporation and condensation cycle	37
Fig. 4. 2 The architecture of neural network modeling all fault indexes in one model.....	40
Fig. 4. 3The architecture of neural network modeling one individual fault index	41
Fig. 4. 4 The contour of probability of density of bi-dimensional feature vectors.....	42
Fig. 4.5 The absolute percentage of deviation between SP and PV	42
Fig. 4.6 The fluctuation of the six residuals of each sample in the training dataset.....	45
Fig. 4.7 The membership functions of the residual of PRC	46
Fig. 4.8 The architecture of constructed fuzzy inference system	47
Fig. 4.9 The changing curve of TEI_TEO under different operating condition.....	49
Fig. 4.10 The changing curve of TO_sump under different operating conditions	50
Fig. 4.11 The residulas of six fault indices under normal operating conditions.....	51
Fig. 4.12 The residulas of six fault indices under 4 levels of severity of FEW.....	51
Fig. 4.13 The residulas of six fault indices under EO fault-existing conditions	52
Fig. 4. 14 Structure diagram of cascaded neural networks scheme.....	59
Fig. 4. 15 The architecture of single hidden layer neural network for fault classification	60

List of Tables

Table 4.1 The accuracy estimation of each of reference models	43
Table 4.2. The fuzzy rules codified for each fault.....	44
Table 4.3 The centers of MFs estimated for every fault index's residual.....	46
Table 4.4. The experimental datasets and involved data files from ASHRAE RP-1043.....	48
Table 4.5 The number of samples left after steady-state filtering	49
Table 4.6 The percentage of abnormal samples detected in each type of subset.....	52
Table 4.7 The accuracy of fault identification with a confidence threshold 0.25.....	53
Table 4.8 The accuracy of fault identification with a confidence threshold 0.50.....	53
Table 4.9 The accuracy of fault identification with a confidence threshold 0.75.....	54
Table 4. 10 Rules in fault diagnostic classifier for centrifugal chillers	57
Table 4. 11 The fitness for fault indexes of two reference models.....	57
Table 4. 12 False alarm rates of the two methods	58
Table 4. 13 Diagnosis results of the two methods for seven chiller faults	58

Abstract

The thesis proposes a quite novel and easily generalized fault detection and diagnosis (FDD) scheme for typical modern process systems, by integrating two powerful machine learning techniques: multilayer neural network and fuzzy inference system. With adequate sensors installed to and monitoring software widely applied to modern process systems, a vast amount of data relating to operating conditions of target processes could be recorded and stored, which provides potentials and foundations for applying new FDD methods. However, due to the facts that all these data are indiscriminate data without being labelled with tags (e.g., normal or abnormal), and that almost all these data are collected under fault-free operating conditions, the characteristics of unlabeled and unbalanced data of process systems also brings challenges for the application of new FDD methods.

By combining neural nets and fuzzy inference the two algorithms, one could make full use of the advantages (e.g., abundant fault-free data), and meanwhile avoid the disadvantages (e.g., few and unclassified faulty data) of the characteristics of process systems. Idea behind this is that, neural nets, especially multi-layer neural nets have powerful function approximating capacity and are able to automatically learn from a huge amount of historic data, thus it could be used to model system normal behaviors by feeding it enough fault-free operating data; meanwhile, fuzzy inference system could well incorporate expert knowledge as well as operators' maintenance experience about the target system in the form of fuzzy linguistic rules, and perform a human-like reasoning process to determine the cause of faults according to symptoms -- the deviations between observed system behaviors and model-predicting behaviors, hence dealing with the problem of lack of faulty data.

The basic procedures of the proposed scheme could be briefly described as: firstly using multi-layer neural network to build a reference model for the target system under healthy operating conditions; secondly, by comparing observed system behaviors and predicted behaviors from the built model, deviations in behaviors, called residuals, could be generated, and a preliminary fault detection is made based on out-of-limit principle; finally, if any fault is detected in last step, then the fuzzy inference system would further analyze the residuals to give an answer about which types of faults might have occurred with corresponding degree of certainty.

A case study on a typical process system, a chiller plant, is carried out where experimental data collected by ASHRAE Research Projects 1043 is utilized to validate the proposed FDD strategy. In addition, comparative study with another two typical FDD approaches is also conducted based on their respective experimental results on the same chiller plant and data to further demonstrate the effectiveness and superiority of the proposed scheme, and meanwhile to investigate the scope of application of the scheme.

Chapter 1 Introduction

1.1 Background and Motivation

In modern process systems engineering, process control systems have been widely applied to various fields out of consideration of plant efficiency and process reliability, from chemical process plant, nuclear power station, mass production lines to building air-conditioning. But even if equipped with highly developed and well-designed automated control systems, these plants often suffer from system degradation, instability, and shutdown even breakdown. It has been proved that most of these sufferings are caused not by control systems themselves but by unexpected faults which originate from control systems' participating components (e.g., sensors, actuators, and equipment). These faults have caused serious productivity decrease, energy waste, maintenance cost increase and even safety problems.

In addition, with scale expanding and complexity increasing, common modern industrial process systems are usually comprised of tens of large-sized automation equipment, hundreds of actuators, and thousands of sensors, accompanied with multiple inputs and outputs. Such a complex system, on the one hand, is vulnerable to faults and failures while, on the other hand, it is difficult, time-consuming and costly to identify, localize and remove faults and failure occurred in the system by means of pure manpower-based maintenance.

For tackling the challenges mentioned above, timely fault detection and accurate fault identification has become an urgent need and a hot issue in both industrial and academic fields. Timely detecting faults occurred in systems and then accurately localizing the faults means real-time monitoring a system, determining whether faults have occurred, which kinds of faults have occurred, further confirming where the faults are.

The method of detecting the occurrence of faults and then localizing the position of faults is usually formally called fault detection and diagnosis (FDD). FDD aims at real-time monitoring and predicting system abnormalities, and at automatically detecting and diagnosing early faults before systems become uncontrollable. A typical FDD-based operation and maintenance process includes the following four distinct functional processes [1]: 1) detect anomalies through monitoring performances of target systems; 2) diagnose the causes and identify the locations; 3) evaluate the severity level and make decisions; 4) make alerts or recommendations to the facility managers, and in some cases even automatically generate working orders or take corrective actions directly within control systems.

Recent years have seen an increasingly growing interest in researches and applications of fault detection and diagnosis as a result of the demand for higher performance, efficiency, reliability and safety in process control systems [2]. But most of existing schemes of FDD are only targeted

to a certain object or some particular systems or plants. Even though there are some proposals attempting to provide a general solution, they are either not quite feasible to implement due to lack of sufficient recognitions of practical systems, or merely provide a limited performance due to the constraints of methods they adopted. Therefore, providing a feasible and effective FDD strategy applicable to most of typical process control systems will be a meaningful research area exactly where I would like to research to go deep into.

1.2 Literature Review

FDD techniques focus on how to detect the occurrence of a fault as early as possible and how to identify the type of a fault as accurately as possible. In the past decades, a lot of effort has been put into the development of FDD schemes for applying to different industrial systems, for example, chemical plant, building air-conditioning and thermal power station.

1.2.1 FDD Methods Overview

Reviewed FDD methods can be roughly divided into three categories [3-5, 17]: signal-based methods, model-based methods, and data-driven based methods. Next, a brief introduction to each type of methods will be conducted.

- Model-based FDD

At the early stage, FDD method merely checks sensor measurements and system outputs with predefined limits. However, the simple over-threshold checking method becomes invalid as target systems become gradually complex. To overcome difficulties with limit checking raised by system complexity, model-based FDD was proposed based on the sophisticated state-space modeling and system identification techniques [7-8]. A parameterized mathematical model, derived from first principle of the target system, is the basis for all branches of this kind of method. Since a typical parameterized mathematical model mainly consist of state variables, system parameters as well as output variables, based on the derived explicit model, one of these three variables could be computed while taking the other two as known variables. That is, either state estimates, parameter estimates, or output estimates could be generated.

Due to the fact that the existence of faults tends to cause the changes in these variables, by comparing these estimated values with expected normal values, a particular residual pattern could be generated, based on which the type of a fault could be classified. Accordingly, three main branches of model-based FDD are respectively developed based on the estimation of each of these three variables, for example, parameter estimation method adopted in [9], parity relation approach employed in [12] and observer -based approach used in [10]. In addition, there have been not a few related works adopting the model-based FDD methods targeting to different application scenes, for instance, [11], [7], [13], [14] by Isermann, Patton, Frank, and Ding, respectively.

- Signal-based FDD

Signal-based FDD method was first proposed as a result of the dramatical development and improvement of signal processing techniques. The signal-based FDD method mainly investigates the correlation between signal patterns and faults. Since the output variables of the target system are usually directly influenced by the faults within the system, the signal used in the majority of signal-based FDD methods is the output variables. Therefore, unlike model-based methods, signal-based FDD is mainly based on the analysis of the output signals, and there is no need for building system reference models. According to the signal processing techniques and the types of signal patterns, the signal-based FDD methods could be categorized into three classes, namely, time-domain, frequency-domain, and joint-time-frequency methods. Verma et al. [19] proposed a time-domain signal-based FDD method for automatic diagnosis of rolling element bearing defects. As one of the most successful frequency-domain signal-based FDD applications, Li et al. [20] extracted motor current signals' frequency-domain features to form signal signatures, and then by analyzing motor current signatures, identified different types of faults occurred in electrical motors.

- **Data driven based FDD**

In recent years, with the rapid development of smart sensors, data storage techniques as well as communication networks, especially with the wide deploy of distributed control systems and integrated monitoring systems, the ability to collect and store a mass of process data has been remarkably improved [17]. What's more, aided by the gradually increasing computation power of computers and enabled by the recently booming artificial intelligence (AI) techniques, the capacity to handle and utilize data also has been strengthened. The improved data collection and storage ability as well as the strengthened data handling and utilization capacity provide potentials and foundations for applying new FDD methods, called data driven based FDD. Data driven based FDD, as its name implied, relies on data, more specifically, mass data or big data, to perform FDD.

The data driven based FDD methods employ AI techniques, especially machine learning algorithms, discovers the hidden information buried in enormous historical data, and automatically learn rule identification rules from labelled samples [22]. Unlike model-based FDD method and signal-based FDD method, the data driven based method relies on mass historical data and is entirely explicit model or priori patterns free. By far, many machine learning algorithms have been successfully introduced to FDD field to deal with the huge number of data. Typical examples are k-means, principle component analysis, neural networks, fuzzy logics, etc. For example, reference [23] suggested using k-means clustering technique to isolate different classes of faults in the aluminum smelting process, Wen et al. proposed principle component analysis based FDD applied to building air-conditioning systems for fault detection and diagnosis [24], reference [26] [27] [28] presented similar approaches to design feedforward neural networks to perform fault diagnosis in chiller plants, electric motors, chemical plants, and Chang et al. adopted fuzzy-logic based fault diagnosis strategy for feedback control loops [29].

Every type of FDD method has its own strength and shortcomings. For model-based FDD, it requires less data and computation, but demands an explicit mathematical model. However,

explicit mathematical models for most of large-scaled complex process systems is becoming harder and harder to derive according to first principles, which causes the failure of use of model-based FDD to many modern industrial systems [30]. For signal-based FDD, it avoids the procedures of derivation of reference models, and is usually simplified to merely extract signal signatures of system output signals. For data driven based FDD, it seems that it does not involve in exact system models or rely on priori patterns, but in fact, these knowledge and information are implicitly extracted from mass data by automated learning. Therefore, the performance of data-driven methods heavily relies on both the number of process data and the quality of these data.

Due to the explosion in system complexity and size in modern industry, either the derivation of exact system models or the summary of signal patterns is becoming more and more impractical. On the contrary, thanks to the development of sensor network techniques and machine learning methods, the problem of obtaining sufficient number of data as well as making sure the quality of collected data is gradually alleviated. Therefore, the data-driven FDD method has shown apparent advantages over model-based and signal-based methods recently [17]. As an alternative to the complex physical models, data-driven methods are derived only from readily available sensor data. It generates satisfactory performance under the big data background. This dissertation mainly focuses on studying data-driven FDD techniques.

1.2.2 Review of Data Driven Based FDD

Previous FDD works have explored a wide variety of machine learning techniques as data-driven methods, for instance, Primary Component Analysis (PCA) [24,31-34], Statistical Process Control (SPC) [12,13,35], multiple linear regression (MLR) [22], Bayes Classifier [36-39], Multilayer Neural Networks (MLP)[26-28,40], Fisher Discriminant Analysis (FDA) [41], Gaussian Mixture Model (GMM)[42-43], K-Means [23], Support Vector Data Description (SVDD) [44-45], and Support Vector Machines (SVM) [46-48]. Among these explored methods, PCA, SPC K-means, and GMM belong to the category of unsupervised learning that does not depend on prior knowledge for fault labeling, while others like MLP and SVM are supervised learning methods that require the availability of labeled fault samples as training data.

In supervised learning FDD, there should be abundant data which is first labeled with tags indicating the operating conditions and symptoms of target systems, such as ‘healthy’, ‘faulty’ (with specified fault type) [17]. The task of supervised learning algorithms is to find out the mapping relationship between data patterns and system conditions. In unsupervised learning, data provided is unlabeled. Unsupervised algorithms group pieces of data as a cluster according to the similarity between them. Different clusters usually associate with different types of faults.

1.2.3 Review of Integrated FDD Schemes

For one thing, from the perspective of functionalities of FDDs, a complete FDD scheme should be able to finish the two tasks, determining whether a fault existed, and identifying the type of the fault if a fault did exist, namely, fault detection and fault diagnosis, respectively. One single

technique is quite hard to complete two tasks at the same time (In theory, regarding normal conditions and all types of faulty conditions as a pattern recognition and classification task is feasible but experiments have demonstrated that its performance is somehow very poor perhaps because of lack of enough numbers of data or high dimensional input features). For another, from the perspective of techniques of FDD, FDD techniques have their strengths and shortcomings, especially when both the types of data and the number of data available are limited. Thanks to the reasons mentioned above, it has been a trend that many classic techniques are combined to achieve a better performance [15].

Many FDD schemes have been proposed targeting to different applications under different conditions. Basically, most of FDD schemes are composed of two steps, each step using an individual algorithm. First is to generate residuals, based on which fault detection is implemented; the second step is to classify residuals to confirm the type of a fault. For example, in [49], Vedam et al. proposed a principle component analysis (PCA) and signed digraphs (SDG) integrated scheme for automatically detecting and identifying faults in chemical plants. Fault detection is performed using PCA which generates residual vectors. Whenever an abnormality is indicated, residual vectors are presented as input to the SDG to perform fault diagnosis. In [50], Prabakaran et al. proposed a two neural networks cascaded scheme for fault detection and isolation in control valves. The two neural nets are used for fault detection and fault diagnosis respectively. The front one provides expected system behaviors, with which the observed system behaviors are compared to generate residuals, while the back one taking residuals as inputs classifies the type of a fault. Similarly, in [51], J. Cui proposed a multiple linear regression (MLR) and expert system combined scheme for fault detection and identification for centrifugal chiller systems, where the multiple linear regression algorithms work like neural networks offering references for observations, and expert systems comprised of rules are used for distinguishing different faults. In fact, not only are various data driven based FDD techniques combined, but also there also exist some schemes integrating methods crossover categories to alleviate the drawbacks of individual approaches, for instance, [52], incorporating both model-based and data-driven based FDD into a distributed aero-engine health-monitoring system. Because the focus of this thesis is mainly on data-driven based techniques, therefore no more references to signal-based or model-based integrated FDD schemes is given here.

1.2.4 Problems of Existing FDD Schemes

Although plenty of integrated FDD solutions have been proposed by previous literatures, most of them are targeted to and effective on a specific system under particular conditions. If applied to other systems or the same systems but different conditions, they often fail to work. Complete FDD schemes applicable for general process systems with general conditions are still rarely mentioned and researched in literatures. In particular, problems related to fault type diagnosis have long been ignored or over-simplified in existing works.

For better discussing the problems of existing schemes, the thesis first summarizes four typical characteristics of modern process systems: 1) despite being unable to be clearly expressed by the explicit mathematical formulas, the entire control system still has an underlying physical model

[23]; 2) with adequate sensors installed and data acquisition systems deployed, large volumes and high dimensions of process data relating to system states, inputs and outputs are available; 3) Due to the facts that all these data are indiscriminately stored without being labelled with tags (e.g., normal or abnormal), and that almost all of these data are collected under fault-free operating conditions, data of process systems are unlabeled and unbalanced; 4) based on long-term maintenance experience and records, common seen types of faults and corresponding symptoms could be easily surveyed.

According to the first feature of process systems, due to the existence of underlying physical models, the most accurate way of measuring the deviation of system behavior under different operating conditions is to compare the observed system behaviors with the normal ones which could be predicted by system reference models. In addition, according to the second feature, the availability of huge amounts of data under normal operating statuses provides foundation for learn-by-example regression algorithms. Therefore, for generating accurate residuals indicating the occurrence of faults and types of faults, the key point is to build a reference model for the target system by means of fitting data using machine learning algorithms. In existing works FDD schemes, PCA-based residual generation [50] is not suitable because it does not build a system reference model mapping between inputs and outputs but a statistical model representing data distribution, while MLR-based methods [22] have very limited approximation capacity for complex systems. The most appropriate method, as used in [51], may be neural networks based system modeling, especially multilayer neural networks which have the ability to appreciate any functions.

However, according to the third feature of typical process systems that only a few abnormal data could be collected in practice and that abnormal data are usually mixed with normal data, collecting and labelling data under each type of fault-existing condition is not realistic. It means in the phase of fault classification, the way of using faulty data to train a fault classifier to classify each type of failure does not make sense, unless just as [51] did, artificially introducing each kind of fault to the target system, and then collecting faulty data individually. But artificially introducing faults to systems is often costly, sometime infeasible because it not only stops plants' normal production activities but also likely causes damages to devices. Exactly because of the reason that there are only normal operational data and lack of kinds of labeled faulty data, the methods proposed for classifying residual patterns are still very rare. The classification methods adopted in [50] and [22], SGD and If-Then rules respectively, stand for the mainstream way of thinking. By making use of IF-THEN rules as the basis of fault diagnosis, relating deviation patterns to types of faults, it indeed eases the problem of the absence of faulty data to some extent, and works with a not bad accuracy in conditions that residual patterns caused by different types of faults are apparently different that classification is easy. However, in the scenarios where residuals change in the same direction only with a little discrepant magnitude, it works with unacceptable accuracies, and even does not work at all because of different faults' similar effects on residuals. In addition, the classification result of IF-THEN rules is also quite unsatisfactory, which can only give a binary result, true or false, rather than list probability of occurrence for each fault.

1.3 Contributions

The main contributions of this dissertation are based on the typical process systems oriented FDD method study. Firstly, through investigating the general characteristics of modern industrial process systems from the perspective of data, the thesis explains why most of existing FDD methods lack practicability and generalization. According to the data characteristics of typical process systems, a novel FDD scheme is proposed by combining two classic machine learning techniques, which possesses the theoretical potentials of tackling the challenges posed by lack of faulty data. Then the thesis designs the framework of the scheme and describe all technical details and procedures to implement the scheme. After that, for validating the feasibility and evaluating the performance of the scheme, a case study is conducted based on a chiller plant. The experimental results of the case study concretely demonstrate the effectiveness of the scheme. In addition, we also compare the scheme with another two currently popular FDD methods to further proves its superiority and to find out its application scope. The contributions are summarized as follows:

- 1) The proposal of a novel and widely applicable FDD scheme
- 2) The design of a framework for the scheme as well as the presentation of related techniques and procedures to implement the scheme.
- 3) The validation of the effectiveness of the scheme by a case study
- 4) The comparative study between the proposed method with currently popular two classes of FDD methods.

1.4 Outline of Contents

This report consists of five chapters, which are Introduction, Methodology, Framework Design, Case Study and Conclusion as well as Future Work Plans.

In the chapter of Introduction, the thesis briefly discusses the backgrounds and motivations of the research, review and commend previous researches and papers, and then at last sketch out the scheme we propose in the work which may tackle currently confronted some challenges when applying FDDs to typical process systems.

The remaining part of this report is organized as follows:

Chapter 2 introduces preliminary works about the basics of neural network based on which system is modeled, and the concepts of fuzzy inference system based on which fault diagnosis is performed in our following work. Chapter 3 gives a detailed description of the proposed framework, including ideas of such a design, implementation procedures, and some problems worth noticing. Chapter 4 takes a chiller plant as a concrete example to validate whether the method could work as well as how well it could work on a benchmark dataset, after which comparisons with currently popular two methods are also made to further prove the superiority of our method. Chapter 5 gives conclusions and future research works.

Chapter 2 Methodology

The proposed FDD scheme mainly adopts two kinds of classical machine learning techniques, artificial neural networks and fuzzy inference systems. This chapter will introduce the basics of neural network based on which system is modeled, and the concepts of fuzzy inference system based on which fault diagnosis is performed in the following work.

2.1 Artificial Neural Networks

Artificial neural network works by mimicking the structure and mechanism of human brain. According to the findings of neuroscientists, human brain is comprised of tens of billions of nerve cells called neurons which are connected to each other through wires called axons. Stimuli from external environment or from sensory organs are first accepted by outermost layer of neurons, and then the response of outermost layer is sent to more internal layers in such a way an original input could quickly travel through and be responded by the entire neural network.

Similarly, artificial neural network is also composed of plenty of such nodes, imitating the biological neurons. These neurons are structured in the form of layers and connected by links to neurons standing in their next layers. Each neuron can work individually, receiving inputs, performing arithmetic or logic operations, and sending its operation results to other neurons having interconnection relationships for further processing. Each interconnection or link is associated with a weight. By adjusting the values of weights, a neuron can output different results. Consequently, by adjusting all weights of the population of neurons, neural nets could dramatically differently respond or behave. In other words, if one wants a response from a neural net, just alter its weights accordingly. Moreover, what really makes neural nets powerful is that a neural net is capable of automatically adjusting weights to follow specified expectations or targets based on given stimulus or inputs by minimizing the errors iteratively. The process of automatically adjusting of weights is usually referred to as learning of neural nets.

Many types of neural network have been proposed and developed over the years, and each type of neural net has their own characteristics suitable for different applications. The most important two categories are referred as to feedforward neural network and recurrent neural network. The major differences between the two are that for the former, neurons in one layer can only connect to neurons in its next layer, namely, neurons linked in a single forward direction while for the latter, neurons in one layer are allowed to more freely connect, to next layers, or to previous layers and even to themselves. That is, feedforward neural nets allow signals to travel only one way: from input to output, having no feedback loop as shown on the right side of Fig. 2.1, but recurrent neural nets allow signals to travel in both directions as shown on the left side of Fig. 2.1.

Fig. 2. 1 The general architectures of two typical neural nets.

From the perspective of functionalities, the feedforward neural nets are ideally suitable for modeling relationships between a set of input variables and output variables. In other words, they are appropriate for functional mapping problems in which the tasks are usually to find out the quantitative relationships between input variables and output variables. Due to the existence of feedback loops, the outputs of recurrent neural network not only depend on current inputs but also previous outputs or states, so recurrent neural networks are used for modeling or recognizing kinds of sequence of tasks (e.g., time series predictions). In this work, since the aim is to model the relationships between inputs and outputs of target systems in steady state, therefore feedforward neural nets are chosen as the modeling algorithms.

2.1.1 Multilayer Neural Networks

Multilayer neural network is a kind of feedforward neural network which has an input layer, an output layer, and one or more hidden layers as shown in Fig. 2.2. By introducing hidden layer neurons, multilayer neural network overcome many restrictions of single layer feedforward network, such as the limited approximation capacity for nonlinear complex systems. In fact, researchers have demonstrated that a single hidden neural network has an ability of infinitely approximating any functions and that with increasing numbers of neurons in hidden layers or with deeper hidden layers, multilayer neural networks tend to have more powerful ability to easily fit more complex functions or to differentiate complicated patterns in classification tasks.

Fig. 2. 2 The architecture of multilayer neural network

Due to that process control systems commonly encountered are usually highly nonlinear and

rather complex, for succeeding in modeling such systems, a multilayer neural network with more than one hidden layers should be a better choice, for it has the potential capability of implementing any model equation or function to any desired degree of accuracy.

2.1.2 Learning Process

After talking about the building structure and characteristics of our model, neural nets, another most important issue remained is how to train and optimize the model by adjusting parameters to make it work as expected. To some extent, it could be said that the training and optimization of a model is more significant than the structure of the model.

Compared with conventional modeling methods, one of the fundamental advantages of neural networks is that they have the ability of automated learning and adapting. In short, learning occurs when feeding training examples into the targeted network. In the process, weights and biases are iteratively changed with a goal of minimizing the error between outputs and targets. Such a learning style is always called supervised learning where each training example is a pair consisting of an input vector and a desired output value, and training algorithms drive weights and biases to change in a direction that could make the output of network closer to the desired one and that could reduce the overall error.

2.1.2.1 Cost Function

For almost all AI-based models, the process of learning or training is virtually the process of minimizing errors. Errors are expressed in the form of a parametric function, called objective function or cost function in machine learning, denoted by $J(\theta)$, which is used to estimate the cost of producing an output or measuring the error between actual outputs and desired outputs over training dataset. The most commonly used objective function is a sum-squared error between predicted network outputs and expected outputs.

Given a training dataset:

$$X = \{x_1, x_2, \dots, x_n\}, \text{ and } Y = \{y_1, y_2, \dots, y_n\}$$

Where x_i represents an input vector, and y_i represents its corresponding target output.

Then the equation of cost function can be expressed as:

$$J(\theta) = \frac{1}{2n} \sum_{k=1}^n (h_{\theta}(x_k) - y_k)^2 \quad (2.1)$$

Where $h_{\theta}(x_k)$ is the actual output of the network given an input vector x_k under the model current parameters θ , meanwhile y_k is the desired output, and n is the number of training examples.

Although, $J(\theta)$ is seemingly involving three variables, inputs x , parameters θ , and target outputs y , it is merely a function of parameters, and its values only depend on the values of parameters because x and y are constant here when training dataset has been given. The ideal shape of surface of a cost function should be concave like a bowl as shown in Fig. 2.3, such that there is only one minima point and that the moving toward the minima point will have no other obstacles. But it is

hard to guarantee that we could choose a cost function exactly with such a surface shape, because the complexity of cost function explodes with increasing numbers of parameters.

Fig. 2. 3 The surface shape of an ideal cost function

2.1.2.2 Gradient descent

Gradient descent is a first-order derivative optimization algorithm which also known by the name of steepest descent or the method of steepest descent. The purpose of gradient descent algorithm is to make the error, namely the value of cost function, to be as low as possible, by iteratively updating model's parameters, including weights w and biases b . To minimize the cost function $J(w, b)$, we take partial derivatives of cost function $J(w, b)$ w.r.t. to weights w and biases b , respectively:

$$\begin{aligned}\nabla J_w &= \frac{\partial J}{\partial w} \\ \nabla J_b &= \frac{\partial J}{\partial b}\end{aligned}\tag{2.2}$$

As we stated above, what gradient descent does is update parameters in the direction of the surface slope of the cost function, namely, the gradient. As for how far it moves in the direction, it is up to the value of learning rate, μ , which is probably the reason why learning rate is often referred to as the length of stride in machine learning documentations. The process of updating weights and biases could be formulated as follows:

$$w_{\text{new}} = w_{\text{old}} - \mu \nabla J_w\tag{2.3}$$

$$b_{\text{new}} = b_{\text{old}} - \mu \nabla J_b$$

For a more general expression of updating rules in multilayer neural networks where there are many layers and each layer has a group of weights and a bias. For simplicity, we generalize the above equations in the form of matrix. To do so, the following equation is used:

$$\begin{aligned}w_{ji} &\rightarrow w_{ji}' = w_{ji} - \mu \frac{\partial J}{\partial w_{ji}} \\ b_j &\rightarrow b_j' = b_j - \mu \frac{\partial J}{\partial b_j}\end{aligned}\tag{2.4}$$

Where (j, i) represent the location of weights in the weight matrix of k th layer of the network, and the same notion for biases.

To order to find the minimum of an objective function using gradient descent method, one takes steps in the opposite direction of the gradient of the function at the current point, and iteratively moves toward a set of parameter values that gradually minimize the value of cost function, as shown in Fig. 2.4. The red arrows represent the trajectories in the way of finding the minimum point.

Fig. 2. 4 The trajectories of moving toward the surface minima

Stochastic gradient descent is an improvement for traditional gradient descent algorithm to speed up learning, where it randomly picking out a small number of examples, randomly chosen training input sets, making it a mini batch. If a size of m examples is selected as a batch to train networks, the following equation can be used:

$$\frac{\sum_{j=1}^m \nabla J_{x_j}}{m} \approx \nabla J \quad (2.5)$$

From the relation above, the following equation can be found:

$$\theta \rightarrow \theta' = \theta - \frac{\mu}{m} \sum_j \frac{\partial J_{x_j}}{\partial \theta} \quad (2.6)$$

2.1.2.3 Backpropagation

The fundamental training algorithm for feedforward multilayer neural networks is the Backpropagation algorithm. Backpropagation algorithm provides a prescription about how to change initial weights step by step so that outputs of the network being trained could approximate the desired values as much as possible. For using optimization methods, first one must work out the gradients of the objective function w.r.t. parameters (e.g., weights and biases), only after which one could change them according to update rules. Here is exactly backpropagation algorithm being used. The essential task backpropagation algorithm fulfils is to provide a means of computing gradients of the objective function with respect to parameters to be modified. Taking the minimization of objective function as a goal, combining with gradient descent optimization method, backpropagation is usually regard as not only a way of computing gradients but also a

way of training nets.

There are two passes in the process of implementation of backpropagation, called forward pass and backward pass respectively, which are described as follows:

In the forward pass, input is fed into the network and going through a series of forward calculation from input layer to output layer via hidden layers. At the output layer, it gives an output as the prediction result of the network, which differs from the desired output especially in the first period of training. The differences between actual outputs and desired outputs are then computed by cost function, which marks the end of forward pass and the beginning of backward pass. It is worth noting that in the feedforward pass, all of weights and biases remain unchanged.

In the backward pass, the error in each neuron of the output layer is first calculated, and then is propagated backward to the neurons of previous layers, and then to the neurons of previous layer of previous layer until the input layer. In other words, the error is propagated from output end to the input end in a layer-by-layer manner. During the backward pass, each layer's weights and biases are changed by the error feedback to the layer. The continuous modification of weights and biases is applied to make the real output of network bit by bit close to the desired one.

Next, a three-layer neural network with one output node is taken as an example to formulate and to illustrate the process of backpropagation learning in both forward pass and backward pass.

Forward-pass propagation:

i) Output of the j th neuron of the hidden layer:

$$a_j = f^j(\sum_i w_{ji} x_i + b_j) \quad (2.7)$$

ii) Output of the neuron of the output layer:

$$z_l = f^l(\sum_j w_{lj} a_j + b_l) \quad (2.8)$$

$$h = z_l$$

iii) Error of the neuron of the output layer given k th input sample:

$$e_k = h_k - y_k$$

Cost function by computing mean square errors of all n samples in training dataset:

$$J(\theta) = \frac{1}{2n} \sum_{k=1}^{k=n} (h_{\theta}(x_k) - y_k)^2 \quad (2.9)$$

Backward-pass propagation:

i) Derivation of output-layer neuron by means of cost function:

$$\begin{aligned} \frac{\partial J}{\partial w_{lj}} &= \frac{\partial J}{\partial z_l} \frac{\partial z_l}{\partial w_{lj}} \\ \frac{\partial J}{\partial w_{lj}} &= \frac{1}{n} \sum_{k=1}^{k=n} (h_{\theta}(x_k) - y_k) * f^{l'} * a_j \end{aligned} \quad (2.10)$$

Often let,

$$\delta_l = (h_{\theta}(x_k) - y_k) * f^{l'}$$

Thus, the above equation could be rewritten to:

$$\frac{\partial J}{\partial w_{lj}} = \frac{1}{n} \sum_{k=1}^{k=n} \delta_l * a_j \quad (2.11)$$

Similarly, the derivatives of output-layer bias could be rewritten:

$$\frac{\partial J}{\partial b_l} = \frac{1}{n} \sum_{k=1}^{k=n} \delta_l * 1 \quad (2.12)$$

Where δ_l is usually referred to as the “local error” in output layer.

ii) Derivation of hidden-layer neurons by means of cost function:

$$\begin{aligned} \frac{\partial J}{\partial w_{ji}} &= \frac{\partial J}{\partial z_l} \frac{\partial z_l}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \\ \frac{\partial J}{\partial w_{ji}} &= \frac{1}{n} \sum_{k=1}^{k=n} \delta_l * w_{lj} * f^{j'} * x_i \end{aligned} \quad (2.13)$$

Similarly, let δ_j as the “local error” in the hidden layer

$$\delta_j = \delta_l * w_{lj} * f^{j'}$$

Thus,

$$\frac{\partial J}{\partial w_{ji}} = \frac{1}{n} \sum_{k=1}^{k=n} \delta_j * x_i \quad (2.14)$$

Similarly, the derivatives of hidden-layer biases could be rewritten

$$\frac{\partial J}{\partial b_j} = \frac{1}{n} \sum_{k=1}^{k=n} \delta_j * 1 \quad (2.15)$$

iii) Updating weights according to the rules of gradient descent:

$$w_l \rightarrow w_l' = w_l - \mu \frac{\partial J}{\partial w_l} \quad (2.16)$$

$$w_{ji} \rightarrow w_{ji}' = w_{ji} - \mu \frac{\partial J}{\partial w_{ji}}$$

$$b_l \rightarrow b_l' = b_l - \mu \frac{\partial J}{\partial b_l}$$

$$b_j \rightarrow b_j' = b_j - \mu \frac{\partial J}{\partial b_j}$$

By updating weights and biases in such a way iteratively, a minima of cost function could be reached when the gradients of both weights and biases are approximately zeros. It works effectively for most cases but there are still some problems needed to take into consideration when training neural nets uses these kinds of methods.

2.2 Fuzzy Inference System

Fuzzy inference system is also known as fuzzy reasoning system, is operated based on fuzzy set theory and fuzzy reasoning method. Its most remarkable ability can incorporate and process fuzzy information. Fuzzy inference system taking fuzzy mathematic and fuzzy logic as the major calculation tool, can achieve a complex nonlinear mapping relationship between input and output while allowing the input and output are numeric values. Fuzzy inference is a behavior-based bionic reasoning method, which is mainly used to solve the complex reasoning problem involved with vague or inaccurate information. Because such information is very common fuzzy reasoning

systems are widely used from fuzzy control, fuzzy signal processing to the recent fuzzy computer vision.

Just like human reasoning which is based on known facts, basic logics and rules, fuzzy inference system incorporates expert knowledge about target systems in the form of IF-THEN rules expressed in natural language [53], building an inference model like what was constructed before using neural networks, converting inputs to outputs. According to the output type, fuzzy inference system could be grouped into two categories:

1) Mamdani fuzzy system

Mamdani fuzzy system has a form like this:

IF x is A **THEN** y is B.

Where x is the input, A and B are fuzzy sets, and y represents the output; for example, **IF** the heating power is high **TEHN** the temperature will increase fast.

2) Takagi-Sugeno fuzzy system

Takagi-Sugeno fuzzy system has a form like this:

IF x is A **THEN** $y = f(x)$.

Where x is the input, A is a fuzzy set, f is a function of x, and y represents the output; for example, **IF** the heating power, x is high **TEHN** the temperature, $t = 2 * x + 1$.

To meet the actual demands, the input and output of a fuzzy inference system must be numeric values, usually called crisp values in fuzzy field. But the values used for inference based on fuzzy rules are fuzzy values, therefore except for inference process. Therefore, there also should include a fuzzification - defuzzification process into the fuzzy inference system. The working mechanism of the fuzzy reasoning system could be briefly described as:

- 1) Firstly, the crisp values of inputs go through a fuzzification process, where inputs would be fuzzified and transformed into a value standing for degree of membership to a fuzzy set which is defined in universe of discourse.
- 2) Then in the rule base, all fuzzy rules would be activated to different degrees according to the degree of match of the input fuzzy facts to antecedents, after which by accumulating the output of each rule by means of a certain aggregation method, a fuzzy result could be obtained.
- 3) Finally, the fuzzy result goes through a defuzzification process where it would be defuzzified to obtain the final precise output.

From what was described above, one could know that a fuzzy inference system at least consists of three modules: fuzzification module, inference module, and defuzzification module, whose architecture can be sketched in Fig. 2.5.

Fig. 2.5 The architecture of a fuzzy inference system

Next, some important concepts applied to the fuzzy inference system will be introduced, which could help better understand the working mechanism, and better combine practical problems with fuzzy inference system.

2.2.1 Fuzzy Logic

Fuzzy logic is a form of many-valued logic where variables' values could range from 0 to 1. It is exploited to represent the concept of partial truth -- a truth value able to range from fully true (1) to fully false (0) [56]. In comparison, in Boolean logic, variables' truth values can only be either 0 or 1. Due to the facts that fuzzy logic allows for handling uncertain and imprecise knowledge and that it provides a powerful framework for reasoning, fuzzy logic has been extended to many fields, such as control theory and artificial intelligence.

2.2.1.1 Fuzzy set

Fuzzy sets were introduced by Zadeh in 1965 to represent and manipulate data and information in possessing non-statistical uncertainties [56]. The concept of a fuzzy set is an extension to classical concept of a crisp set. Fuzzy set theory generalizes classical set theory to allow partial membership.

Each element in a fuzzy set is a two-tuple pair which consists of an element from the data universe, and a degree indicating the membership of the element to the fuzzy set. Fuzzy set can be defined as follows:

$$A = \{ (x, \mu_A(x)) \mid x \in X \}$$

Where x is an element from a certain universe, and $\mu_A(x)$ denotes the degree of membership of the element x in fuzzy set A .

Membership function is a curve mapping each element in defined universe to a value between 0 and 1, where 0 represents that an element does not belonging to a set at all, 1 represents an element does belong to a set fully, and a value between 0 and 1 represents that an element partially belongs to a set. The membership function of a fuzzy set A is usually expressed as follows:

$$\mu_A(x) \in [0,1]$$

It means a fuzzy set A contains an object x to degree $\mu_A(x)$.

There are some commonly used membership functions, including Triangular MF, Trapezoidal MF, and Gaussian MF, their shapes shown as Fig. 2.6.

Fig. 2. 6 Commonly used membership functions

2.2.1.2 Linguistic Variables

A variable able to take words of natural languages as its values, is termed as a linguistic variable. These words are characterized by fuzzy sets defined in the universe of discourse where the variable is defined [58]. Since our human experience and knowledge are represented in the form of linguistic variables, being able to take linguistic variables as inputs becomes the crucial characteristic of fuzzy inference system in terms of highly incorporating expert knowledge to it. For example, when one makes use of sensors to measure a variable, it gives numerical values while, when asking a human expert to estimate a variable, he or she is most likely give us a word, like large or small. By employing the concept of linguistic variables, ambiguous natural language descriptions can be transformed into precise mathematical terms, which is the first and significant step to incorporate human knowledge into engineering systems [57].

As one usually uses more than one word to express different degrees in our daily life, a linguistic variable could take several linguistic items as its values. For improving the accuracy of fuzzy inference, more levels of linguistic values will be adopted, like very weak, weak, moderate, strong, and very strong, etc. Each linguistic value a variable could take is a fuzzy set defined in universe of discourse. Therefore, according to physical meanings of a variable, one should specify a membership function for each of its linguistic values. The membership functions specified for 5 linguistic values (e.g., negative high, negative low, zero, positive low, and positive high) of a certain variable is shown in Fig. 2.7.

Fig. 2. 7 Linguistic values and their membership functions

To order to ensure that input and output behaviors of the target fuzzy reasoning system are continuous and smooth, it is generally required that there are certain overlaps between adjacent linguistic values. The overlap of the membership functions also reflects the ambiguity and uncertainty of different levels of linguistic values. Designing in this way could improve the robustness of fuzzy reasoning system. Another question that must be pointed out is that it does not mean that the more levels of linguistic values is assigned to a variable, the higher inference accuracy can be achieved. It should be set based on the degree of distinction of fuzzy rules. Moreover, setting more linguistic values surely means there are more membership functions to be specified, increasing the complexity of fuzzy inference system.

2.2.1.3 Fuzzy rules

Fuzzy rules are the key component of a fuzzy inference system that can effectively model human expertise in the form of if-then rules. Fuzzy if-then rules are rules whose antecedents, consequences both are fuzzy rather than crisp. A simple fuzzy if-then rule with one-dimensional input has such a form:

IF x is A, THEN y is B

Where x and y are linguistic variables, A and B are corresponding linguistic values, “IF x is A” is called the antecedent part of the rule and “y is B” is called the consequent part of the rule.

The form of fuzzy rules for multi-dimensional input pattern classification problem can be written as follows:

R_q : If x_1 is A_{q1} and x_2 is A_{q2} and ... Then Class is C_q with w_q

Where R_q is the q-th fuzzy rule in fuzzy set, $\{x_1, x_2, \dots\}$ are input variables and $\{A_{q1}, A_{q2}, \dots\}$ are their corresponding antecedent fuzzy sets, C_q is the consequent output representing a class, and w_q is called degree of certainty.

Degree of certainty is a weight assigned to a rule, whose value is in the unit interval [0, 1], representing the degree of confidence to the cause and effect conclusion. Many research works [54, 23] have confirmed that rule weights have a significant influence on the result of classification results. Adding a weight to each rule helps improve the accuracy of fuzzy rule-based classifiers when keeping the membership function of each linguistic value unchanged. In addition, it also provides us an alternative, tuning the parameters of membership functions or tuning rules weights when a little data is available for further improving the accuracy of classifiers.

The consequent class is often referred to as dominant class which has a maximum value by accumulating the outputs of corresponding rules, the value usually called confidence. The confidence of class h could be calculated by the equation below:

$$c(\text{class } h) = \frac{\sum_{x_p \in \text{class } h} \mu_{A_q}(x_p) \cdot w_q}{\sum_{p=1}^{p=m} \mu_{A_q}(x_p) \cdot w_q} \quad (2.19)$$

Where m means there are m classes to be classified, and $\mu_{A_q}(x_p)$ is the compatibility grade of input x_p to antecedent part A_q . In this work, the minimum operator is used as

$$\mu_{A_q}(x_p) = \mu_{A_{q1}}(x_{p1}) \wedge \mu_{A_{q2}}(x_{p2}) \wedge \dots \quad (2.20)$$

Where $\mu_{A_{qi}}(\cdot)$ is the membership function of the antecedent fuzzy set A_{qi} .

In a fuzzy system, all rules fire to some extent. In other word, some fire fully while some fire partially. The firing degree is determined by the degree of compatibility between inputs and antecedents.

2.2.2 Fuzzy Reasoning

Fuzzy reasoning is the core of a fuzzy inference system, mapping the given inputs to an output through a fuzzy logics based reasoning process which consists of a collection of fuzzy if-then rules, linguistic variables and their membership functions. It projects crisp quantities onto fuzzy sets, then activates rules, accumulates their outputs, and at last yields a precise value as a result, whose entire process could be illustrated as Fig 2.8.

Fig. 2.8 The illustration of the process of fuzzy reasoning

As shown above, the entire fuzzy reasoning process could be divided into four steps:

1) Fuzzifying the input variables

Fuzzification is to take crisp inputs and to determine the degree to which these inputs belong to each of the appropriate fuzzy sets.

2) Evaluating the output for each rule

To evaluate the output of each rule, first one needs to apply the fuzzified inputs, to the antecedent part of fuzzy rules. If there exist more one antecedent, fuzzy operations (e.g., AND using minimum operator, OR using maximum operator) should be applied to them to obtain a single value, which is exactly aforementioned the degree of compatibility between inputs and antecedents. The output of each rule is a part of the original consequent membership function, which is obtained by cutting the consequent membership function at the level of compatibility degree.

3) Aggregating all rules' output

Aggregation is the process of unification of the outputs of all rules, where the outputs obtained for each rule in step 2) are combined into a single fuzzy set, using a fuzzy aggregation operator (e.g., maximum operator).

4) Defuzzifying to give a crisp output

Defuzzification is a reverse process of fuzzification, where a fuzzy set obtained in step 3) is transformed into a crisp value as the final output of a fuzzy inference system. There are several methods used for Defuzzification, including Centroid of Gravity method, Center Average method, and Mean-of-Maxima method. Among them, the most popular of is centroid method

which finds a point representing the center of gravity of the fuzzy set. The method can be mathematically expressed as follows:

$$Z_{COG} = \frac{\int_Z \mu_{C'}(z)zdz}{\int_Z \mu_{C'}(z)dz} \quad (2.21)$$

These four steps could be graphically illustrated by Fig. 2.9. As shown in the figure, there are in total two rules codified, each of which involves two input variables (e.g., x, y) and one output variable z. And each linguistic variable has two linguistic values, A_1 and A_2 for x, B_1 and B_2 for y, C_1 and C_2 for z. The two rules can be described as follows:

R1: IF x is A_1 AND y is B_1 THEN z is C_1

R2: IF x is A_2 AND y is B_2 THEN z is C_2

After going through fuzzification, evaluation, aggregation and defuzzification, the final inference result is outputted as a crisp value.

Fig. 2. 9 The example illustration of Mamdani inference process

It should be noted that in our pattern classification task, since the output variable in consequent part has been already a crisp value, therefore the inference process could be simplified into the first three steps of a general inference process.

2.3 Conclusion

In this chapter, two main methodologies are introduced technically as a warming-up before substantially using them to develop the proposed scheme in the next chapter. Firstly, multilayer neural network is comprehensively presented from what it is and how it works, to how to train it. In a nutshell, multilayer neural network has outstanding abilities of approximating complex functions by suitably adjusting its parameters, while backpropagation algorithm provides a prescription of learning its parameters automatically iteratively so that outputs of the network being trained could gradually approximate the desired values. Secondly, fuzzy inference system is introduced from the concept of fuzzy set, and the construction of membership function to the procedure of fuzzy reasoning. Being able to directly incorporate human prior knowledge as the foundation of fuzzy reasoning is of the central importance for fuzzy inference system to perform human-like decision-making and pattern classification.

Chapter 3 Framework Design

A novel two-step FDD scheme combining neural network and fuzzy inference system is proposed in this work. The proposed scheme, considering the characteristics of data collected from modern process systems, makes full use of the advantages of each of two algorithms, providing a feasible and effective solution for FDD in process systems. In this chapter, we will further explain the reasons why such a scheme is designed and then describe in detail the compositions and implementations of the proposed scheme.

3.1 Ideas Behind the Scheme

As put in [1], a FDD system could be thought of as a data processing system where the data and our understanding of the data are the two fundamental elements. First, let us talk about the data, which means what kinds of and what quantities of data our target system could generate and collect. Today's typical process systems, with the help of large numbers of sensors and real-time monitoring software, could generate and collect huge amounts of high-dimensional data. These data usually have two significant characteristics: one is that all these data are stored indiscriminately without being labelled with tags (e.g., normal or abnormal); the other is that almost all these data are collected under fault-free operating conditions because, after all, a fault only occasionally occurred in a system. Therefore, the data virtually available is always unlabeled and severely unbalanced, namely, a vast number of fault-free data and some unclassified faulty data. Our understanding of the data, in my opinion, is based on the understanding of characteristics of the data to choose appropriate algorithms or build suitable models which could make full use of the data to extract useful information as much as possible.

Now that there are large amounts of fault-free data available, then choosing an algorithm which could take advantages of it to build a normal system model is straightforward; meanwhile, due to the lack of faulty data, then that kind of classification algorithm that does not depend on labelled data should be desirable. Among kinds of classical or newly merging machine learning techniques, multilayer perceptron neural network (MLP) is one of the most suitable ones, for it has powerful function approximation capabilities and could readily learn from enormous numbers of data. That's why MLP is selected as system modeling algorithm in this work. As for building a classifier, most of algorithms are data-hungry, except for classical if-then rules and fuzzy logics based fuzzy inference system (FIS). As discussed above, classical if-then rule based classifier has a very limited classification ability, but in contrast, FIS has been proved a strong classification performance. Therefore, FIS is selected as fault classification algorithm in this work.

By combining the two methods, neural nets and fuzzy inference system, one could make full use of the advantages (e.g., abundant fault-free data), and meanwhile avoid the disadvantages (e.g., few and unclassified faulty data) of the characteristics of process systems. Idea behind this is that, neural nets, especially multi-layer neural nets have powerful function approximation capacity and

are able to automatically learn from a huge amount of historic data, thus it could be used to model system normal behaviors by feeding it enough fault-free operating data; meanwhile, fuzzy inference system could well incorporate expert knowledge as well as operators' maintenance experience about the target systems, in the form of fuzzy linguistic rules, and perform a human-like reasoning process to determine the cause of faults according to symptoms -- the deviations between observed system behaviors and model-predicting behaviors, hence dealing with the problem of lack of faulty data.

The basic procedures of the proposed scheme could be briefly described as: firstly using multi-layer neural network to build a reference model for the target system under healthy operating conditions; secondly, by comparing observed system behaviors and predicted behaviors from the built model, deviations in behaviors, called residuals, could be generated, and a preliminary fault detection is made based on out-of-limit principle; finally, if any fault is detected in last step, then the fuzzy inference system would further analyze the residuals to give an answer about which types of faults might have occurred with corresponding degree of certainty.

3.2 Brief Description of the Framework

From the perspective of structure, the proposed scheme mainly consists of two parts: a neural network (NN) based fault detector, and a fuzzy inference based system (FIS) based fault classifier. A NN is used for system modeling, providing a baseline for system behaviors. The reference model should have a high accuracy and a low uncertainty so that small behavior deviations could be used to find the early occurrence of a failure. A fuzzy inference system comprised of a collection of fuzzy rules is constructed to find out the type of fault once if a fault is detected. An overview of the framework is schematically depicted in Fig. 3.1. Note that a fault detector is composed of a NN based behavior predictor, a comparator, and a judger, while a fault classifier is completely comprised of a fuzzy inference system.

Fig. 3. 1 The schematic diagram of proposed FDD scheme

From the perspective of process, the proposed scheme goes through two key steps: residual generation and residual evaluation. And each step also could be divided into offline model training and online FDD using. For the first step, the offline portion includes: confirming inputs and outputs of the reference model and training the reference model by using historic data; the online portion includes: taking real-time data from the target systems and comparing it with

model-predicting ones to generate residuals. For the second step, the offline portion includes: extracting fuzzy rules and constructing a fuzzy inference system; the online portion includes: making preliminary residual analysis using control chart to determine whether there exists an out-of-limit behavior, and if existing, then performing further fuzzy reasoning using FIS to identify the type of fault.

3.3 Collection of Training Data

For any data-driven modelling method, data is one of the most important ingredients for constructing a decent system model, which, in some cases, tends to be more important than the algorithm itself. It means that collecting data should be paid at least as much as attention as selecting a modelling algorithm. For the proposed FDD scheme, in the first stage, system modeling using MLP is exactly a purely data-driven algorithm, which demands a huge amount of valid data to train constructed models to learn numbers of model parameters and to approximate target systems' underlying models.

For a typical process control system, it usually has multiple steady-state working points which means that the normal working conditions are not unique, and often frequently transfers between dynamic states and steady states which means that even though target systems stay under normal working conditions, only those data generated during the steady states can be collected and used for training due to transient data being invalid for building a steady-state system model. Compared with the latter how to obtain steady state data, the former how to obtain data which covers all normal working conditions is more challenging, since whether a system is under steady states can be quantitatively measured (discussed in following sections), but whether a system has gone through all possible normal conditions within a run cannot be measured or be guaranteed. But in order to cover as full range of normal operating conditions of target systems as possible, the following manipulations are often recommended to make:

- Varying system loads from extremely small to extremely large under the condition of system stability. For instance, in [61], the author tries to collect a chiller plant's operating data under different normal working conditions by purposefully changing chiller loads from 0 to 65 tons.
- Running target systems many times under different internal configurations (e.g., varied initial values of controlled variables), and under different external environment (e.g., various ambient temperatures) as done in [62].

3.4 Selection of Fault Indices

Before starting a FDD design cycle, for a specific target system, first, one should identify what kinds of system variables are most sensitive to and indicative of the occurrence of a fault. These fault-sensitive and -indicative variables are often referred to as performance indices (PIs) or fault indices (FIs), which could reflect the target system's operating status and could indicate the occurrence of faults of interest. And what's more, in fact, all the upcoming design works are carried out around the manipulating of these fault indices. Therefore, selecting appropriate fault indices means a lot.

There have some basic principles for the selection of fault indices. First, fault indices should be very sensitive to the occurrence and presence of a fault. In other word, faults could significantly affect the changes of fault indices. Second, fault indices should also be fault selective, which means they should discriminately respond to the occurrence of different faults, namely, for different types of faults, exhibiting different changes in terms of direction and magnitude. In addition, all the selected fault indices should be either directly measurable or explicitly calculable from other measurements.

As for which variables should be selected concretely, it depends on the target system and on the faults of interest. Usually, both target system researching experts and experienced system maintenance operators are able to point out which variables could be selected as effective fault indices for a successful FDD.

3.5 Design of Fault Detector

As illustrated in Fig. 3.2, the design of a fault detector consists of two parts, namely, offline model training and online FDD operation for the target system. Three important procedures involved in the offline reference model training are listed as follows:

- 1) Raw data preprocessing
- 2) Model construction and training
- 3) Residuals uncertainty estimation

The online using of FDD also could be divided into three corresponding steps:

- 1) Monitoring data preprocessing
- 2) Indices prediction and comparison
- 3) Residual preliminary analysis

Next comes to introduce the design details about those tools used to complete the corresponding tasks, including data preprocessor, system modeling, and residual control chart.

Fig. 3.2 The schematic diagram of fault detector

3.5.1 Data Preprocessor

Data preprocessor is comprised of an outlier data filter and a steady-state analyzer, which is utilized to clean raw data and to screen data that are suitable for training steady state system models.

3.5.1.1 Outlier Data Filter

In statistics, an outlier is an observation point that is distant from other observations [55]. Data directly collected and stored in databased is raw data, and occasionally contains outlier data due to measurement errors, incomplete recordings or temporary malfunctions, for example, a strange value with huge size in a sample of data. In addition, as mentioned above, a few faulty data are also stored together with healthy data, which could also be regarded as outliers since seeing from the perspective of a high-dimensional feature space, their clusters are typically distant from the clusters of normal data and are always marginalized by the majority of data.

In order to clear outlier data points, this thesis chooses an unsupervised learning based algorithm, Multivariate Gaussian Distribution (MGD), to realize automated multidimensional outlier filtering. For a multivariate Gaussian distribution of an n -dimensional vector, $X = [x_1, x_2, \dots, x_n]^T$ with mean $\mu \in R^n$ and covariance matrix $\Sigma \in S_{++}^n$, its probability density function could be expressed by

$$\rho(x; \mu, \Sigma) = \frac{1}{2\pi^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right). \quad (3.1)$$

Where the concept of the covariance matrix is quite crucial to understand a multivariate Gaussian distribution since the covariance matrix captures the spread and orientation of a set of n -dimensional data. Covariance matrix could be computed by the equation below:

$$\begin{aligned} \Sigma &= E[(X - \mu)(X - \mu)^T] \\ &= E(XX^T) - \mu\mu^T. \end{aligned} \quad (3.2)$$

Where X is a matrix consisting of all sample vectors.

The essence of MGD based outlier filter is to remove those samples of data which seldom occur in the population of data samples. What Gaussian distribution function does is compute the probability of occurrence of a sample in the population. Due to that faulty data, error data or incomplete data are absolute minority in the population; therefore, they could be easily removed by MGD. Fig. 3.3 shows a probability contour plot quantifying the distribution of data shown in Fig. 3.2., where from interior to exterior the probability of occurrence is decreasing. From the figure, one could see that normal data are fallen within inner circle lines with high density while outliers are located in outer circle lines with low density. Therefore, by setting a probability threshold, they can be separated easily.

Fig. 3.3 The distribution of two-dimensional data with probability contour lines

3.5.1.2 Steady-State Analyzer

It is worth to point out that for process systems, transients (e.g., overshoot or undershoot) occurring among different steady-state operating conditions account for a considerable proportion. Accompanying with transient states, there are transient data which means data collected under system transients. However, the proposed system modeling algorithm is only capable of capturing

the steady-state system characteristics since the output of a neural net only depends on current inputs, not relying on previous states of neurons. Therefore, before feeding data into the network, a steady-state detection step is a must to stop dynamic data from being learned by the network.

The simple and effective method of designing a steady-state analyzer is to just observe changes of controlled variables of the target system. For a typical control system, each controlled variable should have a set-point value. According to control principles, when the deviations between set points (SP) and present values (PV) are less than 5%, one could think of the system as reaching a steady state. Since both SP and PV might keep changing in a system, as shown in Fig. 3.4, one needs to keep track of them all the way when online FDD is performed in the target system. In addition, according to the numbers of controlled variables in a specific application, one also could properly reduce the amount of observed controlled variables because, after all, the conditions where all controlled variables strictly stay in steady states are actually rather rare for a complicated control system having a large number of variables. On the contrary, if there only a few, say, one control variable, one also could add some non-controlled variables whose changes follow the changes of system states, as the judgment basis of a steady condition.

Fig. 3.4 Steady state analyses by tracking set points and present values

3.5.2 System Modeling

After going through outlier filtering and steady-state screening, the data left can be considered as “good” data, good for the neural nets to learn effective parameters. But it is also a common practice to normalize data before they are fed into a network, in order to give all features same scale, avoiding a biased learning caused by the order of magnitudes of different features. But it is not an indispensable procedure.

System modeling is a process to find the underlying relations between inputs and outputs. The first step is to determine what as well as how many the inputs and outputs are. The outputs of the model are the selected fault indices as mentioned above. But what should be the corresponding inputs? All system variables except for target ones, in theory, could be used as candidates of inputs.

But only a subset of them are selected and used as model inputs, called features. In addition, some new features could also be extracted from the combination of original ones. All selected features form a feature space where each feature is a dimension in feature space. Each example in training dataset is a vector of features, which could be regarded as a point in the multi-dimensional feature space. Insufficient input features usually lead to a low-accuracy mapping, which means that outputs cannot be well represented in such a low-dimensional feature space, while excessive input features, on the contrary, always cause the curse of dimensionality, which means that data is becoming sparse in the overly high dimensional space, causing model over-fitting. Therefore, an appropriate number of features should be selected according to domain knowledge.

As for how many hidden layers and how many neurons in each hidden layer should be configured for a neural network, there is no rule of thumb. It is mainly dependent on the complexity of the target system and the volume of data available. Usually, one to three hidden layers, and five to ten neurons in each layer are the commonly seen specifications for a neural network. If m variables are selected as fault indices, and n variables selected as input features, then a single hidden neural network with p neurons in hidden layer could be illustrated in Fig 3.5.

Fig. 3.5 The architecture of a single hidden layer neural network

After determining the architecture of a neural network, the next step is to train it with data prepared in advance. Backpropagation combined with gradient decent is used as the training algorithm for optimizing the constructed network. The whole dataset is divided into three parts: training dataset, validation dataset and testing dataset in the proportion of 70%, 15% and 15%, respectively. Validation set is used to monitor the training data when the error in validation set reaches a minimum, the training process ends, while test data is used to test the trained model's performance in unseen examples.

3.5.3 Residual Control Chart

Model-based FDD methods are dependent on the determination if the residuals of fault indices exceed fault detection thresholds [51]. Residuals of fault indices should be zeroes ideally under normal operating conditions, but due to the uncertainty of reference models themselves as well as noise disturbances, residuals often fluctuate up and down within a small range. Therefore, thresholds are needed to determine whether residuals are normal or not. Fault detection thresholds are referred to as the upper limits or the lower limits of residuals under normal operating

conditions. If residuals exceed these thresholds, then a fault is thought to occur. Threshold estimation is usually developed based on the evaluation of the modeling errors.

Control charts are a statistical process control tool used to determine if a process is in state of control. As shown in Fig. 3.6, one can use it to determine if a residual is in a normal range. The center line (CL) for the control chart is the mean μ (it's close to zero in this work) of a kind of residual, and the upper control line (UCL) and the lower control line (LCL) are times of standard deviation σ of the residual, which could be expressed as:

$$UCL = \mu_0 + L \cdot \sigma \quad (3.3)$$

$$LCL = \mu_0 - L \cdot \sigma$$

Where L is the times of standard deviation, also the width of the control chart, which is usually set to be 3 (the corresponding confidence level is 99.73%).

Fig. 3.6 The schematic diagram of control chart

By statistically estimating the residuals of fault indices under normal operating conditions, one could obtain a pair of thresholds (i.e., LCL and UCL) for each kind of residual. Once a residual exceeds its thresholds in the phase of online FDD as shown in Fig 3.6 where the red point represents an out-of-limit behavior), it could be said that there might have existed a fault in system. Then the residuals are fed into the fuzzy inference system for further determination and identification.

3.6 Design of Fault Classifier

The generation of residuals has been completed by now. The task left is to make a fault classifier used for further analyzing and identifying what kinds of faults have occurred once a fault is detected in the detection phrase. It is an essential step for a successful FDD, only with which a FDD tool can tell system maintainers what the types of faults are more likely to occur, thereby helping engineers localize the faults efficiently and quickly.

The non-data-driven based classification of residuals often requires rules, classical if-then rules or fuzzy if-then rules. On the basis of a collection of rules, a decision begins to be possible to be made when residuals are given. Rules are from both standard literatures with relatively accurate numbers and human experience (e.g., experienced system maintainers or domain experts) [53],

which are usually expressed informally, inaccurately, with some uncertainties. Generally, the latter is highly specific for target systems, more useful for building a rule-based classifier, while the latter is also quite difficult even impossible to accurately code in classical binary rules, not to mention traditional mathematical equations. Thanks to this, fuzzy logic becomes a more appropriate choice than the other methods.

In this work, the fault classifier is comprised of a fuzzy inference system. A typical fuzzy inference system usually consists of a fuzzification module, an inference module, an aggregation module and a defuzzification module. One may replace the defuzzification module with a max selector, which chooses the one with maximum confidence value as the final output. The structure of the proposed fault classifier could be illustrated in Fig. 3.7. It has an n-dimensional residual vector as inputs, a fuzzifier comprised of numbers of membership functions, a rule base consisting of p fuzzy rules, and m classes of faults as the candidate output.

Fig. 3. 7 The structure of a FIS-based fault classifier

Fig. 3. 8 The typical architecture of a RBF Network

3.6.1 The Comparative Explanation of How FIS Works

As mentioned in the Chapter 2, in this work, there are mainly three steps involved in the process of FIS performing a fuzzy inference. From Fig. 3.7, the structure of the FIS classifier also can be decomposed into three layers: input fuzzification layer, hidden rule-activation layer, and aggregation output layer. To be specific, firstly in the first layer, a crisp input is transformed into a new representative, a degree of membership to a fuzzy set, which can be considered as a type of data mapping. Secondly, in the hidden rule-activation layer, rules in the rule base are fired to certain degrees according to the degree of compatibility between inputs and antecedents, which means that ‘mapped’ inputs activate rules to some degrees, depending on how closely the inputs resemble or meet the rules required inputs or conditions. Thirdly, in the aggregation output layer, by accumulating all weighted outputs of rules related to a class, the final score of the class could be obtained, which also could be rephrased as the (unnormalized) probability of inputs belongs to a class could be computed by linear combination of rule outputs and corresponding weights. Typically, a classification decision is made by assigning the input to the category with the highest score.

The reason why in the above paragraph, each module (or layer) of FIS is described doubly, namely, using two different kinds of technical expressions is that a comparison with a more commonly used and easier to understand classification algorithm named radial base function network (RBFN) is intended to be introduced for better explaining how a FIS classifier works. The second description for a FIS classifier is almost what each of three layers does in RBFN, particularly for the last two layers. A typical RBF Network consists of three layers, input layer, hidden activation layer and output layer, whose architecture can be illustrated in Fig. 3.8. A RBFN performs classification by measuring the input’s similarity with stored ‘prototypes’ in the hidden layer, and then linearly combines each similarity degree in the output layer as final outputs. Just like what each rule is activated to some degree in a FIS, each RBF neuron storing a ‘prototype’ vector is also similarly activated by inputs and outputs a value between 0 and 1 as a measure of similarity or a measure of degree of match. In other words, fuzzy rules stored in the rule base of FIS behaves like ‘prototypes’ stored in hidden neurons of RBFN, both of which measure to which degree the inputs comply with the specified ‘rules’. The last layer of both classifiers is used to aggregate hidden layer units’ outputs with corresponding weights and assigns each category a score.

One of reasonable approaches to selecting ‘prototypes’ when building a RBFN classifier is looking for the cluster centers of data in training set by means of clustering algorithms (e.g., k-means), which is exactly one of methods for extracting fuzzy rules for a FIS classifier when a little faulty data is available as discussed in the following section. In addition, as an alternative, randomly selecting data points from training set as ‘prototypes’, handed-crafted rules may seem more effective and less dependent on weights.

Through the comparison between FIS classifiers and RBFN classifiers, it may be revealed that a FIS classifier is nothing but one of nonlinear classifiers just like a RBFN classifier. The only difference is that it is unsupervised learning, or a kind of transfer learning, directly transferring human learned knowledge from a task to building it without fine-tuning.

It is worth pointing out that since a fuzzy inference system is a type of classification method, just like what traditional classifiers can do, it can only classify classes that are predefined and are specific in advance and cannot identify classes undefined but potentially existing. Therefore, FIS classifiers cannot capture all possible faults occurring in the target system unless all of them are specified beforehand.

In addition, another interesting and noteworthy property of FIS is that it allows for fine-tuning. If abnormal data is starting to be available after operating the FDD tool on the target system, the labelled faulty data can be used to improve the built FIS classifier. When the number of faulty data available is small, it can be used to fine-tune the parameters of membership functions (i.e., means and variances of Gaussian functions) of linguistic variables in the fuzzy rules or/and to fine-tune the weights (i.e., confidence values) assigned to fuzzy rules. When the number of faulty data is becoming relatively moderate, it can be used to extract new fuzzy rules more suitable for the target system, further enhancing the classification accuracy of the FIS classifier. From above two cases, it can be stated that fuzzy inference system is also compatible with online learning framework.

3.6.2 The Extraction of Fuzzy Rules

A fuzzy inference system is essentially a kind of expert system, which runs based on prior knowledge. For a mature plant in industry, there are always many specialized maintenance engineers, domain experts and well-documented maintenance records available, from which commonly seen system faults, their root causes, effects of each fault on system behaviors and so on, all could be extracted and summarized. These linguistic expressions about types of faults, root causes and their effects form the foundation of fault diagnosis, based on which one could write if-then statements, and codify fuzzy rules.

Apart from directly extracting rules by surveying human experts and reading domain articles, another effective and convincing method is to extract rules from data, especially labelled faulty data. Unlike training a fault classifier which needs a mass of faulty data, only when just a few labelled faulty data are available, we can either simply visualize these data in 2D or 3D space to look at the correlations between data distribution and corresponding fault classes or employ clustering algorithms to analyze data structures and to associate a certain structure with a certain fault type. For example, the following two rules might be extracted from some class-specified data's distribution in a residual1 and residual 2 comprised feature space as shown in Fig. 3.9.

Rule 1:

IF residual 1 is small and residual 2 is not large, THEN fault is class 1 with certainty 0.50.

Rule 2:

IF residual 1 is small and residual 2 is large, THEN fault is class 2 with certainty 0.80.

Fig. 3. 9 The space distribution of class specified data

Usually, for detecting faults of different levels of severity, two or more than two rules should be given for each type of faults, the first several for detecting early faults based on not quite significant symptoms with low degree of certainty as shown in Rule 1 or 3, where linguistic values used are words like “small”, the remaining rules for detecting serious faults based on significant symptoms with high degree of certainty as shown in Rule 2, where linguistic used assigned are words like “large”.

3.6.3 The Selection of Linguistic Values

A linguistic value is a natural language term which could be used as a value of a linguistic variable to describe the behaviors of a variable in a semi-quantitatively way. In general, these linguistic terms are categorized into three groups: primary terms, which are labels of fuzzy sets, for example, “small”, “medium”, “large”; complement, “not”, and connection “and” and “or”; hedges, such as “slightly”, “extremely”. In this work, according to the characteristics of residuals of fault indices (FIs), “positive” and “negative” terms are chosen to represent the direction deviation of FIs, and 3 levels of severity are utilized to represent the magnitude of deviation of FIs, namely, “small”, “medium” and “large”, and meanwhile using “zero” stands for a normal deviation.

3.6.4 The Construction of Membership Functions

After selecting linguistic values, one also needs to construct a membership function (MF) for each of linguistic values of each linguistic variable. The MFs of same linguistic value but different linguistic variables may differ largely. For example, as shown in Fig. 3.8, the MF for “small” used in residual 1, may considerably differ with the MF for “small” used in residual 2. In this work, Gaussian function is adopted as the base membership function for all linguistic values. A Gaussian function has a smooth shape, and its exact location and spread are determined by two parameters,

μ , σ , respectively, which means that for constructing a MF, one needs to estimate the two parameters for it.

Fig. 3. 10 The MF of linguistic value “ZERO” under normal operating conditions

For instance, in a certain application five linguistic values are employed, i.e., NL (negative large), NS (negative small), ZE (zero), PS (positive small) and PL (positive large). The centers (μ) of MFs of ZE under every variable are set as zero since the means of each of residuals under normal operating conditions tend to be zero (i.e., $\mu_{ZE} = 0$), as shown in Fig. 3.10. As for the centers of MFs of NS or PS, they can be set to be plus or minus 3 times standard deviations each of residuals under normal operating conditions, namely, $\mu_{NS/PS} = \pm 3\sigma$. But the centers of NL and PL are often chosen based on process knowledge or operation experience, for example, plus or minus 5 times standard deviations. Note that compared with other MFs, NL and PL are not the complete bell curve, but a half of bell curve, called left-open or right-open.

In addition, for a linguistic variable, its two adjacent MFs (e.g., PS and PL) are conventionally shifted to overlap each other to some degree, to keep the uncertainty. A typical overlap rate between two adjacent MFs is 0.5 [53]. By applying this constraint, the value of δ for each MF after knowing μ could be determined.

3.7 Conclusion

In this chapter, before diving deep into the introduction of the proposed FDD framework, the core ideas motivating such a design of framework is specifically stated. To be specific, NN-based reference model could provide a quite accuracy baseline for normal system behaviors, so when the target system operates in normal operating conditions, the residuals of selected fault indices are very close to zero, fluctuating around zero line up and down within a small range. But significant residuals are generated when a fault occurred since the underlying system changed. Given residuals, FIS could perform human-like logic reasoning based on prior knowledge represented in the form of fuzzy rules and decide what kinds of faults have occurred with a degree of confidence.

Therefore, the combination of the NN-based system modeler and the FIS-based pattern classifier has the potential of becoming an effective FDD tool.

After the statement of design thoughts, the framework architecture of the proposed FDD is described in detail, accompanied with major procedures and important techniques to implements the framework. The proposed framework mainly consists of two stages, namely, the design of MLP based fault detector, and the design of FIS based fault classifier. In addition, some necessary data preprocessing techniques are also introduced, including outlier filtering, steady state detector, to remove outlier data and dynamic data, respectively.

Chapter 4 Case Study

Last chapter has discussed the frameworks and procedures of the proposed FDD scheme. For validating the feasibility and effectiveness of the proposed scheme, subsequently, this thesis will take as a concrete example a typical process system, chiller plant, which is widely used in building air conditioning field in industry, to demonstrate the superiority of this scheme. In this chapter, a brief introduction to the chiller plant is given first, followed by concrete implementation of and applying of the constructed scheme to the target system; at last, this thesis will analyze the experimental results, and compare the results of the method with that of another two popular methods to investigate the strengths and shortcomings of the method compared with others.

4.1 Chiller Plant

Heating Ventilation and Air Conditioning (HVAC) system is an important part of any building structures, may it be residential structures, commercial structures, military structures, or even medical structures. In all HVAC system components, chiller is one of the most power-consuming components, which usually accounts for more than half of the total electricity consumption of a HVAC system. Hence, any fault that occurs in a chiller might increase the power consumption of the chiller and lead to inefficiency of the overall HVAC system. In order to maintain chiller's energy efficiency, faults occurred in a chiller plant are required to be detected and removed timely. To achieve this, in recent years, kinds of FDD strategies have been applied to chiller systems, and in fact, FDD researchers are tending to give more and more attention to this equipment.

4.1.1 Introduction of Chiller System

A chiller is a machine that removes heat from a liquid via a vapor-compression refrigeration cycle [38]. Refrigeration making use of substances called refrigerant, deprives a material or space of heat by means of making the temperature to be lower than its surroundings. Refrigerant absorbs unwanted heat from its surroundings, which then raises its own temperature into saturation temperature. Saturation temperature is reached when addition and reduction of heat causes a change of state. In this case, refrigerant changes its state from liquid to gas after absorbing unwanted heat, which is called evaporation process. As opposed to the evaporation process, a reverse process called condensation which is used to release the heat of the refrigerant, changes the gas-state refrigerant back to liquid state, thus keeping the cycle to be continued constantly. Fig. 4.1 illustrates the entire process.

Fig. 4.1 The Diagram of evaporation and condensation cycle

In the figure above, the aforementioned two key cycles occurred in chiller system, evaporation and condensation processes take place in evaporator marked in No. 4 and condenser marked in No. 2, respectively. The high pressure and high temperature vapor state refrigerant entering the condenser encounters cool water or air which is flowing or blowing across the outside of its tube where heat transferred from the refrigerant to the cooler water or air due to the existence of temperature difference, thus, heat being removed accompanied with changes in refrigerant state from hot gas state to cool liquid. As the low-pressure liquid and low temperature refrigerant enters the evaporator, warm water to be chilled comes across the refrigerant in the evaporator's finned coils, similarly, heat transferred from the warmer to the cooler, thus, the water being cooled and the refrigerant being "boiled", changing the state into low pressure gas state. From the figure, it could also be seen that except for condenser and evaporator, a chiller system also has another important component called compressor. Compressor compresses the low-pressure refrigerant vapor from evaporator into the high-pressure refrigerant vapor which is prepared for the entering of condenser. The leaving of evaporator and the entering of condenser are by means of two lines in compressor, one called suction line and the other discharge line, the former bringing low pressure vapor into compressor while the latter bringing the high pressure vapor out of the compressor.

4.1.2 Faults and Effects

Identifying the commonly occurred faults as well as their effects on the behaviors of the target system is one of the prepared tasks for a success of FDD. Only after knowing what kinds of faults are most potential to occur and knowing their effects on system behaviors, one could begin to select fault indices and to confirm their corresponding input features, to extract fuzzy rules and to link them to each of faults. Identifying and summarizing these faults are usually by means of surveying experienced system operators or inquiring domain experts.

For chiller systems, there have been already summarized commonly seen faults and their effects by pioneer researches, which greatly accelerate the usage of FDD in chiller systems, allowing us to mainly focus on the development of FDD algorithms. According to the survey of chiller operators conducted in [1], there are 7 commonly seen faults in a typical chiller system, which also have been previously mentioned in [24, 28, and 44]. These faults are listed below, and their corresponding effects are also described in a way that using “increase” or “decrease” represent the change in direction and using “slight” or “significant” represent the change in magnitude, given as follows:

- 1) Reduced evaporator water flow (FWE);
 - Significant increase in Condenser water temperature difference (TCO_TCI)
 - Slight increase in Condenser Pressure (PRC)
 - Slight increase in Power Consumption (kW)
- 2) Reduced condenser water flow (FWC);
 - Significant increase in Evaporator water temperature difference (TEI_TEO)
 - Slight increase in Evaporator pressure (PRE)
 - Slight increase in Evaporator approach temperature (TEA)
- 3) Excess oil (EO);
 - Significant increase in Oil temperature (TO_sump)
 - Significant increase in Compressor power (TO_feed)
- 4) Refrigerant leak (RL);
 - Slight decrease in Condenser pressure (PRC)
 - Slight or Significant decrease in Condenser approach temperature (TCA)
 - Slight or Significant decrease in Refrigerant sub-cooling (TRC_sub)
- 5) Refrigerant overcharge (RO);
 - Slight increase in Condenser pressure (PRC)
 - Slight or Significant increase in Condenser approach temperature (TCA)
 - Slight or Significant increase in Refrigerant sub-cooling (TRC_sub)
- 6) Condenser fouling (CF);
 - Slight increase in Condenser pressure (PRC)
 - Slight increase in Condenser approach temperature (TCA)
 - Slight decrease or increase in Refrigerant sub-cooling (TRC_sub)
- 7) Non-condensable (NC).
 - Slight or Significant increase in Condenser pressure (PRC)
 - Significant increase in Condenser approach temperature (TCA)
 - Slight or Significant increase in Refrigerant sub-cooling (TRC_sub)

There are in total 8 kinds of system abnormal behaviors observed under different types of faults, from which one could select the more sensitive and selective ones as the fault indices.

4.1.3 Fault Index Selection

As mentioned above, there are totally 8 variables involved, either from direct measurements (e.g., TCA, PRC, TRC_sub, TO_sump, TO_feed), or from indirect calculations (e.g., TCO_TCI, TEI_TEO, kW). By comparing their actual measurement or calculation values with reference

model prediction values, one could get their residuals. And the residuals of these variables could reflect the existence of potential faults.

There are some principles of selection of fault indices. On one hand, for the purpose of better classification of faults, one should try to select fault-sensitive and fault-selective variables as many as possible such that potential faults could be easily observed and discriminated. On the other hand, for building a relatively reduced system model and for simplifying the fuzzy rules, one also should try to select fault indices as few as possible for each fault. Therefore, there may always have to be a compromise between performance and simplicity. In this study, taking these two aspects into consideration, 6 out of 8 variables are selected as a fault index set for 7 potential faults. The fault indices selected are listed as follows:

- TEI_TEO
- TCO_TCI
- TO_sump
- PRC
- TCA
- TRC_sub

Among these fault indices, either a single one or combined several ones could not only indicate the occurrence of a fault and could discriminate the fault from the others.

4.1.4 Experimental Data

In order to validate whether the FDD scheme proposed by us has those advantages as claimed before, the data sets from a standard evaluation organization named ASHRAE are adopted in this study. Just like many benchmark datasets used for evaluating the performance of different methods in machine learning filed, such as CIFAR-10 and MNIST, the ASHRAE-provided specialized data set, named ASHRAE 1043-RP, has also been used by many researches [22, 24, 40, and 47] as a standard data set to study faults in chiller and to develop and evaluate FDD methods for chiller. This data contains both normal and seven typical faults data which were generated and collected through conducting experiments on a 90-ton water-cooled centrifugal chiller with an expansion valve. Within experiments, the system cooling load is designed to change in the range of 25% to 100% of the cooling capacity. Seven typical chiller faults are considered in the experimental tests. These faults are exactly those as introduced above defined by Comstock and Braun [2]. In experiments, these faults were artificially introduced, and each fault was monitored in four severity levels, i.e., from severity level 1 to severity level 4, which are measured in percentage. There are three datasets under each operating status, including “Complete dataset” which contains 5191 samples recorded every 10 seconds, “Recued dataset” which contains 433 samples recorded every 2 minutes, and “Steady-state dataset” which contains 27 samples corresponding to 27 operating conditions.

4.2 Implementation of the Proposed FDD Scheme

The implementation of the proposed FDD scheme mainly includes two steps, constructing and training system model, and building fuzzy inference system.

4.2.1 Input Feature Selection

Literature review showed that three variables have been widely used as input features for system modeling in many previous researches of chiller FDD [24,28,44,51], which are physically meaningful for chiller systems in terms of thermodynamics and energy conversation. These three variables are TEO, TCI, and Q_{ev} , which represent evaporator water leaving temperature, condenser water entering temperature, and cooling load, respectively. In addition, previous researches with varied system modeling methods also demonstrated that these three variables could effectively make up a feature space based on which lots of other system variables could be represented and approximated. Therefore, here this thesis also adopts these three variables as input features of the reference model to be constructed. After determining input features and fault indices, the next step will be to use a multilayer neural network to construct the system reference model.

4.2.2 Neural Network Configuration

From the perspective of inputs and outputs, the model should have 3 inputs, (TEO, TCI, Q_{ev}), and 6 outputs, (TEI_TEO, TCO_TCI, TO_sump, PRC, TCA, TRC_sub). The most straightforward method is to build a neural network with 3 inputs and 6 outputs as shown in Fig 4.2. Meanwhile, there is another modeling method -- building 6 neural networks, each one with 3 inputs and 1 output, namely, constructing an individual network for each of six fault indices as shown in Fig. 4.3. Despite that each of the two methods could achieve comparatively accurate reference models, there exist some differences. For example, the former has a simpler implementation style, which allows completing the task of system modeling in one operation, while the latter allows us to flexibly adjust the architecture for each network, and to get insight to individual modeling accuracy of each of these fault indices. In this study, for the convenience of checking every fault index's modeling uncertainty, the method of simultaneously constructing six neural networks with the same architecture, each one having three input nodes and one output node, will be chosen.

Fig. 4. 2 The architecture of neural network modeling all fault indexes in one model

Fig. 4. 3The architecture of neural network modeling one individual fault index

As for the number of hidden layers, in consideration of the volume of training data (roughly five thousand in the complete dataset), we utilize two hidden layers, in first hidden layer with 10 neurons and the second hidden layer with 5 neurons. In addition, tanh function is chosen as the activation function for neurons in two hidden layers while linear function chosen as the activation function for neurons in the output layer. After finishing the configuration of neural networks, next step will be to have it learn from samples.

4.2.3 Data Preprocessing

As mentioned above, in the dataset of ASHRAE 1043-RP, there are 3 kinds of datasets under fault-free operating conditions, “Complete Dataset” with 5191 samples, “Reduced Dataset” with 433 samples and “Steady-state Dataset” with 27 samples. For feeding neural nets with enough samples, this thesis chooses the “Complete Dataset” in the data sheet named “Normal” as the training dataset to train configured neural nets. It should be noted that not all data contained in selected dataset are valid for training, since they are just raw data, containing both outlier data and transient data. Therefore, before starting the formal training process, a data preprocessing procedure is needed to remove these invalid samples from the entire training dataset. There are two tools developed for getting rid of corresponding samples, outlier filter for removing outliers and steady-state analyzer for removing transients.

For developing the outlier data filter, multivariate Gaussian distribution algorithm is utilized to separate abnormal data from normal data. In the training dataset, each sample consists of 9 variables, including variables as model inputs and variables as model outputs. First, we take each sample in the training dataset as a point or a vector in a 9-dimensional space, and then compute the data population’s statistical parameters, including mean vector μ , and covariance matrix Σ . Then by applying probability density function to each sample, one could obtain the relative likelihood of the existence of each of samples in the training dataset. As mentioned before, in a raw dataset, almost all data are normal data and only a very small proportion may be faulty data.

And according to some researches like [55], the proportion of outliers usually does not exceed 2%. Therefore, one could transform the problem of outlier sample removal into the problem of clustering of samples, where data might be divided into two clusters, one cluster with very high likelihood of existence while the other with very low likelihood of existence.

In this study, in order to simplify the problem, just first 95% of samples with relatively high likelihood of existence are taken as non-outliers and the remaining 5% of samples with relatively low likelihood of existence in the training dataset are discarded. Due to the high dimensionality of feature vectors consisting of all variables, it is impossible to illustrate the distribution of samples in a 9-dimensional space, only 2 variables are taken out of 9 variables (e.g., TEI and Q_{ev}), as feature vectors to illustrate their distribution within which outliers are located at the edge of their probability of dense function. Fig 4.4 illustrates data distribution in the form of discrete circular rings where the wider of rings, the more data reside in. Data represented by the outmost dark blue ring is more likely to be discarded than the other. After outlier filtering, there are about 5000 samples left in the training dataset.

Fig. 4. 4 The contour of probability of density of bi-dimensional feature vectors

Except for outliers, dynamic samples also should be removed. One could make use of the conclusion in classical control principle that when the present values (PV) approach set points (SP) and stay within 5% deviation, the target system could be roughly considered as reaching a steady state.

Fig. 4.5 The absolute percentage of deviation between SP and PV

For a chiller system, the major controlled variable is generally the evaporator water leaving temperature (TEO). Therefore, one could monitor and track the relative changes between the set outlet water temperature (TEO_set) and the real outlet water temperature (TEO_real). When their difference keeps within 5% in six consecutive intervals, one could regard the six samples as steady state data and keep them in training dataset, otherwise, removing them out of the training dataset. The absolute deviation between TEO_real and TEO_set could be shown in Fig 4.5. After the procedure of steady-state screening, there are roughly 3300 samples left in the training dataset.

4.2.4 Training of Reference Models

The training of a neural network creates the nonlinear correlation between the inputs and corresponding output, which is done by adjusting weights and bias of the network so that the error between the targeted output and the actual output for all samples in the training dataset is minimized. In this study, backpropagation algorithm combined with gradient descent are adopted to update the weights and biases. The training for six neural networks is performed individually. The entire training dataset is also divided into three subsets with a specified proportion, training set with 70% (2300 samples), validation set with 15% (500 samples), and test set with 15% (500 samples).

For FDD, whether reference models built could predict system's behaviors with high accuracy or not is one of the key criteria to determine the performance of the FDD method. There are two commonly standards to assess the accuracy of a regression model, namely, mean square error in testing set and the coefficient of determination (denoted by R^2). The former's magnitude is affected by the magnitude of the outputs, since it is directly measuring the difference between actual outputs and desired outputs, while the latter is fallen between 0 and 1, indicating the extent to which the dependent variable (output) is predictable form independent variables (inputs). Here both results are shown in Table 4.1, to give a full picture of how accurate each fault index could be predicted.

Table 4.1 The accuracy estimation of each of reference models

Fault Index Name	Error	R^2
TEO_TEI	0.00059	0.9999
TCO_TCI	0.0033	0.9996
TO_sump	0.48	0.9813
PRC	0.226	0.9994
TCA	0.0729	0.9743
TRC_sub	0.094	0.9846

4.2.5 Codification of Fuzzy Rules

According to the description of causes and effects of all potential faults, and combing the principles of using fault indices, selecting linguistic values, and designing rules as mentioned in previous paragraphs, the thesis uses most fault-sensitive and fault-selective fault indices, sets 7 linguistic values for each fault index variable, and codifies two fuzzy rules with different degree of

certainties for each of the seven potential faults. These codified rules are described in Table 4.2.

Table 4.2. The fuzzy rules codified for each fault

Type of Faults	Fuzzy Rules	Degree of Certainty
FWE	1) IF R_{TEI_TEO} is positive small, THEN fault is FWE.	0.5
	2) IF R_{TEI_TEO} is positive {medium or large}, THEN fault is FEW.	1.0
FWC	1) IF R_{TCO_TCI} is positive small, THEN fault is FWC.	0.5
	2) IF R_{TCO_TCI} is positive {medium or large}, THEN fault is FWC.	1.0
EO	1) IF R_{TO_sump} is positive small, THEN fault is EO.	0.5
	2) IF R_{TO_sump} is positive {medium or large}, THEN fault is EO	1.0
RL	1) IF R_{PRC} is negative small or zero, and R_{TCA} is negative {small or medium or large}, and R_{TRC_sub} is zero, THEN fault is RL.	0.5
	2) IF R_{PRC} is negative small or zero, and R_{TCA} is negative {small or medium or large}, and R_{TRC_sub} is negative {small or medium or large}, THEN fault is RL.	1.0
RO	1) IF R_{PRC} is positive zero or positive small, and R_{TCA} is positive {small, medium}, and R_{TRC_sub} is positive {small or medium}, THEN fault is RO.	0.5
	2) IF R_{PRC} is positive {small or medium}, and R_{TCA} is positive {medium, large}, and R_{TRC_sub} is positive {medium or large}, THEN fault is RO.	1.0
CF	1) IF R_{PRC} is zero or positive small, and R_{TCA} is zero or positive small, and R_{TRC_sub} is negative small or zero or positive small, THEN fault is CF.	0.5
	2) IF R_{PRC} is zero or positive small, and R_{TCA} is positive {small or medium}, and R_{TRC_sub} is negative small or zero or positive small, THEN fault is CF	0.5
NC	1) IF R_{PRC} is positive {small or medium}, and R_{TCA} is positive {medium or large}, and R_{TRC_sub} is positive {medium or large}, THEN fault is NC.	1.0
	2) IF R_{PRC} is positive large, and R_{TCA} is positive large and R_{TRC_sub} is positive large, THEN fault is NC.	1.0

Firstly, all subscripted fault indices used in the table are not referring to the fault indices themselves but to the residuals of fault indices, which are computed by comparing the actual measured values with the predicted values of reference models. For instance, R_{PRC} , it means $Residual_{PRC} = Observed_{PRC} - Predicted_{PRC}$. Secondly, as described in the table above, this thesis merely uses each of three fault indices' residuals, R_{TEI_TEO} , R_{TCO_TCI} , and R_{TO_sump} , for indicating the occurrence of each of three faults, FEW, FWC and EO, while combining residuals of three fault indices, $\{R_{PRC}, R_{TCA}, R_{TRC_sub}\}$, to indicate the occurrence of each of the remaining

four faults, including RL, RO, CF, NC. Each rule in the table is assigned to an estimated degree of certainty, 0.5 representing for a relative low degree of uncertainty for confirming the occurrence of a fault with the corresponding symptoms, while 1.0 for a relative high degree of certainty. In addition, it is also worth noting that 7 levels of values are set to describe both the direction and the degree of residuals of fault indices. These seven linguistic values are negative {large, medium, small}, zero, positive {small, medium large}, where {"negative", "positive"} represent the deviation directions of residuals, {"zero"} represents the normal fluctuation of residuals, and {"small", "medium", "large"} represent the magnitudes of residuals.

4.2.6 Construction of Membership Functions

After codifying fuzzy rules for faults, there is another important thing left, constructing 7 membership functions (MFs) for each of the residuals of six kinds of fault indices. The thesis adopts Gaussian function as the basic shape of all MFs. As for the two parameters of each Gaussian MF is determined, for mean (μ), by the uncertainty of reference model of the corresponding fault index, and for variance (σ), by the adjacent MF half overlapping principle and engineering experience.

Fig. 4.6 The fluctuation of the six residuals of each sample in the training dataset

For determining the center of each Gaussian function, the first thing to do is investigate the residual of each fault index under normal operating conditions, which represents the uncertainty of each reference model. Figure 4.6 shows the fluctuation of the six residuals of each sample in the

training dataset, where the x-axis represents the number of samples, the y-axis represents the volume of deviation of residuals in form of percentage, each blue point represents a sample in training set, yellow line represents mean value of residuals, and two red lines represent plus and minus double times of standard deviation of residuals. In chapter 3, we put that in general, the centers of MFs are set to be several times of standard deviation, which could be seen as another measure of uncertainty. In fact, the values of centers depend not only on the uncertainty of reference models, but also on the changes in the fault indices themselves under normal and abnormal operating conditions. For example, although the uncertainty of the reference model of TEO_TEI is quite small, 1%, its actual deviation under faulty operating conditions could reach several tens of percent. Therefore, the centers of its corresponding MFs are set to be 10% for “small”, 20% for “medium”, and 30% for “large”, respectively. In contrast, the uncertainty of reference model of TCA is quite large, 20%, but its actual deviation under faulty operating conditions are also several tens of percent. Therefore, the centers of MFs are set to be a few times of uncertainty, which are 30%, 50%, 70%, for “small”, “medium” and “large”, respectively.

According to the actual deviation conditions of every fault indices in the chiller system as well as the uncertainty of reference models, the centers of MFs for each residual are set as shown in Table 4.3. Accordingly, the shapes of MFs of residuals of PRC could be illustrated as shown in Fig. 4.7.

Table 4.3 The centers of MFs estimated for every fault index's residual

Residuals/ Linguistic Value	MF Center of Zero (%)	MF Center of Small (%)	MF Center of Medium (%)	MF Center of Large (%)
TEO_TEI	0	10	20	30
TCO_TCI	0	10	20	30
TO_sump	0	5	10	15
PRC	0	5	10	15
TCA	0	30	50	70
TRC_sub	0	20	40	60

Fig. 4.7 The membership functions of the residual of PRC

4.2.7 Architecture of Constructed Fuzzy Inference System

After finishing the codification of fuzzy rules and the construction of membership functions, the fuzzy inference system for identifying the type of the fault occurred in the chiller system is also completed. Its architecture could be illustrated in Fig. 4.8. It has 6 residuals as inputs, a number $C \in \{1, 2, \dots, 7\}$ representing the class of fault as well as a decimal $cl \in [0, 1]$ standing for the confidence level of the fault as outputs.

Fig. 4.8 The architecture of constructed fuzzy inference system

In this study, in order to investigate the influence of the confidence threshold on both false alarm rate and classification accuracy, three common thresholds, 0.25, 0.50, 0.75 are set, respectively as the threshold of confirmation of a fault, meaning that only when the fault's confidence level is more than the threshold, a fault is finally confirmed and sent to the operators of the chiller system.

4.3 Experimental Results

In the previous section, the implementation of the proposed FDD scheme is completed by employing the fault-free data provided by ASHRAE to build reference models and by extracting chiller system knowledge from literatures to codify fuzzy rules and to construct membership functions. In this section, this thesis will apply the implemented scheme to the experimental data collected under fault-free operating conditions and under each of seven faults individually existing operating conditions to validate the effectiveness of the proposed FDD scheme.

4.3.1 Experimental Datasets

Except for validating its effectiveness of the proposed FDD, the thesis also plans to investigate the sensitivity of the FDD to faults; therefore, 4 levels of severity of faulty samples are used to individually combine with the normal data set to form 4 experiment datasets, which are named "Dataset of Severity Level 1", "Dataset of Severity Level 2", "Dataset of Severity Level 3", and "Dataset of Severity Level 4", respectively. The detailed composition of each experimental dataset

is shown in Table 4.4. Each experimental dataset is comprised of 8 types of samples, including samples under normal conditions, and samples under each of 7 fault-existing conditions. And the level of severity of fault in every experimental dataset is identical, and the higher level represents the more serious fault, for instance, level 1 for lowest level of fault, and level 4 for the highest level of fault. The data stored in ASHRAE RP-1043 “Reduced” data sheets is imported, which collected in the interval of 2 minutes from a chiller system. Originally, each data sheet contains rough 430 samples.

Table 4.4. The experimental datasets and involved data files from ASHRAE RP-1043

Sample Type	Dataset of Severity Level 1	Dataset of Severity Level 2	Dataset of Severity Level 3	Dataset of Severity Level 4
Normal	normal	normal	normal	normal
FWE	fwe10	fwe20	fwe30	fwe40
FWC	fwc10	fwc20	fwc30	fwc40
EO	eo14	eo32	eo50	eo68
RL	rl10	rl20	rl30	rl40
RO	ro10	ro20	ro30	ro40
CF	cf6	cf12	cf20	cf30
NC	nc1	nc2	nc3	nc5

From experimental Dataset 1 to experimental Dataset 4, each of them will be individually applied to the FDD scheme, to check both the effectiveness and the sensitivity of the FDD scheme.

4.3.2 Results of Fault Detection and Identification

The procedures of applying FDD to each experimental dataset could be briefly described as follows:

- 1) Data preprocessing: applying the steady-state analyzer to the target dataset, picking out those samples which are collected under system steady states;
- 2) Baseline evaluation: feeding the preprocessed samples into trained neural networks to evaluate their corresponding model-predicted values;
- 3) Residual generation: comparing the predicted values given by reference models with the observed values given within samples, to generate the residuals of each of six fault indices;
- 4) Preliminary detection: comparing these residuals computed in 3) with their corresponding thresholds of residuals which are 3 times of standard deviation of each kind of residual under healthy conditions, and taking those with any one dimension of residual exceeding their thresholds as abnormal samples.
- 5) Fault identification: feeding these abnormal samples detected in 4) into the constructed fuzzy inference system for further identification of fault, and then obtaining the identification results.

Next will show the processing or analyzing results step by steps.

- Data preprocessing

After going through steady state filtering, the number of samples left in each type of sample subset is listed in Table 4.5.

Table 4.5 The number of samples left after steady-state filtering

Sample Size	Dataset of Severity Level 1	Dataset of Severity Level 2	Dataset of Severity Level 3	Dataset of Severity Level 4
Normal	315	315	315	315
FWE	347	331	312	327
FWC	329	322	309	286
EO	270	279	276	263
RL	270	274	273	296
RO	261	261	242	222
CF	291	315	273	286
NC	259	241	259	229
The total Number of Samples	2342	2338	2259	2224

- Baseline evaluation

Here as shown in Fig 4.9 and 4.10, the thesis draws out their respective changing curves of two fault indices, TEI_TEO and TO_sump, to illustrate the baselines provided by respective reference models for the corresponding observed values under different operation states. The left side of each figure represents the normal operating conditions while the right side represents one of FEW and EO fault-existing conditions (faults used here are the third level of severity of faults). In these figures, the blue lines represent the actual values of fault indices, and the red dotted lines represent the predicted values of fault indices. From these figures, one could see that under normal operating conditions, the observed values are very close to the predicted ones while under fault-existing conditions, the observed values are far away from the predicted ones.

Fig. 4.9 The changing curve of TEI_TEO under different operating condition

Fig. 4.10 The changing curve of TO_sump under different operating conditions

- Residual generation

As shown in Fig 4.11, Fig.4.12 and Fig. 4.13, the three figures respectively show the residulas of six fault indices in three different operating statues, including normal operating conditions, FEW fault-existing conditions, and EO fault-existing conditions. Every point in these figures are representative of a sample in the dataset, where blue points stand for normal samples, magenta for abnormals. Two red lines in each subplot of thse figures stand for two reference thresholds for the correspodng residual. The upper one represents the upper threshold of a residual while the lower one represents the lower threshold of the residulas. In addition, the middle yellow lines, very close to zero, represent the mean values of every kind of residual.

It can be seen from Fig. 4.11 that almost all points in all of the six subplots are fallen within the range of upper and lower limits, which means that the residuals of these fault indices rarely deviate beyond its thresholds under healthy operating conditions. However, from Fig. 4.12 to Fig. 4.13, there exist at least one subplots in each figure where the majority of points are fallen beyond control limits, for instance, the subplot of residual of TEO_TEO in the upperleft of Fig.4.12, and the subplot of residual of TO_sump in the middle of Fig. 4.11. That means that under faulty operating conditions, at least one of these six residuals will change significantly and tend to deviate beyond its thresholds.

In addition, from the last two figures, one also could see that with the increase fault levels, the fault-susceptible residuals will change increasingly significantly, deviating farther away from their limit lines.

Fig. 4.11 The residulas of six fault indices under normal operating conditions

Fig. 4.12 The residulas of six fault indices under 4 levels of severity of FEW

Fig. 4.13 The residulas of six fault indices under EO fault-existing conditions

- Preliminary detection

The residual thresholds as marked using red lines in Fig. 4.11 to Fig. 4.13 were usually computed in the phase of reference model training and testing, when for making the statistical parameter more representative, samples in the whole dataset including training set, validation set, and testing set were involved. After comparing the residuals with their thresholds, this thesis takes those samples with one or more than one dimensions of residuals exceeding their thresholds as abnormal samples. That is, in the preliminary stage, as long as there is one residual out of 6 residuals deviating beyond its thresholds, the corresponding sample is thought as an abnormal sample. In the table 4.6, the percentage of abnormal samples detected in each type of sample subset of each experimental dataset is shown in detail.

Table 4.6 The percentage of abnormal samples detected in each type of subset

Abnormal Percentage	Dataset of Severity Level 1	Dataset of Severity Level 2	Dataset of Severity Level 3	Dataset of Severity Level 4
Normal	13.3%	13.3%	13.3%	13.3%
FWE	100.0%	100.0%	100.0%	100.0%
FWC	100.0%	100.0%	100.0%	100.0%
EO	84.4%	99.3%	100.0%	100.0%
RL	63.3%	79.2%	99.6%	100.0%
RO	81.6%	90.8%	95.5%	100.0%
CF	67.7%	73.3%	90.1%	99.3%
NC	100%	100.0%	100.0%	100.0%

From Table 4.6, one could see that in the normal sample subset, there are still about 13% of samples being detected as abnormal samples after going through preliminary fault detection. However, even in faulty sample subsets, there are a small proportion of samples not being detected as the abnormal, but following such a trend that with the increase in the level of severity, faulty samples are more likely to be detected, especially for fault types of RL, RO and CF. For fault types including FEW, FWC, and NC, from severity level 1 to severity level 4, all faulty samples are 100% detected, it might imply that these three types of faults could greatly affect their fault indices, driving fault indices far away from normal values and forming large residuals.

- Fault identification

For confirming anomalies detected in last step, these abnormal samples should be fed into the fuzzy inference system for further analysis, by identifying their most likely types of faults as well as their corresponding probabilities or confidence levels. From Table 4.7 to Table 4.9, in every table accuracies of fault identification for both the normal sample subset and 7 types of faulty sample subsets under 4 levels of severity are listed. To investigate the influence of the threshold of confidence level on identification accuracy and false alarm rate, the thesis also sets 3 common confidence thresholds, namely, 0.25, 0.50, and 0.75. From one single table, one could see that with a confidence threshold, what percentage of samples is correctly identified in each type of sample subset as well as the overall identification accuracy under a level of severity. For example, in Table 4.7, with confidence threshold 0.25, 88.5% of normal samples are finally identified as the normal, which means that the false alarm rate will be 11.5%. And the overall classification accuracy under each level of severity is roughly 55%, 75%, 85%, and 90%, respectively.

Table 4.7 The accuracy of fault identification with a confidence threshold 0.25

Identification Accuracy	Dataset of Severity Level 1	Dataset of Severity Level 2	Dataset of Severity Level 3	Dataset of Severity Level 4
Normal	88.5%	88.5%	88.5%	88.5%
FWE	89.1%	99.7%	98.7%	100.0%
FWC	76.3%	100.0%	97.4%	98.9%
EO	15.6%	82.8%	96.0%	98.9%
RL	23.0%	44.5%	89.7%	91.9%
RO	52.9%	66.7%	79.1%	78.4%
CF	43.0%	46.7%	58.3%	66.8%
NC	52.5%	71.8%	71.8%	100.0%
Dataset Overall Accuracy	57.3%	76.1%	86.5%	90.6%

Table 4.8 The accuracy of fault identification with a confidence threshold 0.50

Identification Accuracy	Dataset of Severity Level 1	Dataset of Severity Level 2	Dataset of Severity Level 3	Dataset of Severity Level 4
Normal	93.2%	93.2%	93.2%	93.2%
FWE	68.0%	99.7%	98.7%	100.0%
FWC	59.6%	100.0%	97.4%	99.0%
EO	6.3%	72.8%	96.0%	98.9%

RL	19.4%	33.6%	76.6%	82.4%
RO	35.3%	46.0%	72.7%	77.0%
CF	7.7%	9.2%	29.0%	51.8%
NC	47.1%	70.5%	70.0%	99.6%
Dataset Overall Accuracy	43.9%	66.7%	80.2%	87.9%

Table 4.9 The accuracy of fault identification with a confidence threshold 0.75

Identification Accuracy	Dataset of Severity Level 1	Dataset of Severity Level 2	Dataset of Severity Level 3	Dataset of Severity Level 4
Normal	97.6%	97.6%	97.6%	97.6%
FWE	0.0%	99.7%	98.7%	100.0%
FWC	6.8%	90.1%	97.4%	98.9%
EO	1.2%	8.7%	79.0%	97.7%
RL	6.0%	10.0%	25.6%	45.3%
RO	4.6%	8.1%	35.5%	78.4%
CF	2.4%	4.3%	8.0%	12.6%
NC	12.4%	32.4%	49.4%	97.0%
Dataset Overall Accuracy	16.9%	46.2%	63.4%	75.0%

From Table 4.7 to Table 4.9, one could see that the classification accuracy of samples is heavily affected by the level of severity of samples, the higher level of severity of fault, the higher classification accuracy, for example, in Table 9, 43.9% for level1, 66.7% for level 2, 80.2% for level 3, and 87.9% for level 4. In addition, it can also be seen from the three tables that the larger confidence thresholds used, the lower false alarm rates the scheme would generate, as shown in tables, 11.5% for 0.25, 6.8% for 0.5, and 2.4% for 0.75. However, with the increase in confidence threshold, the overall classification accuracy of the system is also adversely influenced, for example, the overall classification accuracy under 3th level of severity being 86.5% for 0.25, 80.2% for 0.50, and 63.4% for 0.75.

4.4 Comparison with Existing Methods

As mentioned at the introduction chapter, a variety of FDD methods have been proposed in the past years and not a few of them do exhibit satisfactory performances. However, some of them merely focus on the presentation of fault detection and leave subsequent fault identification unattended, which can only be viewed as FD (fault detection) rather than FDD (fault detection and diagnosis), for instance, the method of MLP-based fault diagnosis for actuators of multi-tank system presented in [60]; and although truly FDD methods, most of them are also seriously constrained to the fundamental assumption that there are both sufficient normal and sufficient faulty data being able to collect from target systems. A typical representative of this kind of

methods is the cascaded neural networks scheme proposed by Prabakaran et al. [50] for control valves FDD.

Fortunately, some researches realized the problem of practicability of those methods and proposed some more practical methods. For example, J. Cui [51] proposed the MLR and expert system combined FDD method for centrifugal chiller systems, using MLR for fault detection and using expert system for fault diagnosis. For example, Vedam et al. [49] proposed the PCA and signed diagraphs integrated FDD method for chemical plants in which fault detection is performed using PCA and fault classification are performed using the SDG. It is worth noting that this kind of methods also demand normal data to train system models, but they do not require any faulty data, instead, employing empirical knowledge as the basis of fault classification.

In this section, for comparing the performance differences between existing methods and the proposed one, this thesis will select two typical existing FDD methods, and apply them to the aforementioned chiller plant, respectively. There will be two groups of comparisons taken place, one happening between the proposed method and J. Cui [51] 's MLR and expert system combined method, the other between the proposed method and Prabakaran et al. [50]'s cascaded neural networks method. Both selected methods are representative in research and commonly used in practice.

In addition, for better evaluating and comparing the performance of tested methods, the thesis makes use of the two metrics, false alarm rate and fault diagnosis accuracy, to measure how well a method works. It should be clarified again that false alarm rate is the ratio between the number of normal samples classified as faulty samples and the total number of normal samples. A FDD method with high false alarm rate will alert numerous false fault alarms which always brings big troubles to operators. A low false alarm rate alone cannot testify the effectiveness of FDD method because it may mean classifying all samples including truly faulty samples as normal samples. Therefore, fault diagnosis accuracy is used as a tradeoff. Fault diagnosis accuracy is the number of truly-classified faulty samples over the total number of faulty samples, which indicates the diagnosis precision on abnormal samples.

4.4.1 Configurations for Competing Schemes

Before starting the two pairs of comparisons mentioned above, there are configuration instructions applied to both two competing methods needed to explicitly list for conducting fair, convincing and meaningful comparisons.

In the phrase of system modeling, the following several crucial training and evaluating configurations will be kept the same with the proposed system modeling method:

- Selecting the same number and the same types of system variables as model inputs.
- Selecting the same or more amount of training data for training models (in the consideration that perhaps some algorithms such as MLR need more training data to avoid overfitting).
- Applying similar data preprocessing methods and procedures (i.e., outlier filtering, steady state detection, and data normalization) to training data before feeding them into models.

- Selecting respective appropriate optimization algorithms (e.g., for MLR using least square method, and for MLP using backpropagation) to optimize respective models for several times and then choosing that one giving best performance as the final trained model, for trying to make sure that the power of each model has been fully maximized.

In the phrase of building fault classifiers, for the first group comparison, due to both fuzzy inference system and traditional if-then rule based expert system using the same prior knowledge merely with different representation forms, the thesis will make the equivalent rules for the expert system according to the Faults and Effects described in Section 4.1.2. But for the second group of comparison, since the data-driven based fault classifier is almost totally different from the proposed FIS classifier in terms of data usage vs rule extraction, model training vs parameter estimation, there are very few things one can do to make them a fair comparison. What's more, the main purpose of this group of comparison is also not on strict performance contrast under a level playing field but on investigating the scope of application of the proposed method when labelled faulty data is starting to become available.

4.4.2 Comparison I

In this subsection, the MLR and expert system combined method will be briefly described and then implemented according to the configuration described above to try to keep the comparison fair and meaningful.

4.4.2.1 Introduction of Method I

It is aimed to generate accurate benchmarks of the fault indexes that reference models of these proposed fault indexes are built, characterizing the fault-free operations of a chiller system [51]. As stated in experiments of previous section, the thesis also uses the three variables (TEO, TCI, Q_{ev}) as model inputs, and use these variables (TEI_TEO, TCO_TCI, TO_sump, PRC, TCA, TRC_sub) as model outputs. Instead of using multiplayer perceptron neural networks (MLP) to model system, the method adopts multivariate linear regression (MLR) to perform system modeling. The chiller model is presented as follows [51]:

$$\begin{aligned}
 Y &= f(\text{TEO}, \text{TCI}, Q_{ev}) + \varepsilon \\
 Y &= b_0 + b_1 * \text{TEC} + b_2 * \text{TCL} + b_3 * Q_{ev} \\
 &\quad + b_4 * \text{TEC} * \text{TCL} + b_5 * \text{TEC} * Q_{ev} \\
 &\quad + b_6 * \text{TCL} * Q_{ev} + b_7 * Q_{ev}^2 + \varepsilon
 \end{aligned} \tag{4.1}$$

Where Y is the model output are assigned to be these six fault indexes of concern, ε is the error term following residual normal distribution $\varepsilon \sim N(0, \sigma^2)$, and coefficients ($b_0, \dots, b_7$) are the model parameters which can be obtained from regression training by applying the popular least square method.

As mentioned above, classic if-then rule based expert system is adopted to classify seven different types of faults according to six fault indexes different changes under their impacts. The characteristics of the fault indexes will change when a fault occurs. It must be noted that although the changes of fault indexes include both changes in the directions and in the magnitudes, only the

changes in directions are used since it is hard to quantitatively indicate variations of magnitudes. Table 10 lists the rules used to link the seven faults to the six fault indexes, where, the sign “—” represents that there is no noticeable trend found in the fault index, “ ∇ ” represents that the fault index tends to decrease as the severity of a fault increases, and “ Δ ” represents that the fault index tends to increase as the severity of a fault increases.

Table 4. 10 Rules in fault diagnostic classifier for centrifugal chillers

Types of Faults	Classic Rules				
FWE	R_{TEI_TEO}	Δ			
FWC	R_{TCO_TCI}	Δ			
EO	R_{TO_sump}	Δ			
RL	R_{PRC}	—	R_{TCA}	∇	R_{TRC_sub} ∇
RO	R_{PRC}	—	R_{TCA}	Δ	R_{TRC_sub} Δ
CF	R_{PRC}	—	R_{TCA}	∇	R_{TRC_sub} —
NC	R_{PRC}	Δ	R_{TCA}	Δ	R_{TRC_sub} Δ

4.4.2.2 Comparisons with the Results of Method I

Just as this thesis adopts, the method developer J. Cui also adopts the data set ASHRAE 1043-RP to determine model parameters, and he chooses using least square method to optimize the model instead of gradient descent technique probably because the model to fit is linear and least square method is already capable. Next comes to use the same dataset as used in the processes of training and testing the multilayer neural networks model to fit the MLR model. These data will also go through the same data preprocessing procedures to remove outliers and dynamic samples. But in order to keep the comparison concise, this section will not investigate the sensitivity of the FDD method again. Therefore, faulty samples from four levels of severity will be equally selected and then evenly mixed together as a united test set.

- Comparison of Modeling Accuracy

Table 4.11 shows the fitness for six fault indexes of multiple linear regression (MLR) model represented by the R-square values (R^2). Meanwhile, for comparing with multilayer perceptron neural network (MLP) model, the fitness of MLP model is also listed again.

Table 4. 11 The fitness for fault indexes of two reference models

Reference models	TEO_TEI	TCO_TCI	TO_sump	PRC	TCA	TRC_sub
MLR	0.9996	0.9995	0.9741	0.9903	0.9486	0.9524
MLP	0.9999	0.9996	0.9813	0.9994	0.9743	0.9846

From the table above, it can be found that MLR model could very well model the behaviors of six fault indexes and that in comparison, the accuracy of MLP model is slightly better than that of MLR model, especially for fault indexes TCA and TRC_sub.

- Comparison of False Alarm Rate

For increasing generalization and reliability, instead of using 430 samples, more samples are selected from normal dataset and fed into the MLR models. Similarly, by comparing the predicted values with actual values, residuals are generated. Residuals exceeding control limits are inputted

into the rule based expert system to determine which fault types it belongs to. If no fault class is assigned, then samples are regarded as normal samples. The experimental results of both this method and the proposed method (taking 0.5 as confidence level) are shown in Table 4.12.

Table 4. 12 False alarm rates of the two methods

Tested methods	No. of normal samples	No. of misclassified samples	False alarm rate
Method I	3,000	346	11.53%
Proposed	3,000	162	6.40%

The table above demonstrates that the proposed method has a much lower false alarm rate than that of the method I. The false alarm rate of the method I is almost two times of the proposed one, which convincingly justifies the superiority of the proposed method in terms of fault detection.

- **Comparison of Fault Diagnosis Accuracy**

For each of seven faults, 400 samples will be evenly selected from its corresponding four severity levels of faulty datasets, namely, 100 samples per fault type per severity level, and 2,800 samples in total. Every sample will be assigned either a normal class or one of seven fault classes after going residual generation and residual identification processes. The fault classification results of both methods are shown in Table 4.13.

Table 4. 13 Diagnosis results of the two methods for seven chiller faults

Tested methods	FWE	FWC	EO	RL	RO	CF	NC	Overall
Method I	100%	70.20%	18.48%	100%	14.36%	10.04%	100%	59.01%
Proposed	91.82%	90.54%	76.33%	63.80%	63.19%	40.26%	83.64%	72.80%

From the table above, it could be seen that for some fault types like FEW and NC, Method I could achieve a 100% diagnosis accuracy but for some other fault types like RO and CF, its diagnosis accuracy drops to about 10%. The reason may be that classic if-then rules based expert systems are usually confined to make use of the changes of directions of fault indexes. If two faults cause the identical changes in directions but different changes in magnitudes, then expert systems will fail to distinguish the two faults just like the case of fault RO (14.36%) and fault NC (100%), both of which increase the values of fault indexes (e.g., TCA, PRC, TRC_sub) to different extent, but the classifier owes most of such a condition to fault NC. In contrast, fuzzy classifier seems easier to tackle this kind of case (RO 63.80%, NC 83.64%). From the perspective of overall diagnosis accuracy, the proposed method could achieve 72.8% accuracy while the method I only reaches 59.01%. The proposed method significantly outperforms the method I.

4.4.2.3 Conclusions of Comparison I

From above comparisons of experiment results, the proposed method could achieve lower false alarm rate (6.4% vs 11.5%) and higher fault diagnosis accuracy (72.8% vs 59.0%) compared with Method I. Therefore, the conclusion could be drawn that the proposed method has remarkable advantages over the method I in terms of false alarm rate and diagnosis accuracy.

4.4.3 Comparison II

In this work, although our discussion is mainly focused on FDD methods for systems without faulty data or systems with no enough faulty data to train data-driven classification models, it will be interesting and beneficial to investigate the boundary, within which the proposed method works better than data-driven based methods, and outside which data-driven based methods begin working better. After all, it must be admitted that data-driven based methods possess powerful learning ability and classification potentials and that as the number of labelled faulty samples available increases, data-driven based methods will surely perform better and better while the proposed method is bound to fail to keep such a rising trend. In this subsection, a two-stage cascaded neural networks strategy as a typical representative of data-driven methods will be briefly introduced and implemented to find out the number of faulty samples above which the advantages of the proposed method will start to lose compared with the selected data-driven method, named Method II.

4.4.3.1 Introduction of Method II

The cascaded neural nets method consists of two individual neural networks. The first one is a regression model for system modeling while the second one is a classification model for pattern recognition. On runtime, the former stage provides expected system behaviors with which the observed system behaviors are compared to generate a residual vector, while the latter stage taking the residual vector as inputs relates a residual pattern to the most likely fault type. The structure diagram of the method could be depicted in Fig. 4.14.

Fig. 4. 14 Structure diagram of cascaded neural networks scheme

For the convenience of the subsequent comparison, the first neural network will be designed to have the same architecture with the one used in previous chiller plant experiments as shown in Fig. 4.3. As for the second neural network, for keeping the universal approximation potential of neural works and meanwhile keeping the number of parameters as less ass possible to prevent it from overfitting, a single hidden layer neural network with 5 hidden neurons is adopted as the classifier. Its architecture is shown in Fig. 4.15. It takes as inputs the residuals of 6 fault indexes and outputs 8 score values for the normal class and 7 faulty classes, respectively.

Fig. 4. 15 The architecture of single hidden layer neural network for fault classification

4.4.3.2 Comparisons with the Results of Method II

For observing the performance change curves of Method II as the number of training data gradually increases, the thesis will make 10 groups of experiments of varied sample size (not including samples used for system modeling which is fixed and sufficient). It is assumed that the maximum number of samples available to train classifier is 50,000 and that these samples are evenly selected from 8 classes including one normal class and seven fault classes, namely, 6250 samples per class. And for every fault class, its samples are also equally taken from 4 levels of severity. The original sample size is set to be 5,000 in the first experiment, and then gradually rises to 50,000 in the last experiment at the interval of 5,000.

Since the variations of the number of faulty samples have no impact on the proposed method, therefore, false alarm rate and fault diagnosis accuracy of the proposed method could be regarded as being unchanged. For simplicity, let them be 6.4% and 72.8%, respectively as they were in Comparison I. The value changes of false alarm rate and fault diagnosis accuracy during the 10 groups of experiments are curved in Fig. 4.11 and Fig 4.12, respectively.

From the two images below, one could see that only when the number of training samples reaches about 45, 000, the false alarm rate of Method II could be lower than the proposed method's, and then it requires roughly 35,000 training samples to make the fault diagnosis accuracy of Method II to exceed that of the proposed method. According to the trend of the two curves, if one feeds more training samples to Method II, its false alarm rate will be very likely to decrease further, and its fault diagnosis accuracy will be also very likely to increase more.

Fig.4.11 The changing curve false alarm rate of Method II with increasing samples

Fig.4.12 The changing curve of fault diagnosis accuracy of Method II with increasing samples

4.4.3.3 Conclusions of Comparison II

From the comparisons shown above, it could be seen that when the number of training data (i.e., labelled faulty samples) is inadequate, the proposed method exhibits evident advantages over data-driven based methods. But as the quantity of training data increases, such advantages will gradually disappear (e.g., when there exists 40K of faulty samples for the researched chiller plant), and when training data becomes sufficiently available, data-driven based methods tend to win with a remarkable margin.

As mentioned above, data-driven based classifiers, especially those having high level nonlinearity and plenty of trainable parameters like the MLP based fault classifier implemented in the above comparison experiment, inherently possess powerful classification potentials. With sufficient and high-quality training data available, data-driven based classifiers always exhibit better performances than other methods. One reason why the well-trained MLP classifier outperforms

the proposed FIS classifier in the comparison is that in essential, the FIS classifier can only classify faults semi-quantitatively and fuzzily while those data-driven based classifiers perform fault classification tasks fully quantitatively and clearly. However, it must be noted that the FDD scheme proposed by this thesis is only targeted to systems that lack labelled fault data rather than systems that could provide abundant labelled fault data. Therefore, the validity and superiority of the proposed scheme is not challenged by the competing scheme. What's more, in practice, the situation of labelled faulty samples being sufficiently available quite rarely exists in industrial applications.

4.5 Conclusion

In this chapter, the FDD scheme proposed by this thesis is concretely implemented and practically applied to a typical process control system, a chiller plant, to evaluate its performance and to verify its effectiveness. The experimental results show that the proposed method could detect and diagnose typical chiller faults with the state of the art fault classification accuracy as well as a very low false alarm rate under the condition of no faulty data available, which well proved the good performances and fully verified the effectiveness of the proposed method.

In addition, for comparing the differences in performance between those existing methods and the proposed one, and meanwhile for investigating the scope of application of the proposed method, this thesis also selects currently popular two FDD methods (i.e., the MLR plus expert system scheme, and the cascaded NNs scheme), and apply them to the aforementioned chiller plant, respectively. The experimental results not only demonstrate the superiority of the proposed method in comparison with the former, but also clarify the scope of application of the proposed method in comparison with the latter.

Chapter 5 Conclusion and Future Work

5.1 Conclusion

In this work, the thesis firstly discusses the necessity and significance of applying FDD to complex process systems, review the existing FDD solutions and analyze both of their strengths and drawbacks. Aiming at the drawbacks of existing methods and the challenges posed by complex industrial process systems, a novel FDD scheme is proposed based on the recognition of the characteristics of typical process systems. The proposed FDD strategy is a two-stage strategy. At the first stage, a data driven based system modeling method (i.e., MLP) is adopted to provide the benchmarks for system behaviors, thereby generating residuals by comparing with observed system behaviors; at the second stage, a fuzzy reasoning based method (i.e., FIS) is adopted as classifiers to classify the residuals formed in last stage, thus achieving the links from the observation of abnormal system behaviors to the inference of the occurrence of a known fault. Due to the facts that the data driven based modeling methods do not rely on any exact physical model of target systems but can model target systems accurately, and that the fuzzy reasoning based classification methods do not require any labelled faulty data but can provide semi-quantitative diagnosis, the proposed scheme, for one thing, has the capacity to be applicable to general process systems rather than merely being applicable to a specific system or process; for another, it has ability of improving the accuracy of fault diagnosis under the condition of lack of or complete unavailability of faulty data, which is also well proven by a chiller plant based case study and by two groups of comparison experiments with two popular FDD schemes.

5.2 Future work

In this dissertation, experimental data collected by ASHRAE Research Projects 1043 is utilized to validate the proposed FDD strategies and algorithms. However, since different process systems have various working mechanisms and potential faults, comprehensive experiments on different process systems, such as power stations, and chemical plants, are needed in order to further explore the performances and generalizability of the proposed FDD scheme.

In addition, from the comparative study one could see that currently, both two categories of schemes, the pure data driven based one (e.g., cascaded NNs), and the data-driven and expert system combined hybrid one (e.g., MLR plus classic rules, and the proposed MLP plus FIS), still cannot give human-level fault diagnosis accuracy. The major reason is that they cannot efficiently detect and diagnose the early faults (e.g., faults of severity of level 1 or 2). When early faults occur, the residuals generated are still small, only slightly different from those residuals under normal working conditions. Therefore, perhaps there is a need to specifically look for other approaches more sensitive to small deviations and able to distinguish between normal operating conditions and early abnormal operating conditions of target systems, and then integrate it into the entire FDD scheme.

Finally, since the performance of fuzzy inference system could be improved promisingly by fine tuning its parameters (e.g., degrees of certainty of fuzzy rules, centers of membership functions) and by extracting more accurate rules from data structure, the paradigm of online learning may be worthy of being added to the FDD scheme so that with the accumulation of class-confirmed faulty samples, the deployed FDD tool also could keep learning and give better performance.

References

- [1] S. Katipamula and M. R. Brambley, "Review article: methods for fault detection, diagnostics, and prognostics for building systems review, part i," *HVAC&R Research*, vol. 11, no. 1, pp. 3-25, 2005.
- [2] A. Schumann, J. Hayes, P. Pompey, and O. Verscheure, "Adaptable fault identification for smart buildings," in *Artificial Intelligence and Smarter Living*. AAAI Workshop, 2011.
- [3] X. Li, C. P. Bowers, and T. Schnier, "Classification of energy consumption in buildings with outlier detection," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3639-3644, 2010.
- [4] Z. O'Neill, X. Pang, M. Shashanka, P. Haves, and T. Bailey, "Model-based real-time whole building energy performance monitoring and diagnostics," *Journal of Building Performance Simulation*, vol. 7, no. 2, pp. 83-99, 2014.
- [5] J. M. House and G. E. Kelly, "An overview of building diagnostics," in *National Conference on Building Commissioning*, Kansas City, MO, Conference Proceedings
- [6] J. J. Gertler, "Survey of model-based failure detection and isolation in complex plants," *IEEE Control systems magazine*, vol. 8, no. 6, pp. 3-11, 1988.
- [7] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results," *Automatica*, vol. 26, no. 3, pp. 459-474, 1990.
- [8] ---, "Analytical and qualitative model-based fault diagnosis a survey and some new results," *European Journal of control*, vol. 2, no. 1, pp. 6-28, 1996.
- [9] C. Bianchini, E. Fornasiero, T. N. Matzen, N. Bianchi, and A. Bellini, "Stator fault detection for multi-phase machines with multiple reference frames transformation," in *Proc. 35th Annu. Conf. IEEE Ind. Electron.*, 2009, pp. 3467-3470.
- [10] A. Bouzida, O. Touhami, R. Ibtouen, A. Belouchrani, M. Fadel, and A. Rezzoug, "Fault diagnosis in industrial induction machines through discrete wavelet transform," *IEEE Trans. Ind. Electron.*, vol. 58, no. 9, pp. 4385-4395, Sep. 2011.
- [11] R. Isermann, "Process fault detection based on modeling and estimation methods, A survey," *Automatica*, vol. 30, pp. 397-404, 1984.
- [12] E. Y. Chow and A. S. Willsky, "Analytical redundancy and the design of robust failure-detection systems," *IEEE Trans. Autom. Control*, vol. AC-29, no. 7, pp. 603-614, Jul. 1984.
- [13] P. M. Frank and X. Ding, "Survey of robust residual generation and evaluation methods in observer-based fault detection systems," *J. Process Control*, vol. 7, no. 6, pp. 403-424, 1997.
- [14] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis part I: Quantitative model-based methods," *Comput. Chem. Eng.*, vol. 27, no. 3, pp. 293-311, 2003.
- [15] J. Chen and R. J. Patton, *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Boston, MA, USA: Kluwer Academic, 1999.
- [16] Russell, E. L., Chiang, L. H., & Braatz, R. D. Data-driven techniques for fault

- detection and diagnosis in chemical processes, London: Springer.
- [17] X. Dai and Z. Gao, "From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis," *IEEE Transactions on Industrial Informatics*.
 - [18] H. Chilton, "Performance of natural-draught water-cooling towers," Proceedings of the IEE-Part II: *Power Engineering*, vol. 99, no. 71, pp. 440-452, 1952.
 - [19] A. Srividya, A. K. Verma, and B. Sreejith, "Automated diagnosis of rolling element bearing defects using time-domain features and neural networks," *Int. J. Mining, Reclamation Environ.*, vol. 23, no. 3, pp.206–215, 2009.
 - [20] S. Nandi, H. A. Toliyat, and X. Li, "Condition monitoring and fault diagnosis of electrical motors: A review," *IEEE Trans. Energy Conv.*, vol. 20, no. 4, pp. 719–729, Dec. 2005.
 - [21] S. Rajagopalan, J. A. Restrepo, J. M. Aller, T. G. Habetler, and R. G. Harley, "Non-stationary motor fault detection using recent quadratic time-frequency representations," *IEEE Trans. Ind. Applic.*, vol. 44, no.3, pp. 735–744, May–Jun. 2008.
 - [22] Yang, Xue-Bin, Xin-Qiao Jin, Zhi-Min Du, and Yi-Bo Guo., "A Hybrid Model-Based Fault Detection Strategy for Air Handling Unit Sensors," *Energy and Buildings*, vol. 57, pp.132–143, 2013.
 - [23] K. Jyoti and S. Singh, "Data clustering approach to industrial process. Monitoring, fault detection and isolation," *Int. J. Comput. Applic.*, vol. 17, no. 2, pp. 41–45, 2011.
 - [24] Y. Hu, H. Chen, J. Xie, X. Yang, and C. Zhou, "Chiller sensor fault detection using a self-adaptive principal component analysis method," *Energy and buildings*, vol. 54, pp. 252-258, 2012.
 - [25] Hoskins, J. C., & Himmelblau, D. M. "Artificial neural network models of knowledge representation in chemical engineering", *Computers and Chemical Engineering* 12, 881-890.
 - [26] B. Fan, Z. Du, X. Jin, X. Yang, and Y. Guo, "A hybrid fdd strategy for local system of ahu based on arti_cial neural network and wavelet analysis," *Building and environment*, vol. 45, no. 12, pp. 2698-2708, 2010.
 - [27] Y. Zhu, X. Jin, and Z. Du, "Fault diagnosis for sensors in air handling unit based on neural network pre-processed by wavelet and fractal," *Energy and buildings*, vol.44, pp. 7-16, 2012.
 - [28] Z. Du, B. Fan, X. Jin, and J. Chi, "Fault detection and diagnosis for buildings and hvac systems using combined neural networks and subtractive clustering analysis," *Building and Environment*, vol. 73, pp. 1-11, 2014.
 - [29] S. Chang, C. Chang, "A fuzzy-logic based fault diagnosis strategy for process control loops", *Chemical Engineering Science*, vol. 26, pp. 335–341,2003.
 - [30] S. Ding, Model-based fault diagnosis techniques: design schemes, algorithms, and tools.*Springer Science & Business Media*, 2008.
 - [31] M. A. Kramer, "Nonlinear principal component analysis using auto associative neural networks", *AIChE J.*, vol. 37(2), pp. 233-243, 1991.
 - [32] S. Li and J. Wen, "A model-based fault detection and diagnostic methodology based on pca method and wavelet transform," *Energy and Buildings*, vol. 68, pp.

- 63-71, 2014.
- [33] S. Wu and J.Q. Sun, "Cross-level fault detection and diagnosis of building systems", *Building and Environment*, vol. 46, no. 8, pp. 1558-1566, 2011.
 - [34] W. Svante, K.Esbensen, P.Geladi., "Principal component analysis", *Chemometrics and Intelligent Laboratory Systems*, vol.21, pp.37-52, 1987.
 - [35] H. Wang, Y. Chen, C. W. Chan, and J. Qin, "An online fault diagnosis tool of vav terminals for building management and control systems," *Automation in Construction*, vol. 22, pp. 203-211, 2012.
 - [36] D. J. Hill, B. S. Minsker, and E. Amir, "Real-time bayesian anomaly detection for environmental sensor data," in *Proceedings of the Congress-International Association for Hydraulic Research*, vol. 32, pp. 503-507, 2007.
 - [37] G.I. Webb, J. Boughton, Z. Wang., "Not So Naive Bayes: Aggregating One-Dependence Estimators," *Machine Learning*, vol. 58, pp. 5-24, 2005.
 - [38] Y. Zhao, F. Xiao, and S. Wang, "An intelligent chiller fault detection and diagnosis methodology using bayesian belief network," *Energy and Buildings*, vol. 57, pp. 278-288, 2013.
 - [39] F. Xiao, Y. Zhao, J. Wen, and S. Wang, "Bayesian network based fdd strategy for variable air volume terminals," *Automation in Construction*, vol. 41, pp. 106-118, 2014.
 - [40] M.C. Comstock, J.E. Braun, Fault detection and diagnostic (FDD) requirements and evaluation tools for chillers, ASHRAE 1043-RP, Purdue University, 2002.
 - [41] Z. Du and X. Jin, "Multiple faults diagnosis for sensors in air handling unit using Fisher discriminant analysis," *Energy Conversion and Management*, vol. 49, no. 12, pp. 3654-3665, 2008.
 - [42] J. Sun, Y. Li, and C. Wen, "Fault diagnosis and detection based on combination with gaussian mixture models and variable reconstruction," *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference*, pp. 227-230, 2009.
 - [43] Nelwamondo and Marwala, "Faults detection using Gaussian mixture models, mel-frequency cepstral coefficients and kurtosis," *SMC., IEEE International Conference*, vol. 1, pp. 290-295, 2006.
 - [44] Y. Zhao, S.Wang, and F. Xiao, "Pattern recognition-based chillers fault detection method using support vector data description," *Applied Energy*, vol. 112, pp. 1041-1048, 2013.
 - [45] Y. Zhao, F. Xiao, J. Wen, Y. Lu, and S. Wang, "A robust pattern recognition based fault detection and diagnosis method for chillers," *HVAC&R Research*, vol. 20, no. 7, pp. 798-809, 2014.
 - [46] J. Liang and R. Du, "Model-based fault detection and diagnosis of hvac systems using support vector machine method," *International Journal of refrigeration*, vol. 30, no. 6, pp. 1104-1114, 2007.
 - [47] H. Han, Z. Cao, B. Gu, and N. Ren, "Pca-svm-based automated fault detection and diagnosis for vapor-compression refrigeration systems," *HVAC &R Research*, vol. 16, no. 3, pp. 295-313, 2010.
 - [48] K.-Y. Chen, L.-S. Chen, M.-C. Chen, and C.-L. Lee, "Using svm based method

- for equipment fault detection in a thermal power plant," *Computers in industry*, vol. 62, no. 1, pp. 42-50, 2011.
- [49] A. Beghi, R. Brignoli, and F. Simmini, "Data-driven Fault Detection and Diagnosis for HVAC water chillers," *Control Engineering Practice*, vol. 35, pp. 79-91, 2016.
- [50] K. Prabakaran, and D. Prakash, "Fault Diagnosis in Process Control Valve Using Artificial Neural Network," *International Journal of Innovation and Applied Studies*, vol. 3, pp. 138-144, 2013.
- [51] J. Cui, S. Wang, "A model-based online fault detection and diagnosis strategy for centrifugal chiller systems", *International Journal of Thermal Sciences*, vol.44, pp. 986–999, 2005.
- [52] T. Jackson, J. Austin, M. Fletcher, M. Jessop, B. Liang, A. Pasley, M. Ong, X. Ren, G. Allan, and P. J. Fleming, "Distributed health monitoring for aero-engines on the GRID: DAME," *Proc. IEEE Aerosp. Conf.*, pp. 3738–3747, 2005.
- [53] Ishibuchi H, Nakashima T., "Effect of Rule Weights in Fuzzy Rule-Based Classification Systems," *IEEE Transactions on Fuzzy Systems*, vol. 9, pp. 506-515, 2001.
- [54] O. Cordón, M. J. del Jesus, and F. Herrera, "Genetic learning of fuzzy-rule-based classification systems cooperating with fuzzy reasoning methods," *Int. J. Intell. Syst.*, vol. 13, no. 1011, pp. 1025–1053, 1998.
- [55] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: Algorithms and applications". *The VLDB Journal, the International Journal on Very Large Data Bases*, pp. 237–253, 2000.
- [56] D.Dubois and H.Prade. "Fuzzy sets and systems: theory and applications", *Academic press*, New York, 1980.
- [57] L. A. Zadeh, "Fuzzy sets", *Information and Control*, vol. 8, pp. 338-353, 1965.
- [58] J. S. Roger Jang, "Neuro-Fuzzy and Soft Computing", *Prentice Hall*, Upper Saddle River, 2nd edition, 1997
- [59] S.W. Wang, and J.B. Wang, "Automatic sensor evaluation in BMS commissioning of building refrigeration systems," *Automation in Construction*, vol. 11, pp. 59-73, 2002.
- [60] M.Mrugalski, J.Korbicz, "Neural network-based robust actuator fault diagnosis for a non-linear multi-tank system", *ISA Transactions*, vol. 61, pp. 318-328, March 2016.
- [61] Bailey, M.B., "The Design and Viability of a Probabilistic Fault Detection and Diagnosis Method for Vapor Compression Cycle Equipment," *Ph.D. Thesis*, School of Civil Engineering, University of Colorado, 1998.
- [62] Peitsman, H. and V. Bakker, "Application of Black-Box Models to HVAC Systems for Fault Detection," *ASHRAE Transactions*, Vol. 102, Pt. 1, pp. 628-640, 1996.