12-2002

Development of a Data Driven Multiple Observer and Causal Graph Approach for Fault Diagnosis of Nuclear Power Plant Sensors and Field Devices

Ke Zhao
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Ke Zhao entitled "Development of a Data Driven Multiple Observer and Causal Graph Approach for Fault Diagnosis of Nuclear Power Plant Sensors and Field Devices." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.

Belle R. Upadhyaya, Major Professor

We have read this thesis and recommend its acceptance:

J. Wesley Hines, Laurence F. Miller

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Ke Zhao entitled "Development of a Data Driven Multiple Observer and Causal Graph Approach for Fault Diagnosis of Nuclear Power Plant Sensors and Field Devices." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.

Belle R. Upadhyaya
Major Professor

We have read this thesis and
recommend its acceptance:

J. Wesley Hines

Laurence F. Miller

Accepted for the Council:

Anne Mayhew
Vice Provost and Dean of
Graduate Studies

(Original signatures are on file with official student records.)

Development of a Data Driven Multiple Observer and Causal Graph Approach for Fault Diagnosis of Nuclear Power Plant Sensors and Field Devices

A Thesis

Presented for the Master of Science Degree

The University of Tennessee, Knoxville

Ke Zhao

December 2002

Dedication

This thesis is dedicated to my parents and my wife for their always inspiring me and encouraging me to reach higher in my career.

Acknowledgments

I thank all those who helped me in completing the Master of Science degree in Nuclear Engineering. I thank Dr. B. R. Upadhyaya for his invaluable guidance all through the research work. I thank Dr. J. W. Hines for introducing me to statistical modeling and adaptive network fuzzy inference modeling. I thank Dr. L. F. Miller for serving on my committee.

Dr. H. L Dodds and Dr. B. R. Upadhyaya of the Nuclear Engineering Department of the University of Tennessee are especially appreciated for providing the research assistantship.

Abstract

Data driven multiple observer and causal graph approach to fault detection and isolation is developed for nuclear power plant sensors and actuators. It can be integrated into the advanced instrumentation and control system for the next generation nuclear power plants.

The developed approach is based on analytical redundancy principle of fault diagnosis. Some analytical models are built to generate the residuals between measured values and expected values. Any significant residuals are used for fault detection and the residual patterns are analyzed for fault isolation.

Advanced data driven modeling methods such as Principal Component Analysis and Adaptive Network Fuzzy Inference System are used to achieve on-line accurate and consistent models. As compared with most current data-driven modeling, it is emphasized that the best choice of model structure should be obtained from physical study on a system.

Multiple observer approach realizes strong fault isolation through designing appropriate residual structures. Even if one of the residuals is corrupted, the approach is able to indicate an unknown fault instead of a misleading fault. Multiple observers are designed through making full use of the redundant relationships implied in a process when predicting one variable.

Data-driven causal graph is developed as a generic approach to fault diagnosis for nuclear power plants where limited fault information is available. It has the potential of combining the reasoning capability of qualitative diagnostic method and the strength of quantitative diagnostic method in fault resolution. A data-driven causal graph consists of individual nodes representing plant variables connected with adaptive quantitative models. With the causal graph, fault detection is fulfilled by monitoring the residual of each model. Fault isolation is achieved by testing the possible assumptions involved in each model. Conservatism is implied in the approach since a faulty sensor or a fault actuator signal is isolated only when their reconstructions can fully explain all the abnormal behavior of the system.

The developed approaches have been applied to nuclear steam generator system of a pressurized water reactor and a simulation code has been developed to show its performance. The results show that both single and dual sensor faults and actuator faults can be detected and isolated correctly independent of fault magnitudes and initial power level during early fault transient.

Table of Contents

List of Figures

List of Tables

Acronyms

ANFIS	Adaptive network fuzzy inference system
API	Application program interface
DOF	Degree of fulfillment
FDI	Fault detection and isolation
FCV	Feed water control valve
GED	Graphic editor
GMDH	Group method of data handling
GUI	Graphic user interface
NPP	Nuclear power plant
NR	Narrow range
Picasso	A user interface development software
PCA	Principal component analysis
PLS	Partial least square
PWR	Pressurized water reactor
RTM	Real time management
SDG	Sign directed graph
SG	Steam generator
SimPWR	Code name for PWR system simulation
TCV	Turbine control valve
UIMS	User interface management system
UTSG	U-tube steam generator
WR	Wide range

Chapter 1

Introduction

1.1 Background and Motivation

Fault Detection and Isolation (FDI) has been considered as an important strategy to improve operational performance in a variety of industries for a long time. A fault in a process is defined as any malfunction of sensors, controllers and field devices at the initial stage, which may ultimately affect the operational performance. The most important objectives of fault detection and isolation are to prevent a sudden equipment failure, collect information on malfunctions, improve maintenance planning, and have a better plant control such that optimal operational performance can be achieved (Himmelblau, 1978). It brings about significant benefits through minimizing the downtime, enhancing the safety and reducing the manufacturing cost (Upadhyaya, 1999).

In nuclear power plants (NPPs), FDI becomes increasingly emphasized with the strategic development of advanced plant instrumentation and control. Owing to the revolution of digitization, the abundant available measurements have provided the opportunity to automate FDI. The incorporation of FDI as an indispensable part of modern instrumentation and control system has begun to be further speeded up.

In current nuclear power plants, some major deficiencies of plant instrumentation and control design which are affecting the economic performance and safety features, are as follows (White, 1994):

- *Unscheduled plant trips are not rare due to component failures.*
- *Important indications of abnormal conditions are masked by many less important alarms during some transients.*
- *Operators face difficulties to determine which alarms are due to an important initiating event and which alarms are due to operation action such as out-of-service components undergoing maintenance.*
- *Due to fault propagation, the fault alarms may occur in an order different from that the fault occurs.*

In order to overcome these problems, the advanced instrumentation and control system has been defined with the following features (EPRI, 1994):

- *Fault-tolerant systems should be introduced to avoid misinformation.*
- *Digital systems should enable the plant to have self-diagnostics and on-line replacement. Failed equipment can be replaced and fixed on non-outage time.*
- *"Adaptive tuning, drift-free operation, and nonlinear compensation" should be achieved to avoid human errors.*

In fact, the automation of FDI has become an important measure to differ advanced instrumentation and control system from a traditional one in nuclear power plants. Figure 1.1 illustrates the interface between FDI system and the advanced instrumentation and control system. On the one hand, the FDI system is able to provide fault information to either an operator support system or a plant surveillance system. This information assists operators in making optimal maintenance planning and fault management. On the other hand, the FDI system provides inputs to some software driven protection logic and software driven control algorithms either to compensate for fault effects or to implement safeguard.

The overall objective of the thesis is to develop some FDI algorithms that can be integrated into the advanced instrumentation and control system. Because the algorithms aim at on-line implementation for a nuclear power plant, the following performances must be satisfied:

- The fault detection module can detect an incipient fault but will not trigger a false alarm during any normal plant transients.
- A fault can be correctly diagnosed regardless of its fault magnitude and the initial plant condition when the fault occurs.
- If one of the fault signatures is corrupted or degenerated due to process noise or measurement noise, the algorithm will indicate an unknown fault instead of being misdiagnosed as another fault.
- A fault must be detected and isolated during its fault transient rather than after a new steady state has arrived in order to overcome the problem with controller compensation.

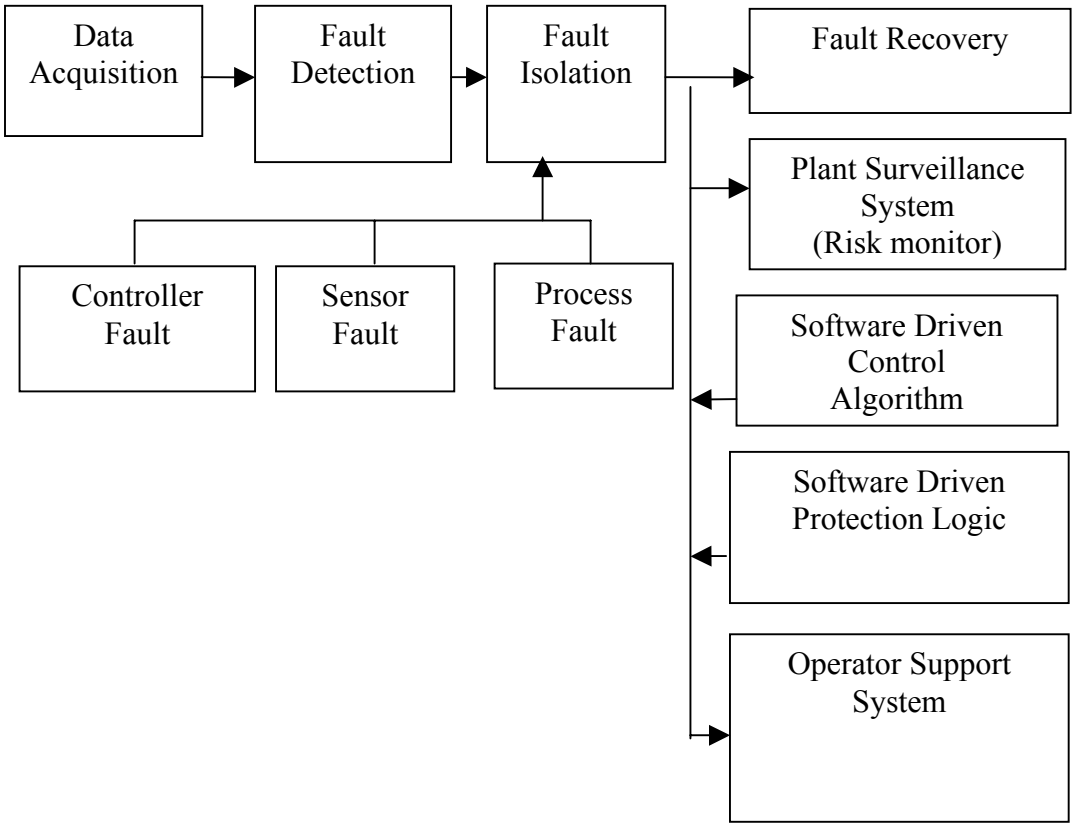

Figure 1.1. A proposed schematic of on-line incipient fault detection and isolation (FDI) for nuclear power plants (NPPs).

- In order to facilitate modularization in implementation, a decision should be made only based on local evidences.

Since a large safety critical system is being dealt with, the following constraints are imposed on the development:

- The possible faults may not be enumerable.
- The fault signatures can be obtained only from limited amount of fault data.

In order to achieve the FDI algorithm with the above technical specification under the above constraints advanced statistical inference based modeling such as Principal Component Analysis (PCA) and artificial intelligence methods such as Adaptive Network Fuzzy Inference System (ANFIS) are studied in order for adaptive modeling. As compared with most current data-driven modeling, the best choice of model structure is obtained from physical study on a system. After a systematic reviewing on the available FDI methods in other industries and the implementation of PCA based approach and ANFIS model based approach, data driven model causal graph is proposed as a general approach to automatic FDI for nuclear power plants. This approach can naturally arrive at efficient data driven modeling. The fault isolation is based on cause effect analysis on model residuals. Therefore, it is unnecessary to enumerate faults and define the associated fault signatures for fault isolation. Moreover, it also enables to isolate simultaneous faults thanks to its excellent reasoning capability. In addition, since fault isolation is considered as a process of confirming which fault candidate can fully explain all the observed abnormal fault symptoms, the decision logic is inherent with conservatism. It is concluded that the data driven model causal graph approach is applicable to be integrated into the advanced instrumentation and control system for nuclear power plants.

1.2 Statement of the Problem

Although many researches on FDI have been performed in other industries, some special issues must be seriously addressed when those experiences are applied to nuclear systems (Kaistha and Upadhyaya, 2001).

1.2.1 Multi-operational regimes

A nuclear power plant may operate at numerous operational points such as start-up operation, the change in power demand, the evolution of fuel cycle, performance change of components throughout its lifetime, the change in system configuration to meet safety requirements, etc. The designed FDI system must be adaptable to all these operational regimes. For instance, the FDI system should be able to correctly isolate a fault under all these operational conditions. A normal operational transient such as a power change, a chemical-volume-control-system startup or shutdown, a steam generator blow-down system startup or shutdown, will not trigger a false alarm. This requires that the developed FDI system be able to adaptively adjust its models at all the operational points.

1.2.2 Dynamic process behavior

A nuclear power plant always experiences some internal disturbances such as the vibration of machinery components and turbulence induced fluctuation, and some external disturbances such as the change in power demand. Therefore, all the state variables and/or the measured variables are random variables due to measurement disturbances or process disturbances.

Unlike a process dynamics, an electric circuit exhibits static behavior. Once an electrical circuit is around its operation point, a set of algebraic equations can always be found to characterize the relationship of the voltage, the current and the resistance among certain nodes. By systematically checking the consistency of all the algebraic equations, it is not difficult to detect a faulty component and isolate it within the circuit.

For a dynamic process, a set of algebraic equations may not be able to characterize the relationship among process variables. Different initial conditions may result in different sets of relationships. A set of differential equations may usually be required to characterize a dynamic system.

Non-linearity results in additional difficulties in modeling the behavior of a dynamic system especially for a nuclear power plant where many nonlinear components such as valves, pumps, and controllers with dead band and saturation limits are utilized.

The existence of non-linearity will also weaken certain good features of many FDI design schemes.

1.2.3 Controller feedback effects

Many distributed feedback controllers are present in a nuclear power plant in order to maintain the operation within the designed operation regimes. Power regulating system controls the reactor power such that the power generation from the core matches the desired power output of the plant. Steam generator (SG) water level control system controls the feed water control valve position such that SG level is maintained at the set point level. Pressurizer level and pressure control system manipulates the power of electric heaters and spray flow rate such that the level and pressure is maintained at the set point level.

Because of feedback controller, a sensor fault or an actuator fault will propagate throughout the system. The fault propagation would create challenges to designing an effective FDI system as described below (Dash and Venkatasubramanian, 2000):

- *Data reconciliation approach is not applicable.*
- *A minor fault is harder to be detected and isolated.*
- *A fault may propagate from one subsystem to another subsystem through a control system bridging them.*
- *A comparison between set points and measured values after a new steady state cannot reveal the occurrence of a sensor fault that is involved in the feedback control loop.*

1.2.4 Complexity of fault natures

In a large system such as a nuclear plant, the natures of possible faults are very complicated because many different components may be involved. From the FDI methodology point of view, these components may be categorized as sensor fault, actuator fault, controller fault, and process fault. With regard to fault effects on the measurements, a fault can be classified as additive fault and multiplicative fault. The

time dependence of fault magnitude allows categorizing a fault as abrupt fault, drift fault and intermittent fault. None of the available FDI approaches has acceptable performance for all the different types of faults.

1.2.5 Multiple faults

The importance of multiple fault diagnosis should not be underestimated simply because its probability is much lower than single faults. In practice, in a facility such as nuclear power plants where safety is always placed at the top, multiple fault diagnosis plays a role as important as single fault diagnosis because the risk contribution due to multiple faults is much higher than single faults. A good example that simultaneous faults may have significant consequence is the Three-Mile-Island accident. One of the major reasons for multiple faults is a common cause failure.

Multiple fault diagnostics is challenging because of the interacting nature of most faults (Dash and Venkatasubramanian, 2000). In a complex process, the interaction of different faults through a closed control loop would make the fault symptoms more difficult to delineate. System non-linearity makes it even harder to develop analytical methods to infer multiple faults simply based on the information contained in single faults.

1.2.6 Complex systems

When most FDI techniques are applied to a complex system such as a nuclear power plant, some serious difficulties may occur. These difficulties are:

- Many input variables may be involved in a model such that its accuracy may deteriorate significantly.
- Faults in many subsystems may have the same symptoms.
- The system interaction or controller interaction among the subsystems may make the causal-effect relation very complicated.

1.3 Current Solution

1.3.1 Hardware redundancy

Serious consequences of a failure in an instrument system have resulted in great conservatism in the design of nuclear power plants. Hardware redundancy is the traditional design scheme to achieve this conservatism. When an instrument measurement is used for system control, a voting logic based on several redundant sensors is used to detect and isolate a faulty sensor. More conservatism has been imposed upon the reactor protection systems. The designed safety-critical control system must satisfy:

- Adequate redundancy.
- Adequate independence.
- Physical isolation.

Adequate redundancy means that multiple sensors or instrumentation channels should be used. Adequate independence means the measurements should be performed based upon multiple different principles. Physical isolation means that the sensors or instrumentation channels should be physically isolated. The second and the third criteria aim at defending common cause and common mode failures.

Hardware redundancy is the most effective way to detect and isolate an instrument fault. However, it is too expensive to extend the philosophy to the whole plant including all the auxiliary systems of a nuclear plant.

1.3.2 Reactor spectral analysis

Reactor spectral analysis is a widely used signal processing technology to detect and isolate a fault at the component level.

"Reactor spectral analysis is basically a statistical technique for extracting information on reactor system dynamics from the fluctuations of measured instrumentation signals during steady state operation. The small fluctuations of measurable process signals are the results of stochastic effects inherent in physical process such as heat transfer, boiling, coolant flow turbulence, fission process, structural

vibrations and pressure oscillations. Reactor noise analysis can monitor and assess the conditions of technological processes and the instrumentation in the nuclear reactor in a non-intrusive, passive way" (Glockler and Tublett, 1995).

Some successful applications of reactor spectral analysis are summarized as follows:

- Detection of abnormal operation of an instrument or an actuator.
- On-line monitoring of slowly changing parameters such as fuel-to-coolant heat transfer coefficient, sensor degradation.
- Monitoring reactor stability and stability margin.
- Vibration analysis of reactor internals and components.
- Sensor response monitoring and failure detection.

Reactor spectral analysis is successful in detecting a sensor fault or an actuator fault at the component level. Frequency spectrum analysis assumes that plant measurements have a standard frequency spectrum under normal operations and any deviation from the standard spectrum indicates an abnormal condition. One drawback of reactor spectral analysis technique is the difficulty in its extension to the level of a system or a plant. For example, if the steam generator water level control system is to be monitored, an individual signal-processing unit must be designed for each signal. Another drawback of reactor spectral analysis is that the signal characteristics and its dependence on operation conditions of the system must be known for the faults of concern.

Modern FDI technique, with analytical redundancy being the representative, has a significant feature that the fault signature for a fault is not dependent on system operation states. There would be long-term benefits to the operation of nuclear plants if modern FDI technologies can be combined with the traditional reactor spectral analysis.

1.3.3 Analytical redundancy analysis

Analytical redundancy analysis is the foundation of modern FDI. It borrows the idea from hardware redundancy. It takes advantage of the redundant information

inherent in a physical system. First-principle or data-driven models of fault-free systems are built to relate different measurements. These models can function as soft sensors providing an additional redundancy to the measurement systems if none of the model inputs is corrupted. In general, if the relations are violated, a conclusion can be drawn that either the process or the measurements are not correct. This information can be used as an indicator to fault detection. The deviation pattern can further be analyzed for fault isolation.

Analytical redundancy based FDI approaches have some incomparable advantages as compared with the traditional methods:

- The fault signature is independent of fault magnitudes.
- The fault signature is independent of the operation conditions when a fault occurs.

Therefore, the fault signature collected once for a fault is sufficient to characterize the fault so that it can be isolated. The application of FDI technology has been significantly speeded up since the analytical redundancy was introduced. It is evident that FDI approaches dependent on large amounts of fault data to characterize a fault have little value in engineering application.

However, as a principle, analytical redundancy does not give information about how to generate and analyze fault signatures for fault detection and isolation. Depending on the form of system knowledge available and the technical specification of the designed FDI system, a variety of implementation strategies exist. A detailed description of these approaches is given in Chapter 2. In the thesis, the difficulties of utilizing the principle on FDI for nuclear power plants are addressed and engineering applicable implementation strategies are pursued.

1.4 Technical Approach and Task Definition

The FDI process aims at inferring the root causes from the symptoms observed in the measurements. These symptoms are the basis for an operator to make fault diagnosis. The concept of an automatic FDI follows the same logic as a human being in making a decision — feature extraction, fault detection and fault isolation. Feature extraction

compresses fault symptoms into a low dimensional space. Fault detection detects any changes in the feature space. Fault isolation classifies the fault signatures into separate fault classes.

Fault symptoms are observed in many measured variables. They can be represented in three ways. The first one is the values of individual measurements. They can be compared with certain set points dependent on operation conditions. The second one is the change of measurements before and after a fault. A third one, the most sophisticated one, is the residuals between measured values and the expected values based on some analytical models. The analytical models can take advantage of the redundancy of measurements inherent in a process such as energy conservation, momentum conservation, mass conservation, etc. These models can be considered as additional soft sensors available for checking consistency. In the thesis, the third representation is used.

Analytical redundancy approach is used as the basis to develop FDI methods for nuclear power plants. Plant models are built to characterize the relationship among plant variables. Fault signatures are generated as deviations from the models. Different modeling methods and different approaches to defining fault signatures have been studied for fault detection and fault isolation.

A successful FDI system depends on accurate modeling. Accurate modeling implies that the developed model is able to characterize a system with high accuracy. However, because a nuclear power plant is so complicated, the available first principle models are usually not accurate enough for FDI. For this reason, data driven modeling method needs to be used which is able to learn a model from data. Many data driven modeling techniques have been developed such as time series models, Kalman filtering algorithms, Principal Component Analysis (PCA), Partial Least Squares (PLS) models, Group Method of Data Handling (GMDH) and Artificial Neural Network. Different modeling methods have their own advantages and disadvantages. In the thesis, PCA and ANFIS are utilized.

Appropriate choice of model structure is essential to building data-driven models. For example, if many input variables are involved in a neural network model, it is very

difficult to train it. Moreover, more input variables need more measurements for training, which may cause a delay in fault detection. In addition, the co-linearity contained in the inputs may result in model instability. In the thesis, it is emphasized to take advantage of the available system knowledge so that the most parsimonious model structure can be obtained.

Three different types of simultaneous faults exist (Lee, 1999). Independent dual faults result in symptoms on different variables. The effects of either fault will be different on different variables. For masked dual faults, the fault symptoms of one of the dual faults are a subset of the symptoms of another fault. For dependent dual faults, one of the dual faults competes with the other. The resulting symptom is unpredictable depending on which fault dominates the process. In this case, neither of the dual faults can explain all the symptoms because of mutual amplification and diminution.

The challenge to multiple fault diagnosis is to appropriately select fault signatures in order to avoid fault masking. For a dual fault whose fault symptom masks one of its elemental faults, the only possibility that they can be distinguished is to derive some new fault signatures to avoid the fault masking. Otherwise, the dual fault cannot be isolated from its element faults.

Figure 1.2 shows the overall technical approach taken in this thesis. Data driven modeling such as ANFIS and PCA is used to obtain system models. The system knowledge is used to define model structures. The plant measurements are used to parameterize the models. The process of fault diagnosis involves residual generation and residual analysis. The physical residual is defined as the difference between the measured values and their true values. Based on the available physical models, the physical residuals can be approximated as the difference between the measured values and the predicted values. A significant residual can be used to detect a fault. Three methods are used for residual analysis. If the possible faults and their associated fault signatures are known, residual direction and residual structure can be defined to characterize a fault for fault isolation. In the more general case when possible faults are not known and their associated fault signatures are not available, cause graph approach based on the cause effect analysis on the residuals can be used for fault isolation.

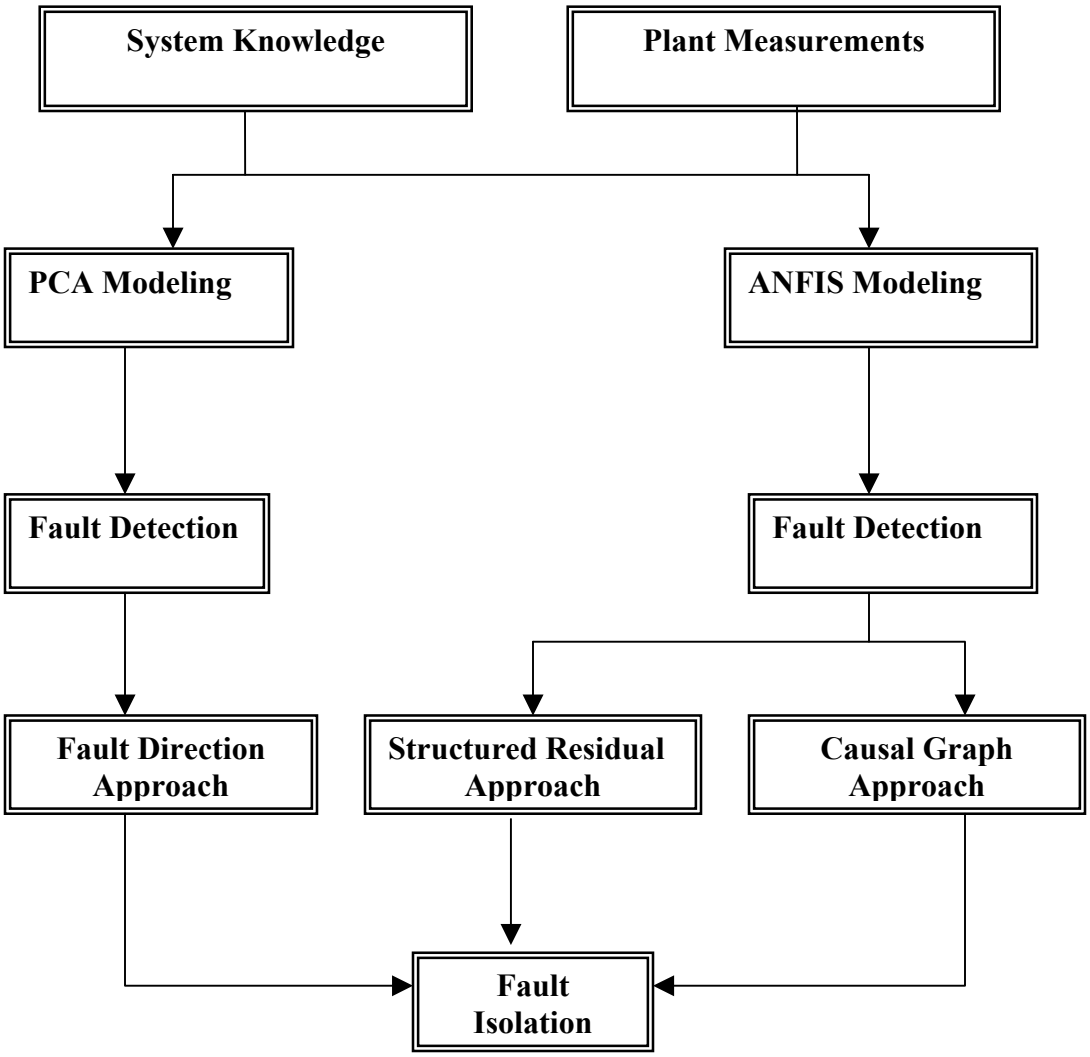

Figure 1.2. Schematic diagram of the overall technical approach.

To accomplish the objectives of the thesis based on the technical approach discussed above, the following tasks, demonstrating the independent work performed for the thesis research are completed:

- Review modern FDI techniques for process monitoring.
- Study data driven modeling for a linear static system using PCA.
- Study data driven modeling for a nonlinear dynamic system using ANFIS.
- Analyze the fault responses of sensor and actuator faults for PWR steam generator system.
- Implement PCA based FDI to detect and isolate the faults for PWR SG system.
- Implement structured residual design approach for PWR SG system using ANFIS models.
- Develop data driven model causal graph approach to fault detection and isolation for nuclear power plants and apply it to PWR SG system.
- Design a user interface to demonstrate the efficiency of the designed FDI system.

1.5 Contributions of the Thesis

The contributions of the thesis are as follows:

a) A detailed review of the modern approaches to fault detection and isolation.

Qualitative model based approaches such as Sign Directed Graph (SDG) and bond graph approach, and quantitative model based approaches such as parity space approach, state space approach, parameter estimation approach, and pattern recognition approach are reviewed. Some comparison study is then performed.

b) Implementation of PCA based FDI algorithm.

A statistical inference method such as PCA algorithm is implemented for fault detection and isolation. This approach is shown to have inherent connection with parity space approach. The linear relationship among measured variables implying analytical redundancy can be consistently represented by the eigenvectors corresponding to the

trivial components. Any deviation in either model space or residual space indicates a fault. The fault direction jointly defined both in the model space and in the residual space provides better fault isolability. When applied to nuclear plant steam generator system, it is able to detect and isolate the selected thirteen single and dual faults.

c) Implementation of ANFIS based FDI algorithm.

Given that the possible faults are known for the plant based on engineering judgment, a set of ANFIS models can be built to characterize the nonlinear relationship among plant measurements. Through appropriate choice of model structures, structured residuals can be achieved for fault isolation. When applied to nuclear plant steam generator system, it is able to detect and isolate the selected thirteen single and dual faults.

d) Development of data driven model causal graph based FDI algorithms

Data driven model causal graph is proposed as a generic approach to fault diagnosis for nuclear power plants. This approach is able to combine the reasoning capability of qualitative model based method and the strength in fault resolution of quantitative model based method. The causal graph consists of individual nodes representing plant variables connected with quantitative models. To facilitate on-line implementation, ANFIS is used as an adaptive modeling tool. Fault detection is fulfilled by monitoring the residual of each model. Fault isolation is achieved by cause effect analysis on the residuals. The developed approach is demonstrated using data obtained from a simulation code for a Pressurized Water Reactor (PWR) (Doster, 2000). The developed approach is able to detect and isolate single faults and dual faults with fault propagation regardless of fault magnitudes and initial power level during early fault transient.

e) Development of a real time demonstration system for FDI

In order to show the effectiveness of the developed FDI methods for nuclear power plants, a graphic user interface is developed under the environment of Picasso-3, a user interface management system. The software is able to (1) create a fault by changing the fault characteristic parameters; (2) display key parameters on the schematic of a reactor system; (3) exhibit residual patterns specific to a fault; (4) trend process variables relevant to a fault; and (5) echo FDI results. The software has integrated SimPWR, a reactor system analysis code in FORTRAN, and the FDI code in Matlab, and the C++ code to control the GUI.

1.6 Organization of the Thesis

The overall objective of the research is to develop an approach to automated FDI for nuclear power plant systems.

In Chapter 2, major modern FDI techniques under the principle of analytical redundancy are described. Qualitative model based approaches such as SDG and bond graph, and quantitative model based approaches such as parity space approach, state space approach, parameter estimation approach, and pattern recognition approach are reviewed. Some comparison studies in their applications are also made.

In Chapter 3, the nuclear steam generator system and the available measurements are described for a PWR reactor. Some discussions about how to enumerate faults are also made. Some considerations on preparing data for building models are then described. Finally, the system responses to these faults are analyzed.

In Chapter 4, linear PCA algorithms and the relationships between PCA and parity space approach for FDI are described. The PCA algorithm is then implemented to detect and isolate the faults for a PWR nuclear steam generator system. The $\boldsymbol{T}^2$ statistics and Q statistics are used for fault detection. Fault direction is used as fault signatures for fault isolation.

In Chapter 5, ANFIS is introduced as an advanced tool for modeling nonlinear systems. Its structure and learning algorithm are discussed. ANFIS model based FDI is implemented to isolate the selected faults for the steam generator water level system. Structured residual design approach proves to be an efficient method for fault isolation. The approaches to structured residuals are studied.

In Chapter 6, data driven model causal graph is developed as a generic approach to fault detection and isolation for nuclear power plants. The structure of a model causal graph is described. The reasoning algorithms are then described in order to achieve fault isolation. It has also been shown that the causal graph is in full agreement with efficient data driven modeling. The developed approach is successful when applied to nuclear steam generator system.

Chapter 7 describes a graphical user interface and its design to demonstrate the efficiency of the designed FDI system.

Concluding remarks and recommendations for future work are presented in Chapter 8.

Chapter 2

Literature Review

This chapter describes the principles of fault detection and isolation through a systematic literature review.

The FDI approaches differ in what kind of knowledge is used and how the knowledge is used. In broad sense, FDI approaches can be quantitative knowledge based or qualitative knowledge based. Depending on how to develop models, FDI approaches can be first-principle model based or historical data driven model based. In terms of how to use the knowledge, many FDI approaches exist. The major approaches are reviewed in this chapter.

2.1 Data Reconciliation

Data reconciliation is a technique used for detecting and isolating a measurement error or a process fault that results in measurements inconsistent with energy balance equations, mass balance equations, and other balance equations. Its goal is to reconstruct the measurements so that the balance equations are not violated. This approach provides an efficient way to handle a large system (Albuquerque and Biegler, 1996).

For a given system under fault conditions, the measurement vector y can be represented as a function of the actual values of some system variables denoted by the vector z, which is given by:

$$y = f(z) + \Delta y$$

where

Δy = measurement error and fault error.

The actual values of the system variables satisfy certain constraint relationship, which is given by:

$$g(c, z) = 0$$

where

c = some other system variables.

Data reconciliation aims at determining a vector $z*$ such that

$$J(z^*) = (y - f(z^*))R^{-1}(y - f(z^*)$$

is minimized on condition that

$$g(c, z^*) = 0$$

where

R = a weighting matrix reflecting the accuracy of different measurements.

The reconstructed measurements can then be obtained by:

$$y^* = f(z^*)$$

Therefore, $y - y^*$ can be used as fault signatures for fault detection and isolation.

Data reconciliation approach is powerful in solving a large system with co-linearity. However, it is only applicable for steady state conditions since the involved optimization may become unmanageable during transient process. Therefore, there might be some undesirable time lag for fault detection and isolation. In addition, for nuclear power plants where many controllers are involved, the steady state information may not be enough to isolate some faults.

2.2 Model Based Approach

The foundation of model-based approach is analytical redundancy. It assumes that first principle or data-driven models can be used to represent the relationships among plant variables during fault free conditions. These models provide the same function as some redundant soft sensors. The physical residuals can then be approximated as the difference between the measured values and the model prediction. A significant residual can be used to detect a fault and the residuals can be analyzed for fault isolation. Model based approaches differ in how to generate and analyze residuals.

2.2.1 Parity space approach

For a given system, the relationship between the measurement vector y and the state vector x is given by:

$$y = Cx + \Delta y$$

where

Δy = measurement error.

C = system matrix.

$x = x^* + \Delta x$

Δx = disturbance of state variables.

Parity space approach (Chow and Willsky 1984, and Frank, 1990) aims at generating a residual vector that is influenced only by measurement error. The physical residual o(t) of the system has this property, which is given by:

$$\text{o(t)} = y - Cx^*$$

This physical residual is not directly available. Therefore, a parity vector must be introduced. The parity vector p(t) is obtained through a linear transformation V^T of the physical residuals given by:

$$p = V^T o(t) = V^T (\Delta y + C\Delta x) \tag{2.1}$$

The parity vector will not depend on the disturbance of the state variables if the following constraint is imposed:

$$V^T Cx = 0$$

Consequently,

$$V^T C\Delta x = 0$$

Therefore,

$$p = V^T \Delta y$$

Hence, the parity vector p is influenced only by the measurement error. It is defined in the parity space that is usually smaller than the original measurement space. A nonzero component of the parity vector indicates a faulty measurement.

For example, a system has five measurements and three state variables. The dimension of the measurement vector y is 5 by 1, the dimension of the measurement matrix C is 5 by 3, and the dimension of the parity vector will be 2 by 1 since the system has two linear dependent relationships among the measurements if the rank of the matrix is three. Therefore, two of the five measurements can be derived from the other measurements. Correspondingly, the dimension of the matrix V is 5 by 2.

Such a FDI approach has a great advantage in its robustness to disturbances. However, the linear transformation approach is not applicable for a non-linear system.

2.2.2 State estimation approach

State estimation approach is considered as a more general approach than parity space approach. When the residuals are generated for fault detection and isolation, the simplest model to estimate a measured signal is identical to the plant model functioning as a simulator in parallel without using the information of the system outputs. However, because the simulator type of model is an open loop system, the solution may not be stable or convergent if the plant operates beyond its designed region.

In order to overcome the problem, the state space model is introduced in the FDI. The linear state space model of a system is defined as follows:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{2.2}$$

For fault free condition, a state observer with feedback matrix H can be designed as follows (Simani, 2000):

$$\begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + Bu(t) + He(t) \\ e(t) &= y(t) - C\hat{x}(t) \end{aligned} \tag{2.3}$$

such that:

$\varepsilon(t) = x(t) - \hat{x}(t)$ approaches zero asymptotically.

$\varepsilon(t+1) = (A - HC)\varepsilon(t)$ approaches zero asymptotically.

where

$e(t)$ = output estimation error.

$\varepsilon(t)$ = state estimation error.

If an additive fault $f(t)$ occurs in the system, the dynamic behavior of the system can be characterized by:

$$\begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + Bu(t) + D_1 f(t) \\ y(t) &= C\hat{x}(t) + D_2 f(t) \end{aligned} \tag{2.4}$$

Then the output error is given by:

$$e(t) = C\varepsilon(t) + D_2 f(t)$$

Both state estimation error and the output error are not zero and show dynamic behavior after a fault occurs. Both can be used for fault detection. To generate stable

and sensitive residuals, the feedback matrix H must be carefully designed. State estimation error vectors can also be designed with directional characteristics for fault isolation (Jones, 1973).

Because the dynamic behavior of state estimation error and the output error are different from $D_1 f(t)$ and $D_2 f(t)$ if the signals are affected by additional noise, it is a challenging task to design the feedback matrix such that the estimation error is sensitive to a fault and insensitive to noise.

2.2.3 Parameter estimation approach

Some process parameters can be estimated using some input variables and/or some output variables. The difference between the normal process parameters and the estimated parameters can then be used for fault detection and diagnosis (Chen and Patton, 1993).

Given a single input single output (SISO) system of order *n* defined in the matrix form as follows:

$$y(t) = x'\theta \tag{2.5}$$

where

θ = parameters related to the linear model.

$x' = [y(t-1),...y(t-n),u(t-1),...u(t-n)]$

The least square estimate of θ can be computed as:

$$\theta = (x'x)^{-1} y \tag{2.6}$$

If the knowledge is known about what is the mapping between a fault and the parameter θ, the values found from the measurements through system identification can then be used for fault isolation.

Parameter estimation is a powerful approach to detecting and isolating a process fault for a linear system. However, for a nonlinear system, it may be very difficult to estimate the parameters with enough accuracy and define a one-to-one relationship between a parameter change and a fault in the physical process.

2.3 Pattern Recognition

Statistical inference and neural network are two major applications of pattern recognition techniques in fault detection and isolation. The task of pattern recognition is to extract some features and set up a mapping between a class of objects and their features. This is in conformance with the task of fault diagnosis. If appropriate fault signatures can be obtained to characterize fault symptoms, they can then be used as features to infer the fault.

Mathematically, the principle of fault detection and isolation based on pattern recognition can be described by Bayes's rule:

$$P(F_i \mid S) = \frac{P(F_i)P(S \mid F_i)}{\sum P(S)P(F_i)} \tag{2.7}$$

where

$P(F_i \mid S)$ = the probability that a fault with symptoms S is F_i.

$P(F_i)$ = the prior probability of F_i.

$P(S \mid F_i)$ = the conditional probability of symptom S given fault F_i.

Each fault defines a specific region in the feature space. For a given observation, the likelihood of the observation falling into each region corresponding to all the possible faults in the feature space can be computed. The observation is assigned to the fault class that gives the largest likelihood.

Quite a few statistical methods such as PCA, PLS, multivariate auto-regression modeling can be used to capture the features. The critical point to feature extraction is to compress the information describing the relationship among variables in a reduced dimensional space without significant loss of information.

Artificial neural network is another powerful technique to perform pattern recognition. It is trained such that all the fault patterns are memorized. When a new fault comes, the network is then able to classify its pattern into correct fault class.

Pattern recognition based FDI has the advantage in its possibility of on-line implementation. However, a large amount of training data is needed in order to characterize the features of the possible faults. For nuclear power plants where many

components are involved, it is unrealistic to collect such data and define fault characteristics for the possible faults of interest.

2.4 Sign Directed Graph

A signed directed graph (SDG) is a graphic representation of the causal relationship among plant variables. In a SDG, these variables are individual nodes and some directed arcs are used to represent the causal relations between the nodes. A node can take qualitative values, denoted as 0, +, and –, which correspond to nominal, high and low, respectively. The directed arc signs may take + or – depending on whether the causal relation is in the same direction or in the opposite direction. A root node is linked with at least one effect node but is not connected to any causal node. The process of fault diagnosis using SDG is to find a single path from a root node to all the abnormal measurement nodes, which satisfies the qualitative constraints defined by the signed directed arcs for the system (Lee, 1999).

The process of single fault diagnosis using the SDG method can be summarized as the following steps (Vedam and Venkatasubramanian, 1995):

- *Identify all the fault candidates by tracking consistent arcs from the effect nodes to the causal nodes starting from all the abnormal measurement nodes.*
- *For each of the fault candidates, check if an effective causal path exists to explain the observed abnormal measurements.*
- *A fault candidate is confirmed to be the fault origin if reasonable causal paths can be defined to interpret all the abnormal measurements.*

The SDG method has a distinct feature in fault diagnosis in finding out all the possible root causes capable of explaining all the abnormal measurements.

In order to represent explicitly the knowledge of the system behavior contained in SDG, a rule based SDG approach to FDI has been developed (Kramer and Palowitch, 1987). The rule base variation is more concise and can be easily incorporated into an expert diagnostic system.

Recently fuzzy-SDG has been developed (Zakarian and Kusiak, 2000). A node of fuzzy-SDG is a variable that may take several fuzzy values such as low, medium low,

normal, medium high and high. A branch represents the causal-effect direction and strength. The connection strength $e(x_{j+1} \to x_j)$ from x_{j+1} to x_j is determined by:

$$\mathrm{e}(\mathrm{x}_{\mathrm{j+1}} \to x_j) = S_{j,j+1} \frac{Rx_{j+1}}{Rx_j} \tag{2.8}$$

where

Rx_{j+1} , Rx_j = the value range of node x_{j+1} and x_j.

$S_{j,j+1}$ = the sensitivity coefficient.

$$S_{j,j+1} = \frac{\partial x_j}{\partial x_{j+1}}$$

The architecture of fuzzy-SDG not only has the good features of traditional SDG methods but also has the learning capability from data.

2.5 Bond Graph

When a fault happens, the system may involve some dynamic behavior because of external disturbances and control changes. In most cases, a model developed under fault free conditions may not be able to track the model changes caused by faults. However, the transient behavior caused by faults contains rich information for fault isolation. In order to make most use of this information, a diagnosis inference method, bond graph approach has been proposed to model the dynamic behavior, reason about the temporal attributes of system parameters and relate behavior changes to component parameters (Mosterman and Biswas, 1997):

Bond graph is a causal behavior graph developed for qualitative modeling (Amsterdam, 1992). In a bond graph, the individual nodes represent effort variables in the system. The directed arcs represent the relations between flow and effort variables. The characterization of the relationship is achieved in analogy to some electrical element. For a resistance element, the relationship is characterized with the inverse of its electric resistance. For a capacity element, the relationship is characterized with the inverse of the time integral of its capacity. Because a bond graph model makes it possible to keep track of the magnitude change and change rates of all the measured variables, the temporal

response of a fault manifested in measurements can then be used as fault signatures for fault isolation during transients.

The diagnostic procedure using bond graph is summarized as follows (Mosterman and Biswas, 1997):

- *Generate fault candidates by propagating the observed values backward to some root node.*
- *Predict the fault behavior for each fault candidate and make a comparison between the predicted values and measured values.*
- *If all the measurements are consistent with the predicted values, the candidate fault is chosen as the true fault. Otherwise, the candidate fault is rejected*.

This chapter has reviewed the major modern approaches to fault detection and isolation developed in other industries. Analytical redundancy has been recognized as the foundation of modern FDI approaches. Under this principle, qualitative model and quantitative model based approaches can be derived. Data reconciliation approach, model based approach and pattern recognition approach are quantitative model based approaches. Sign directed graph approach and bond graph approach are qualitative model based approach. The former approach has better resolution in fault isolation and the latter has better reasoning capability. In the thesis, accurate models required by quantitative model based approach will be achieved by data driven modeling such as PCA and ANFIS. Furthermore, in order to organically combine the reasoning capability of qualitative model based approaches and the good resolution of quantitative model based approaches, data driven model causal graph will be developed as a generic approach to fault diagnosis for nuclear power plants.

Chapter 3

General FDI Design for Nuclear SG System

The U-tube steam generator (UTSG) water level control system of a typical four loop PWR is selected to study the FDI methods.

3.1 Description of nuclear SG system

The SG water level control system in PWR has a three-element controller to control the water level in the steam generator as is shown in Figure 3.1. The three elements are steam flow, feed water flow, and steam generator water level. The reference water level is a function of the turbine load and the steam dump rate through steam dump valves. The SG level error signal is the measured level minus the reference level. The flow mismatch error is the fractional steam flow rate minus the fractional feed water flow rate. The combination of the SG level error and the flow mismatch error is used as the input to the controller. The controller output is used to manipulate the feed water control valve position. Because the main control purpose of the SG level control system is to control the water level, the level error has been multiplied by a gain in order to dominate the flow mismatch error signal. In addition, feed water temperature is used to take into account its effect on SG level.

Another control system involved in the nuclear SG system is the control over the speed of the main feed water pump. The objective of this control system is to maintain the feed water control valve position approximately at its midpoint so that the best control performance can be achieved. The system obtains the collected steam flow rate from all the steam generators and generates a reference pressure difference. The error signal is generated from the difference between the reference pressure difference and the actual pressure difference between the collected steam line and the collected feed water line. The error signal is used to control the pump speed. The pressure on the feed water line is

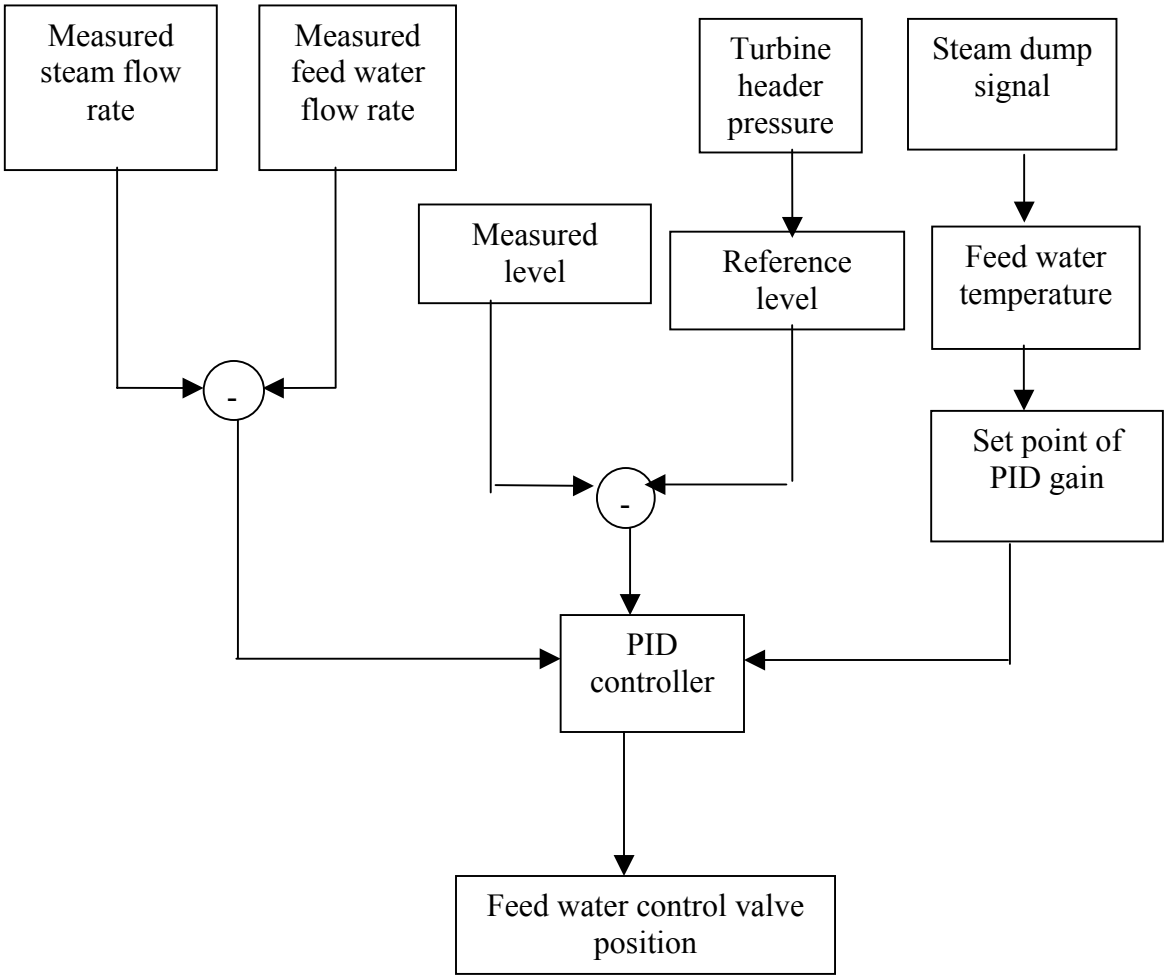

Figure 3.1. SG water level control system.

measured at the upstream of feed water control valve. The pressure on the steam line is measured at the downstream of the main steam isolation valve.

3.2 Available Measurements

Table 3.1 shows the major available measurements relevant to the FDI task for the steam generator system of a nuclear power plant.

3.3 Enumeration of Single Faults

For a large system where thousands of components may be involved, it is usually difficult to know all the possible faults and incorporate them into a fault dictionary. However, in nuclear power plants, the sophisticated reliability analysis can provide rich information about the possible faults from engineering point view at a specified operation lifetime. In addition, probabilistic risk assessment can provide detailed information about the faults of safety concern under different operation conditions. Therefore, it is still of great value to develop FDI methods with the assumption that the faults can be enumerable. In the thesis, the approaches described in Chapter 4 and Chapter 5 deal with the situation where the possible faults are known. Data-driven model causal graph approach developed in Chapter 6 deals with the situation where the possible faults are unknown.

3.4 Enumeration of Simultaneous Faults

Even when the possible single faults are known, their combination will generate many possible simultaneous faults. To keep the number of possible simultaneous faults manageable, a common approach is to ignore those simultaneous faults with very low probability based on engineering experiences. However, these engineering practices are usually very subjective and even so there still might remain a very large number of possible simultaneous faults.

Table 3.1. Available measured variables for the FDI of nuclear SG system

1	Thermal power	23	FCV1 position
2	Reference temperature	24	FCV1 controller output
3	Programmed reference temperature	25	SG1 WR indicator
4	Primary pressure	26	SG 1 NR indicator
4	Cold leg 1 temperature	27	FCV2 position
5	Cold leg 2 temperature	28	FCV2 controller output
6	Hot leg 1 temperature	29	SG NR reference
7	Hot leg 2 temperature	30	SG2 WR indicator
8	Pressurizer pressure	31	SG2 NR indicator
9	Pressurizer heater output	32	SG1 temperature
10	Pressurizer level	33	SG2 temperature
11	Pressurizer reference level	34	TCV1 position
12	Pressurizer spray flow rate	35	TCV1 flow rate
13	Charging flow are	36	TCV2 position
14	Letdown flow rate	37	TCV2 flow rate
15	Surging flow rate	38	TCV3 position
16	SG1 pressure	39	TCV3 flow rate
17	SG2 pressure	40	TCV4 position
18	Feed water temperature	41	TCV4 flow rate
19	SG1 steam flow rate	42	Turbine header pressure
20	SG2 steam flow rate	43	Turbine output
21	Feed water flow rate to SG1	44	Turbine load
22	Feed water flow rate to SG2	45	Turbine RPM

In the thesis, qualitative knowledge based FDI is proposed to limit the number of possible simultaneous faults. For an online FDI system, the qualitative knowledge based FDI module is used to generate only a few fault candidates and their reference fault signatures are generated on-line using a simulator in parallel with the plants. In fact, only one simulation is required to generate data so as to characterize a fault if analytical redundancy based FDI approach instead of pattern recognition method is used.

Qualitative knowledge based FDI needs to have certain system knowledge in order to obtain system causal graph. However, this is not a major problem for nuclear power plants, where the system responses to significant faults can be available from a variety of sources such as system design manual, system operation manual or standard safety analysis report.

The following section shows an example on how to use sign directed graph, a typical qualitative knowledge based approach to generate the reasonable combination of simultaneous faults for the steam generator system.

Figure 3.2 shows the sign directed graph for the steam generator system in which three single faults, feed water flow meter offset fault, SG level sensor offset fault, and steam flow meter offset fault, are considered. Correspondingly, only three root nodes (node 1, node 8 and node 11) and the other eight process nodes are involved. In the graph, the cause effect relationships between variables are represented by the positive and the negative directional connections. Because a feedback control loop is involved, the directional variation of the effect variable during the transients induced by a disturbance on the causal variable is used to represent the cause effect relationship.

The fault candidates can be identified simply by tracking consistent arcs from the effect nodes to the causal nodes starting from all the abnormal measurement nodes. As an example, if the observed symptom is that the feed water flow indicator is low and the SG narrow range level indicator is high, node 2 takes a negative value and node 7 takes a positive value. If the negative value of node 2 is back propagated immediately to node 1, a negative feed water flow meter offset fault can be identified as a possible root cause.

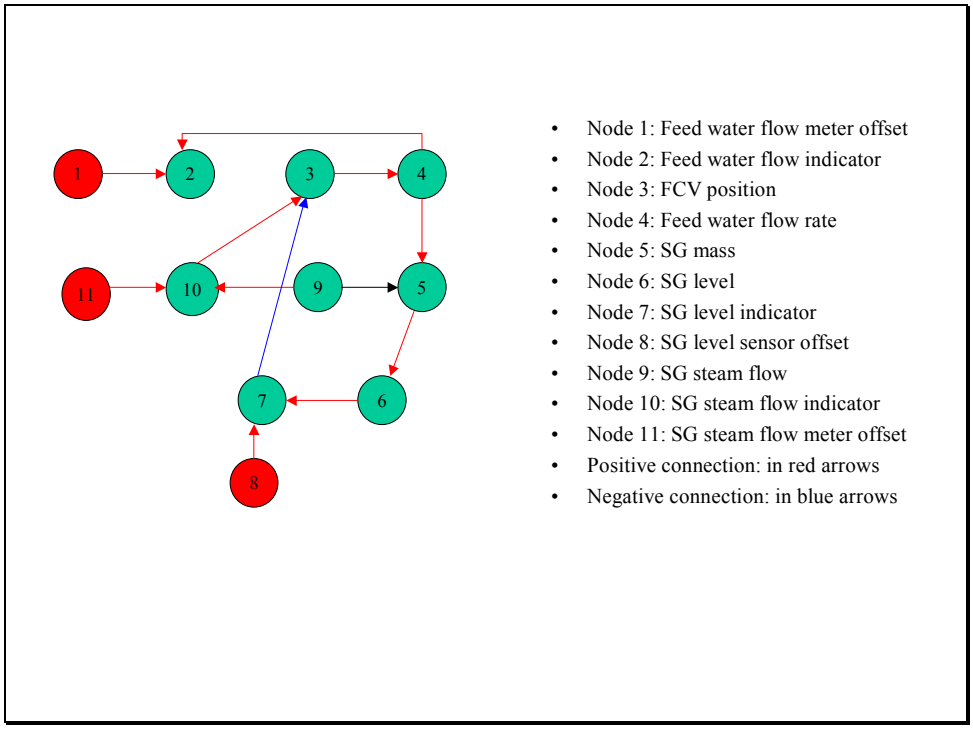

Figure 3.2. SDG graph of nuclear SG system.

If the negative value of node 2 is back propagated to the root node 11 along node 4, node 3, the node 10, a negative steam flow meter offset can be identified as a possible root cause. If the negative value of node 2 is back propagated to node 8 along the node 4, the node 3, the node 7, a positive steam generator level sensor offset fault can be identified as a possible root cause.

If the positive value of node 7 is back propagated immediately to the root node 8, a positive steam generator level sensor offset fault can be identified as a possible root cause. If it is back propagated to the root node 11 along node 6, node 5, node 4, node 3, and node 10, a positive steam flow meter offset fault can be identified as a possible root cause.

In the sign directed graph method, fault localization is a process of determining one or several root nodes whose predicted fault symptoms are in agreement with the measured fault symptoms. In the example, after the possible fault candidates have been identified, the fault symptoms can then be predicted by propagating forward from the root nodes to the measured or the unmeasured nodes. If there is only single fault in the system, a positive SG level offset fault will be isolated as the fault origin since it is able to fully explain the observed fault symptoms. However, if dual faults are considered, the possibility of a negative feed water flow meter offset fault cannot be excluded since its combination with a positive steam generator level sensor offset may also cause the same symptoms.

From this example, it can be seen that the number of possible simultaneous faults can be significantly reduced based on a preliminary SDG analysis. In fact, three single faults and three simultaneous faults need to be considered without SDG analysis.

3.5 Scope of the Studied Faults

Although SDG can be introduced to generate fault candidates for both single and dual faults, the focus of the thesis is on how to detect and isolate these single faults and dual faults.

To design a FDI algorithm that is able to deal with simultaneous faults, the number of the faults has to be manageable from research point of view. Only those

simultaneous faults with meaningful probability in engineering sense have been taken into account. For instance, triple faults occur with much lower probability than dual faults. It is usually enough to incorporate only dual faults. Moreover, inappropriate inclusion of simultaneous faults may degrade the performance of a designed FDI system since not all the possible simultaneous faults can be isolated. Raich and Cinar, 1996 reported that a random variation fault is highly likely to mask with another random fault or another step fault or a ramp fault. Therefore, random variation fault is not taken into account in order to avoid fault masking due to inadequate information.

The following single faults and dual faults are defined in the FDI design for the steam generator system:

- Feed water flow meter offset fault.
- Steam flow meter offset fault.
- Feed water flow meter offset fault and steam flow meter offset fault.
- Feed water flow meter offset fault and SG level sensor offset fault.
- Steam flow meter offset fault and SG level sensor offset fault.
- SG pressure sensor offset fault.
- Feed water flow meter offset fault and SG pressure sensor offset fault.
- Feed water flow meter offset fault and FCV offset fault.
- Steam flow meter offset fault and FCV position offset fault.
- Feed water control valve position offset fault.
- SG pressure sensor offset fault and SG level sensor offset fault.
- SG level sensor offset fault.
- Steam flow meter offset fault and SG pressure sensor offset fault.

3.6 Data Preparation for Modeling

In order to develop appropriate data driven models with enough accuracy for fault detection and isolation, the quality of data preparation is very important. The collected data must cover the entire normal operation range and the entire faulty operation range.

Otherwise, the data driven models will perform extrapolation and the prediction will be unreliable.

The following operation conditions are considered in order to prepare the data for SG modeling:

- Slow power transients at forty power levels beginning from 20% to 100% at an interval of 2% power increase.
- Large power transients from 20% to 100% power level.
- Large power transients from 100% to 20% power level.

The data are collected during slow transient conditions because such data are appropriate to build dynamic models. For a system with feed back control loop, the static information may not enough to detect some faults within the control loop. In addition, it is difficult to generate sufficient amount of data to build data driven models if only steady state data are used. The data collected during large power transients are used to build data driven models to predict the controller output and the control valve position.

When a FDI system is developed for a real application, more process conditions need to be covered. Some of the conditions may be different stages of the reactor life cycle, different ranges of heat transfer rate from the primary side to the secondary side, etc.

A short sampling interval makes it possible to detect a fault faster, but it may bring some high frequency noises into the data and result in time dependence between adjacent observations. This dependence may result in a more complex model to characterize the system behavior. The sampling time is chosen to be 1 second.

The fault data are generated for all the considered faults at 100% power level under the following conditions:

- Drift faults with drift rate at 1%/hr, 2%/hr, 3%/hr, -1%/hr, -2%/hr, -3%/hr.
- Bias faults with offset at 1%, 2%, 3%, -1%, -2%, -3%.

For both the drift faults and the bias fault, the data set for only one fault magnitude is used to provide reference fault signatures and all the other data sets are used to test the reliability of the developed FDI approaches. Although most of the test cases

are based on 100% full power, one special test case is designed to show the performance of the developed FDI algorithm in dealing with a fault at another power level 80%.

3.7 Fault Response Analysis

Before a FDI system is designed, it is necessary to have some qualitative fault response analysis. This analysis facilitates to determine the structure of the models to be developed. The value range of the affected variables can also be examined after the faults of interest occur so that the developed data driven models have reliable generalization capability. In addition, the analysis can help to examine the effects of controller feedback.

3.7.1 Feed water flow meter positive offset fault

When the feed water flow meter has a positive offset fault, the initial responses are as follows:

- The indicated feed water flow meter increases.
- The FCV valve position decreases.
- The SG water level decreases.
- The SG pressure does not change.
- The steam flow rate has a very slight increase.

The reason why steam flow rate does not change significantly is that the steam flow rate is mainly determined by the power demand.

When a new steady state is reached, the system responses are as follows:

- The indicated feed water flow rate increases.
- The FCV position returns to its initial level.
- The SG water level decreases.
- The SG pressure does not change.
- The steam flow rate returns to its initial value.

After the new steady state, the steam flow rate is equal to the actual feed water flow rate instead of the measured feed water flow rate. The error signal to the controller

becomes zero. This is achieved by the fact that the SG water level has decreased although the indicated feed water flow rate has increased. To maintain the constant reactor power output during the fault, the steam flow does not change.

In the process, the indicated feed water flow rate will be consistently higher than the true value. A model capable of calculating the true value of feed water flow rate can be used to isolate the feed water flow meter fault.

3.7.2 Steam water flow meter positive offset fault

When the steam flow meter has a positive offset fault, the initial responses are as follows:

- The feed water flow meter increases.
- The FCV valve position increases.
- The SG water level increases.
- The SG pressure does not change.
- The indicated steam flow rate increases.

When a new steady state is reached, the system responses are as follows:

- The feed water flow rate returns to its initial level.
- The FCV valve position returns to its initial level.
- The SG water level increases.
- The SG pressure does not change.
- The indicated steam flow rate increases.

After the new steady state, the actual steam flow rate instead of the indicated steam flow rate is equal to the feed water flow rate. The error signal to the controller becomes zero. This is achieved by the fact that the SG water level increases although the indicated steam water flow rate has increased.

In the process, the indicated steam flow rate will be consistently higher than the actual value. A model capable of calculating the actual value of the steam flow rate can be used to isolate the steam flow meter fault.

3.7.3 Steam generator pressure sensor offset fault

When the steam generator pressure sensor has a positive offset fault, the speed of the main feed water pump will change. The proportional integral controller of main feed water pump uses the pressure difference between the steam line and the feed water line as input to control the valve position of feed water control valve approximately at its midpoint. The initial response to SG pressure sensor positive offset fault is that the feed water flow rate will increase due to the increase in the feed water pump speed. To be followed is that the SG water level will increase. After a new steady state is reached, the feed water flow rate will go back to its initial level. The steam flow rate does not have significant changes. The SG level will go back to its initial level after the new steady state is reached.

3.7.4 Feed water control valve position fault

When the feed water control valve has a positive offset fault during its actuation when a command is received from the controller output, the initial responses are as follows:

- The feed water flow meter increases.
- The FCV valve position increases.
- The SG water level increases.
- The SG pressure decreases.
- The steam flow rate doe not change.

When a new steady state is reached, the system responses are as follows:

- The feed water flow rate does not change.
- The actual FCV valve position returns to its initial value.
- The SG water level increases.
- The SG pressure does not change.
- The steam flow rate does not change.

When the feed water control valve has a positive offset fault, the initial response is that the feed water flow rate increases. The controller will then reduce the open-width of the feed water control valve using the mismatch signal between the FCV flow rate and the SG steam flow rate. To be followed is that the SG water level increases and the FCV controller will reduce the open-width of FCV valve position again. In the end, the FCV flow rate and the FCV valve position return to their initial levels.

The SG level changes as the integral effect of the change of the FCV flow rate during the fault progression. The SG final level is greater than the initial level.

Because the command has a positive bias, the controller output will be negative so that the actual actuation of the feed water control valve is zero because the SG water level has increased when a new steady state is reached.

3.7.5 SG narrow range level sensor fault

When the SG narrow range level sensor has a positive offset fault, the initial responses are as follows:

- The feed water flow meter decreases.
- The FCV valve position decreases.
- The indicated SG water level increases.
- The SG pressure has a slight decrease since the actual water level decreases.
- The steam flow rate has a very slight increase.

When a new steady state is reached, the system responses are as follows:

- The feed water flow rate returns approximately to its initial level.
- The FCV valve position returns approximately to its initial level.
- The indicated SG water level returns to its initial level.
- The SG pressure has a slight decrease.
- The steam flow rate returns approximately to its initial level.

When the steam generator level sensor has a positive offset fault, the initial action of the controller is to reduce the open-width of the feed water control valve. The feed

water flow rate decreases at the beginning and tends to go back to its initial level in the end.

The steam generator indicated level would go back approximately to its initial level after steady state is reached. However, the SG true level would be lower than its initial level with an offset equal to the magnitude of the sensor bias as is shown in Figure 3.3. In the fault process, the reactor power load does not change and nor does the SG narrow range reference level. After the new steady state, the steam flow rate is equal to the feed water flow rate.

Table 3.2 summarizes the fault responses to the five single faults. Based on the table, if only steady state information is used, controller output signal must be used in order to detect FCV position fault. Process redundancy is not sufficient in order to detect SG narrow range level sensor fault. Instead, measurement redundancy should be taken into account.

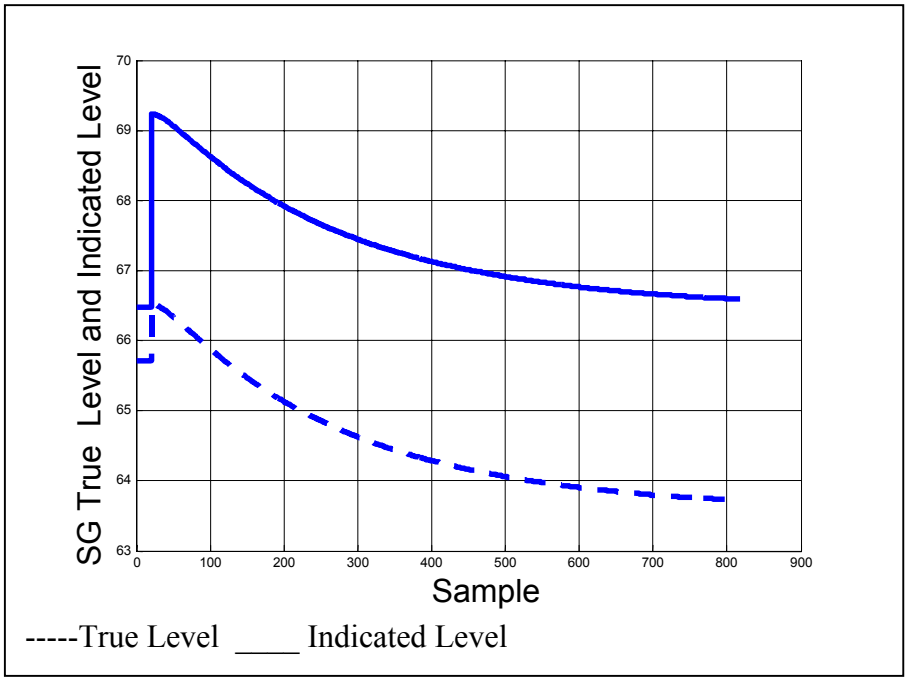

Figure 3.3. SG Level measurement responding to SG NR level sensor fault.

Table 3.2. Summary of the steady state responses to the single faults

	Feed water flow rate	SG level	Steam flow rate	SG pressure	Controller output
Feed water flow meter positive offset fault	↑	↓	0	0	0
Steam flow meter positive offset fault	0	↑	↑	0	0
SG pressure sensor positive offset fault	0	0	0	↑	0
FCV position positive offset fault	0	↑	0	0	↓
SG narrow range level sensor positive offset fault	0	0	0	0	0

Chapter 4

Principal Component Analysis for Fault Diagnosis

4.1 PCA Algorithms

From the point of view of data, PCA is a dimensional reduction method. The original data can be represented in a lower dimensional space without significant loss of the variability. From the modeling point of view, PCA transforms correlated variables into uncorrelated ones and determines the linear combinations with large and low variability (Flury, 1988).

Before the original data are transformed into a lower dimensional space, they are mean centered because only the variability of the data is of interest. The data are standardized with unit variance so that equal weights are given to all the variables as far as their variability is concerned.

For the original data matrix X associated with n observations and m measured variables, the first principal component is obtained by finding out a linear transformation column vector P_1 such that the scores t_1 of the original data along this component has maximized variance. In mathematical term, this is given by:

$$t_1 = XP_1$$

$$E_1 = \text{var}(t_1) = \text{var}(XP_1) = P_1'X'XP_1 = P_1'\Sigma P_1 \quad \text{is maximized.}$$

where

$$\Sigma = X'X = (n-1)S$$

$$S = \frac{1}{n-1}(X'X)$$

$S =$ sample covariance matrix.

An additional constraint on the transformation column vector is to normalize its length.

In order to maximize the variance of t_1 with the above constraints, we can set the derivative of E_1 to zero and include a Lagrange multiplier λ_1 to ensure the constraint is satisfied as well:

$$\frac{\partial}{\partial P_1}(E_1 + \lambda_1(I - P_1'P_1)) = (\Sigma'+\Sigma)P_1 - 2\lambda_1 P_1 = 0$$

Therefore,

$$\Sigma P_1 = \lambda_1 P_1 \tag{4.1}$$

In the above derivation, we have used the formula for the derivative of a matrix with respect to a vector:

$$\frac{\partial}{\partial x}(x'Ax) = (A'+A)x$$

$$\frac{\partial}{\partial x}(yx) = y'$$

The variance of the original data along the first principal component is then given by:

$$E_1 = \text{var}(t_1) = \lambda_1 P_1'P_1 = \lambda_1 \tag{4.2}$$

Equation (4.1) and Equation (4.2) show that the transformation vector to obtain the first principal component is actually the eigenvector of $X'X$ corresponding to its maximum eigenvalue.

In order to obtain the j^{th} principal component, the following constraints must be satisfied (Jackson, 1991):

$$E_k = \text{var}(t_k) = P_k'\Sigma P_k \text{ is maximized}$$

$$P_k'P_k = 1$$

$$XP_k \text{ is orthogonal to } XP_j \text{ for } j = 1,2....k-1$$

In order to maximize E_k with the above constraints, we can set the derivative of E_k to zero, include a Lagrange multiplier λ_k to ensure $P_k'P_j = 1$ and $j-1$ Lagrange multipliers ϕ_j to ensure XP_k is orthogonal to XP_j for $j = 1,2....k-1$:

$$\frac{\partial}{\partial P_k}(E_k + \lambda_k(I - P_k'P_k) + X\sum_{j=1}^{k-1}\phi_j(0 - P_j'P_k) = (\Sigma'+\Sigma)P_k - 2\lambda_k P_k + X\sum_{j=1}^{k-1}\phi_j P_j = 0 \tag{4.3}$$

If we left-multiply Equation (4.3) by P_j' for $j = 1,2....k-1$ and notice the orthogonal property of P_j, then we have:

$$\phi_j = 0 \text{ for } j = 1,2,.....k-1$$

$$\Sigma P_k = \lambda_k P_k$$

$$E_k = \lambda_k$$

If the score vectors are combined, then we have:

$$T = XP$$

The matrix P is constructed by the columns of the eigenvectors of $X'X$ whose left-columns correspond to larger eigenvalues than the right columns. The matrix P is called loading matrix in the sense that the original data can be loaded to a lower dimensional if its dimension is chosen to be fewer than the number of variables in the original data.

The loading matrix can be used to reconstruct the original data without loss of much information given by:

$$X = TP' \tag{4.4}$$

From the equation, it can be seen that all the information contained in the original data set has been compressed into the loading matrix. If the loading matrix is known, the original data can then be reconstructed.

4.2 PCA for Fault Detection

During normal operation, the sample covariance of the measured data is governed by the physics in the process. Its structure will change if a fault occurs in the process. If a PCA model is used to describe the covariance structure of the measured data for the fault free condition, a fault can be detected when the model cannot explain the new observed data. Two cases may make the PCA model fail to explain the new data. The first case is that the new observation deviates from the mean of the normal operation defined by the effective region of the PCA model in the score space. The other case is that the residual of the model has changed significantly. The model residual represents the noise and the redundant information of a system. If a fault occurs, the characteristics of the noise and the redundancy are expected to change.

4.2.1 T^2 statistics

If the score of a new observation is significantly outside the region defined by the scores of the fault free data, a fault may have occurred. If the scores of the fault free data satisfy multivariate normal distribution, the decision ellipse can be given by:

$$T'\Sigma^{-2}T < T_\alpha^2 \tag{4.5}$$

where

$$T_{\alpha}^{\ 2} = \frac{p(n-1)}{n-p} F_{p,n-p,\alpha} \tag{4.6}$$

and

p= number of variables.

n= number of observations.

α = significance level.

The disadvantage of T^2 statistics is that it may be oversensitive to the small elements of Σ_a and result in high false alarm rate.

4.2.2 Q statistics

Q statistics can be used to test whether the principal component model can still explain a new observation. The random variable used for this testing is the sum squared error R of the original PCA model defined by the following equation:

$$R = r'r \tag{4.7}$$

where

$$r = (I - PP')x$$

If the sum squared error measuring the total sum of the variation in the residual space exceeds the Q threshold, it indicates that the original PCA model cannot explain the new data. The threshold of Q statistics Q_{α} is defined as follows (Jackson and Mudholkar, 1979)

$$Q_{\alpha} = a(b + cz)^d \tag{4.8}$$

where

z= the critical value for standard normal distribution at a given significance level.

$$a = \sum_{i=k+1}^{p} \lambda_i$$

$$b = 1 + [\theta_2 h_0 (h_0 - 1)] / a^2$$

$$c = (\sqrt{2\theta_2} h_0) / a$$

$$d = 1 / h_0$$

$$\theta_2 = \sum_{i=k+1}^{p} \lambda_i^{\ 2}$$

$$\theta_3 = \sum_{i=k+1}^{p} \lambda_i^{\ 3}$$

$$h_0 = (1 - 2a\theta_3)/(3\theta_2^{\ 2})$$

4.3 PCA for Fault Identification

The task of fault identification is to determine what are the most affected variables once a fault happens. These variables are usually most relevant to fault diagnosis. Fault identification is useful because it can help operators focus their attention on a reduced number of variables. The out-of-status score can be approximated by (Russell and Chiang, 2000):

$$(\frac{t_i}{\sigma_i})^2 > \frac{T_\alpha^2}{k}$$

where

σ_i = singular value, $i = 1,2,.....s$.

$s =$ the number of scores considered to be responsible for the out-of-control status.

$k =$ the number of principal components.

The contribution of one original variable to one of the out-of-control scores can be expressed as follows:

$$C_{i,j} = \frac{(t_i)(x_j P_{ji})}{\sigma_i^{\ 2}}$$

where

$C_{i,j}$ = the contribution of variable x_j to the out-of-control scores t_i.

The total contribution of the j^{th} variable x_j to the out-of-control status can be given by:

$$\mathrm{C_j} = \sum_{i=1}^{s} C_{i,j}$$

C_i = the contribution of variable x_j to the out-of-control status.

The fault identification measure can also be defined based on the normalized error R_j given by (Russell and Chiang, 2000):

$$R_j = \frac{r_j}{\hat{s}_j}$$

where

$$r_j = x_j - P_j P_j' x$$

$$\hat{s}_j^{\,2} = \sum_{i=k+1}^{m} P_{ij} \sigma_i^{\,2} \tag{4.9}$$

4.4 PCA fault Isolation Versus Parity Space Approach

Gertler et al (1999) reported that there is an inherent consistency between PCA approach and parity space approach when used for fault isolation. A linear static system can be described as follows:

$$y(t) = Au(t) + B\begin{pmatrix} \Delta u \\ \Delta y \end{pmatrix}$$

where

$y(t)$ = the observed outputs.

$u(t)$ = the controlled inputs or the measured inputs.

Δu = the disturbances or the unknown faults related to $u(t)$.

Δy = the disturbances or unknown faults related to $y(t)$.

A and B= known system matrices.

If we combine all the measured variables $u(t)$ and $y(t)$ in a column vector denoted as $x(t)$, a set of residuals can then be defined as:

$$o(t) = [-A, I]x(t) = B\Delta x \tag{4.10}$$

When PCA is performed, the residual vector is given by:

$$o(t) = (x + \Delta x) - PP'(x + \Delta x) = (I - PP')(x + \Delta x) = QQ'(x + \Delta x(t)) \tag{4.11}$$

where

P = the eigenvectors that span the principal component space.

Q = the eigenvectors that span the residual space.

In the above derivation, we have used the property of orthonornal matrices:

$$PP'+QQ'=I$$

In addition, we have assumed that the variances of the scores on the eigenvectors corresponding to trivial components are approximately zero:

$$\text{var}(t_i) = \text{var}(Q'x) \approx 0$$

Correspondingly, if the original data are mean centered, then

$$Q'x = 0$$

Therefore,

$$o(t) = QQ'\Delta x(t)$$

The residual vector $o(t)$ generated in this way can be directly used as a parity vector for fault isolation since any nonzero component of the parity vector corresponds to only one faulty measurement. Such a FDI approach has great advantages in its easy implementation. However, the linear PCA algorithm and the method of obtaining residual vector for fault isolation are not applicable for a non-linear system.

4.5 PCA Fault Isolation Based on Fault Direction

Yoon and MacGregor (2001) reported that the fault directions both in the model space and in the residual space should be used in order to isolate a complex fault.

If a fault occurs in a control loop, the fault effects may propagate within the control loop after a new steady state is reached. Therefore, the developed PCA model from fault free conditions cannot be used to characterize the new relationship. This has twofold implications. The first one is that the linear redundant relationships between the variables have changed. The second one is that the system status has changed. The former can be represented by the residual change in the residual space and the latter can be represented by the score change in the model space.

Combining the system status change and the model structure change, a fault vector can be characterized by the superposition of two fault vectors defined in the model space and in the residual space as follows:

$$\vec{f} = f_1\hat{u} + f_2\hat{v} \tag{4.12}$$

where

$f_1\hat{u}$ = the fault vector defined in the principal component space.

$f_2\hat{v}$ = the fault vector defined in the residual space.

The developed PCA model for fault free conditions can be used to decompose a measurement vector x into two spaces, one component $x_1\hat{u}$ in the model space and the other one $x_2\hat{v}$ in the residual space, that is:

$$x = x_1\hat{u} + x_2\hat{v} \tag{4.13}$$

Therefore, the fault direction in the residual space can be defined as:

$$\hat{v} = \frac{x_2^{post} - x_2^{initial}}{\| x_2^{post} - x_2^{initial} \|} \tag{4.14}$$

where

$x_2^{post} = (I - PP')x^{post}$

$x_2^{initial} = (I - PP')x^{initial}$

P =the loading matrix of the developed PCA model for fault free conditions.

x^{post} =the measurement obtained after a new steady state has reached since a fault.

$x^{initial}$ =the measurement before a fault.

Since $x_2^{initial} \approx 0$, then

$$\hat{v} = \frac{x_2^{post}}{\| x_2^{post} \|} \tag{4.15}$$

The direction defined in the residual space characterizes the change of the model structure after a fault. However, the fault direction defined in the residual space may not be sufficient for fault isolation. The system status change before and after a fault also provides significant information to characterize a fault. The direction starting from the initial plant condition before a fault and pointing to the condition after a fault in the principal component space can be used to define the fault direction in the model space:

$$\hat{u} = \frac{x_1^{post} - x_1^{initial}}{\| x_1^{post} - x_1^{initial} \|} \tag{4.16}$$

where

$x_1^{post} = PP'x^{post}$

$x_1^{initial} = PP'x^{initial}$

After the fault signatures have been defined by fault directions, fault isolation can be achieved based on the angle of the fault vector both in the model space and in the residual space between a detected fault and some reference faults. A fault is isolated as one defined in a reference fault dictionary whose fault direction is most collinear with that of the detected fault both in the model space and in the residual space.

4.6 Determination of the Number of Constraints

Since PCA is in full agreement of parity space approach when used for fault detection and isolation, it is crucial for a successful FDI system to find out all the constraints inherent in the process system. In the context of PCA based FDI, the constraint equations are implicitly represented by the eigenvectors spanning the residual space. Therefore, the correct choice of the number of principal components is important for PCA based FDI.

The most commonly used criteria are cumulative percent variance, Scree plot, average Eigenvalue, and cross validation (Wold, 1978). Cumulative percent variance method selects the number of principal components by setting a subjective threshold of cumulative percent variance so that the model fitness and the parsimony in using principal components are balanced. Scree plot method is based on the plot of the fraction of variance explained by each principal component. The plot orders the principal components from the one that gives the largest amount of explanation to the one that gives the least amount of explanation. This method considers the beginning point of the Scree as the most reasonable number of principal components. Average eigenvalue method assumes that all the components whose corresponding eigenvalues are less than the average value should be discarded. In addition, cross validation can also be used.

4.7 Recommended PCA Based FDI Procedure

The procedure to implement a PCA based FDI is proposed as follows:

(1) Become familiar with the system.

(2) Get information on the operation history of the system and collect the operation experiences of similar plants.

(3) Select faults of interest from an engineering point of view. The reliability data of the components, the environment of the components, the consequences of the component failure etc., should be taken into account.

As far as dual faults are concerned, the selection is mainly safety oriented.

(4) Study the fault responses of the selected faults.

(5) Collect data and evaluate its quality for fault free conditions.

(6) Build a PCA model that is able to characterize the relationships among the measured variables.

(7) Quantitatively define the fault directions for all the faults and save them in a fault dictionary. In effect, only one experiment or one simulation is needed in order to determine the fault direction for each fault.

(8) Implement PCA fault detection using both Q statistic and T^2 statistic.

(9) Implement PCA fault isolation based on fault directions defined both in the model space and in the residual space.

4.8 Application to Nuclear Plant SG System

The PCA based FDI algorithm has been implemented for a PWR steam generator system.

4.8.1 Development of PCA model

A good model to characterize the relationships between the measured variables plays an essential role in PCA based FDI algorithm.

Table 4.1 lists the fifteen measured variables used to develop the PCA model for the SG system. Before the simulated data are used to build a model, some Gaussian noise is added to the data based on the measurement errors of the corresponding sensors.

Figure 4.1 shows the fractions of the variance contained in the data explained by the 15 eigenvectors. If a threshold of 98% percent is chosen, the number of principal components is then determined to be eight. It should be noted that too few principal components will decrease the accuracy of model prediction and too many principal components will increase the complexity of the model. A complicated model is able to reduce the training error, but it will lose the capability of generalization because some of the degrees of freedoms are only used for modeling the noise.

Figure 4.2 shows the predicted SG narrow range level and the actual values. It can be seen that the model can predict the trend of the actual data. The choice of more principal components may increase the accuracy of predicting the training data, but it may result in over-fitting.

The eight eigenvectors to define the model space are as follows:

-0.2706	-0.0013	-0.0970	-0.0946	-0.0818	-0.2759	0.1669	0.3104
-0.2673	0.0053	-0.2591	0.3724	-0.3567	0.6139	-0.2662	0.3839
-0.2701	-0.0006	-0.1373	-0.0628	-0.0795	-0.1317	0.0112	-0.0808
0.2499	-0.0442	-0.6176	0.1526	0.0334	-0.1397	0.0593	-0.0636
-0.2666	-0.0229	-0.2695	-0.4125	0.7342	0.3625	-0.0562	0.0975
-0.2709	0.0074	-0.0580	0.0030	-0.1170	0.1376	0.2739	-0.3698
-0.2709	0.0064	-0.0576	0.0014	-0.1221	0.1314	0.3564	-0.3026
0.0268	0.9976	-0.0610	-0.0125	0.0149	-0.0075	-0.0000	0.0017
-0.2693	0.0166	0.1404	0.7813	0.5015	-0.2066	0.0428	-0.0255
-0.2709	0.0025	-0.0661	-0.0614	-0.0998	-0.2263	-0.5195	-0.2465
-0.2706	-0.0006	-0.0990	-0.0795	-0.0403	-0.2202	-0.5298	-0.2459
-0.2706	-0.0014	-0.0968	-0.0956	-0.0829	-0.2842	0.1756	0.3432
-0.2706	-0.0014	-0.0968	-0.0956	-0.0829	-0.2848	0.1756	0.3427
-0.2709	0.0074	-0.0580	0.0030	-0.1171	0.1380	0.2736	-0.3691
0.2497	-0.0441	-0.6198	0.1270	0.0154	-0.1386	0.0584	-0.1204

Table 4.1. Measured variables used to develop PCA model

Variable number	Variable Description
1	Thermal power
2	Cold leg 1 temperature
3	Hot leg 1 temperature
4	SG1 pressure
5	Feed water temperature
6	SG1 steam flow rate
7	Feed water flow rate to SG1
8	FCV1 position
9	FCV1 controller output
10	SG1 WR indicated level
11	SG 1 NR indicated level
12	SG WR reference
13	SG NR reference
14	TCV1 flow rate
15	SG1 temperature

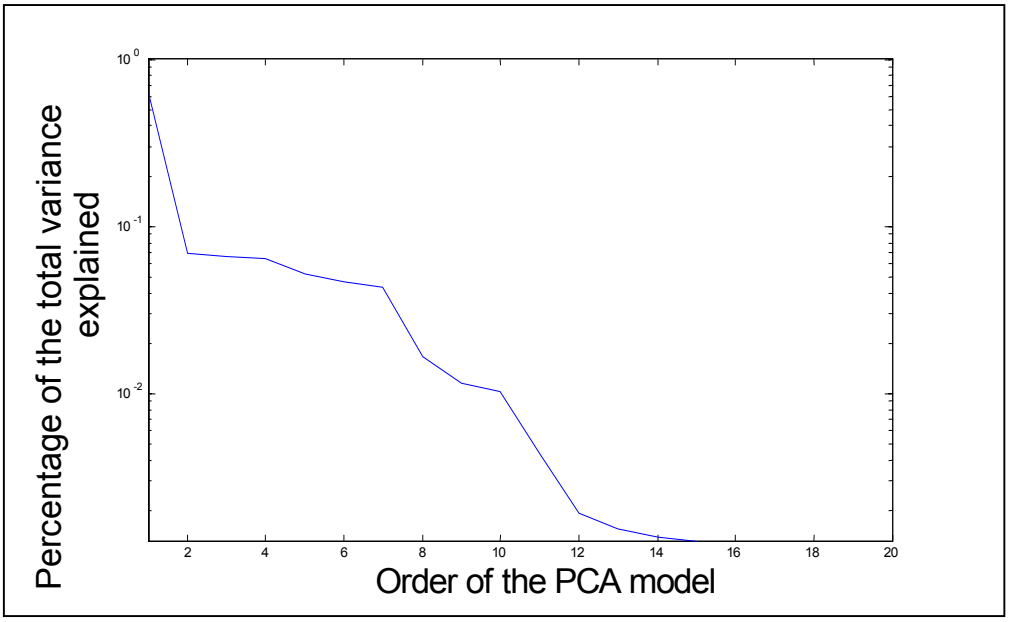

Figure 4.1. Fractions of the variance explained by different PC components.

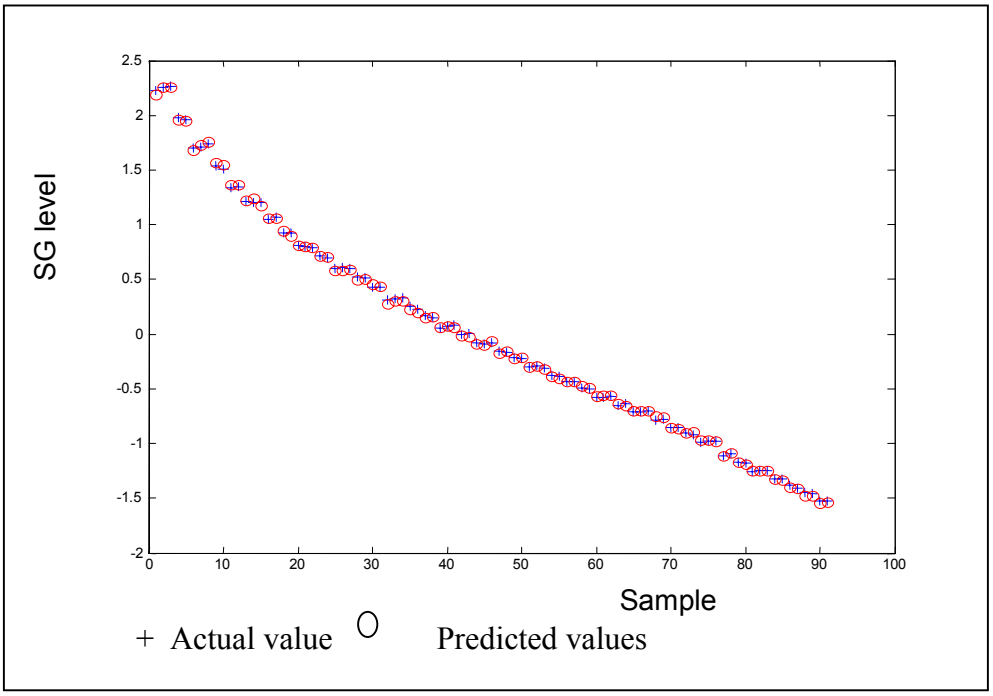

Figure 4.2. Comparison between the predicted SG NR level and the actual values.

The seven eigenvectors to define the residual space are as follows:

0.0142	0.1209	0.0868	-0.5418	0.0572	0.6171	-0.0379
0.0165	-0.0024	0.0097	-0.0023	-0.0003	0.0020	-0.0004
-0.0437	-0.9318	0.0606	0.0029	-0.0031	-0.0036	-0.0002
-0.7060	0.0635	-0.0159	0.0004	0.0008	0.0001	-0.0002
0.0059	0.0185	-0.0457	-0.0030	0.0006	0.0031	-0.0001
-0.0116	0.1161	0.0205	0.0127	-0.3934	0.0088	-0.7140
-0.0099	0.1070	-0.0597	-0.0433	-0.4195	0.0017	0.6925
-0.0001	0.0011	-0.0002	-0.0000	-0.0004	0.0000	0.0006
0.0196	-0.0074	-0.0060	-0.0004	0.0001	0.0006	0.0000
-0.0077	0.1015	-0.7152	-0.0456	0.0102	0.0513	-0.0007
-0.0051	0.1931	0.6822	0.0506	-0.0591	-0.0618	0.0824
0.0165	0.1168	-0.0467	0.7910	-0.0176	0.1729	0.0018
0.0160	0.1173	-0.0453	-0.2680	0.0117	-0.7629	-0.0445
-0.0112	0.1127	0.0315	0.0470	0.8136	-0.0286	0.0212
0.7056	0.0058	-0.0143	-0.0006	-0.0002	0.0004	0.0004

From the eigenvectors in the residual space, the following approximate linear relationships among the measured variables can be derived:

-0.7140*steam flow rate+0.6925*feed water flow rate=0

0.6171*power+0.1729*WR reference level -0.7629*NR reference level=0

-0.3934*steam flow rate-0.4195*feed water flow rate+0.836*TCV flow rate=0.0

-0.5418*power+0.7910*WR reference level-0.2680*NR reference level+0.047*TCV flow rate=0.0

0.0868*power+0.0606*hot leg temperature-0.0597*Steam flow rate-0.7152*WR level+0.6822*NR level=0.0

0.1209*power-0.9318*hot leg temperature-0.0635*SG pressure+0.1161*Steam

flow rate+0.1070*Feed water flow rate+0.1015*WR level+0.1931*NR level+0.1168*WR reference level+0.1173*NR reference level+0.1127*TCV flow=0.0

-0.7060*SG pressure + 0.7065*SG temperature=0.0

These equations can be used to reveal the linear relationship among variables. The corresponding physical relations can be written as follows:

Steam flow rate = feed water flow rate

SG wide range reference level =f(SG narrow range reference level, power)

Steam flow rate + feed water flow rate = TCV flow rate

SG reference level = f (power, TCV flow rate)

SG narrow range level =f(SG wide range level, power, hot leg temperature, feed water flow rate)

Hot leg temperature = f (power, SG pressure, SG NR level, SG flow rate)

SG temperature =f (SG pressure)

As can be seen, all the above equations have clear physical meanings. However, PCA model cannot capture the nonlinear relationship among variables. For example, the PCA model cannot reveal the relationship between FCV valve position and FCV flow rate. Another point that should be emphasized in using PCA for FDI is that the measurements must be carefully selected before a PCA model is to be built. If the available measurements do not allow finding out some relations among variables that are the basis to isolate some faults, these faults will hence not be isolated.

4.8.2 Fault detection

Fault detection can be performed based on the developed PCA models.

Figure 4.3 and Figure 4.4 shows the T Square and Q statistics for the fault free data, respectively. The red lines in the two figures are the T square or the Q statistical limits corresponding to 99% confidence level. If the corresponding statistics exceeds the limit, the confidence to state that the fault free model cannot explain the data is at a level greater than 99 %. The two figures illustrate that all the fault free data are well below the limit lines. The probability of false alarms due to process disturbance is low.

Figure 4.5 and Figure 4.6 show the T square and Q statistics based fault detection for feed water flow meter and steam flow meter drift faults. If the confidence level is chosen to be 99%, the false alarm rate and missing detection rate is shown as follow:

Detecting Fault: Normal Operation
PCA detection
False alarm rates by T2+Q testing = 0.04
False alarm rate by T2 testing = 0.03
False alarm rate by Q testing = 0.01
Detecting Fault: Feed Water Flow Meter Drift Fault
PCA detection
missing detection rate by T2+Q testing = 0.000000
missing detection rate by T2 testing = 0.012547
missing detection rate by Q testing = 0.095358
Detecting Fault: Steam Flow Meter Drift fault
PCA detection
missing detection rate by T2+Q testing = 0.000000
missing detection rate by Q testing = 0.077792
missing detection rate by T2 testing = 0.011292
missing detection rate by Q testing = 0.100376
Detecting Fault: Steam Flow Meter Feed Flow Meter Drift Faults
PCA detection
missing detection rate by T2+Q testing = 0.000000
missing detection rate by T2 testing = 0.010038
missing detection rate by Q testing = 0.115433
Detecting Fault: Feed Flow Meter Drift Fault and SG Level Sensor Drift Fault
PCA detection
missing detection rate by T2+Q testing = 0.000000
missing detection rate by T2 testing = 0.013802
Detecting Fault: Steam Flow Meter Drift Fault and SG Level Sensor Drift Fault

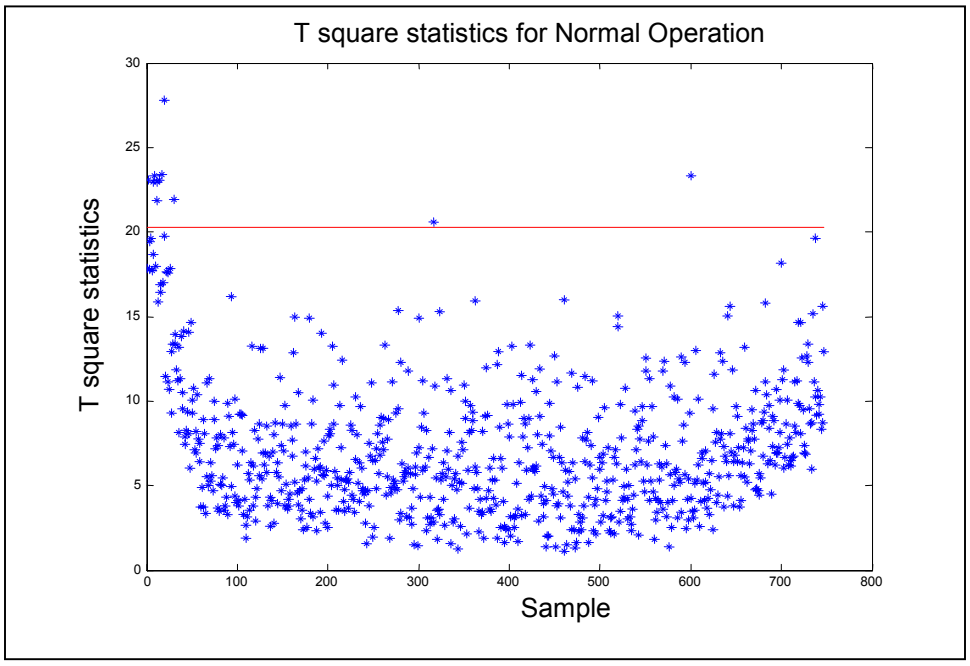

Figure 4.3. T square statistics for the normal data.

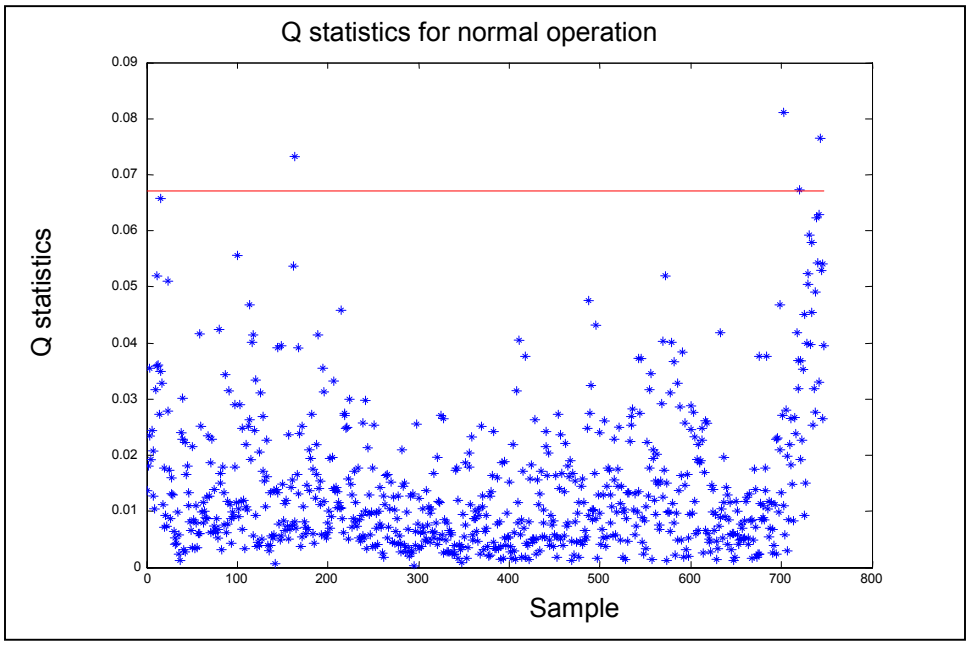

Figure 4.4. Q statistics for the normal data.

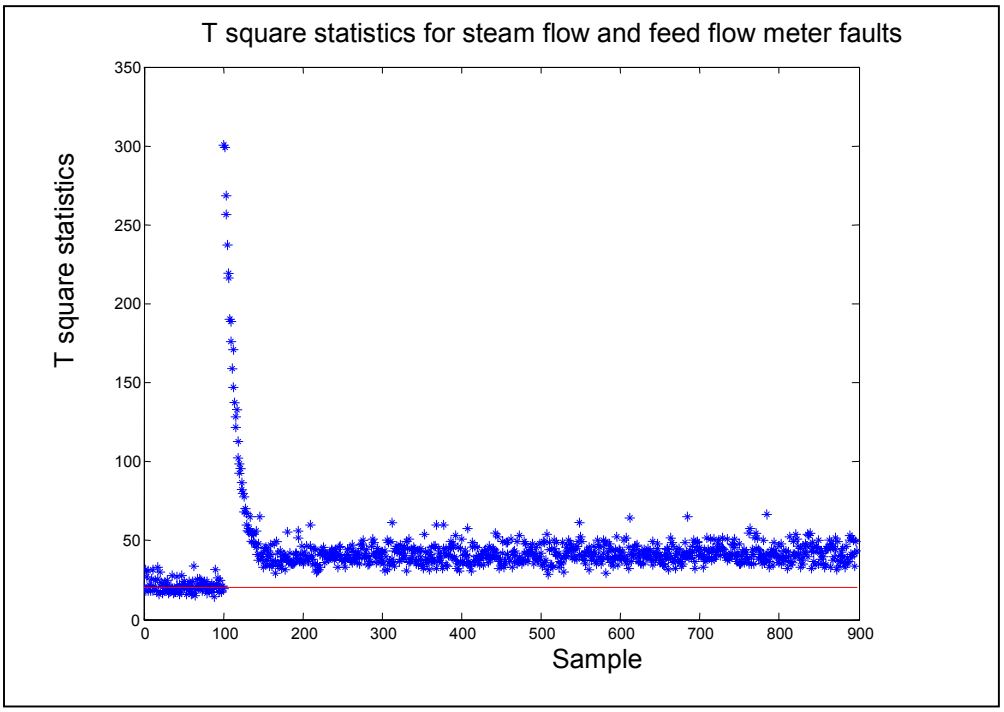

Figure 4.5. T square statistics to detect steam flow meter and feed water flow meter drift faults.

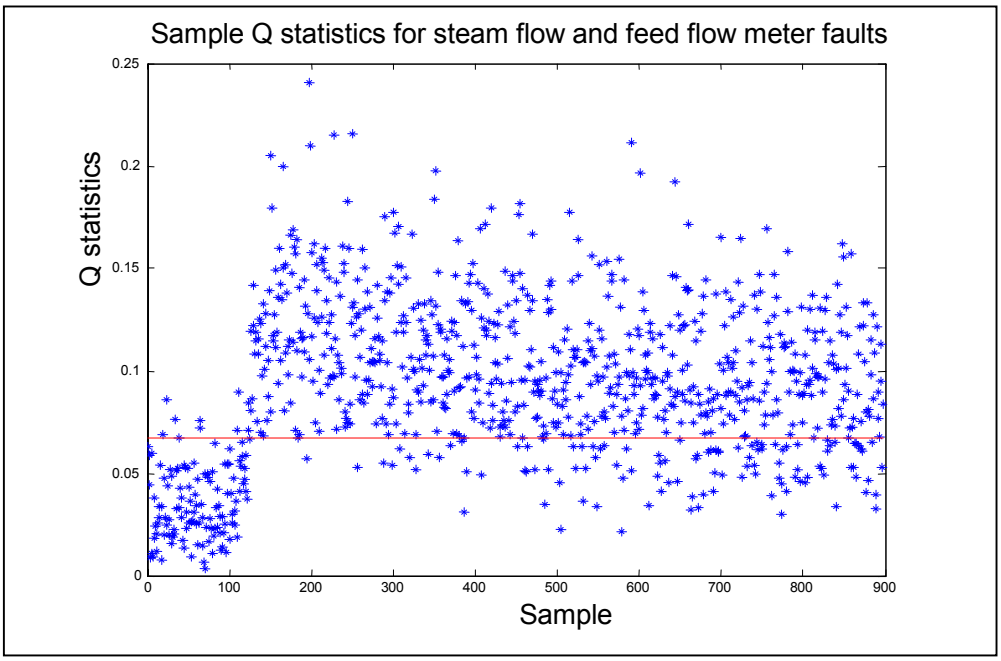

Figure 4.6. Q statistics for steam flow meter drift fault and steam flow meter drift fault.

PCA detection
missing detection rate by T2+Q testing = 0.000000
missing detection rate by T2 testing = 0.006274
missing detection rate by Q testing = 0.042660
Detecting Fault: SG Pressure Sensor Drift Fault
PCA detection
missing detection rate by T2+Q testing = 0.000000
missing detection rate by T2 testing = 0.002789
missing detection rate by Q testing = 0.068042
Detecting Fault: Feed Water Flow Meter Drift fault & SG Pressure Sensor Drift Fault
PCA detection
missing detection rate by T2+Q testing = 0.000000
missing detection rate by T2 testing = 0.001255
missing detection rate by Q testing = 0.097867
Detecting Fault: SG Level Sensor Drift Fault & SG Pressure Sensor Drift Fault
PCA detection
missing detection rate by T2+Q testing = 0.000000
missing detection rate by T2 testing = 0.006274
missing detection rate by Q testing = 0.096612
Detecting Fault: SG Level Sensor Drift Fault

It can be seen that the missing detection rate is small for all the selected faults. It should be kept in mind that both T^2 and Q statistics must be used for fault detection. Either statistics being violated will signify that a fault has happened. The violation of T^2 statistics represents that the system operates at an abnormal state beyond the model space. The violation of Q statistics represents that some of the constraint equations defined in the residual space are violated and the system is abnormal.

PCA can only deal with steady state condition or a slow dynamic process. The algorithm to perform PCA based fault detection is only applicable to steady state condition. When the false alarm rate and the missing detection rate are carefully examined, the false alarm rate and the missing detection rates are not equal to the expected value of one percent. The reason is that the probabilistic distribution underlying the data used to build the model is not normal. Therefore, it is reasonable that the false alarm rate and the missing detection rate are not equal to the specified significance level. The significance level should be determined using experiences obtained from testing the FDI design on the process system.

It should also be noticed that the confidence level would affect the false alarm rate and the missing detection rate. A higher confidence level tends to result in a smaller false alarm rate but a higher missing detection rate. In a real application, the confidence level needs to be adjusted according to the operation requirements.

4.8.3 Fault identification

Figure 4.7 and Figure 4.8 show the contribution plots of the abnormal scores and the abnormal residuals for the feed water flow meter drift fault, respectively. The contribution plots show that the most affected variables for the feed water flow meter drift fault as follows:

- Reactor power
- Feed water temperature
- Feed water flow rate
- Steam flow rate
- SG NR level
- SG temperature

All the identified variables are in agreement with the analysis of the fault responses. The feed water flow rate has been successfully identified as important variables of concern.

The fault identification does not give immediate results to isolate faults. It only provides information about what variables significantly contribute to the residuals. This is true especially in the case that a feed back controller is involved since all the measurements within the control loop may be affected by a fault in the closed loop.

4.8.4 Fault isolation

The objective of fault isolation is to determine whether the fault is known in the fault dictionary and to determine which fault is the most likely one after the fault has been detected.

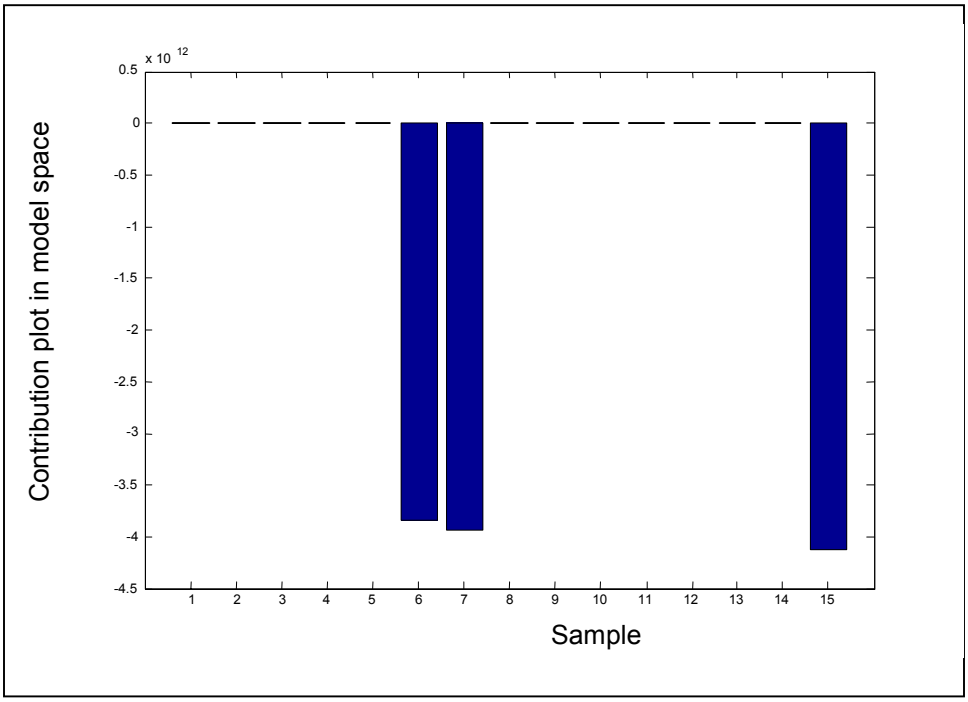

Figure 4.7. Contribution plot in the model space for feed water flow meter fault.

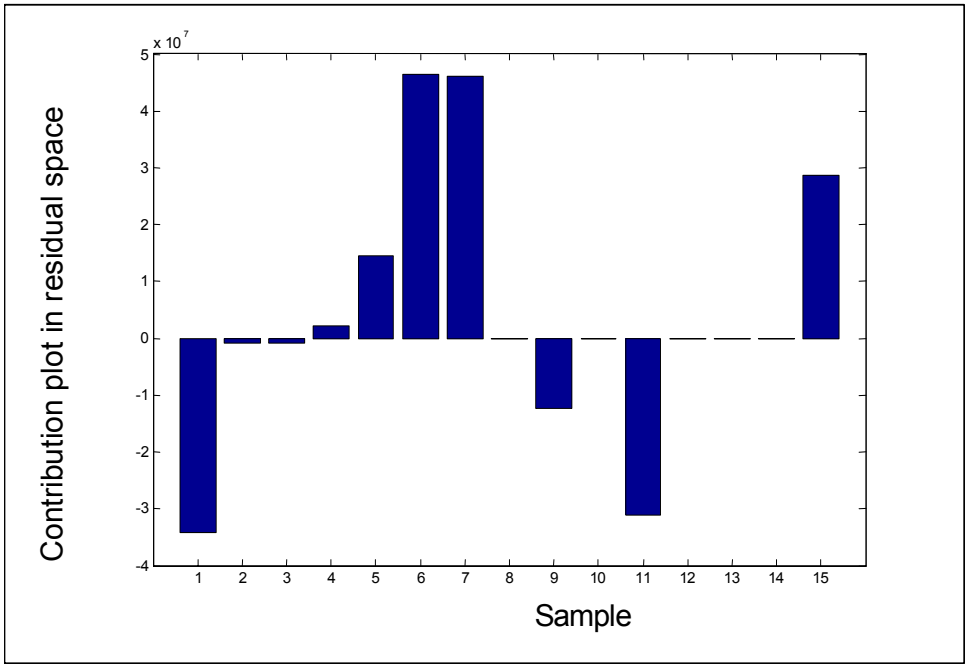

Figure 4.8. Contribution plot in the residual space for feed water flow meter fault.

The fault direction jointly defined in the model space and in the residual space has been used as fault signature for fault isolation. The fault direction is represented by the cosine angle of the fault directions between the unknown fault and all the 13 reference faults. These reference faults are numbered as follows:

- Feed water flow meter offset fault.
- Steam flow meter offset fault.
- Feed water flow meter offset fault and steam flow meter offset fault.
- Feed water flow meter offset fault and SG level sensor offset fault.
- Steam flow meter offset fault and SG level sensor offset fault.
- SG pressure sensor offset fault.
- Feed water flow meter offset fault and SG pressure sensor offset fault.
- Feed water flow meter offset fault and FCV offset sensor offset fault.
- Steam flow meter offset fault and FCV position offset fault.
- FCV valve position offset fault.
- SG pressure sensor offset fault and SG level sensor offset fault.
- SG level sensor offset fault.
- Steam flow meter offset fault and SG pressure sensor offset fault.

Figure 4.9 shows the fault direction in the model space and in the residual space for SG NR level sensor fault and feed water flow meter sensor fault without using SG wide range level. The feed water flow meter fault cannot be distinguished from feed water flow meter sensor fault plus SG level sensor fault. This is because the symptoms of the former fault envelope all those of the latter fault. Therefore, no additional information can be used to uniquely isolate SG NR level sensor fault.

Figure 4.10 shows the fault direction for the feed flow meter fault after the SG wide range level sensor has been used. After SG wide range level signal is used, feed water flow meter fault can then be isolated from feed water flow meter sensor fault plus SG level sensor fault. Therefore, in order to isolate all the selected thirteen faults

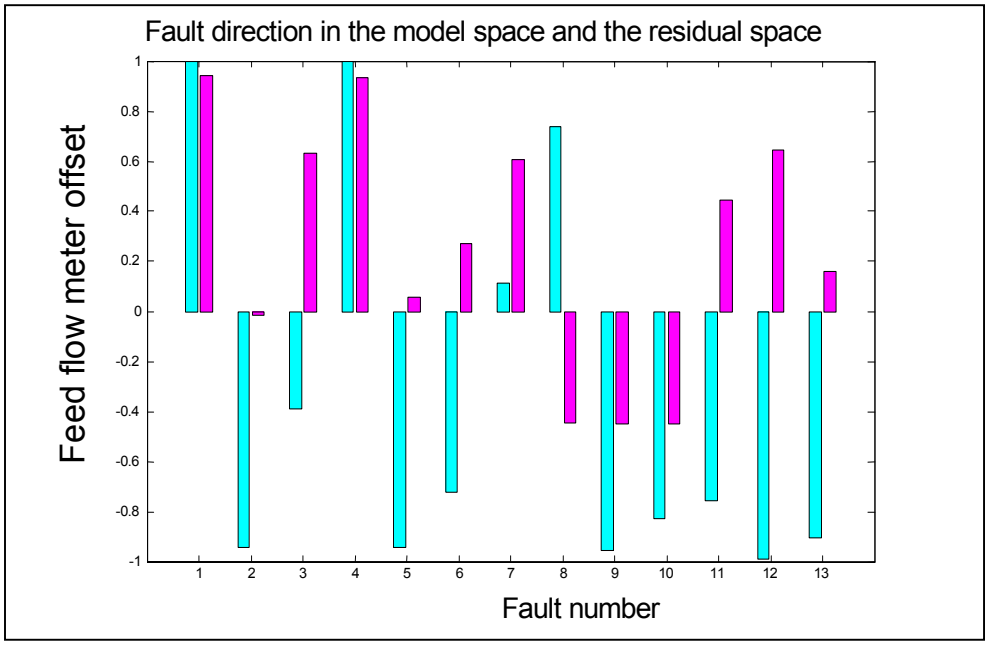

Figure 4.9. Fault direction for feed water flow meter offset fault and SG NR level sensor offset fault without using SG WR level signal.

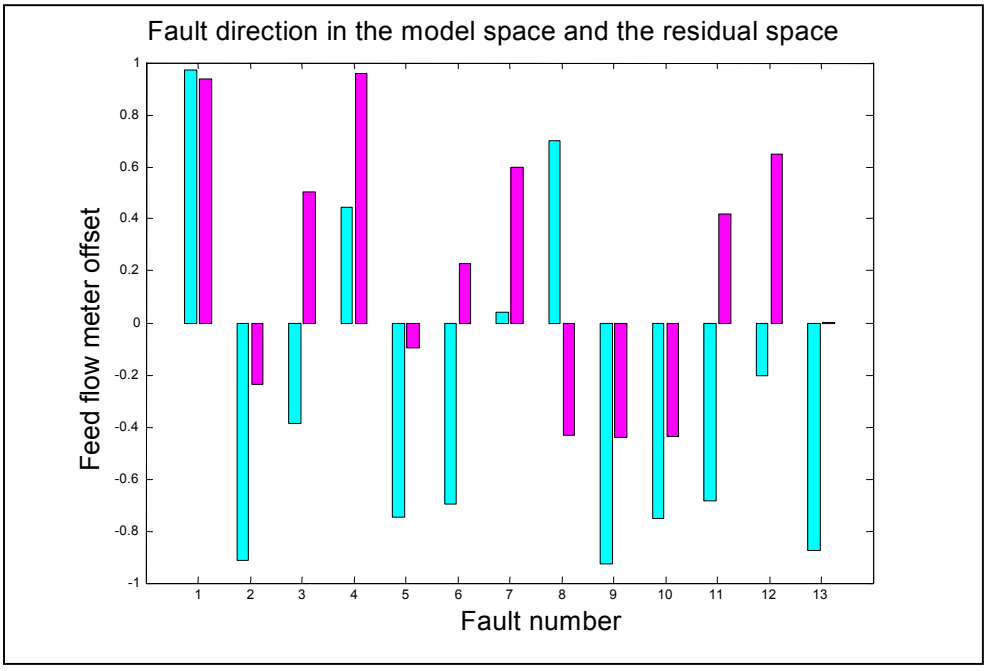

Figure 4.10. Fault direction for feed water flow meter offset fault.

including dual faults, measurement redundancy must be used to avoid the compensation effect of controller feedback.

Furthermore, an important criterion to judge if the designed FDI scheme is successful or not is to test the stability of the fault signatures in different fault magnitudes and under different initial operation conditions. For this reason, a set of data in fault magnitude of three percent under the initial power level at 80% full power, which are unknown to the fault dictionary, are generated to test the reliability of the designed FDI system.

Figure 4.11 to Figure 4.22 show the fault direction both in the model space and in the residual space for the defined 13 faults respectively in magnitude of three percent under the initial power level at 80% full power. As can be seen, the fault direction in either model space or residual space is sometimes not enough to isolate dual faults. For example, the fault direction in the model space for steam flow meter offset fault is similar to that for steam flow meter offset fault plus SG NR level sensor fault (See Figure 4.11). Nonetheless, the fault direction in the residual space for steam flow meter offset fault is quite different from steam flow meter offset fault plus SG NR level sensor fault. An opposite example is that the fault direction in the model space helps to isolate a fault. The fault direction in the residual space for steam flow meter offset is similar to that for steam flow meter offset fault plus FCV position fault (See also Figure 4.11).

Nevertheless, the fault direction in the model space for steam flow meter offset fault is quite different from steam flow meter offset fault plus FCV valve position offset fault. Therefore, when the joint fault direction is used, there is more possibility to isolate faults.

The cosine of the angle between the fault direction of an unknown fault and those of the reference faults can also be used as confidence level when a decision is to be made. Figure 4.15 shows that no significant margin exists to isolate a SG pressure sensor fault from a SG pressure sensor fault plus a steam flow meter offset fault. Figure 4.19 shows that there is no significant margin to isolate a FCV position fault from a steam flow meter

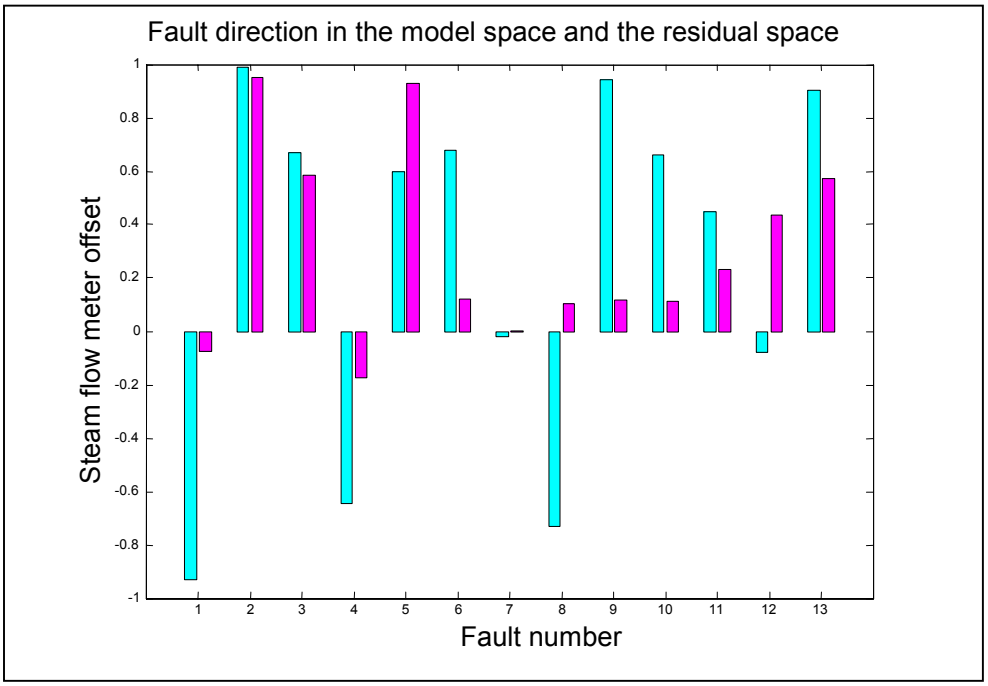

Figure 4.11. Fault direction for steam flow meter offset fault.

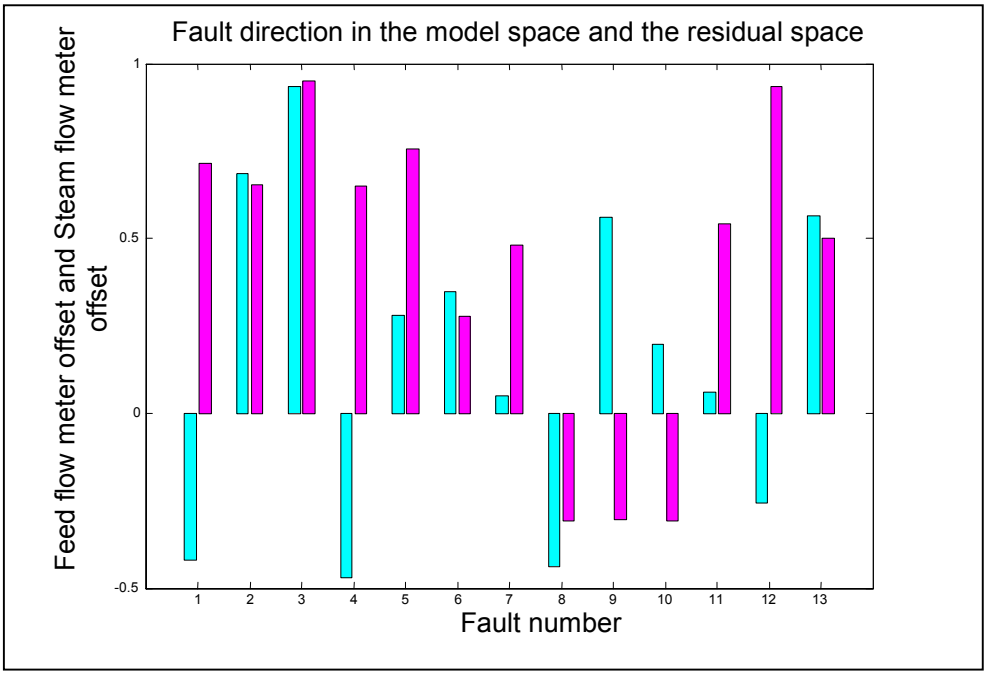

Figure 4.12. Fault direction for feed water flow meter offset fault and steam flow meter offset fault.

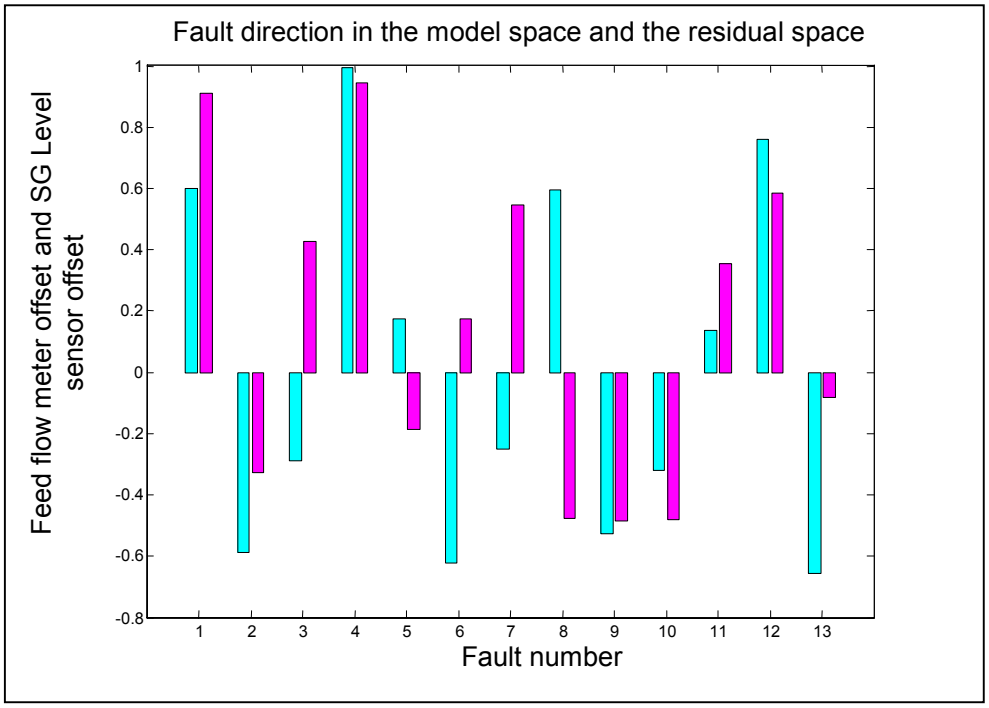

Figure 4.13. Fault direction for feed water flow meter offset fault and SG NR level sensor offset fault.

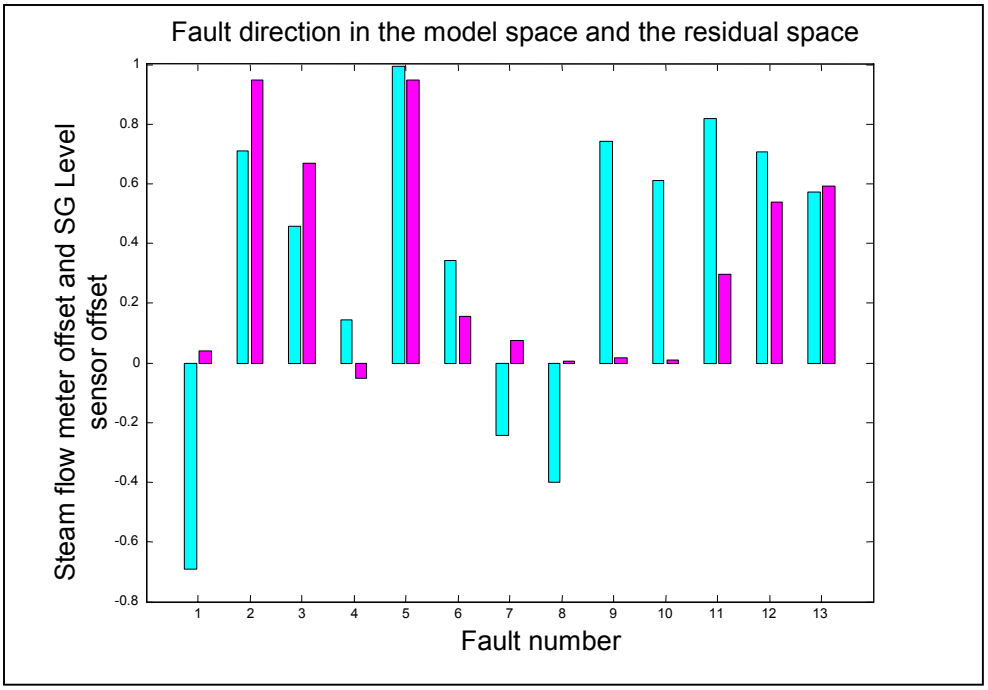

Figure 4.14. Fault direction for steam flow meter offset fault and SG NR level sensor offset fault.

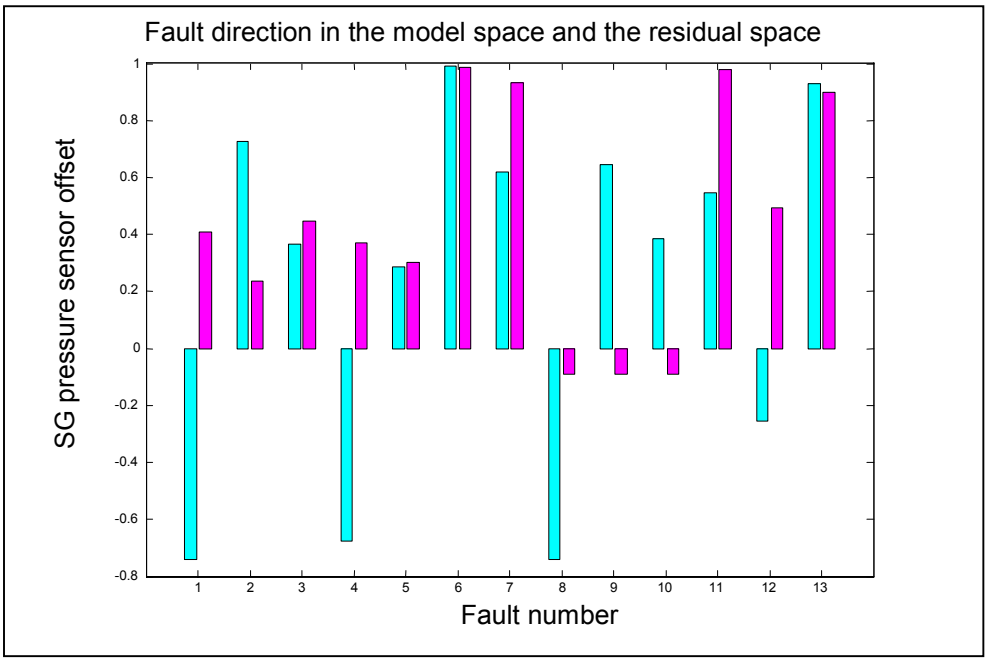

Figure 4.15. Fault direction for SG pressure sensor offset fault.

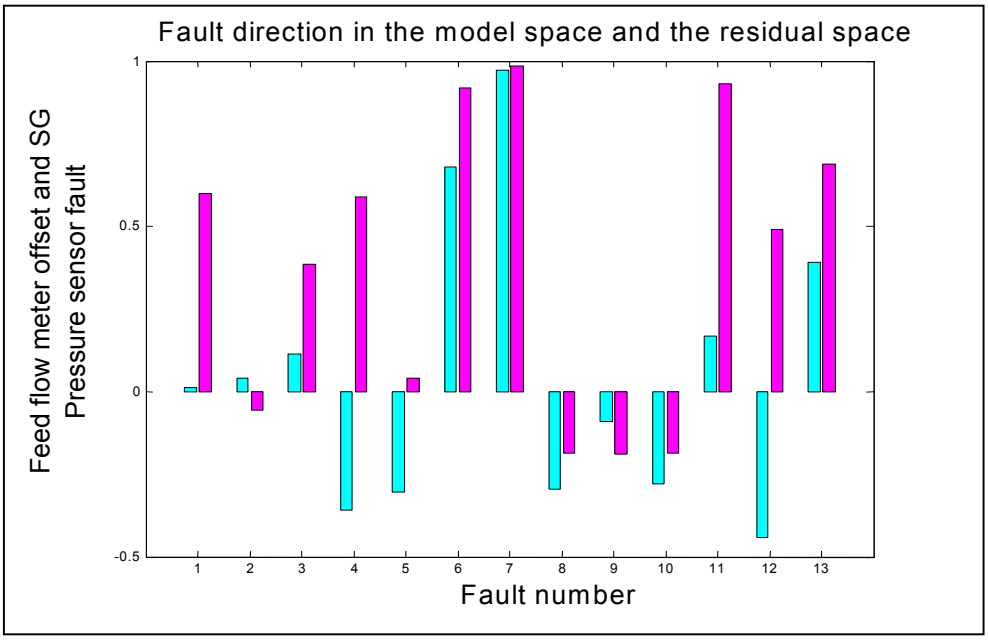

Figure 4.16. Fault direction for feed water flow meter offset fault and SG pressure sensor offset fault.

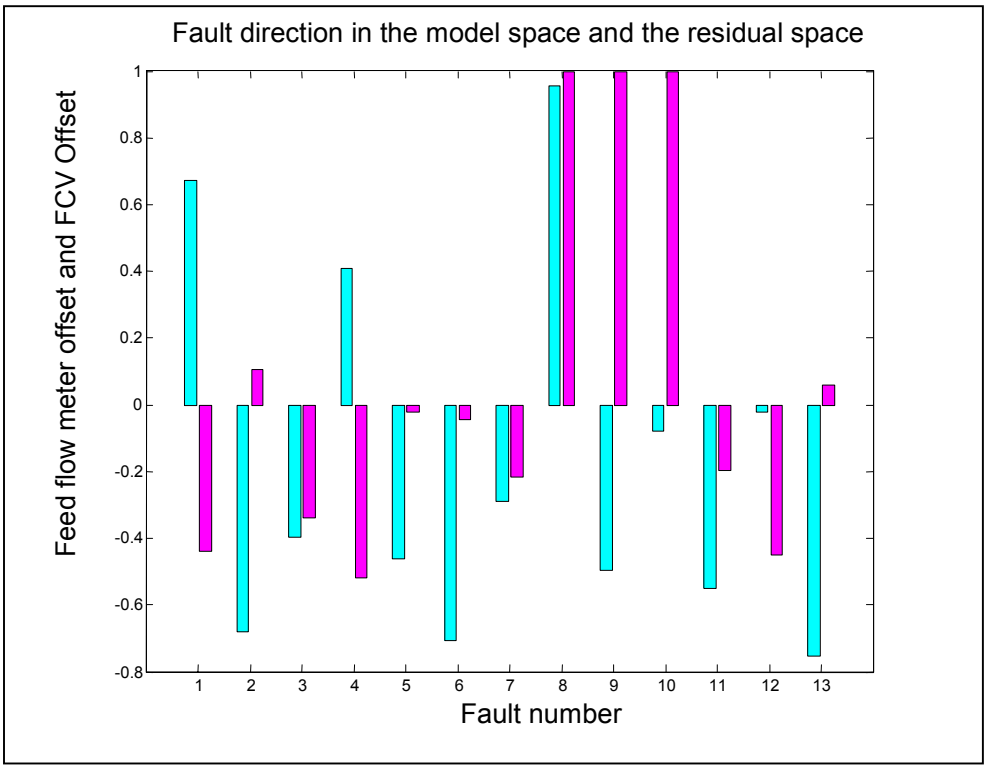

Figure 4.17. Fault direction for feed water flow meter offset fault and FCV position offset fault.

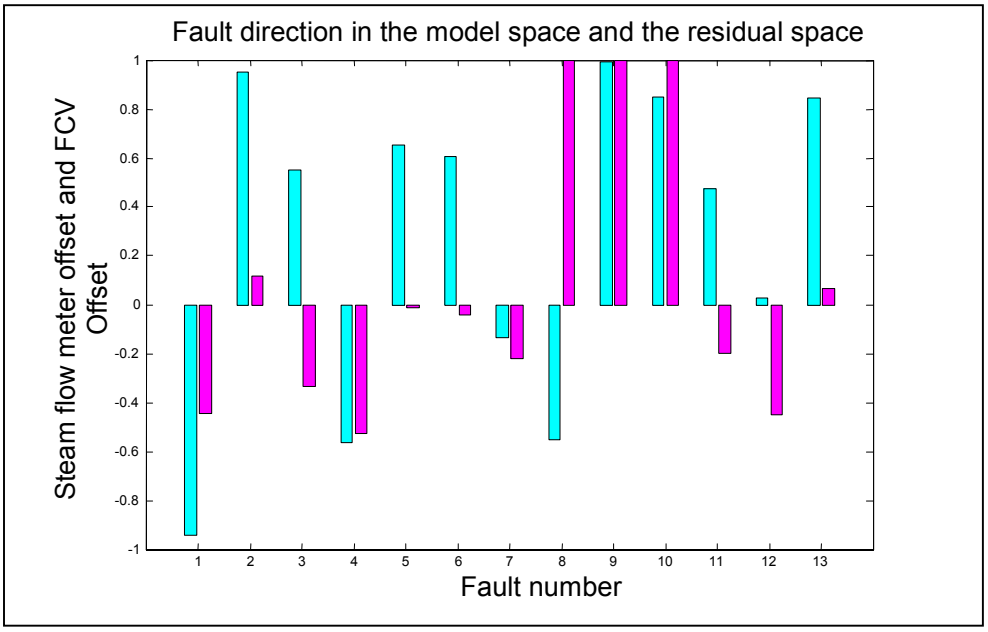

Figure 4.18. Fault direction for steam flow meter offset fault and FCV position offset fault.

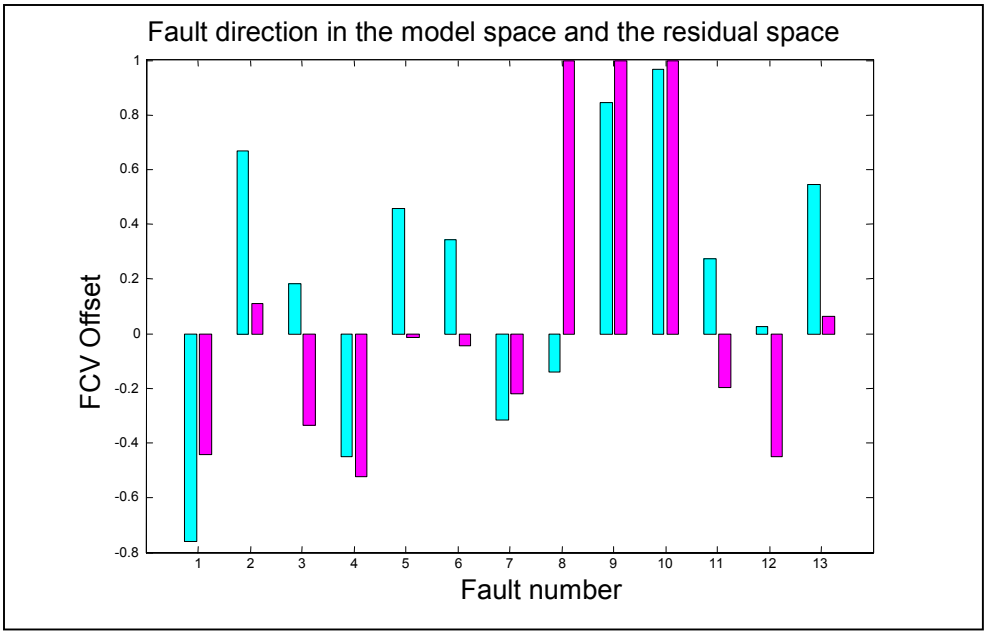

Figure 4.19. Fault direction for FCV position offset fault.

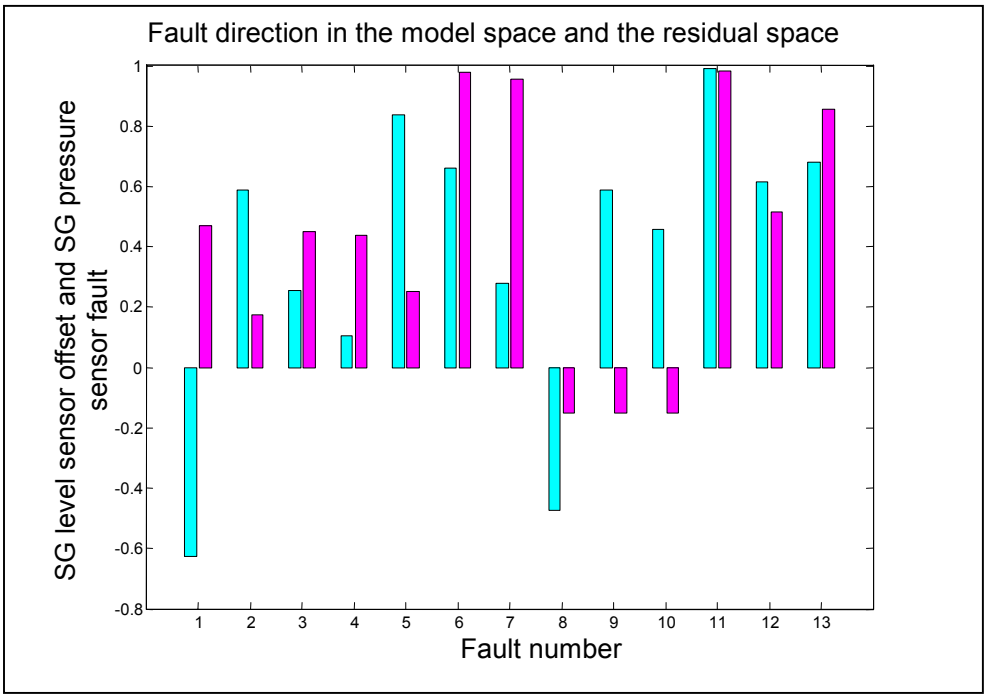

Figure 4.20. Fault direction for SG level sensor offset fault and SG pressure sensor offset fault.

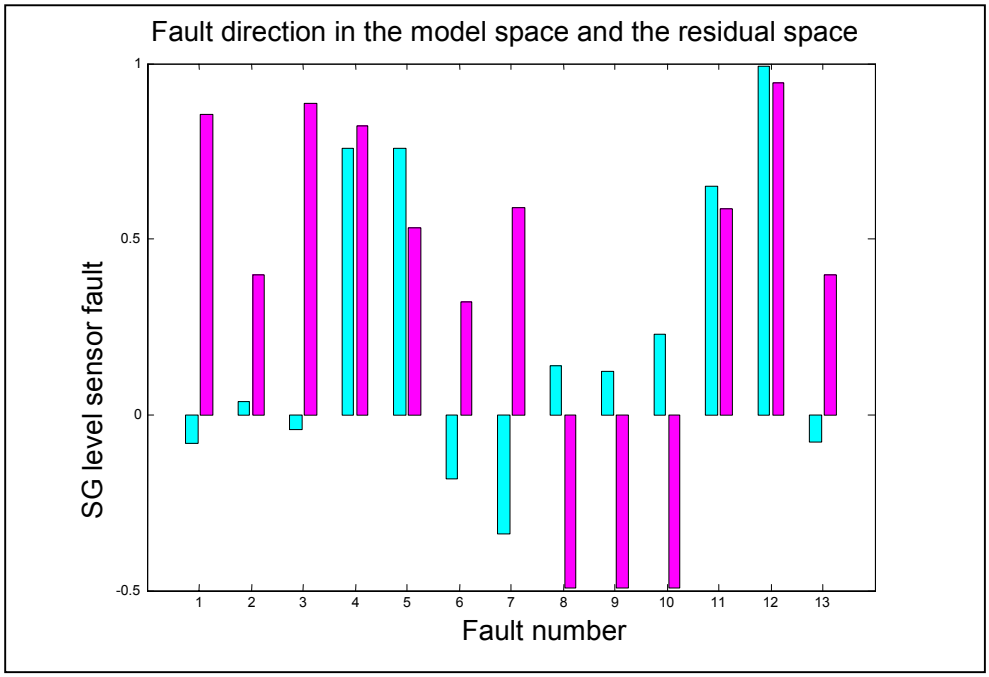

Figure 4.21. Fault direction for SG level sensor offset fault.

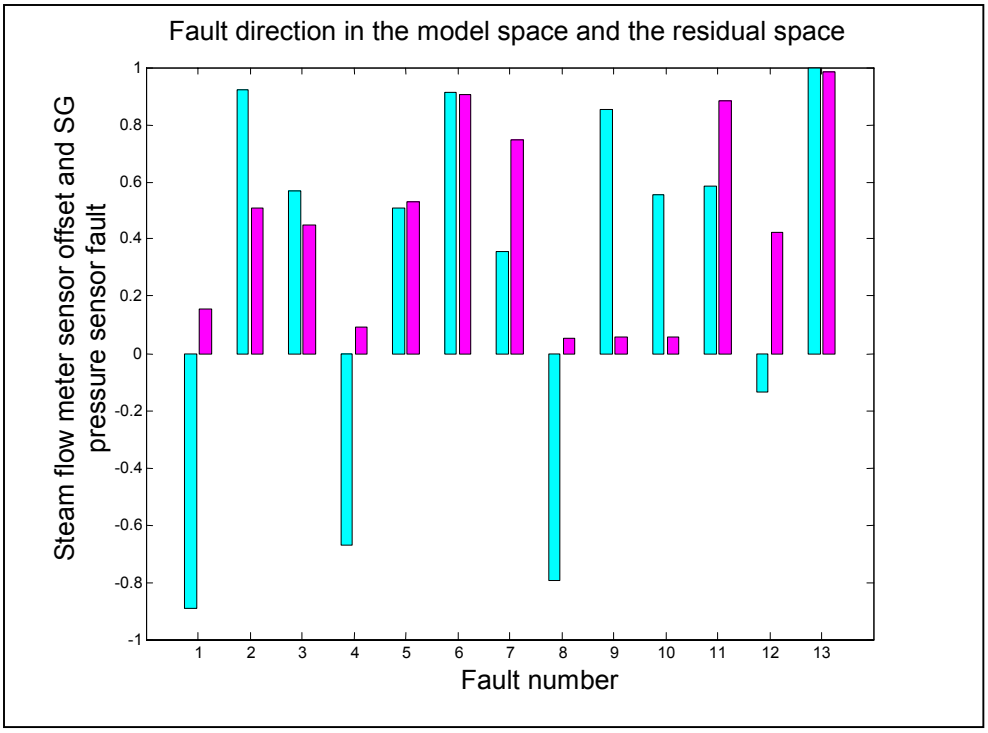

Figure 4.22. Fault direction for steam flow meter offset fault and SG pressure sensor offset fault.

offset fault plus FCV position fault. Therefore, when decisions are made, the confidence level to isolate these two faults should be taken into account.

4.9 Discussions

This chapter has presented the PCA approach to fault detection and isolation and its application to PWR steam generator system. The PCA approach is shown to be in agreement with parity space approach. The linear relationship among measured variables implying analytical redundancy can be consistently represented by the eigenvectors corresponding to the trivial components. The fault directions jointly defined both in the model space and in the residual space is a sensitive fault signature for fault isolation.

PCA approach needs the least information about a system when applied to FDI. It is simple to achieve on-line implementation. It provides an ideal tool to supervise plant status without too much investment. However, PCA approach has many inherent weaknesses. From the viewpoint of modeling, linear PCA is only applicable to a linear static system. It is difficult to develop a nonlinear PCA model for a dynamic system accurate enough to reveal the analytical redundancy inherent in a physical system. With regard to fault isolation, the fault characteristics must be defined from fault data for the enumerated faults. This exerts heavy burden on engineering application. In addition, the fault isolation is a process of classification, so the decision has poor interpretability.

Because PCA approach has inherent connection with parity space approach, it is very important to validate the constraint equations extracted from PCA modeling. If process variables are not appropriately chosen, some constraint equations necessary for fault isolation may not be obtained. If the number of principal components is chosen incorrectly, the residual direction cannot be used to characterize a fault.

Chapter 5

Adaptive Network Fuzzy Inference System for Fault Diagnosis

A PCA model cannot take advantage of the available system knowledge. For this reason, sometimes it is very difficult to build an appropriate model with low model uncertainty. Some of these difficulties are as follow:

- When a large number of variables are involved, it is hard to make sure that all the measurements are well excited in order to obtain a model with reliable generalization capability.
- When noisy data is involved, its effects on how the constraint equations are extracted from the noisy data are unknown to the analyst.
- When nonlinear behavior is involved, it is hard to have a tradeoff between choosing more principal components to have a better approximation and preclude the disturbance of noises.

In order to overcome these problems and keep the good feature of historical data based FDI approach, ANFIS is implemented to generate models for FDI. This method can take advantage of the available system knowledge and captures the most relevant relationships among measured variables to characterize a fault.

5.1 ANFIS Architecture

ANFIS is a fuzzy inference system implemented in the framework of artificial neural network (Jang, 1990). It is able to combine the reasoning capability of fuzzy logic and the learning capability of neural network. It is efficient in building a model with only a few inputs and one output. A fuzzy inference system implements inference procedure using fuzzy rules. A fuzzy rule can be expressed linguistically as follows:

$$\text{If x is A then y is B} \tag{5.1}$$

A fuzzy rule is analytically an implication relation *R* between its antecedent x and its consequent y, which can be expressed as:

$$R(x,y) = \int_{(x,y)} \mu(x,y)/(x,y) \tag{5.2}$$

where

$\mu(x,y)$ = membership function.

The implication relationship *R(x,y)* can also be explained as the membership function of a fuzzy set defined in a two dimensional universe of discourse *(x,y)*. It can be computed using implication operator ϕ as follows:

$$R(x,y) = \phi(\mu_A(x), \mu_B(y)) \tag{5.3}$$

The most commonly used implication operators are Larsen product and Madamni min.

If there are several input variables, it is necessary to have several antecedents connected with fuzzy operators. In general, a fuzzy inference system uses a set of fuzzy rules connected with connectives forming fuzzy algorithms.

Fuzzy inference of Generalized Modus Ponens is stated as the following problem:

If x is A then y is B

$$x = A' \Rightarrow y = B' \tag{5.4}$$

In the above problem statement, the known part is *R(x,y) and A',* the unknown part is *B'* associated with *A'.* This inference procedure is a fuzzy composition given by:

$$B' = A' \circ R(x,y) \tag{5.5}$$

The most commonly used fuzzy composition operators are Max-Min if Madamni Min implication relationship is used and Max-Product if Larsen Product implication relationship is used.

A fuzzy inference system has the following four components (Jang, 1994):

- *A rule base containing if-then rules.*
- *A database defining the membership functions used by the fuzzy rules.*
- *A decision-making unit performing inference operations on the rules.*
- *A unit to fuzzify the inputs and a unit to defuzzify the fuzzy outputs.*

Five steps need to be taken in a fuzzy inference system as follows:

- Fuzzify the inputs.
- Apply fuzzy operator.

- Apply implication Method.
- Aggregate all the outputs.
- Defuzzify the output.

The first step is to fuzzify the crisp inputs. In this step, the membership values of each input variable are computed. The second step is to apply fuzzy operators to compute the degree of the fulfillment (DOF) of the whole antecedent for each fuzzy rule by combining the membership values of all the fuzzy inputs. The result is the firing strength of its corresponding rule. In the third step, the membership of the consequent for each rule is computed based on the DOF of the antecedent for the corresponding rule by applying appropriate composition method. The fourth step is to aggregate the membership of the fuzzy outputs for all the rules. The final step is to defuzzify the output using methods such as centroid, maximum criterion, etc.

The simplest fuzzy inference model is of Sugeno type. It has the following form of fuzzy rules:

$$\text{If x is A and y is B then}$$
$$z=f(x) \tag{5.6}$$

where A and B are fuzzy sets and z=f(x) is a crisp function.

In this model, the consequent of each fuzzy rule is simply a crisp function rather than a fuzzy set. It can significantly simplify fuzzy reasoning. In general, aggregation and defuzzification will involve matrix operation in high dimensional space. However, for a Sugeno fuzzy model, only a simple arithmetic function is involved in computing the output of each rule. Hence, the aggregation and defuzzification can be combined into a weighted sum (Hines and Wrest, 1997).

A Sugeno fuzzy model evolves into its first order form if the defined function is of first order. Given that there are two inputs and one output, two of the fuzzy rules can be represented by:

$$\text{Rule 1: If x is A1 and y is B1 then } z=ax+by+c$$
$$\text{Rule 2: If x is A2 and y is B2 then } z=px+qy+r$$

The output *f* can then be obtained as the sum of the two crisp output *f1* and *f2* weighted by the firing strength ratio *w1* and *w2*. That is

$$f = \overline{w}_1 f_1 + \overline{w}_2 f_2 \tag{5.7}$$

When the fuzzy inference system is implemented using an adaptive network, the network system consists of layers of nodes capable of adapting parameters to map the desired input-output relation using fuzzy inference mechanism. For each node, there are several inputs and one output. The processing inside each node is nothing but performing some function computation. There are no weights designated to the connection between two nodes, but there is directional indication.

A classical ANFIS architecture, developed by Jang, 1994, is shown in Figure 5.1. Two fuzzy rules are involved in the four layers of network. The nodes in the first layer take crisp inputs and compute the DOF of the fuzzy sets (A1, A2, B1, B2). These fuzzy sets are parameterized fuzzy sets. Their membership functions can be adjusted easily by changing a set of parameters. The two nodes in the second layer correspond to the two fuzzy rules. All the nodes in this layer take two inputs to give an output, w_1 or w_2, representing the firing strength of each rule based on the product of the two membership values being involved. The third layer is responsible for calculating the relative importance of each rule ($\overline{w}_1$ and $\overline{w}_2$), the ratio of one rule's strength to the sum of the firing strengths of all the rules. Each node in the fourth layer contains a node function to calculate the consequent multiplied by the ratio calculated in the third layer. The output layer gives the final output by summing all its inputs.

5.2 ANFIS Learning Rule

Hybrid learning algorithm is developed for training ANFIS, which combines the gradient descent method and the least square method (Jang, 1994). The training process involves tuning parameters such that the desired input-output mapping is achieved. The tuned parameters are classified into two sets. One set describes the linear relationship between the inputs and the outputs, which contains the parameters of the crisp function to describe the consequent of each rule. The other set of parameters describes the non-linearity between the inputs and the outputs, which involves those parameters defining membership functions.

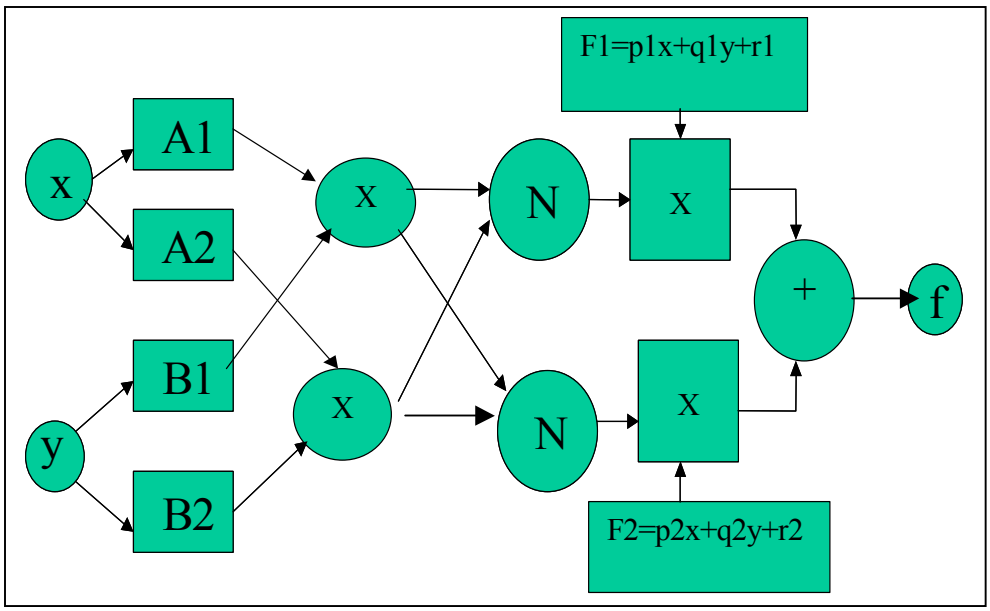

Figure 5.1. Schematic for Sugeno-type ANFIS System.

In the forward pass, the parameters describing the linear relationships are upgraded by sequential least square training. After the error is computed, the gradient descent training is used, which makes the error propagated from the output layer to the input layer. In this backward pass, the parameters describing the nonlinear relationship are upgraded. The training process does not end until the desired error goal is reached or the designated maximum number of epochs is exceeded.

For the ANFIS structure with two inputs and one output, the system output can be expressed as follows:

$$f = \overline{w}_1(A1, B1) * (p_1 x + q_1 y + r_1) + \overline{w}_2(A2, B2) * (p_2 x + q_2 y + r_2)$$

The parameter space S is partitioned into two subspaces S_1 and S_2 given by:

$$S = S_1 + S_2$$

where

$S_1 \supset (A1, B1, A2, B2)$

$S_2 \supset (p_1, q_1, r_1, p_2, q_2, r_2)$

During the forward pass with the fixed set S_1, the parameters in the subspace S_2 can be determined by least squares estimate as follows:

$$S_2 = (X'X)^{-1} X'Y$$

where

X = input data set.

Y = target output.

During the backward pass with the fixed set S_2, the parameters in the subspace S_1 can be determined by gradient descent method. For the output layer, the error rate is defined by:

$$\frac{\partial E}{\partial O^5} = -2(T - O^5)$$

where

$O^5 = \overline{w}_1 * (p_1 x + q_1 y + r_1) + \overline{w}_2 * (p_2 x + q_2 y + r_2)$

T = target output.

For the two nodes in the fourth layer, the error rate is defined by:

$$\frac{\partial E}{\partial O_1^4} = \frac{\partial E}{\partial O^5}\frac{\partial O^5}{\partial O_1^4}$$
$$\frac{\partial E}{\partial O_2^4} = \frac{\partial E}{\partial O^5}\frac{\partial O^5}{\partial O_2^4}$$

where

$O^5 = O_1^4 + O_2^4$ =the system output.

O_i^k =the output of the ith node for the kth layer.

For the two nodes in the third layer, the error rate is defined by:

$$\frac{\partial E}{\partial O_1^3} = \frac{\partial E}{\partial O^5}\left(\frac{\partial O^5}{\partial O_1^4}\frac{\partial O_1^4}{O_1^3} + \frac{\partial O^5}{\partial O_2^4}\frac{\partial O_2^4}{O_1^3}\right)$$
$$\frac{\partial E}{\partial O_2^3} = \frac{\partial E}{\partial O^5}\left(\frac{\partial O^5}{\partial O_1^4}\frac{\partial O_1^4}{O_2^3} + \frac{\partial O^5}{\partial O_2^4}\frac{\partial O_2^4}{O_2^3}\right)$$

where

$O_1^4 = O_1^3 f_1$
$O_2^4 = O_2^3 f_2$

For the two nodes in the second layer, the error rate is defined by:

$$\frac{\partial E}{\partial O_1^2} = \frac{\partial E}{\partial O^5}\left(\frac{\partial O^5}{\partial O_1^4}\frac{\partial O_1^4}{O_1^3} + \frac{\partial O^5}{\partial O_2^4}\frac{\partial O_2^4}{O_1^3}\right)\frac{\partial O_1^3}{\partial O_1^2} + \frac{\partial E}{\partial O^5}\left(\frac{\partial O^5}{\partial O_1^4}\frac{\partial O_1^4}{O_2^3} + \frac{\partial O^5}{\partial O_2^4}\frac{\partial O_2^4}{O_2^3}\right)\frac{\partial O_2^3}{\partial O_1^2}$$
$$\frac{\partial E}{\partial O_2^2} = \frac{\partial E}{\partial O^5}\left(\frac{\partial O^5}{\partial O_1^4}\frac{\partial O_1^4}{O_1^3} + \frac{\partial O^5}{\partial O_2^4}\frac{\partial O_2^4}{O_1^3}\right)\frac{\partial O_1^3}{\partial O_2^2} + \frac{\partial E}{\partial O^5}\left(\frac{\partial O^5}{\partial O_1^4}\frac{\partial O_1^4}{O_2^3} + \frac{\partial O^5}{\partial O_2^4}\frac{\partial O_2^4}{O_2^3}\right)\frac{\partial O_2^3}{\partial O_2^2}$$

where

$$O_1^3 = \frac{O_1^2}{O_1^2 + O_2^2}$$
$$O_2^3 = \frac{O_2^2}{O_1^2 + O_2^2}$$

For the jth node in the first layer, the error rate is defined by:

$$\frac{\partial E}{\partial O_j^1} = \frac{\partial E}{\partial O_1^2}\frac{\partial O_1^2}{\partial O_j^1} + \frac{\partial E}{\partial O_1^2}\frac{\partial O_2^2}{\partial O_j^1}$$

where

$O_1^2 = O_1^1 O_3^1$

$O_2^2 = O_2^1 O_4^1$

The derivative of the output error with respect to the parameters used to define the membership functions can be determined by:

$$\frac{\partial E}{\partial S_{1,i}} = \sum_{j=1}^{M} \frac{\partial E}{\partial O_j^1} \frac{\partial O_j^1}{\partial S_{1,i}}$$

where

$S_{1,j}$ =the jth parameter in the space of S_1.

M =the number of fuzzy sets used to define the fuzzy rules.

If Gaussian membership function is used for the jth fuzzy set in the first layer, which is given by:

$$O_j^1 = \mu_j(x) = \exp(-(\frac{x - c_j}{a_j})^2)$$

then

$$\frac{\partial E}{\partial c_j} = \frac{\partial E}{\partial O_j^1} \frac{\partial \mu_j(x)}{\partial c_j}$$

$$\frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial O_j^1} \frac{\partial \mu_j(x)}{\partial a_j}$$

The update of the parameters in the space of S_1 is determined by:

$$\Delta S_{1,j} = -\eta \frac{\partial E}{\partial S_{1,j}}$$

where η is the learning rate.

The hybrid learning algorithm is much faster than gradient descent method only or gradient descent and one pass of least squares method (Jang, 1993). If some membership functions or some rule functions are determined from expert knowledge, the learning algorithms can be easily adapted to develop some hybrid models.

5.3 Structured Residual Design Approach

If structured residual design approach is used for fault isolation, the residual vector is represented by the bit numbers for a set of models. The bit number 1 indicates that the model has a significant residual while the bit number 0 indicates the model has insignificant residual. If there are only single faults in the designed FDI system, it is possible to achieve strong fault isolation if the model structures are carefully chosen. The most straightforward method to obtain residuals is based on the natural redundancy in a process. Table 5.1 shows the residual structure of four models for four faults. The residual pattern for fault 1 is [1,1,0,0]; the residual pattern for fault 2 is [1,0,0,0]; the residual pattern for fault 3 is [0,0,1,1]; and the residual pattern for fault 4 is [0,0,1,0]. If the bit number of model 3 for fault 2 degenerate, fault 2 will be misdiagnosed as fault 1. If the bit number of model 4 for fault 3 degenerate, fault 4 will be misdiagnosed as fault 1. This kind of residual structure can only result in weak isolation between faults. In order to achieve strong fault isolation, which means a fault will not be misdiagnosed as another fault even if one bit number has degenerated, it is necessary to transform the residual vectors into a structured form shown in Table 5.2. For a linear system, the structured residual can be achieved by a linear transformation on the original residuals. However, for a nonlinear system, it is quite difficult to derive new dependent equations by algebraic combinations of the previous equations in order to obtain the desired residual structure (Garcia, et al, 2000).

5.4 Application to Nuclear Plant SG System

Once the possible faults are enumerated based on engineering judgments structured residual design approach with ANFIS models can be implemented for fault diagnosis for nuclear SG system. The study shows different residual structures are required to deal with single faults and dual faults.

Table 5.1. Structured residual design for weak fault isolation

Model	Fault 1	Fault 2	Fault 3	Fault 4
Model 1	1	1	0	0
Model 2	1	0	0	0
Model 3	0	0	1	1
Model 4	0	0	1	0

Table 5.2. Structured residual design for strong fault isolation

Model	Fault 1	Fault 2	Fault 3	Fault 4
Model 1	1	1	0	1
Model 2	1	0	0	0
Model 3	0	0	1	1
Model 4	0	1	1	0

For PWR SG system, the model structures derived from the physical analysis are as follows:

FCV flow rate (t)=f (FCV valve position (t), SG pressure (t))

FCV valve position (t+1)=f (controller output (t), FCV valve position (t))

SG pressure (t)=f(SG temperature(t))

Steam flow rate(t+1)=f(feed water temperature(t), SG pressure(t), hot leg temperature(t), cold leg temperature(t))

SG level (t+1)=f(SG level(t),Feed water flow rate(t), steam flow rate(t),SG pressure(t))

Table 5.3 shows the residual patterns for the 13 faults with the above model structure. In the table, the bit number 0 indicates that the model to predict the specific variable will not generate significant residual while the bit number 1 indicates that the residual is significant. The threshold to distinguish the significance is determined by the model accuracy and the level of plant disturbance. As can be seen from the table, the residual patterns can be directly used to achieve strong fault isolation for the three single faults (feed water flow meter offset, steam flow meter offset and FCV position offset). However, the SG level sensor fault cannot even be detected. In this case, the residuals refer to the values after the new steady state has been reached. Due to the compensation effects of the SG level controller, the relationship among the feed water flow rate, the steam flow rate, the SG pressure, the FCV position and the SG level are always attempted not to change. Therefore, it is usually very difficult to detect a minor fault of the steam generator water level sensor fault based on a steady state model. The table also shows that some residuals become unstable due to fault competition with different fault magnitudes when dual faults are involved. The unstable residuals of the SG level model, denoted by the sign "?" in the table, correspond to the following combination of faults:

Table 5.3. Consistency checking using natural redundant relations

Functional Model / Faults	FCV flow rate	Steam flow rate	SG pressure	FCV valve position	SG level
Feed flow meter offset	1	0	0	0	1
Steam flow meter offset	0	1	0	0	1
Steam flow meter and feed flow meter offset	1	0	0	0	?
SG NR level sensor offset	0	0	0	0	0
Feed flow meter offset and SG level sensor offset	1	1	0	0	1
Steam flow meter offset and SG level sensor offset	0	1	0	0	1
SG pressure sensor offset	1	1	1	0	1
Feed flow meter offset and SG pressure sensor offset	1	1	1	0	1
Steam flow meter offset and SG pressure sensor offset	1	1	1	0	1
SG level sensor offset and SG pressure sensor offset	1	1	1	0	?
Feed water flow meter offset and FCV position offset	1	0	0	1	?
Steam flow meter offset and FCV position offset	0	1	0	1	1
FCV position offset	0	0	0	1	1

- Steam flow meter offset plus feed flow meter offset.
- SG pressure sensor offset plus SG level sensor offset.
- FCV valve position offset plus feed water flow meter offset.

Although the residual structure can be used directly to isolate the three single faults, it is not sufficient to isolate dual faults. These dual faults are:

- Feed flow meter offset plus SG level sensor offset cannot be separated from feed flow meter offset.
- Steam flow meter offset plus SG level sensor offset cannot be separated from steam flow meter offset.
- SG pressure sensor fault plus Feed flow meter offset cannot be separated from steam SG pressure sensor fault.
- SG pressure sensor fault plus SG level sensor offset and SG pressure sensor fault plus SG steam flow meter offset.

In conclusion, although a set of models derived from physical analysis may result in different residual patterns for fault isolation, they are usually not effective to deal with dual faults. When used for dual fault isolation, some dual faults may result in the same residual pattern as their element faults. In addition, the residuals of some models may become unstable for dual faults with different fault magnitudes.

5.4.1 Dedicated residual design for dual faults

Because dual faults usually cannot be strongly isolated from its element faults, dedicated residual structure is designed to isolate dual faults. Dedicated residual structure has the following two properties:

- Each residual is only sensitive to one fault and insensitive to all the other faults.
- Different faults result in different types of residual patterns.

If the possible faults are known, dedicated residual structure can be obtained through appropriately selecting the model structures to generate residuals. The alternative models can be derived based on:

- Natural redundancy

For instance, for a saturated system, the temperature and the pressure has one to one correspondence. Any model involving either variable can be substituted by the other variable.

- Derived redundancy

For instance, if the flow rate is determined by the system pressure and the valve position for a system, any model involving the flow rate can be substituted by the system pressure and the valve position.

- Measurement redundancy

The measurement redundancy is the most primitive one. For the SG system, the SG narrow range level measurement involved in any model can be substituted by the SG wide range level measurements.

In order to isolate the specified 13 faults for the nuclear SG system, the models with dedicated residual structure are defined as follow:

FCV flow rate (t)=f (FCV valve position (t), SG temperature (t))

FCV valve position (t+1)=f (controller output (t), FCV valve position (t))

SG pressure (t)=f(SG temperature(t))

Steam flow rate(t+1)=f(feed water temperature(t), SG temperature(t), hot leg temperature(t), cold leg temperature(t))

SG NR level (t)=f(SG WR level(t),Feed water temperature(t))

In order to isolate dual faults involving the controlled variable from their element faults in a closed control loop, the measurement redundancy has to be used. For the nuclear SG level system, SG WR level sensor has to be used to isolate SG NR level sensor fault from SG NR level sensor fault plus another fault in the control loop.

5.4.2 ANFIS modeling for SG system

The ANFIS modeling has been used to construct the five models for the system during normal operation. Before the ANFIS models are constructed, the input variables need to be appropriately scaled. The purpose to scale the inputs is to give equal importance to all the inputs in the case that the input variables are in different units.

Feed water flow rate model is shown as an example to train the ANFIS model. The network uses two bell-shape membership functions for either input. Two rules have been selected to map the input and output relationship.

Figure 5.2 shows the membership functions for the ANFIS model to predict the FCV valve flow rate before and after training. It can be seen that the training has changed the shape of the membership function for the first input (FCV valve position) significantly. In general, this change reflects the degree of nonlinearly contained in the mapping between the input and the output. After three epochs, the ANFIS model has been trained to reach a training error less than 0.5 %.

5.4.3 Model testing and validation

The residuals generated by some models can be immediately used for fault detection. If the sum square residuals of all the models are greater than a specified threshold, a fault is assumed to have happened.

In order to reduce the false alarm rates for fault detection, these models must be able to correctly characterize the system behavior under all the fault free conditions. However, if the models are fully static, any changes in the plant status or even plant disturbance will cause false alarms because the dynamic behavior of the system is unknown to the system. For this reason, most FDI systems need to use dynamic models. The dynamic models are able to simulate the normal transients such as a normal power transient.

In order to test the performance in characterizing the dynamic behavior, a power transient from 100% power to 90% power is simulated using the ANFIS models built in

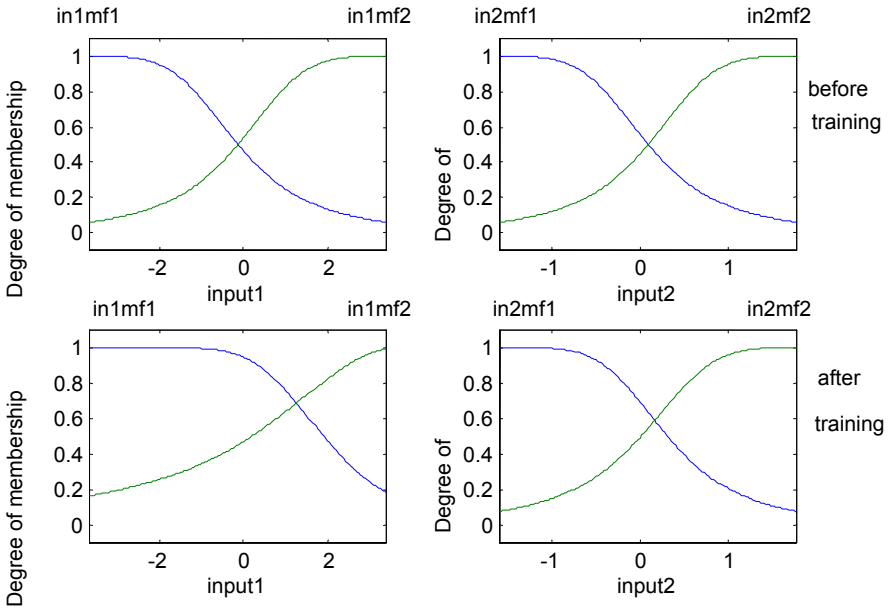

Figure 5.2. Membership for the two inputs to predict feed water flow rate using ANFIS.

the last section. Figure 5.3 to Figure 5.6 show the comparison between the estimation from the ANFIS models and the actual values obtained by the SimPWR simulation code for the following variables:

- SG narrow range level.
- Steam flow rate.
- FCV flow rate.
- FCV valve position.

From these figures, it can be seen that the ANFIS models can correctly simulate the transient process with low errors. When a large complex system with strong interaction is involved, it is usually very challenging to build perfect data driven models. For instance, it would be difficult to build a data driven model for a fast transient due to the fast interaction among systems. For fast transients, it will involve much more complicated model structure and it is harder to collect data to sufficiently excite all the related subsystems.

However, from FDI point of view, the slight errors of these models will not impose a serious problem. First, different thresholds can be set to the residuals for different models depending on the accuracy of the models. Secondly, fast transient is not of major interest for an incipient fault detection and isolation system. A fast power transient is usually under cautious supervision of operators, so operators can easily switch off the FDI system if the expected transient is any faster than the designed level. In general, the ANFIS models should give correct estimation if the relationship between the input variables and the output variable does not change. However, if a fault happens, some input variables may be outside their training range and the model may perform unreliable extrapolation. Hence, the residual of the model may exceed the specified threshold. In order to avoid this problem, it is necessary to evaluate whether the models are excited in all fault cases.

An example is given to show the importance of model testing for SG NR level model. If the data are collected only from 20 % to 100% power ranges, the model will

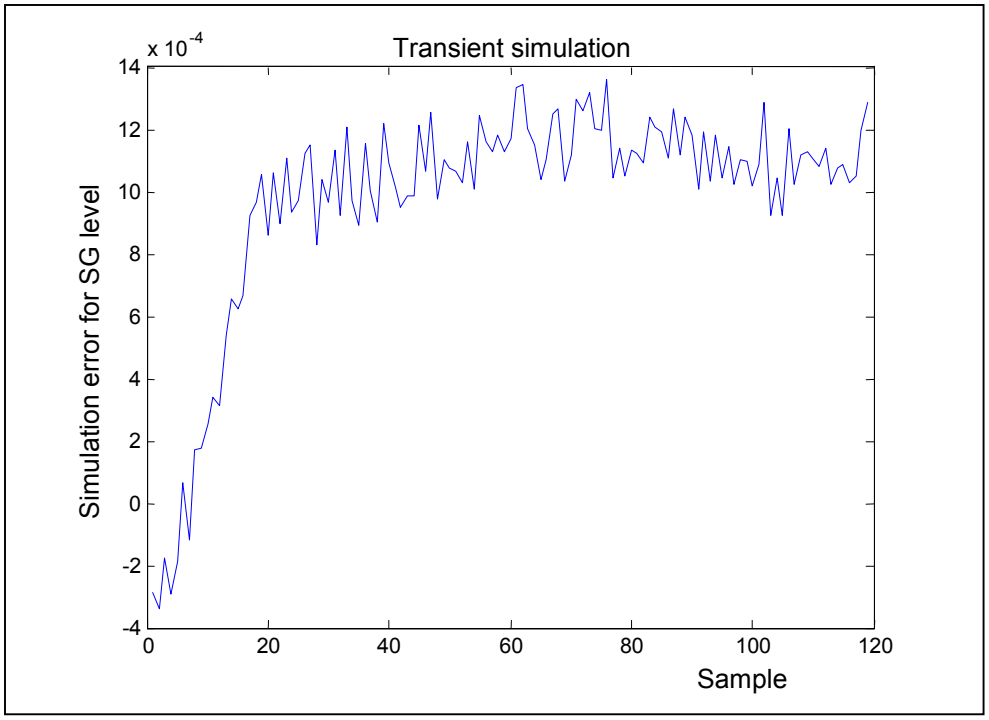

Figure 5.3. Transient simulation of SG NR level using ANFIS model.

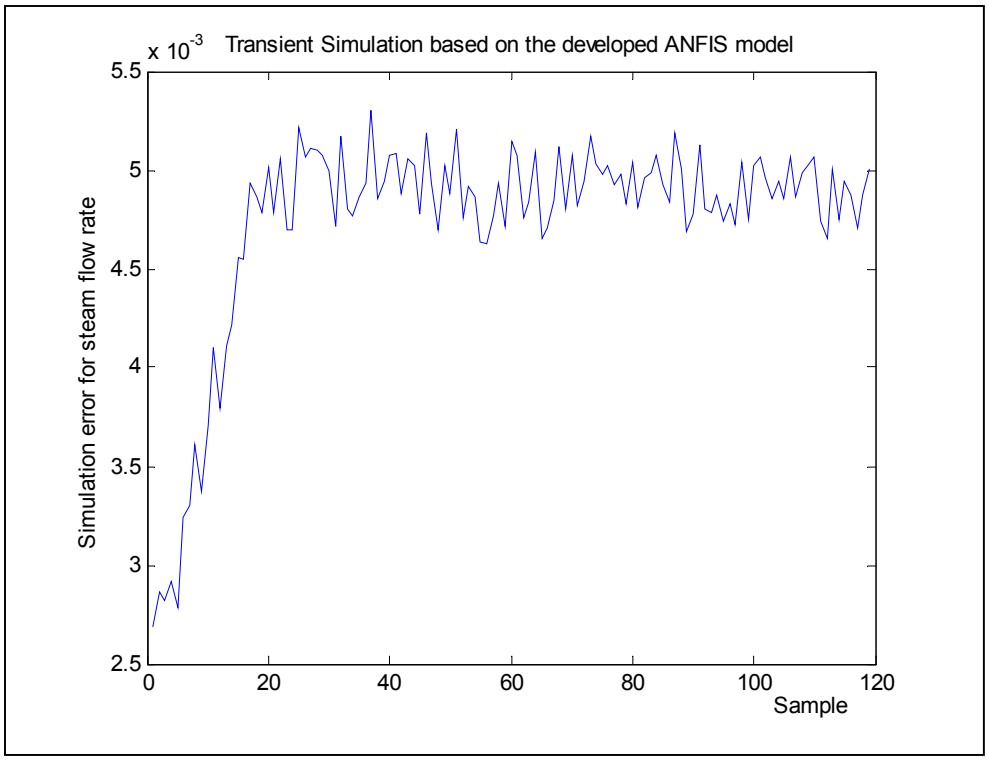

Figure 5.4. Transient simulation of steam flow rate using ANFIS model.

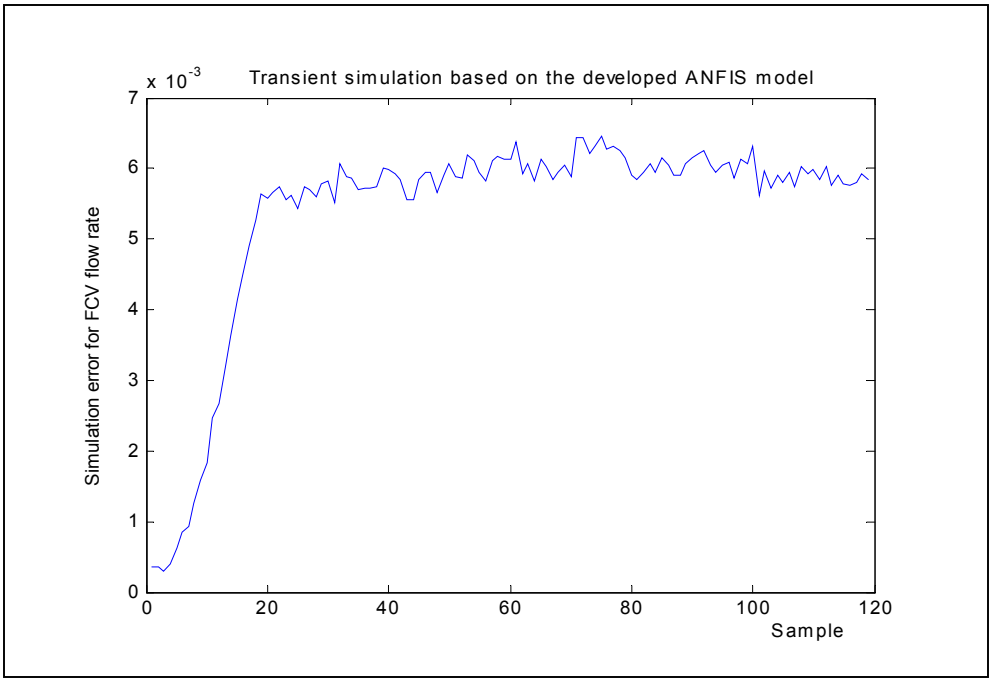

Figure 5.5. Transient simulation of FCV flow rate using ANFIS model.

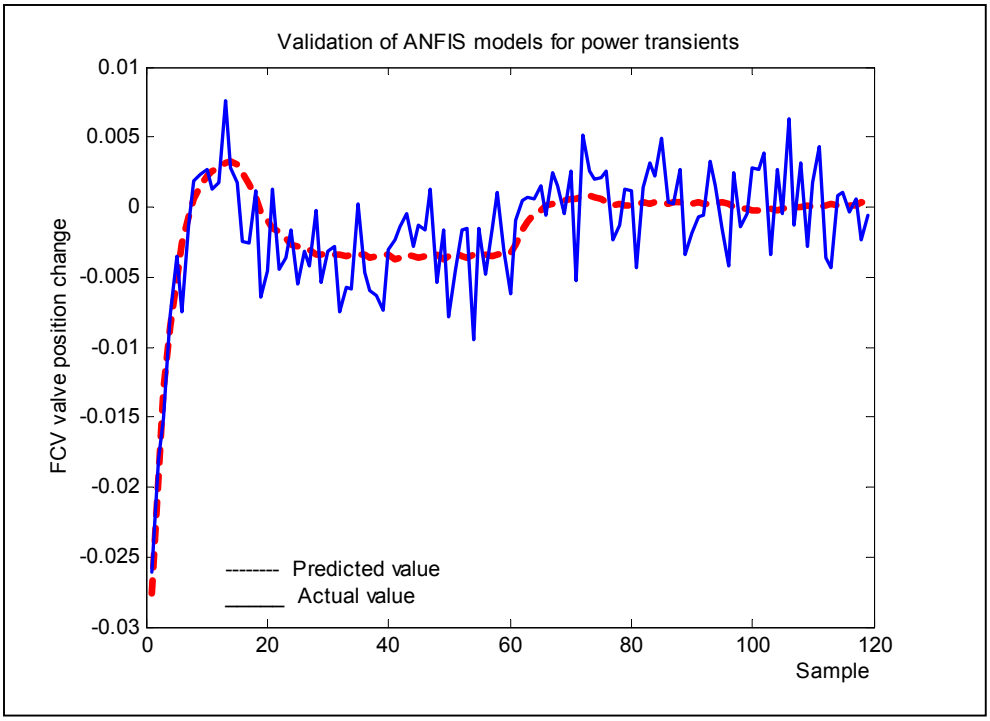

Figure 5.6. Transient simulation of FCV position using ANFIS model.

generate unstable residual patterns for different fault magnitudes. Figure 5.7 shows that when the SG steam flow meter and the SG pressure sensor have less than 2% offset faults, the residual is less than 0.5%. However, when the fault magnitude is 3%, the residuals become unstable. To investigate the causes, the training range of the inputs is examined. The minimum values of the feed water temperature and the SG WR SG level are 313.94 *F* and 76.269% respectively. The maximum values of the feed water temperature and the SG WR SG level are 438.4 *F* and 85.599% respectively. However, for the SG steam flow meter and SG pressure sensor offset faults with 3% fault magnitude, the minimum values of the feed water temperature and the SG WR SG level are *440.42 F* and 83.0% respectively and the maximal values are 440.5 *F* and 86.167% respectively. Apparently, the fault data have exceeded the training range, so the ANFIS model is not able to correctly compute the residual of the SG narrow range level.

After more data is collected to cover the entire range for the faults, the residuals exhibit consistent behavior. Figure 5.8 shows that the residuals of SG NR level models are within 1% for the dual faults (the SG steam flow meter and the SG pressure sensor offset fault) when the SG NR level sensors are healthy.

5.4.4 FDI Results

Table 5.4 shows the dedicated residual structure to isolate the defined 13 single and dual faults. As can be seen, each ANFIS model is dedicated to isolate one fault. For dual faults, the corresponding two models dedicated to the two element faults will generate significant residuals, which provides the full possibility to isolate them.

Figure 5.9 to Figure 5.16 show the residual structures for different fault magnitudes. In these figures, the 13 fault classes correspond to the following faults:

- Fault class 1= Feed water flow meter offset fault.
- Fault class 2= Steam flow meter offset fault.
- Fault class 3= Feed water flow meter offset fault and steam flow meter offset fault.
- Fault class 4= SG level sensor offset fault.
- Fault class 5= Feed water flow meter offset fault and SG level sensor offset fault.

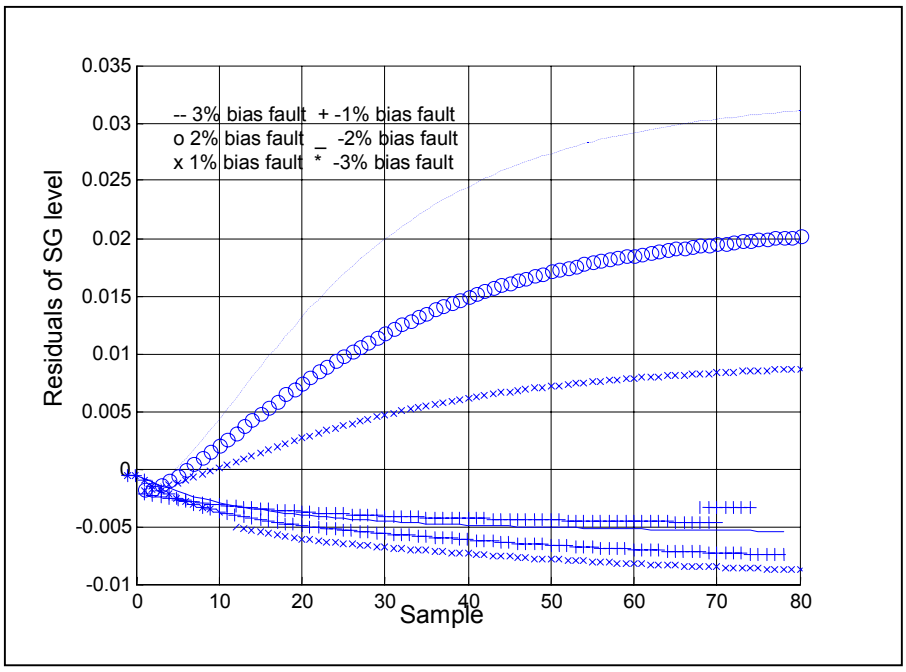

Figure 5.7. Unstable residual of SG level for SG pressure and steam flow meter fault.

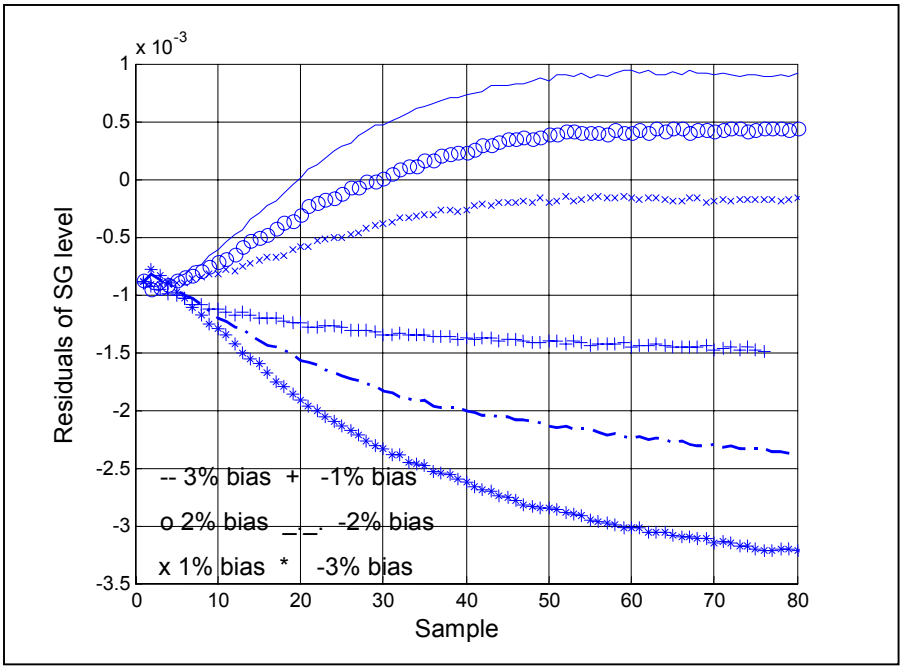

Figure 5.8. Stable residual of SG level for SG pressure and steam flow meter fault.

Table 5.4. Dedicated residual structure for SG system

Functional Model / Faults	Feed water flow rate	Steam flow rate	SG pressure	FCV valve position	SG level
Feed flow meter offset	1	0	0	0	0
Steam flow meter offset	0	1	0	0	0
Steam flow meter and feed flow meter offset	1	1	0	0	0
SG NR level sensor offset	0	0	0	0	1
Feed flow meter offset and SG level sensor offset	1	0	0	0	1
Steam flow meter offset and SG level sensor offset	0	1	0	0	1
SG pressure sensor offset	0	0	1	0	0
Feed flow meter offset and SG pressure sensor offset	1	0	1	0	0
Steam flow meter offset and SG pressure sensor offset	0	1	1	0	0
SG level sensor offset and SG pressure sensor offset	0	0	1	0	1
Feed water flow meter offset and FCV position offset	1	0	0	1	0
Steam flow meter offset and FCV position offset	0	1	0	1	0
FCV position offset	0	0	0	1	0

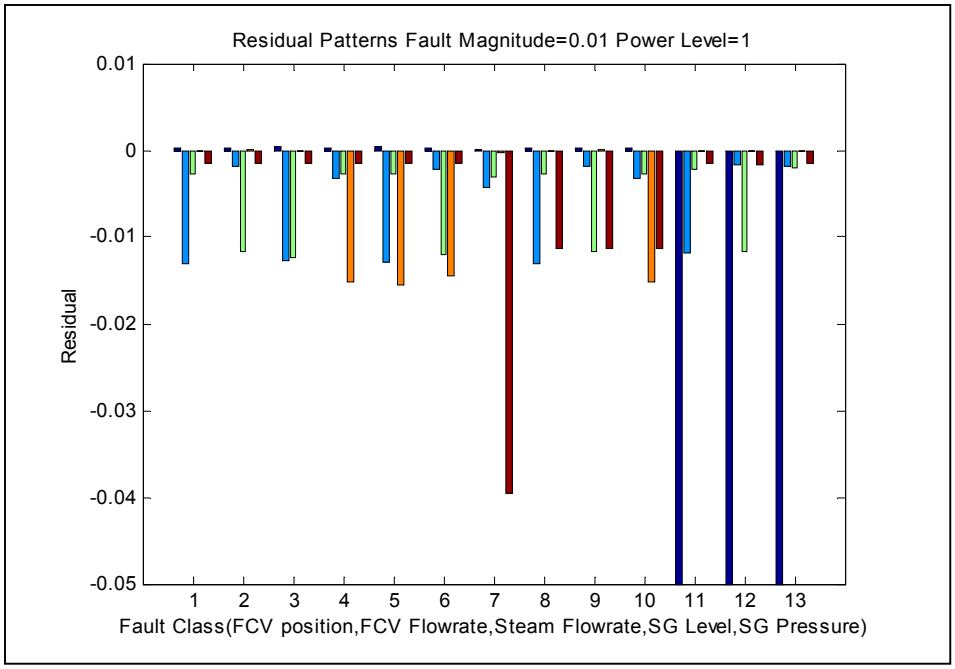

Figure 5.9. Structured residual pattern using ANFIS models (100% Power, 1% offset fault**).**

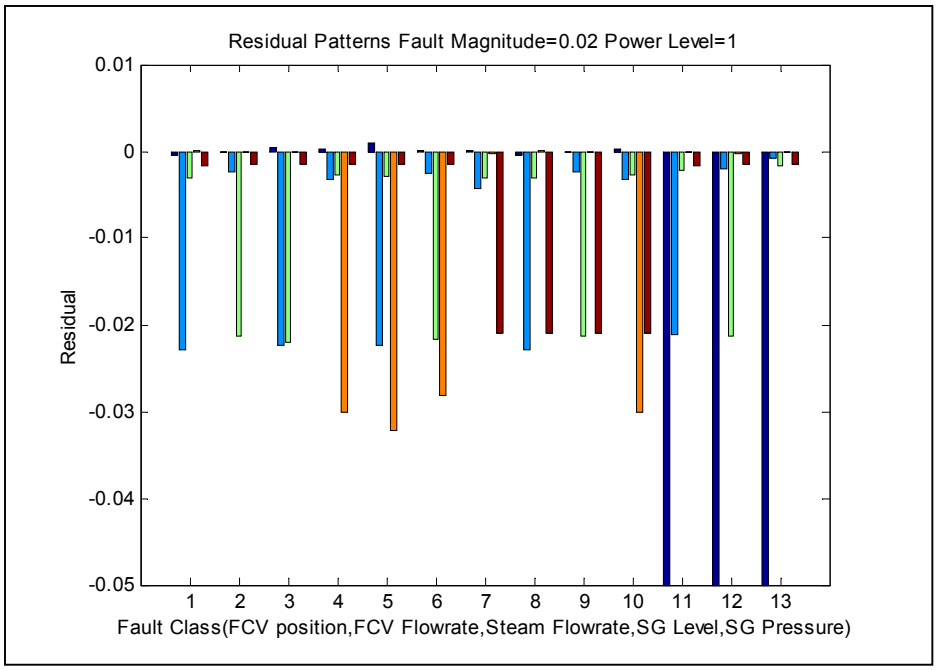

Figure 5.10. Structured residual pattern using ANFIS models (100% Power, 2% offset fault).

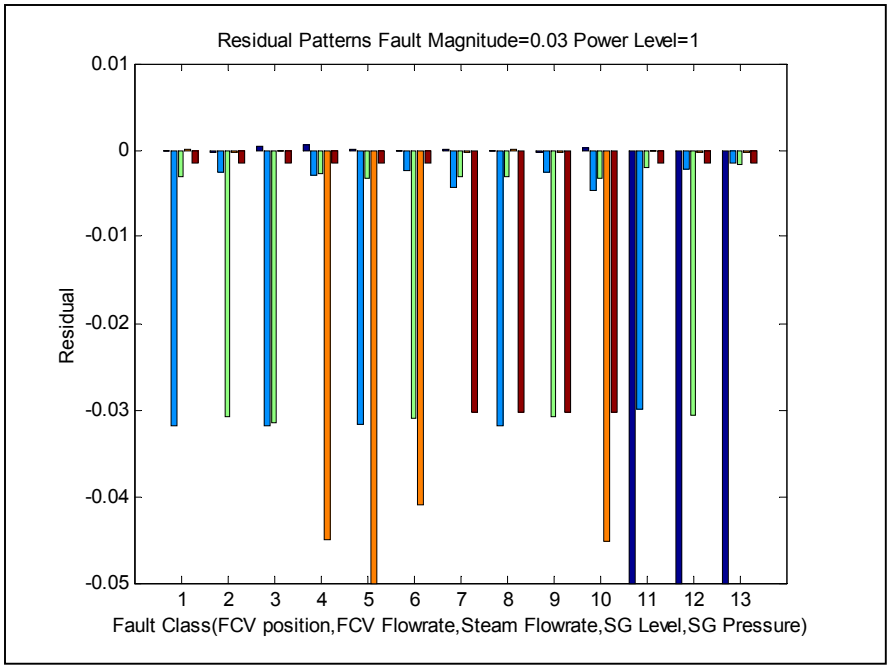

Figure 5.11. Structured residual pattern using ANFIS models (100% Power, 3% offset fault).

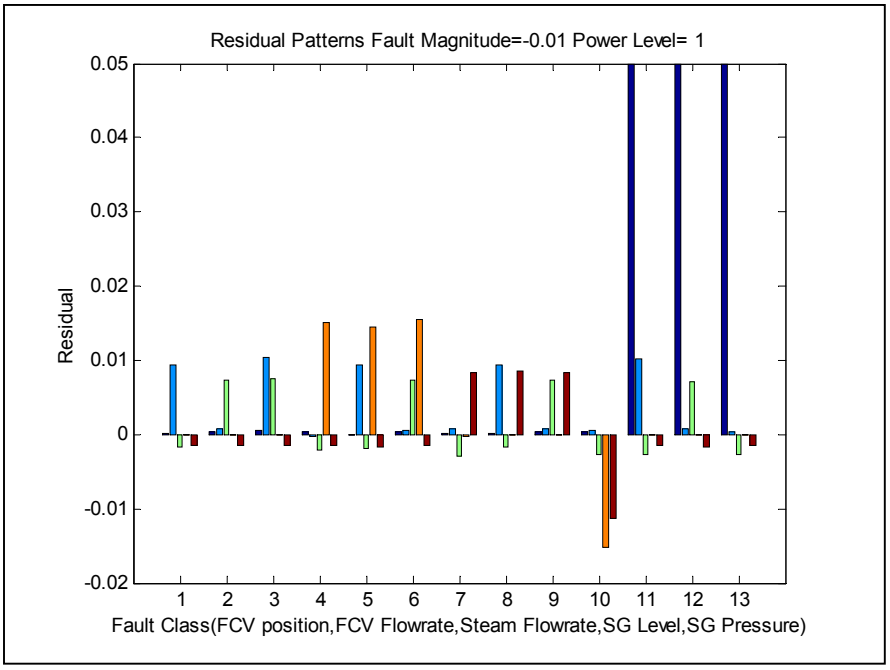

Figure 5.12. Structured residual pattern using ANFIS models (100% Power, -1% offset fault).

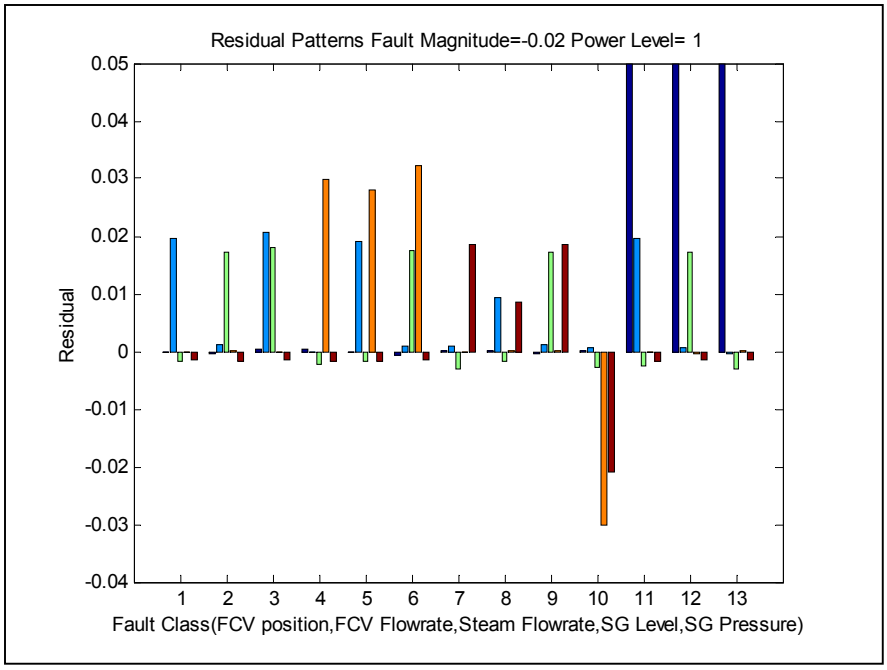

Figure 5.13. Structured residual pattern using ANFIS models (100% Power, -2% offset fault).

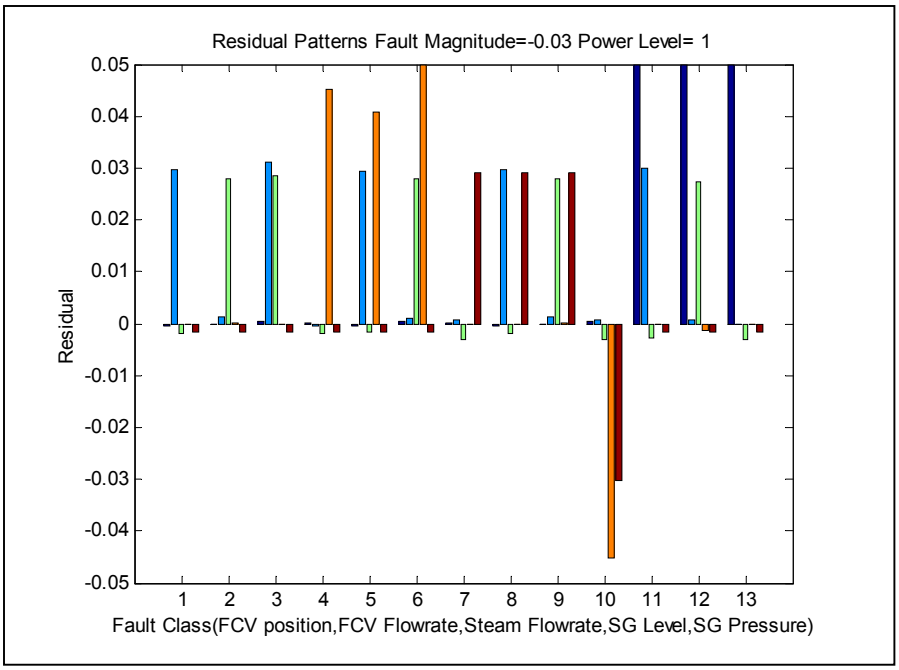

Figure 5.14. Structured residual pattern using ANFIS models (100% Power, -3% offset fault).

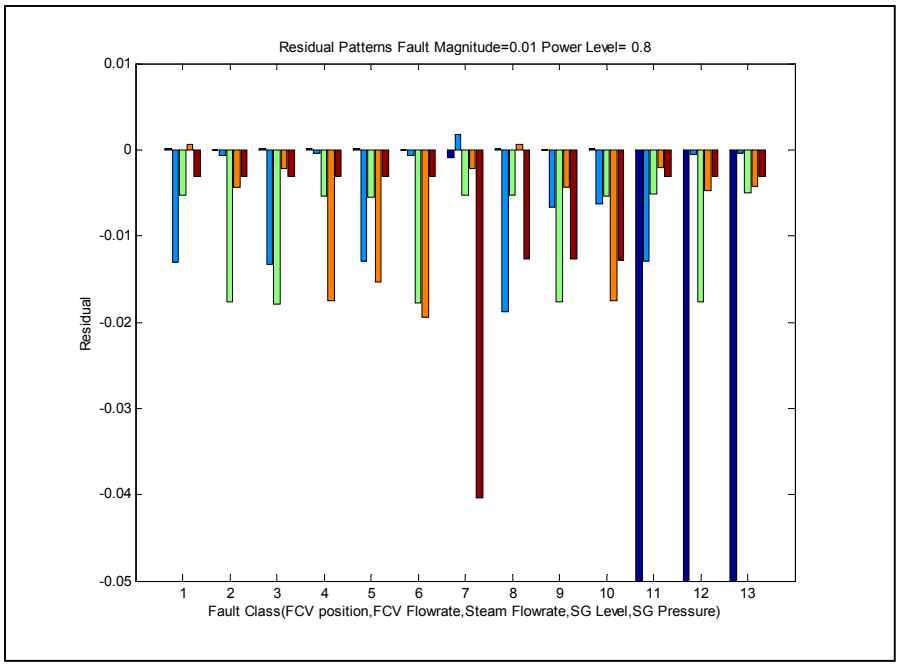

Figure 5.16. Structured residual pattern using ANFIS models (80% Power, 1% offset fault).

- Fault class 6= Steam flow meter offset fault and SG level sensor offset fault.
- Fault class 7= SG pressure sensor offset fault.
- Fault class 8= Feed water flow meter offset fault and SG pressure sensor offset fault.
- Fault class 9= Steam flow meter offset fault and SG pressure sensor offset fault.
- Fault class 10= SG pressure sensor offset fault and SG level sensor offset fault.
- Fault class 11= Feed water flow meter offset fault and FCV offset sensor offset fault.
- Fault class 12= Steam flow meter offset fault and FCV position offset fault.
- Fault class 13= FCV valve position offset fault.

In each figure, the 13 faults have different residual patterns, so they can be isolated. If the residual patterns are compared for different fault magnitudes, their structures are stable. Moreover, the residuals are approximately equal to the fault magnitudes of the sensor faults such as the feed water flow meter fault and the steam flow meter fault. Theoretically, the residuals should be exactly as same as the fault magnitudes. However, due to the modeling errors of these data driven models, some slight differences still exist and these slight differences in FDI are acceptable. The faults occurring at 80% initial power level other than 100% full power are also tested. Figure 5.16 clearly shows that the performance of the FDI system does not degrade. The residual structures keep the same pattern as those faults at 100% power level.

5.5 Discussions

This chapter has presented ANFIS model based approach to fault detection and isolation and the application to PWR steam generator system. ANFIS model based approach combined with structured residual design is shown to be efficient in fault detection and isolation if the possible faults are enumerable. For single faults, strong isolation scheme can be achieved through appropriate choice of the model structures. For dual faults, it is not possible to achieve strong isolation between the dual faults and one of the element faults. Using natural redundancy and derived redundancy, dedicated residual structure can be achieved to isolate dual faults. In order to detect and isolate a fault related to control variable, sometimes it is necessary to use the information about

measurement redundancy itself. ANFIS model based approach combined with structured residual design does not need fault signatures dependent of fault magnitude and initial operation condition. It is in conformance with the principle of modern fault detection and isolation methods. Since ANFIS is able to learn the relationship between variables from data, it has the power of on-line implementation.

However, ANFIS model based approach needs to enumerate the possible faults. This still exert heavy burden on engineering application. In addition, for a non-linear complicated system, structured residual design for fault isolation, especially when data driven modeling is used, is essentially a process of trial and error. This exerts additional difficulties in engineering application.

Chapter 6

Data Driven Model Causal Graph for Fault Diagnosis

6.1 Introduction

In Chapter 5, the dedicated residual structure is achieved based on the assumption that the possible faults are enumerable. With the faults known, the designers may achieve fault isolation through appropriate choice of the models. However, for a large complex system, it will be extremely challenging to enumerate all the possible faults. In order to avoid enumerating possible faults and predefining their associated fault signatures, the development of data driven model causal graph approach has been considered.

Data driven model causal graph is proposed as a generic approach to fault diagnosis for fault isolation. It is able to combine the reasoning capability of qualitative knowledge based method and the strength in resolution of quantitative knowledge based method. To facilitate on-line implementation, ANFIS is used for modeling. Fault detection is fulfilled by monitoring the residual of each model. Fault isolation is achieved by cause effect analysis of the residuals generated from the models.

6.2 Cause Effect Reasoning Using Model Causal Graph

Cause effect reasoning was originally introduced as a reasoning tool to account for the propagation of fault symptoms within a system (Davis, 1983). It has been extended to quantitative model based FDI when mathematical models are available (Montain and Gentil, 2000).

A model causal graph consists of individual nodes connected by quantitative models. The individual nodes represent plant parameters, state variables and measurement variables. The quantitative models represent the cause-effect relationship between the nodes. As compared with sign directed graphs using qualitative knowledge only to describe the relationship between variables, a quantitative model is formally

introduced to express the cause effect relationships. The model causal graph is not a simple network of structural models. It includes the dynamic information about process flow-path, signal flow-path, and control logic so that a fault can be localized based on the cause effect analysis for a process system.

A physical system can be represented by the following set of differential equations:

$$\dot{X}_i = g_i(G_i, X_i) \tag{6.1}$$

where

$\boldsymbol{X_i}$ = the i th system variables.

$\boldsymbol{g_i}$ = a function to estimate $\boldsymbol{X_i}$.

$G_i = \{X_j \mid j \neq i\}$,

$\boldsymbol{G_i}$ = a set of variables as the inputs to $\boldsymbol{g_i}$.

The moving average form of the above differential equation can be used to arrive at the causal graph models, that is:

$$X_i = f_i(G_i) \tag{6.2}$$

If the model causal graph is developed based upon the original process flow and signal flow, the causal relationship between variables will be implicit in it.

The cause effect relationship between the inputs and the outputs of a model has two connotations (Leyal, Gentil and Stephan, 1994). From physics point of view, the cause-effect relation represents the pathway of the signal propagation. Any changes in the model inputs are always before any changes in the model outputs in the time domain. From the computational point of view, the cause-effect relation means that any changes in the model inputs will sufficiently cause some changes in the model outputs and the model outputs will not change without any changes in the model inputs. Figure 6.1 shows a simple example of a model causal graph. In the figure, four models $\boldsymbol{g_1, g_2, g_3, g_4}$ are shown to characterize the system. The six process variables are X1, X2, X3, X4, X5, and X6.

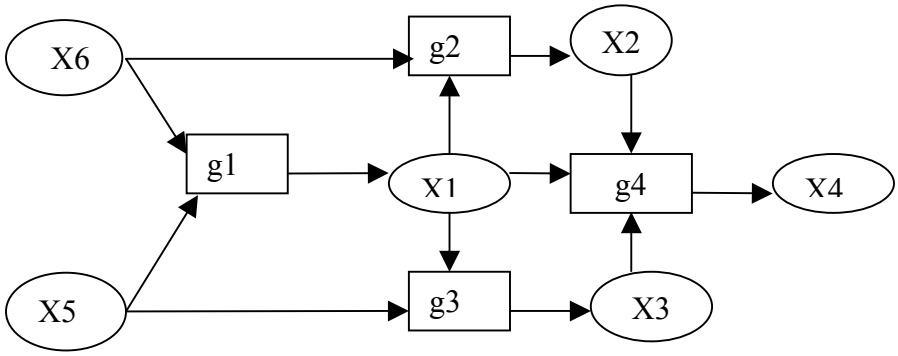

Figure 6.1. A simple example of model causal graph.

Causal effect reasoning can be easily performed based on analyzing the residuals of the individual models. The original residuals are calculated for each measured variable as follows:

$$R_i = X_i - X_i^* \tag{6.3}$$

where

$\boldsymbol{R_i}$ = the residual of the variable $\boldsymbol{X_i}$.

X_i = the measured value of the variable $\boldsymbol{X_i}$.

$\boldsymbol{X_i^*}$ = the estimated value of the variable $\boldsymbol{X_i}$ from the model $\boldsymbol{f_i}$ defined previously.

If $\boldsymbol{R_i}$ is significant, it can be determined that a fault has occurred to the system. However, there are still two possibilities that may explain the abnormal residual:

a) A local fault affecting $\boldsymbol{X_i}$.

b) A consequence of a fault affecting the inputs of the model f_i.

To facilitate fault isolation, a set of reconstructed residuals are calculated as follows:

$$\tilde{R}_i^j = X_i - \tilde{X}_i^j \tag{6.4}$$

where

$\tilde{\boldsymbol{R}}_i^j$ = the residual of $\boldsymbol{X_i}$ after the input $\boldsymbol{X_j}$ of model f_i has been reconstructed.

$\tilde{\boldsymbol{X}}_i^j$ = .the estimated value of $\boldsymbol{X_i}$ after the input $\boldsymbol{X_j}$ of model f_i has been reconstructed

Fault isolation can then be based on the following decision procedure:

a) If $\widetilde{\boldsymbol{R}}_i^j \ll \boldsymbol{R}_i$, $\boldsymbol{X}_j$ has a local fault.

b) If $\widetilde{\boldsymbol{R}}_i^j \approx \boldsymbol{R}_i$ for all $\boldsymbol{X}_j$ that will affect X_i, $\boldsymbol{X}_i$ has a fault.

c) If $\widetilde{R}_i^{j_1} \approx R_i$ and $\widetilde{R}_i^{j_2} \approx R_i$ but $\widetilde{R}_i^{j_1 j_2} << R_i$, then X_{j_1} and X_{j_2} have simultaneous faults.

As an example, the above model causal graph method is used for a typical feed back control loop as shown in Figure 6.2. Four nodes connected by three models are used to represent the control loop. These four nodes are the set point, the controller output, the control variable, and the regulated variable. The three models are the controller model, the actuator model and the plant model. Since a controller always takes the measured value of the regulated variable as input, the controller model can always be used to isolate a controller fault. For the same reason, the actuator model can be used to isolate an actuator fault. However, fault detection and isolation becomes a challenging task when a fault related to the regulated variable is involved. When a new steady state is reached after the fault, the regulated variable will be brought back to its original level. Hence, the steady state information is not enough to detect such a fault.

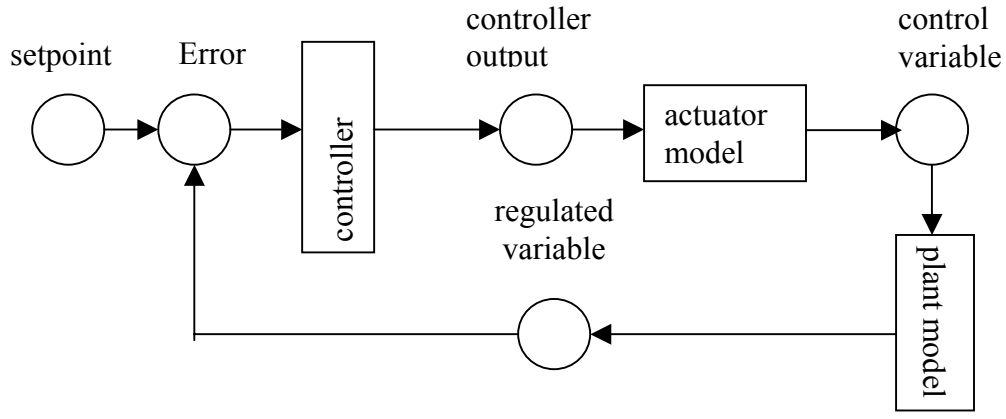

Figure 6.2. Dynamic model causal graph representation of a feedback control loop

Model causal graph method needs to use dynamic models. If a steady state model is used, the developed method can give correct FDI results after a new steady state has been reached. During the fault transient, the steady state models will result in serious false alarms. Moreover, the fault symptoms may become weak after the new steady state due to controller feedback. In addition, dynamic models must be used to isolate a controller fault and some actuator faults such as control valves. By the way, in order to achieve a faster fault detection and isolation for safety concern, dynamic models are also desired.

6.3 Extended Model Causal Graph

6.3.1 Multi-model causal graph

Multiple-model causal graph can be introduced to isolate input faults based on cause effect analysis of model residuals when there are no additional models available to reconstruct these process inputs.

Multi-model approach was proposed for fault isolation (Simani, 2000). The basic idea is to make most use of the knowledge about the process redundancy inherent in a system. For example, in a saturated SG system, there is a one-to-one relation between the SG pressure and the SG temperature. Therefore, any model as a function of SG pressure can always be used to derive a new model as a function of SG temperature. The cause effect analysis on the residuals of these two models can then be performed to isolate the two faults.

Figure 6.3 and Figure 6.4 show two types of multiple models designed to isolate output faults and input faults, respectively. In the design scheme shown in Figure 6.3, one output and all the inputs drive each model. An output measurement fault affects only the residual of the model driven by this output variable. Therefore, the output faults can then be isolated if there is no fault related to the inputs. In the design scheme shown in Figure 6.4, each model is driven by all but one input and all the outputs, which generates a residual sensitive to all but one input fault. Therefore, the input faults can then be isolated if there is no fault related to the outputs.

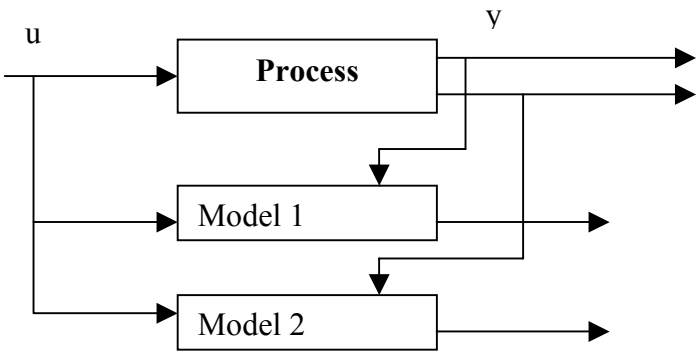

Figure 6.3. Multiple models to isolate output faults.

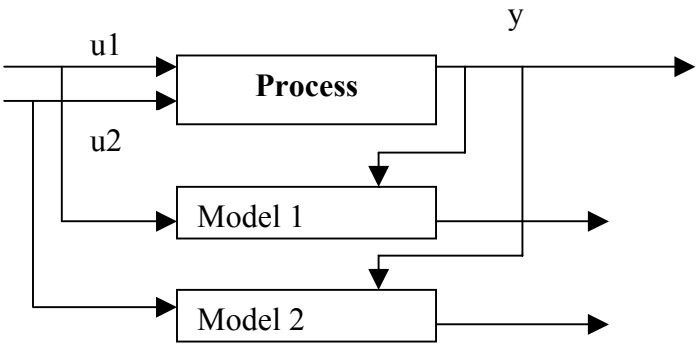

Figure 6.4. Multiple models to isolate input faults.

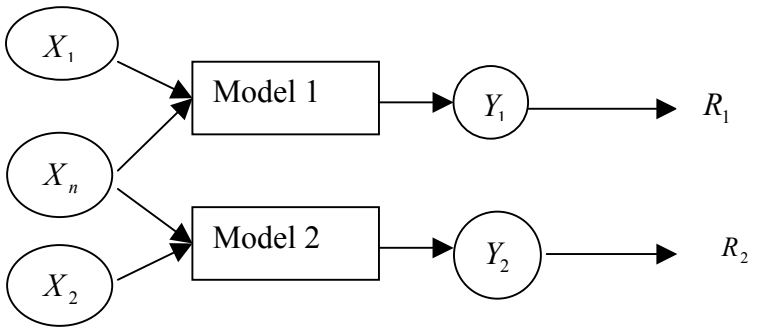

Figure 6.5. An example of multiple-model causal graph.

Figure 6.5 shows an example of how to combine multi-model approach and regular model causal graph to obtain a multi-model causal graph. If X_1 and X_2 cannot be reconstructed from some other models, the multi-model causal graph still enables to perform cause effect analysis on the model residuals. After the subsequent models have confirmed there are no faults related to variable Y_1 and Y_2, the following decision logic can be performed:

a) If $R_1 \approx R_2 \approx R \approx 0$, there is no fault with respect to X_1, X_2 and Y_1 and Y_2.

b) If both R_1 and R_2 are significant, there is a fault with respect to X_n.

c) If $R_1 \approx 0$ but $R_2 \neq 0$, there is a fault with respect to X_2.

d) If $R_2 \approx 0$ but $R_1 \neq 0$, there is a fault with respect to X_1.

6.3.2 Model causal graph with hidden nodes

Model causal graph can also be extended to include unmeasured variables. This is useful to detect and isolate process faults. Figure 6.6 shows an example. In the figure, X1, X2, X3, and X4 correspond to four measured variables and H1 corresponds to an unmeasured variable. In this case, the same reasoning logic can be used except that H1 cannot be used as an independent residual generator. It is necessary to prepare an explicit model instead of a data driven model to estimate the value of a hidden node. If a data driven model is to be used, some special learning algorithm must be developed.

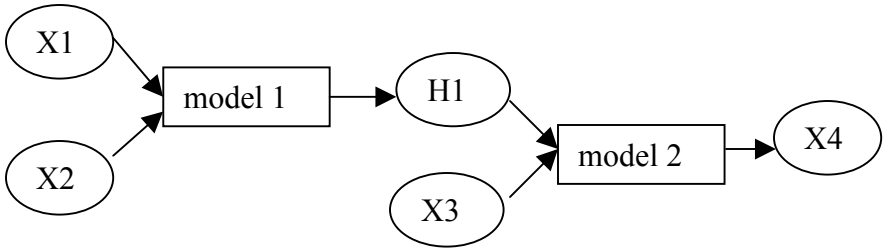

Figure 6.6. Model causal graph with hidden nodes.

6.4 Model Causal Graph Approach with Fuzzy Inference Modeling

Theoretically, an ANFIS model is able to approximate a system to any desired degree of accuracy. However, in real situation, it is not always able to achieve this accuracy because a too complex model may be required. Even if such a model can be obtained, the desired capability of generalization can still not be guaranteed.

In order to achieve a perfect model, the input variables must always be cautiously selected. On the one hand, the co-linearity between the input variables should be avoided since the least square method is used in training ANFIS. On the other hand, the dimensionality of the inputs for the ANFIS models should be as few as possible. In ANFIS, each input variable needs to be fuzzified into problem specific membership functions. When the number of input variables is increased, the number of nodes in the second layer and the third layer of the ANFIS network will be increased exponentially. Correspondingly, the number of rules used in the system will be increased too. This increase will not only have a severe influence on the training speed but also on the stability of the built models because the number of degrees of freedoms may be more than necessary. The principle of choosing the number of input variables is that none of the redundant input variables should be retained in the ANFIS inputs.

An efficient ANFIS model with parsimonious number of rules and membership functions can be achieved only through physically correct choice of inputs. In order to characterize the behavior of a dynamic system, physically driving inputs are much more efficient than purely input delay and output delay. If all the input variables driving the

dynamic process are included in the model, much fewer delays will be required to perform the input-output mapping.

Model causal graph method is in full agreement with the requirement of efficient ANFIS modeling using the system knowledge. The physically involved inputs can be obtained through studying the cause-effect relationship. Therefore, the ANFIS model is able to have most appropriate inputs when combined with cause effect analysis. If known, some important non-linearity can also be directly captured through an unmeasured node in the model graph before it is presented as an input to an ANFIS model. For example, the pressure loss can be assumed as a square function of the flow rate. An unmeasured node can then be designed as the square of the flow rate in the model graph and is used as an input to the ANFIS model to estimate the pressure drop. By explicitly including nonlinear terms in the inputs, fewer membership functions will be needed in the resulting ANFIS models. Model causal graph method helps to decompose a complex model into several small models. The model decomposition can significantly enhance the performance of data driven modeling such as ANFIS when used for FDI. In order to achieve an accurate data driven model, the amount of data required is proportional to the number of inputs. When a complex model is decomposed, much fewer inputs are related to each small model, and correspondingly, much fewer data will be required to train the small model than a complex model with a great number of inputs.

In the case that sufficient knowledge is known about a system so that the rules and the member functions of the inputs can be specified, the training algorithm of the ANFIS system can also be adapted for this purpose. Since ANFIS is a fuzzy model in nature, it is easy to integrate expert knowledge in different forms. Knowledge in different confidence can be represented explicitly by appropriate choice of the shape of membership function.

6.5 Procedures of Model Causal Graph Approach

The following is a summary of the procedures to design a data driven model causal graph based FDI:

- Design the model causal graph structure.

The structure can be obtained from process block diagram and control system design scheme.

- Develop individual models.

In some cases, one model defined in the first step may be decomposed into several models in series. The series of models correspond to the inclusion of some hidden nodes and some multi-model causal graph sub-modules.

- Develop fault detection module.

Appropriate thresholds should be specified for all the nodes. A too small threshold may cause false alarms and a too big threshold may cause missing detection.

- Develop causal reasoning algorithm.

If only single faults are involved, the simple residual reasoning algorithm can be directly implemented. If some dual faults are of concern for the FDI system, some extended reasoning schemes may need to be designed.

The implementation of data driven model causal graph based FDI can be summarized as following steps:

- Fault detection is fulfilled by monitoring the residual of each model.
- For any abnormal model, the possible root causes are identified by tracking backwards until a model gives insignificant residual.
- All the corrupted signals are reconstructed by tracking forward from the identified fault origin to the input nodes of the detected model.
- Finally, cause effect reasoning is performed on the residuals for fault isolation.

6.6 Application to Nuclear SG System

Figure 6.7 shows the model causal graph of steam generator water level system for a PWR nuclear power plant. The models in series can be summarized as follows:

Controller output (t)=f(steam flow rate(t)-feed water flow rate(t), SG level(t)-SG reference level(t))

FCV valve position (t+1)=f (controller output (t), FCV valve position (t))

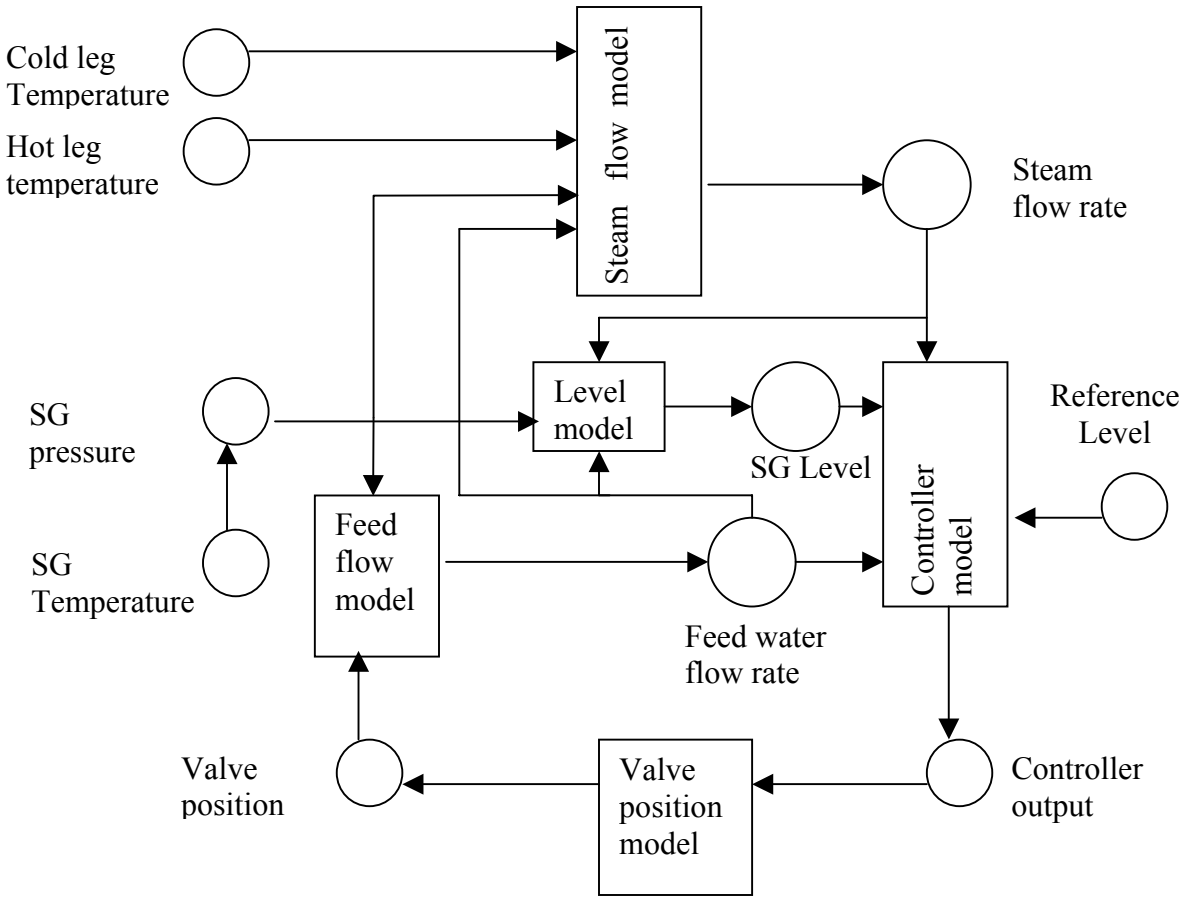

Figure 6.7. Model causal graph of nuclear SG system.

FCV flow rate (t)=f (FCV valve position (t), SG pressure (t))

FCV flow rate (t)=f (FCV valve position (t), SG pressure (t))

Steam flow rate (t+1)=f (feed water flow rate (t), SG pressure (t), hot leg temperature (t), cold leg temperature (t+1))

Figure 6.8 and Figure 6.9 show a comparison between the estimated controller output and the indicated controller outputs for a controller gain offset fault and a feed water flow meter offset fault. As can be seen, the residual can be used to isolate a controller fault as a local fault. If some other faults related to the controller input signals such as feed water flow rate, steam flow rate, or SG level occur, the residual of the controller output remains close to zero. The reason is that the controller model always uses indicated signals. Even if some faults happen to the input signals of the controller, the controller model itself is still not violated. By the way, the capability of isolating the controller fault as a local fault demonstrates that the ANFIS model is precise enough to capture the dynamic behavior of the controller.

Figure 6.10 and Figure 6.11 show a comparison between the estimated valve position change and the indicated valve position change for a valve position offset fault and a feed water flow meter offset fault. As can be seen, the residual can be used to isolate a valve position fault as a local fault. If feed water flow meter offset fault happens, the residual of the valve position change remains close to zero. The reason is that the valve position change is physically determined by the controller output signal.

Figure 6.12 shows the model causal graph to estimate the feed water flow rate. Figure 6.13 shows the residual of feed water flow rate before and after the SG pressure is reconstructed for a feed water flow meter sensor fault. The figure shows that the residual does not change much before and after all the input signals are reconstructed. Therefore, the detected fault can be isolated as a local fault. Figure 6.14 shows the residual of feed water flow rate before and after the SG pressure is reconstructed for SG pressure sensor fault. The reconstruction of SG pressure signal can fully explain the original residual. This indicates that the detected fault is a secondary fault and the fault can be isolated as a SG pressure sensor fault.

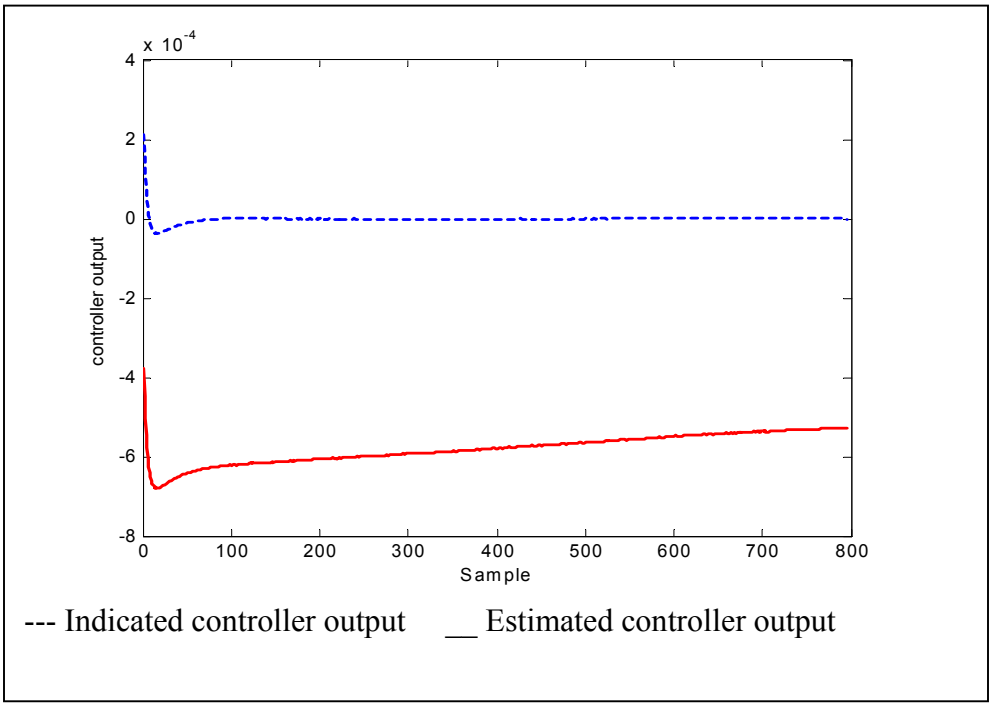

Figure 6.8. Controller output for controller gain offset fault.

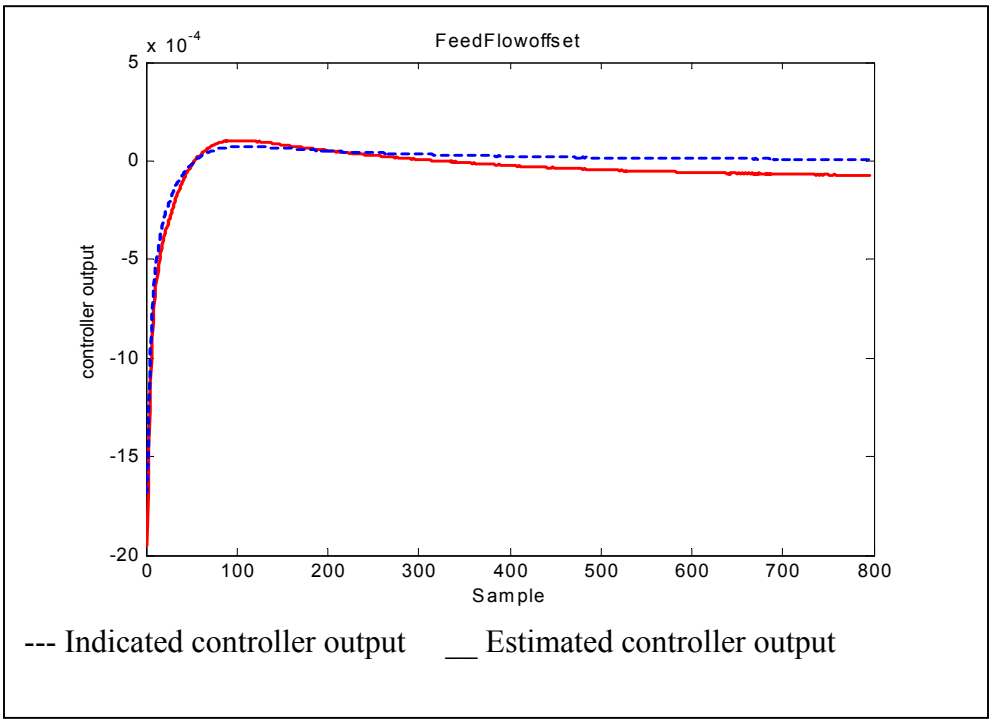

Figure 6.9. Controller output for feed water flow meter sensor fault.

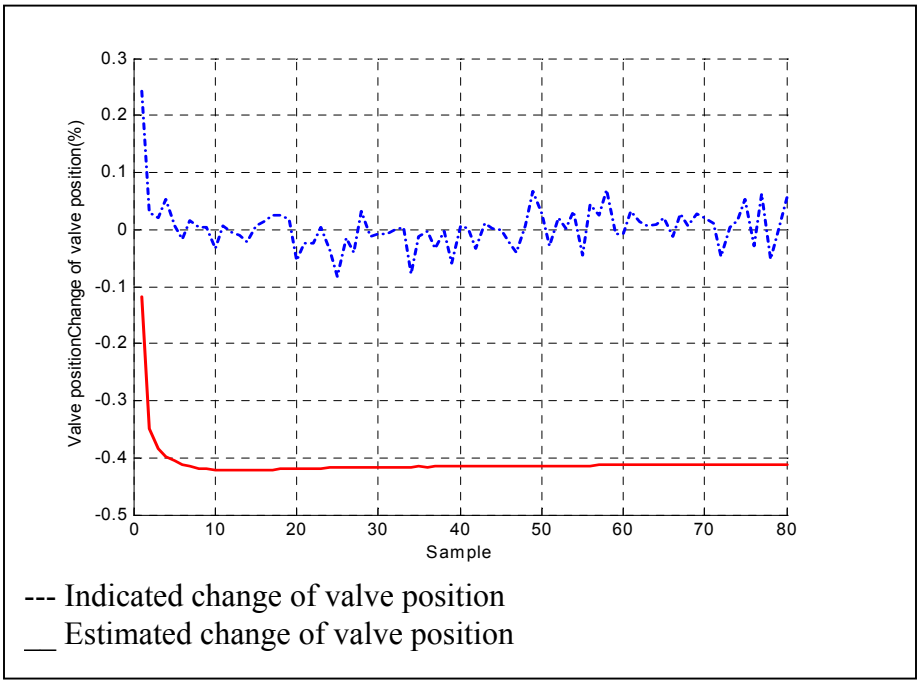

Figure 6.10. Change of valve position for valve position fault.

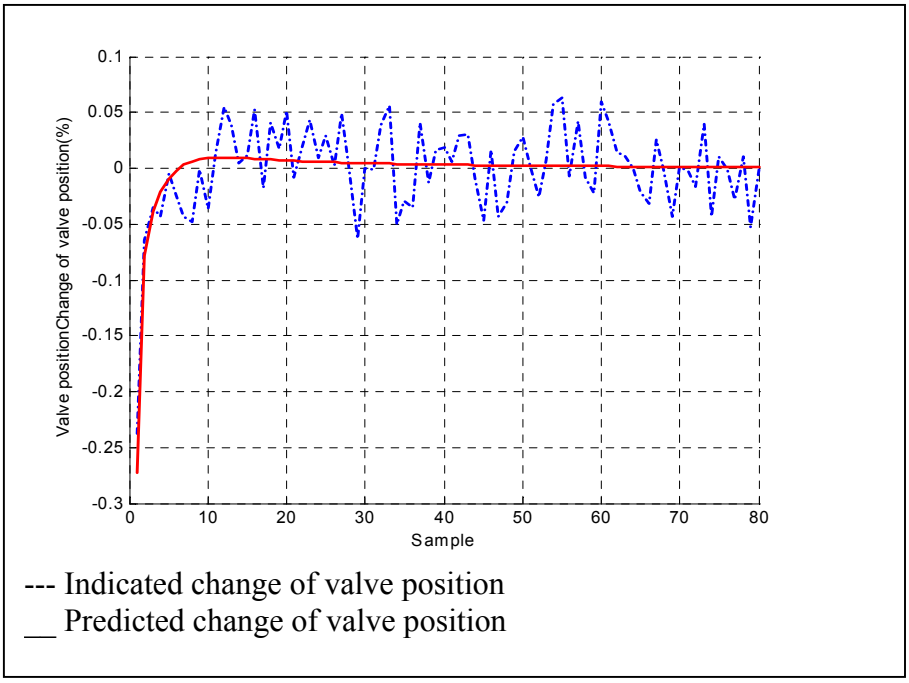

Figure 6.11. Change of valve position for feed water flow meter sensor fault.

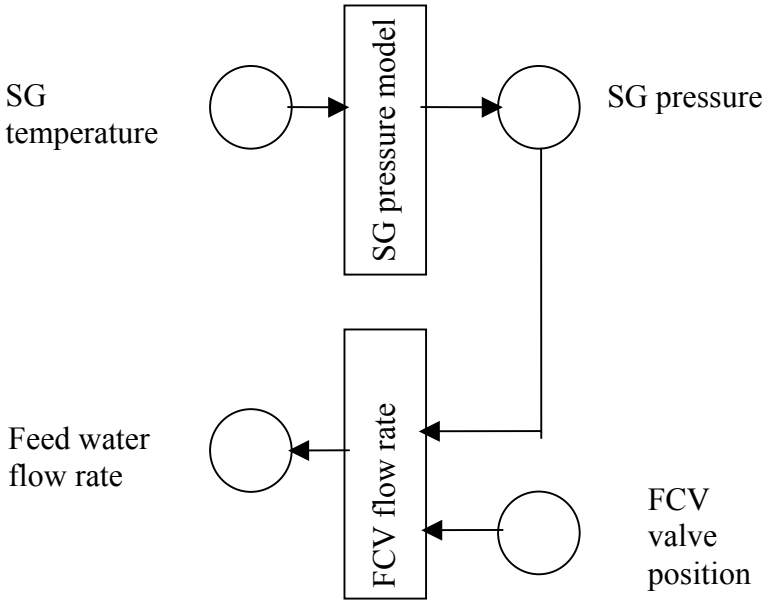

Figure 6.12. Model causal graph of feed water flow rate.

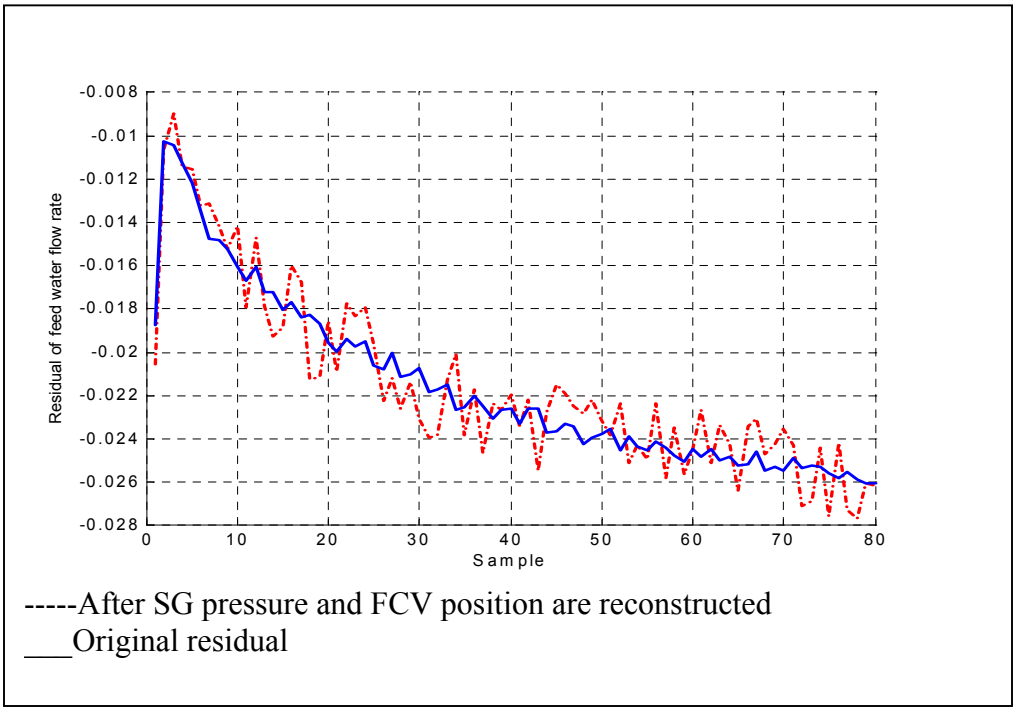

Figure 6.13. Model causal graph approach to isolate feed water flow meter sensor fault.

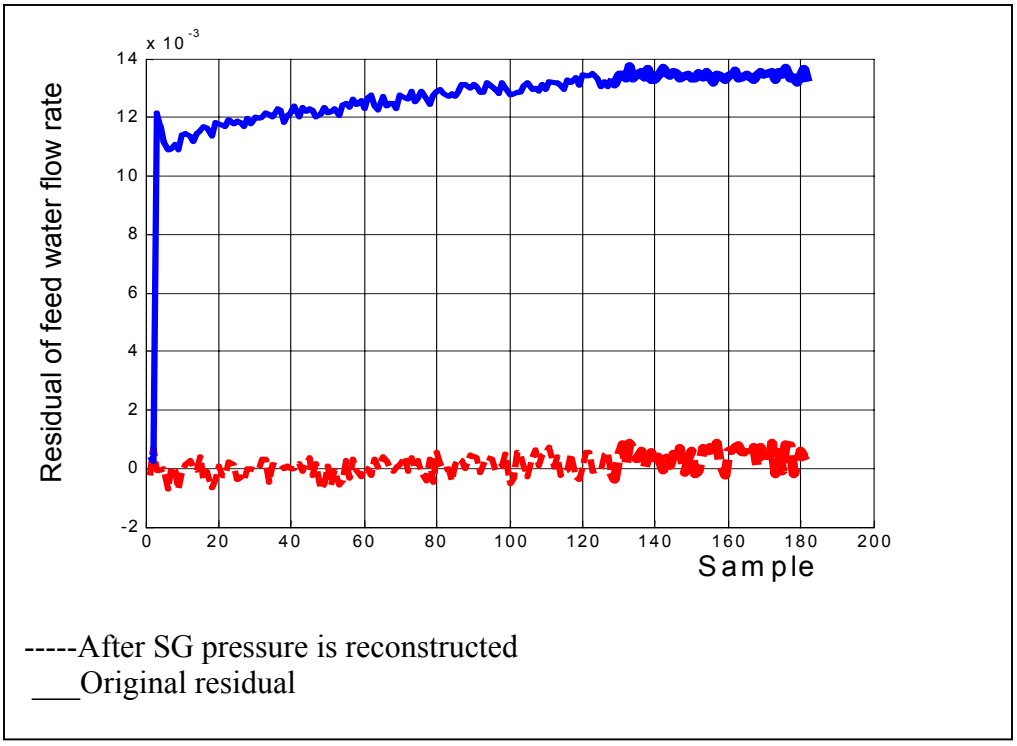

Figure 6.14. Model causal graph approach to isolate SG pressure sensor fault using feed water flow rate model.

Figure 6.15, Figure 6.16 and Figure 6.17 show the residual of the steam flow rate for a steam flow meter sensor fault, feed water flow meter sensor fault and SG pressure sensor fault respectively. Figure 6.15 shows that the residual of steam flow rate does not change much before and after the SG pressure, the feed water flow rate, and the feed water flow rate and the SG pressure is reconstructed. Therefore, the detected fault can be correctly isolated as a steam flow meter sensor fault. Figure 6.16 shows the residual of the steam flow rate for feed water flow meter sensor fault. The reconstruction of feed water flow rate signal can fully explain the original residual. This indicates that the detected fault is a secondary fault and the fault can be isolated as a FCV flow meter sensor fault. Figure 6.17 shows the residual of the steam flow rate for the SG pressure sensor fault, the reconstruction of the SG pressure signal can fully explain the original residual. Therefore, the detected fault is a secondary fault and can be correctly isolated as a SG pressure sensor fault.

In order to detect and isolate the SG narrow range level sensor fault, it is necessary to build a dynamic model to estimate SG level. From physics point of view, The SG level can be determined by the SG mass and the SG thermal parameters. For this reason, an unmeasured node, SG mass, is used to estimate the SG level. It is expressed as a function of SG mass, SG pressure, SG temperature as well as feed water temperature, cold leg temperature and hot leg temperature, shown in Figure 6.18. The SG mass can be estimated as a function of feed water flow rate and steam flow rate. In fact, without using the SG mass as an explicit variable, it is extremely difficult to build a data driven model to estimate the SG level. The reason is that the SG mass is the integral effect of the incoming feed water flow rate and the out-flowing steam flow rate. A given value of SG mass may correspond to any value of FCV flow rate and steam flow rate. In the specific case, a model using delay input does not help to track the dynamic behavior either since the SG indicated level would be ultimately brought back to its normal value after a SG level sensor fault due to the controller feedback.

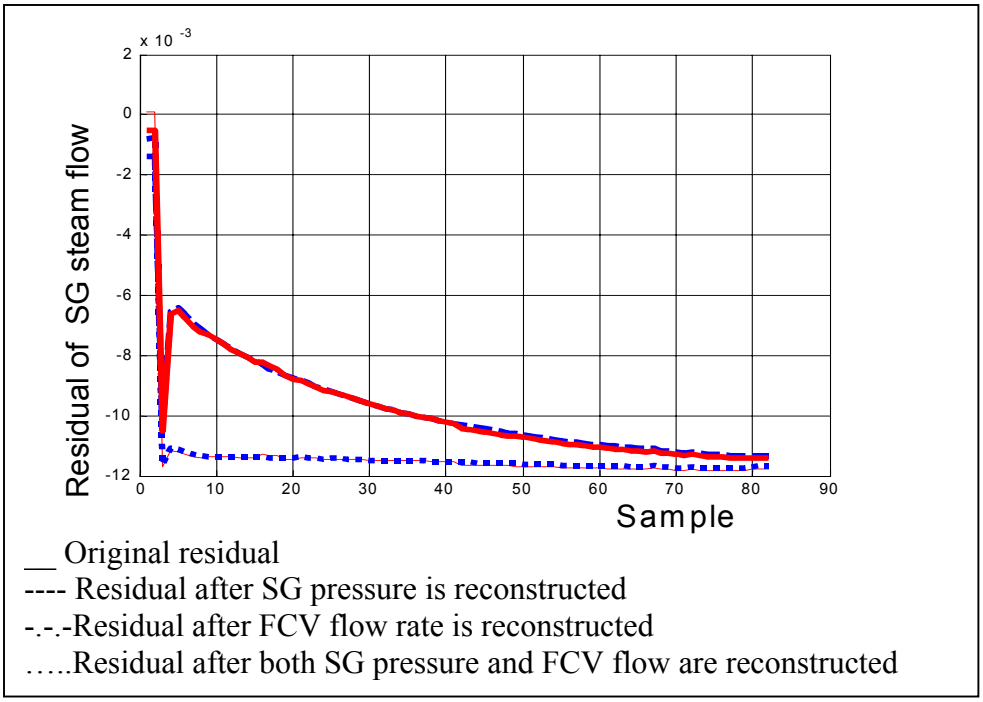

Figure 6.15. Model causal graph approach to isolate steam flow meter sensor fault.

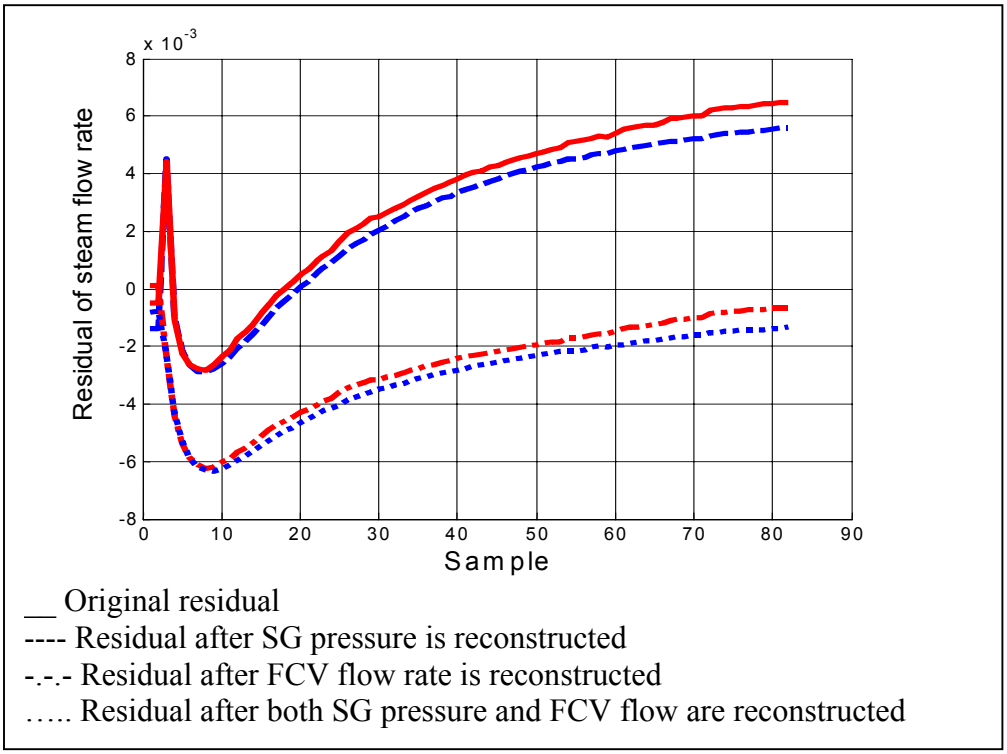

Figure 6.16. Model causal graph approach to isolate feed water flow meter sensor fault.

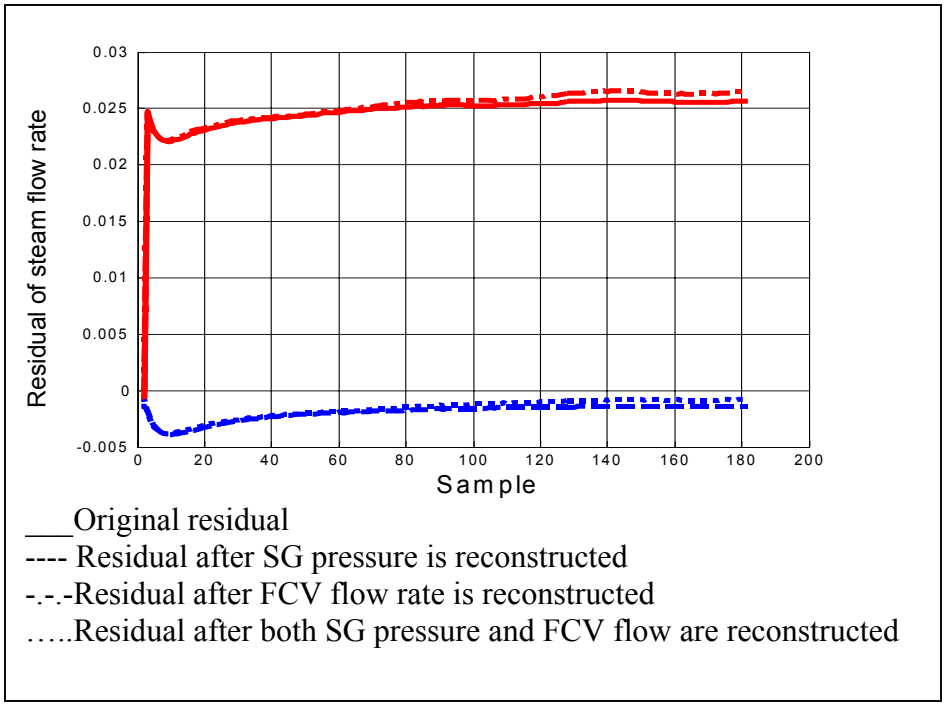

Figure 6.17. Model causal graph approach to isolate SG pressure sensor using steam flow rate model.

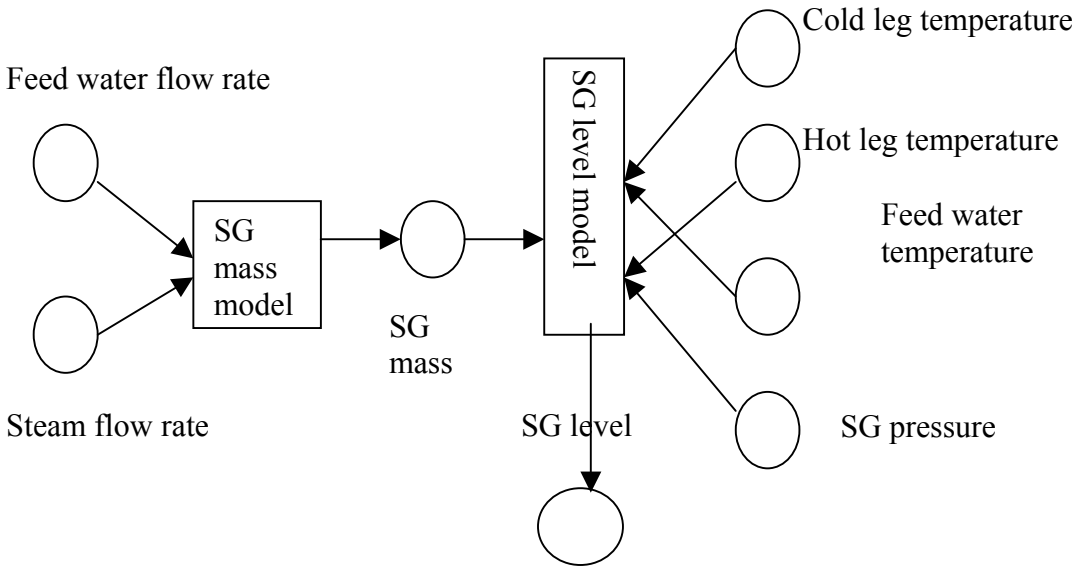

Figure 6.18. Model causal graph of SG level measurement.

Figure 6.19 shows the residual of the SG level for a SG level sensor fault before and after the input signal is reconstructed. The residual does not change because of the input reconstruction for the detected fault. This indicates that the fault is a local fault. The detected fault can then be successfully isolated.

Figure 6.20 shows the residual of the steam flow rate for simultaneous feed water flow meter sensor fault and SG pressure sensor fault. The original residual can be used to detect the fault. In order to isolate the faults, reconstructed residuals are used. When either SG pressure or feed water flow rate is reconstructed, the residual can be reduced. However, the reconstruction of either signal is not enough to explain 100 percent of the original residual. Only when feed water flow rate and SG pressure are reconstructed can the residual reach minimal. Therefore, from explaining maximal fault signature point of view, the simultaneous dual faults can be correctly isolated. Figure 6.21 shows the residual of feed water flow rate for simultaneous feed water flow meter sensor fault and SG pressure sensor fault. The original residual can be used to detect the fault. After SG pressure is reconstructed, the residual of FCV flow rate can be reduced by 50%. After both SG pressure and FCV position are reconstructed, the residual of FCV flow rate cannot be further reduced. It can be concluded that SG pressure sensor is faulty and FCV valve position is healthy. However, the remained residual is still about 0.5%. This fraction of the original residual must be explained by the assumption that the feed water flow meter sensor has a fault.

6.7 Comparison with Other Approaches

Although PCA based FDI and ANFIS model based FDI with structured residual design can be used for fault detection and isolation in some applications, model causal graph approach has some unique features in FDI system design and application to engineering problem.

Both PCA based FDI and ANFIS model based FDI with structured residual design can only be designed when the possible faults are enumerable. However, model causal graph approach isolates a fault based on cause effect reasoning on model residuals,

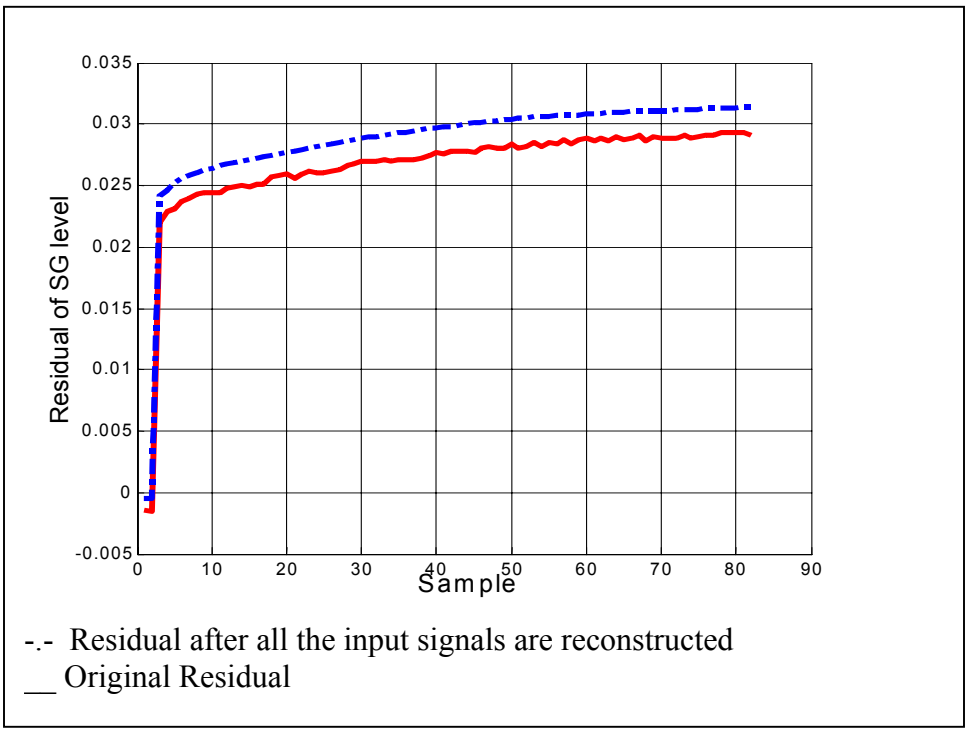

Figure 6.19. Model causal graph approach to isolate SG level sensor fault.

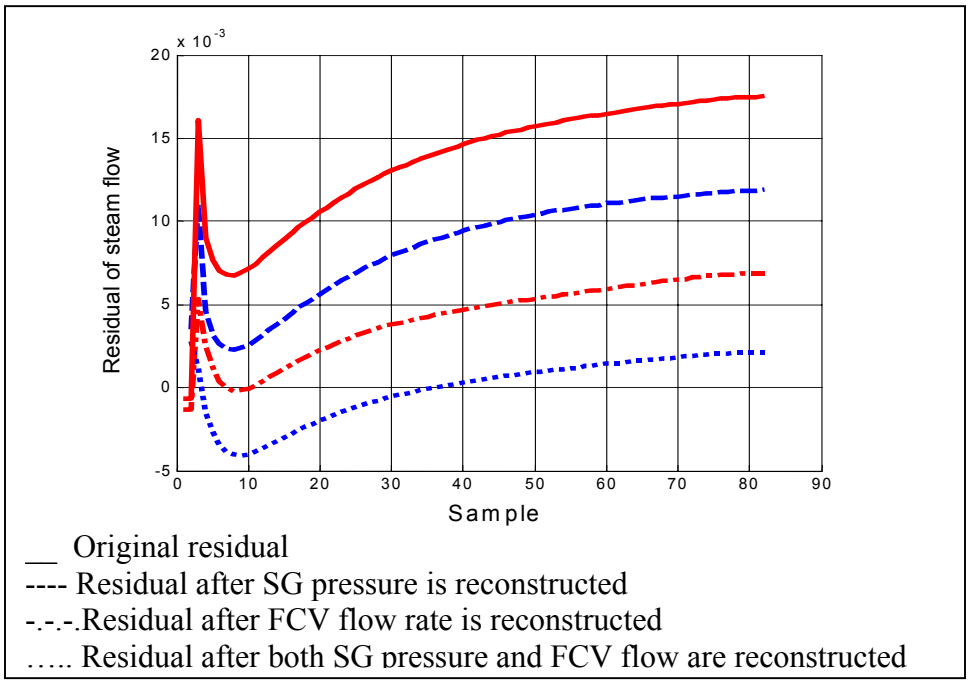

Figure 6.20. Model causal graph approach to isolate feed water flow meter sensor fault and SG pressure sensor fault.

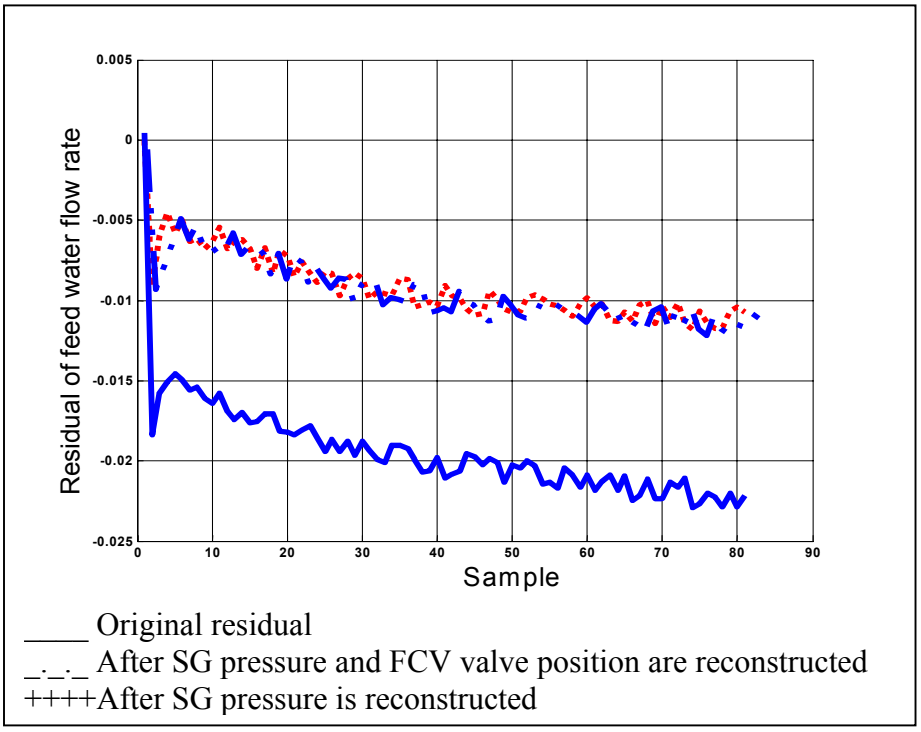

Figure 6.21. Model causal graph approach to isolate SG pressure sensor fault and feed water flow meter sensor fault.

so it is not necessary to predefine the faults and their fault signatures. To incorporate an automated detection and isolation into a large safety critical system such as nuclear power plants, this is a significant step moving forward to engineering application.

Model causal graph approach achieves fault isolation by screening out a fault among all the possible fault candidates so that all the abnormal measurements can be explained. Therefore, there is no problem with misdiagnosing one fault as another. However, both PCA based FDI and ANFIS model based FDI with structured residual design do not have a safeguard against the possibility that some unknown faults may have the same fault signatures as defined for a fault in the fault dictionary. Although structured residual design approach is able to avoid misdiagnosing one fault as another for the enumerated faults through manipulating the residual structures, it still cannot fully solve the problem. In addition, structured residual design is not always achievable especially for a non-linear system.

Data driven model causal graph approach is able to meet the requirement for automation because only normal operation data are necessary to adaptively upgrade system models. The fault signatures used for fault isolation are extracted from the understanding about the physical system instead of time consuming simulation or additional experiments.

Causal graph approach allows accurate data driven modeling. The most parsimonious model structure can be obtained through a model causal graph. Therefore, it can improve the accuracy of the developed data driven models significantly. In addition, the model structure is consistent with the system decomposition, so it helps to arrive at a modular FDI system.

Because fault isolation is based on reasoning about model residuals, data driven model diagraph approach is also able to deal with simultaneous faults.

Chapter 7

Picasso User Interface Design

In order to show the effectiveness of the developed FDI for nuclear power plants, graphic user interface software has been successfully developed under the environment of Picasso-3, a user interface management system. The software has the following functions:

- Create a fault by changing the fault characteristic parameters.
- Display essential parameters on the schematic of the reactor system.
- Display the residual patterns specific to the fault.
- Trend the process variables relevant to the fault, and echo the FDI results.

The software has integrated SimPWR, a reactor system analysis code in FORTRAN, and the FDI code in Matlab, and the C++ code to control the graphic user interface. SimPWR code is the driving code in the software, which makes it possible to advance the simulation time without interruption after the data are flushed to the user interface. The Picasso Real Time Manager is controlled by a C++ code. It keeps running in multithread mode while SimPWR is running so that the performance of the user interface display does not degrade due to possible time delay before the C++ code can get data from the simulation code.

7.1 Introduction

It is important to evaluate the overall performance of a newly developed FDI system. Although quite a few FDI methods have been available, all of these methods have their inherent weakness as compared with the others. This is mainly due to the great challenges to the comprehensive requirements of an FDI system such as early detection and diagnosis, isolability, robustness, novelty identification, multiple fault identifiability, explanation facility, adaptability etc. Not a single FDI method is able to have all these desired characteristics.

Developing a graphic user interface (GUI) provides a convenient and cost effective way to make the evaluation. A simulation code can be used to simulate the process behavior under normal and faulty conditions. A fault can be created without much effort by changing some parameters related to the fault. If only those data measured in actual plants are used for FDI implementation, the data can be reliable substitutes to the real data. Some noises can be easily added to the input and the output of the simulation codes in order that the robustness of the FDI method can be tested. With regard to testing the adaptability of the FDI method, the operational power levels can be modified or some disturbances such as steam generator tube fouling factor can be changed on the GUI. In addition, the GUI can also facilitate evaluating the FDI performance in detecting and isolating a single fault or multiple faults during a transient.

7.2 Picasso Development Environment

Picasso-3 is a User interface Management Systems (UIMS) developed as part of the Halden Reactor Project (Kjell, 1992, Jakobsen, 1994, Kjell, 1994(a), Kjell, 1994(b)). Figure 7.1 shows the schematic of the Picasso-3 system

Picasso-3 has three components. Graphics Editor (GED) is the tool to design the user interface. GED can be used to design some user interface components, draw some pictures, set up some dialogues and define dynamic attributes to some user interface components. The User Interface database is where the GED saves the information containing the complete specification of the user interface. Run-time manager (RTM) actually realizes the application's user interface. Application process is the C++ code written by the user to guide how RTM is to generate the user interface as desired by the user. Application Programmer's Interface (API) is a library of C++ functions that is linked to the application process to enable it to communicate with RTM.

When an application is started, the application process calls functions in the API library in order to connect to the RTM. The RTM responds by loading the application user interface from its database and displaying it on the screen. By calling API functions periodically, RTM will continue to handle incoming events generated by the end user or by the processes.

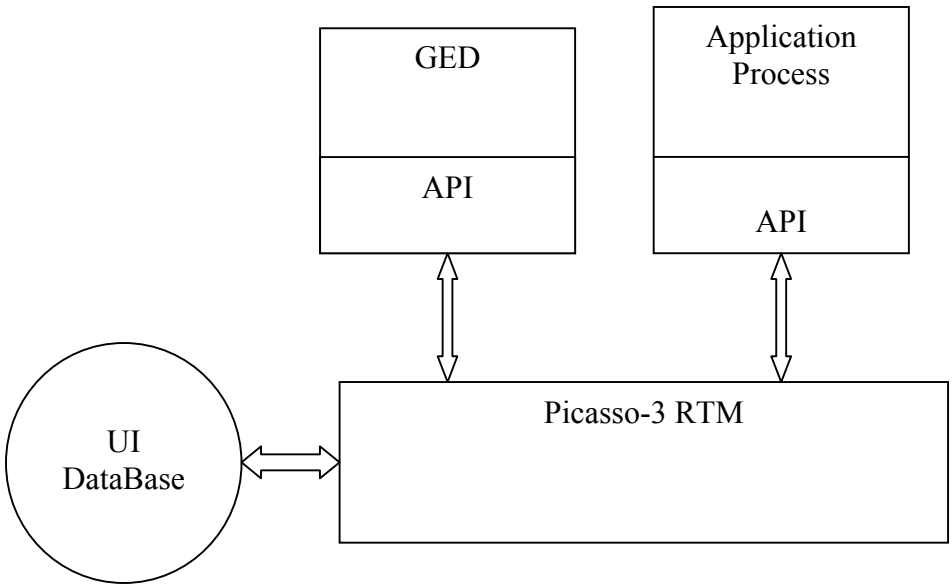

Figure 7.1. Schematic of Picasso-3 system.

7.3 Application Process Design

The schematic of the application process is shown in Figure 7.2. The SimPWR code is the main program. Before entering the main body of the computation, initial_link is called in order to set up a connection with the Picasso RTM. In the meanwhile, some initialization data for SimPWR calculation is transferred to RTM. The main body of SimPWR code is run in a loop. At the end of the loop, the simulation time advances one second. The loop keeps running until the simulation time exceeds the specified maximum time. When SimPWR code steps out of the loop, terminate_link is called in order to end Picasso gracefully.

The application process program is written in multithread operation mode. After it is connected with Picasso RTM, the function process_picasso is called periodically. On the one hand, it flushes the results calculated by SimPWR to Picasso-3 RTM so that they can be displayed on the block diagram of the reactor system and can be trended on trending plots. On the other hand, it detects whether some parameters defined on the screen to create some faults have been changed by the end user. If so, Process_picasso will transfer the changed parameters to SimPWR for a new simulation. If Process_picasso detects the request from the end user for making fault detection and fault diagnosis, it will call the FDI module computing the residuals due to the fault and send the residuals to the RTM for display. The FDI diagnostic results will also be transferred to the FDI diagnostic information window indicating what is the fault according to the FDI algorithm.

The data exchange between Picasso-3 application process and SimPWR simulation code is through global variables. These global variables return a structure in the C++ part of the application process and return a common block in the Fortran part of the application process. The data exchange between the Picasso-3 RTM and the Picasso-3 application process is through process structures and process variables. Both these data exchanges are two-way due to their global attributes.

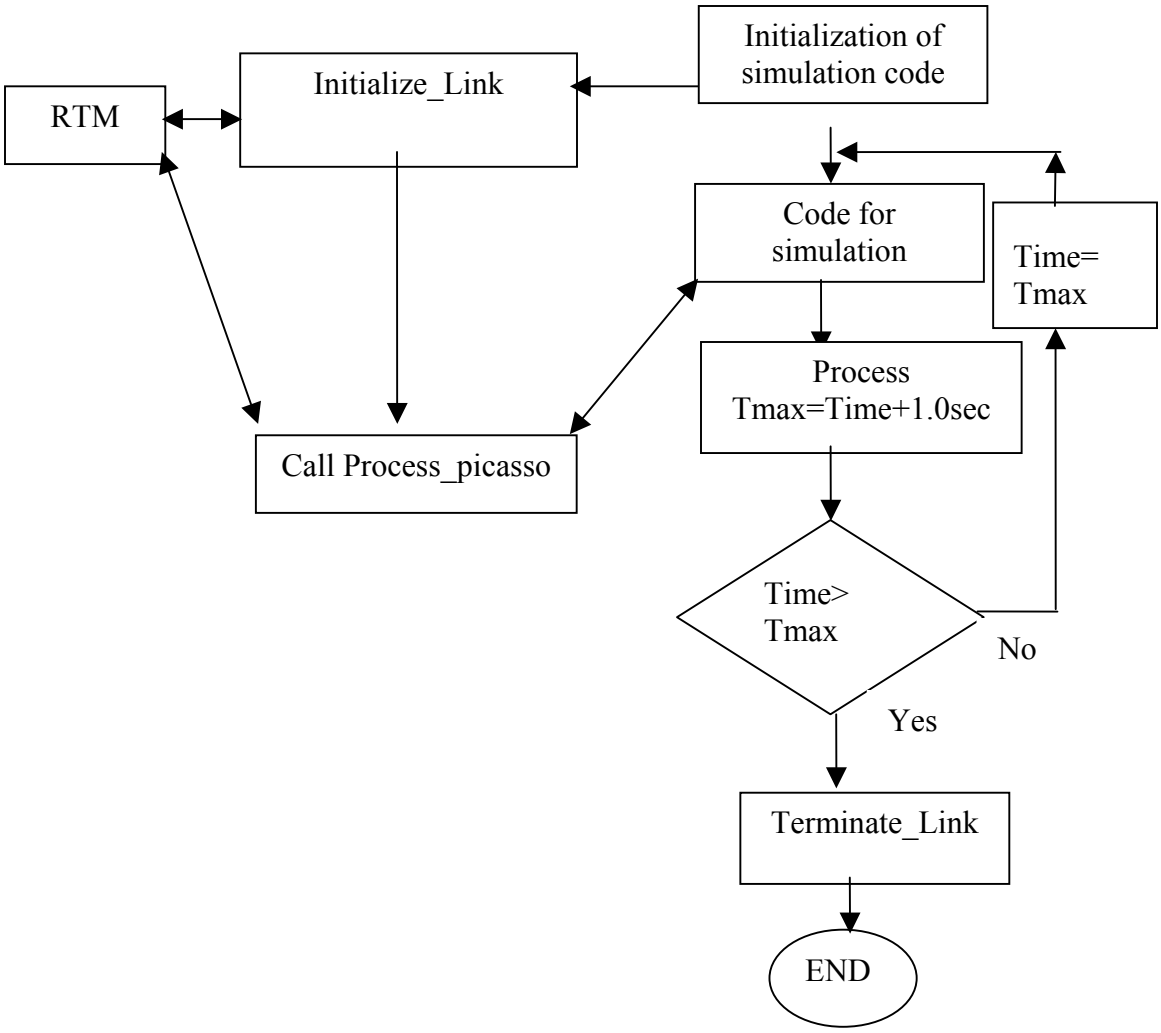

Figure 7.2. The flowchart of Picasso application process.

Two remote functions, stopApplication() and datMount(), are defined. These two remote functions can be called directly by the user interface. The function stopApplication() can help the API code end its task gracefully. The function datMount enables the user to input the samples and sampling interval for SimPWR code to return adequate amount of data for fault diagnosis.

7.4 Descriptions of the Major Functions

A description of the major functions is given in this section.

1) Header

The header files for Picasso API, MFC socket, FDI module, MATLAB as well as the C++ application process itself are included in this part of the code. The header file for the C++ application process declares the function prototypes, structures for sensor characteristics, valve characteristics, controller characteristics, simulation data for display, and residuals. It also defines the global variables or the structures used in the application process. In addition, the global variables used to access the common blocks in the SimPWR FORTRAN code are also declared here.

2) int Initialize_link()

The functions of this function are as follows:

- Initialization of the variables for display on the screen of end user.
- Calling PfInitialize to connect the application process to the RTM.

3) int process_picasso

It is the kernel code to be executed periodically. This code calls PfSend and PfFlush to update the variables in all the user's windows.

The required functions are as follows:

- Calling some functions to transfer data from RTM to SimPWR in order to follow the recent changes in the parameters by end users.
- Sending the most recent SimPWR simulation results to RTM.
- Sending data to the FDI module or extract data from FDI module if FDI is requested.

4) terminate_link

This function calls PfEndLoop to end the picasso application process.

5) int32 createRecords()

It calls PfReadScript to create records and variables according to specification in RecordDefs.pdat.

6) int32 createVariables()

This calls PfCreateVar to create variables locally in API and puts the information into a local buffer to be used by PfFlushCreateVar.

7) void whenRtmConnects()

PfInitialize calls this function. It establishes connections with RTM and calls createRecords, createVariables and registerFunctions to let both RTM and the application process know the declarations of some process variables, structures and functions.

8) int32 registerFunctions()

This calls PfRegisterFunction to register the function stopApplication to terminate the API and the function datMount to receive the user's input of samples and sampling interval from SimPWR code.

9) int32 stopApplication()

This is a function defined in API code but available to RTM as a remote function. Its function is to end the application.

10) int32 datMount()

This is a function defined in API code but available to RTM as a remote function. Its function is to receive the user's input of samples and sampling interval from SimPWR code.

11) void whenRtmDisconnects()

This is a function to give a message if connection has been lost with RTM.

12) RESD class_conversion

This is a function to convert the residual array to the structure type RESD.

13) void Pushdata()

This is a function to convert a double matrix into a mxArray data structure used as input of Matlab function.

14) void Extractdata()

This is a function to convert mxArray data structure used as output of Matlab function to a one-dimensional array.

15) char* faultType()

This is a function to determine the type of faults according to the residuals.

7.5 User Interface Design

The graphic user interface consists of five main windows.

The main window is designed to facilitate switching between functional windows. It provides the following options:

- Switch to the simulation window
- Switch to the trending plot window
- Switch to the FDI diagnostic results window
- Switch to the fault creation window
- End task.

Figure 7.3 shows the options available on the main window of the graphic user interface.

The fault creation window is designed to create faults by changing the parameters of the sensors, controllers, and actuators. The following parameters can be changed on this window:

- FCV1 valve stuck position
- FCV1 offset
- FCV1 time constant
- FCV2 valve stuck position
- FCV2 offset
- FCV2 time constant
- TCV1 valve stuck position
- TCV1 offset
- TCV1 time constant
- TCV2 valve stuck position
- TCV2 offset
- TCV2 time constant
- TCV3 valve stuck position
- TCV3 offset

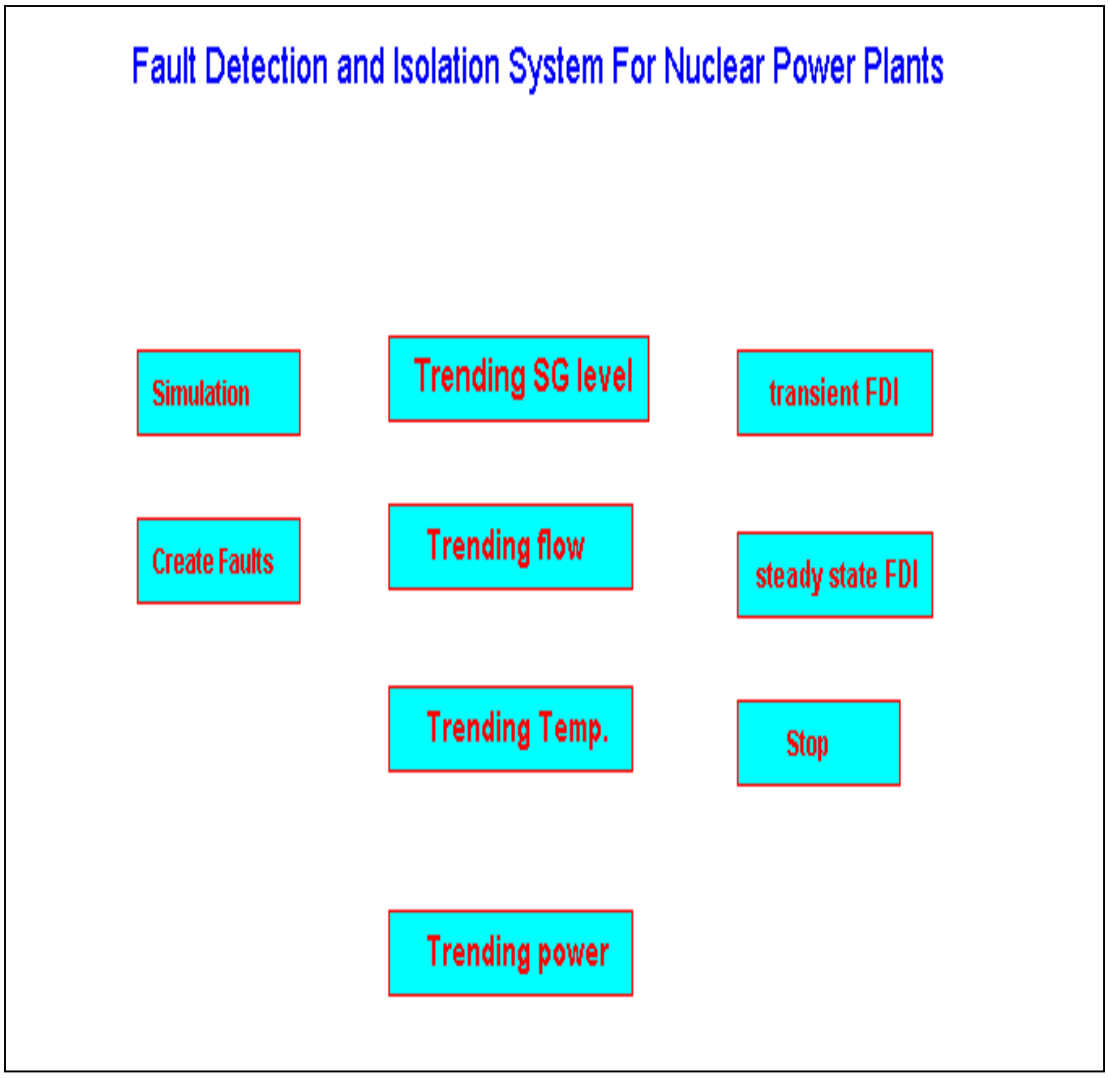

Figure 7.3. The main window of the graphic user interface.

- TCV3 time constant
- TCV4 valve stuck position
- TCV4 offset
- TCV4 time constant
- Reactor power
- FCV1 controller offset
- Proportional gain of FCV1 controller
- Integral gain of FCV1 controller
- FCV2 controller offset
- Proportional gain of FCV2 controller
- Integral gain of FCV2 controller
- SG1 narrow range level sensor drifting rate
- SG2 narrow range level sensor drifting rate
- SG1 flow meter drifting rate
- SG2 flow meter drifting rate

Figure 7.4 shows the fault creation window of choices to change the parameters of sensors, controllers, and actuators.

The trending plot window trends the following plots, which are important to represent the reactor system responses to the created faults:

- Reactor nuclear power
- Reactor power output
- SG 1 water level
- SG 2 water level
- Hot leg temperature
- Cold leg temperature
- Feed water temperature
- Feed water flow rate
- SG 1 steam flow rate
- TCV 1 flow rate
- TCV 2 flow rate
- TCV 3 flow rate
- TCV 4 flow rate.

The fault diagnostic result window shows the residual patterns of the following variables:

- SG1 narrow range water level
- SG2 narrow range water level
- FCV1 flow rate

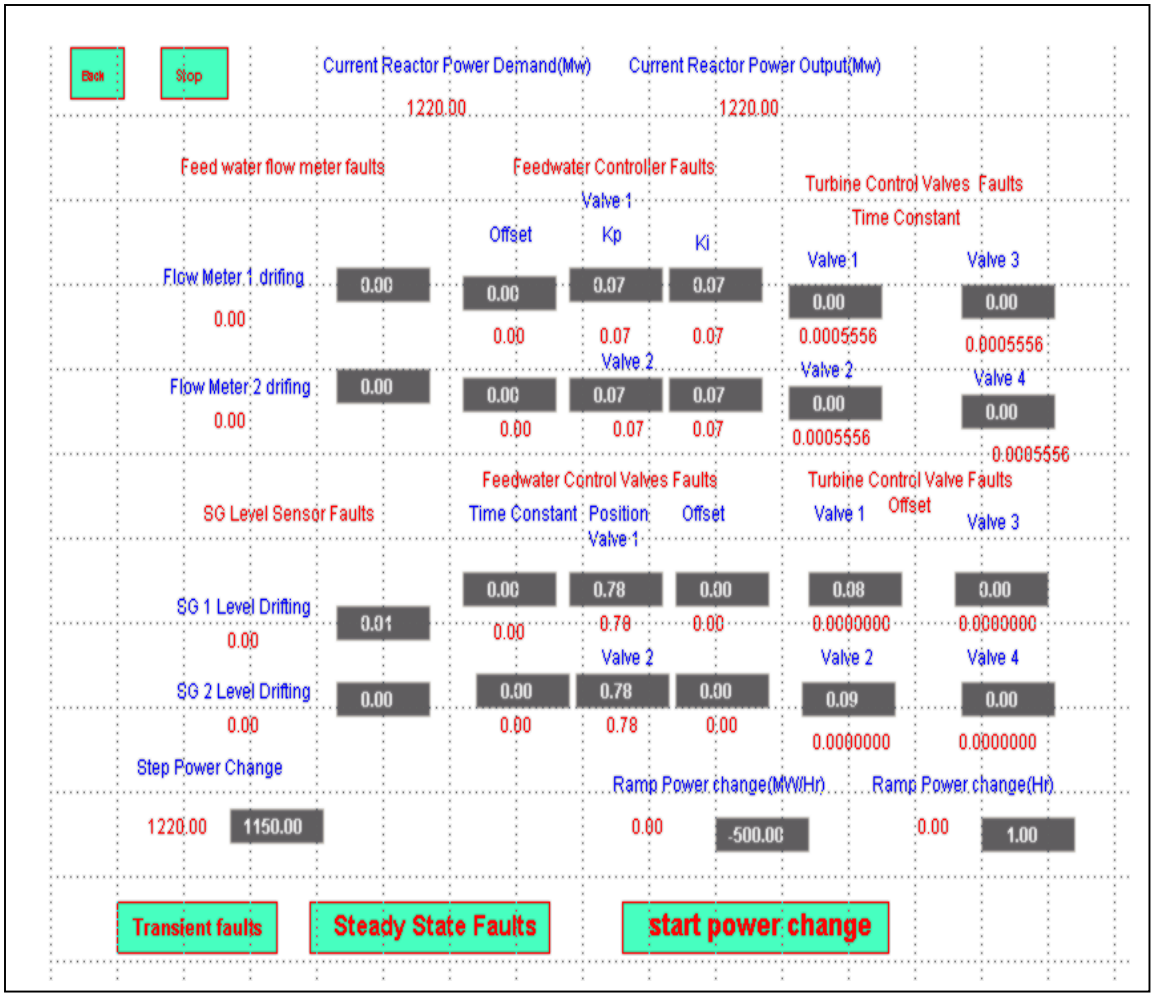

Figure 7.4. Fault creation window to change fault related parameters.

- FCV2 flow rate
- SG 1 steam flow rate
- TCV1 flow rate
- TCV2 flow rate
- TCV3 flow rate
- TCV4 flow rate
- Hot leg temperature
- Cold leg temperature
- FCV1 valve position
- FCV2 valve position
- Feed water temperature (lumped loop)
- Pressurizer temperature
- Pressurizer level.

Figure 7.5 shows the FDI diagnostic window under steady state conditions.

The simulation window shows the following variables on the schematic of the reactor system:

- Reactor nuclear power
- Hot leg temperature
- Cold leg temperature
- Pressure in the pressurizer
- Water level in the pressurizer
- Steam generator water level
- Feed water flow rate to SG1
- Feed water temperature
- Steam flow rate from SG1.

Figure 7.6 shows the simulation window, in which the key parameters of the reactor operation are shown.

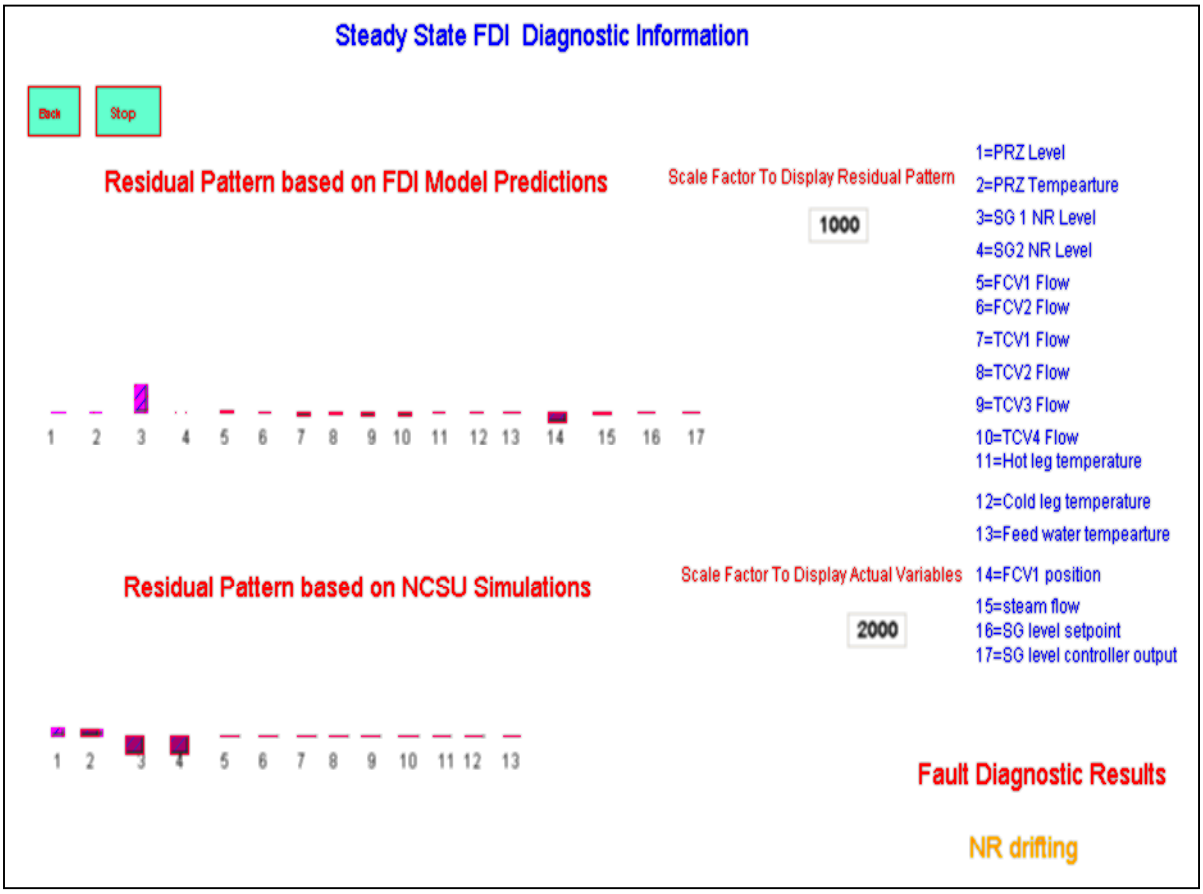

Figure 7.5. Steady State FDI diagnostic window.

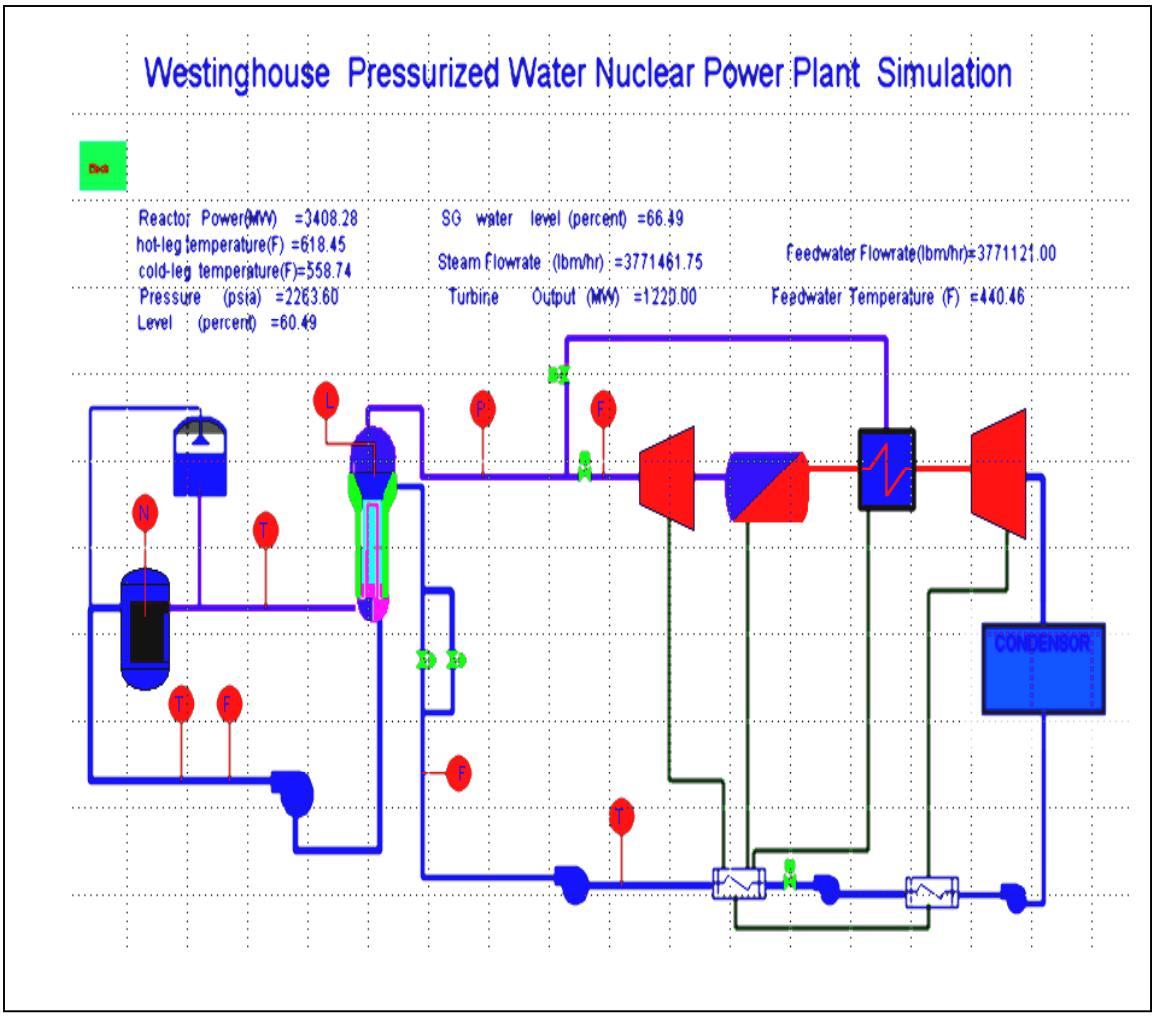

Figure 7.6. FDI simulation window.

Chapter 8

Summary, Conclusions, and Recommendations for Future Work

8.1 Summary

The preceding chapters presented the development of three approaches to fault diagnosis of nuclear power plant sensors and field devices and the applications to a PWR steam generator system.

The application of PCA methods shows that both T^2 statistic and Q statistic need to be used for fault detection in order to achieve low missing detection rate. The fault directions jointly defined in the model space and in the residual space can increase the possibility of fault isolation. This approach requires the least amount of system knowledge. However, the developed fault detection module must be sensitive enough so that the plant measurements can be provided in time to the subsequent fault isolation module to define the fault direction in the model space. In addition, fault isolation cannot be completed until a new steady state condition has been reached after a fault. Because PCA is based on linear projection, this FDI system is only applicable to a linear static system.

ANFIS can be used to learn accurate nonlinear models from plant data. A combination of ANFIS modeling with structured residual design enables us to make fault isolation for a nonlinear system. However, the desired residual structure can only be obtained through derived redundancy relationships. Correspondingly, very complicated model structure may be involved. Moreover, in this approach, it is assumed that the possible faults are known. If some additional faults are to be included, the entire FDI system needs to be redesigned.

Model causal graph is developed as a new approach to FDI for nuclear plant sensors and field devices. The significant feature of causal graph is that model inputs and

model outputs have cause effect relationship. Because model structure is determined by system physics, most parsimonious data driven model structure can then be obtained. The fault isolation is based on cause effect analysis on model residuals, so it is not necessary to predefine possible faults and their fault signatures. Model causal graph approach can also provide diagnosis results with higher confidence because a fault can usually be isolated using several models.

The above three FDI methods are demonstrated with application to PWR UTSG system. In the demonstration, all the selected sensor and actuator faults, including five single faults and eight dual faults, can be successfully detected and isolated.

8.2 Conclusions

The following conclusions are made from the research studies and the results presented in this thesis:

- Analytical redundancy is the basis of modern FDI approaches. It makes it possible to obtain stable fault signatures independent of fault magnitudes and initial operating conditions.
- Data driven models are efficient to characterize the analytical relationship among measured variables. These models can be adaptively upgraded during plant operation.
- The qualification of the data plays a significant role in designing data driven FDI algorithms. Any model extrapolation should be avoided in order to minimize false alarms.
- Quasi-static model contains more information than a static model. FDI algorithm based on quasi-static data enables to achieve earlier fault detection.
- PCA based FDI algorithm has inherent connection with parity space approach. The linear relationship among measured variables implying analytical redundancy can be consistently represented by the eigenvectors corresponding to the trivial components. Any deviation either in the model space or in the residual space will indicate a fault.

The fault direction jointly defined both in the model space and in the residual space provides better performance in fault isolation.

- If the possible faults are known based on engineering judgments, a set of ANFIS models can be built to characterize the relationship between plant measurements. Through appropriate choice of model structures, structured residual design approach can be achieved for fault isolation.
- Data driven model causal graph is a generic approach to fault diagnosis for nuclear power plants. It is able to combine the reasoning capability of qualitative knowledge based approach and the strength in fault resolution of quantitative knowledge based method. Fault detection is fulfilled by monitoring the residual of each model. Fault isolation is achieved by the cause effect analysis on the residuals.
- System decomposition and local residual analysis is not only in full agreement with efficient data driven modeling but also conducive to FDI modularization.
- It is not always possible to distinguish dual faults and one of the element faults. For instance, simultaneous feed water flow meter offset fault and SG narrow range level sensor fault cannot be isolated from feed water flow meter offset fault without using the SG wide range level signal since independent fault signatures are not available. In this case, SG WR level sensor signal must be used such that SG NR level sensor fault can be signified by checking its consistency with SG WR level signal.
- A graphic user interface has been successfully developed to simulate the plant behavior for sensor, actuator and controller faults in nuclear power plants. It provides a convenient environment to demonstrate the performance of any designed FDI algorithms.

8.3 Recommendations for Future Work

Some future work could be launched in order to integrate the proposed FDI algorithm into an engineering instrumentation and control system for nuclear power plants.

a) Development of a unified FDI framework, which is able to deal with system knowledge in different forms.

b) Development of adaptive training algorithm for data driven models.

c) Development of FDI algorithms capable of dealing with sensor faults, actuator faults, controller faults, and process faults simultaneously.

d) Development of automatic causal reasoning algorithm on model residuals.

e) Development of direction based classification algorithm to automate residual analysis.

f) Development of novelty detection based algorithm for fault detection.

In summary, the developed PCA based FDI algorithm and the structured residual design approach to FDI are satisfactory when applied to a PWR steam generator system when the possible faults are known. Data driven model causal graph approach is a more systematic and general approach to fault detection and isolation for a large system where it is difficult to obtain information about the possible faults and their associated fault signatures.

List of References

References

Amsterdam, J. (1992). *Automated qualitative modeling of dynamic physical systems.* Ph.D. Dissertation, MIT.

Albuquerque, J. S., and Biegler L. T. (1996). *Data reconciliation and gross error detection for dynamic systems*, AIChE Journal, Vol. 42, pp. 2841-2856.

Chen, J. and Patton, R. J. (1999). *Robust model based fault diagnosis for dynamic systems*, Kluwer Academic Publisher.

Chow, E. Y. and Willsky A. S. (1984). *Analytical redundancy and the design of robust failure detection systems*, IEEE Trans. on Automatic Control Vol. 29, pp. 603-614.

Doster, M. (2000). *A code for PWR System Thermal Hydraulic Analysis*, North Carolina State University, Personal communication.

Davis, R. (1983). *Diagnosis via causal reasoning: paths of interaction and the locality principle*. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, pp. 88-94.

Dash, S. and Venkatasubramanian, V. (2000). *Challenges in the industrial applications of fault diagnostic systems*. Comput. & Chem. Engng, Vol. 24, pp. 785-791.

Electric Power Research Institute (1994). *EPRI/Utility Program Demonstrates Advanced Nuclear Control System*. Post Office Box 10412, (http://www.nuc.berkeley.edu/thyd/ne161/rtse/softcon.html).

Frank, P. M. (1990). *Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy -- a survey and some new results*. Automatica, Vol. 26, pp. 459-474.

Garcia, F., Izquierdo, J. V., Miguel, L. J. and Peran, J. R. (2000). *Fault diagnostic system using analytical fuzzy redundancy*. Engineering Applications of Artificial Intelligence, Vol. 13, pp. 441-450.

Flury, B. (1989). *Common principal components and related multivariate models*. John Wiley & Sons Publication.

Gertler, J. and Singer, D. (1985). *Augmented models for statistical fault isolation in complex dynamic system*. Proceedings of American Control Conference, pp. 317-322.

Gertler, J. and Li, W. (1999). *Isolation enhanced principal component analysis*. AIChE Journal, Vol. 45, pp. 323-333.

Glockler, O. and Tublett, M. V. (1995). *Application of reactor noise analysis in the Candu reactors of Ontario Hydro*. Progress in Nuclear Energy, Vol. 29, pp. 171-191.

Himmelblau, D. M. (1978). *Fault Detection and Diagnosis in Chemical and Petrochemical Processes*. Elsevier, New York.

Hines, J. W., Wrest, J. D., and Uhrig, R. E. (1997). *Signal Validation using an adaptive neural fuzzy inference system*. August, Nuclear Technology, pp.181-193.

Jones, H. L. (1971). *Failure detection in linear systems*. PhD thesis, Dept. of Aeronautics, MIT, Cambridge, Mass.

Jackson, J. E. and Mudholkar, G. S. (1979). *Control procedures for residuals associated with principal components analysis*. Technometrics, Vol. 21, pp.341-349.

Jackson, J. E. (1991). *A user guide to principal components*. John Wiley & Sons, New York.

Jang, J. R. and Sun, C. (1993). *ANFIS: Adaptive-Network-based Fuzzy Inference Systems*. IEEE, Transactions on Systems, Man, and Cybernetics Vol. 23, pp.665-685.

Jakobsen, O. (1994). *The picasso-3 user interface management system*, the Enlarged Halden Programme Group Meeting, Storefjell, Norway, 8-11 March 1993.

Kaistha N. and Upadhyaya B. R. (2001) *Incipient fault detection and isolation of field devices in nuclear power systems using principal component analysis*, Nuclear Technology, Vol. 136, pp221-230.

Kjell, A. B. (1992). *Picasso: a user interface management system for real time applications*. OECD Halden reactor project report.

Kjell, A. B. (1994(a)). *Developing graphics applications in an interactive environment*. SCS Simulation Multiconference, San Diego, California.

Kjell, A. B. (1994(b)). *Implementation of graphical user interfaces in nuclear applications*. ENS Topical Meeting on I&C of VVER, Prague, Czech Republic.

Kramer, M. A. and Palowitch, Jr., B. L. (1987). *A rule based approach to fault diagnosis using the signed directed graph*. AIChE Journal, Vol. 33, pp. 1067-1078.

Lee, G. (1999). *Multiple fault diagnosis under uncertain conditions by quantification of qualitative relations*. American Chemical Society, Vol. 38, pp. 988-998.

Leyval, L., Gentil, S. and Stephan, F. (1994). *Model based causal reasoning for process supervision*. Automatica, Vol. 30, pp 1295-1306.

Montmain, P. J. and Gentil, S. (2000). *Dynamic causal model diagnostic reasoning for online technical process supervision*. Automatica, Vol. 36, pp.1137-1152.

Mosterman, P. J. and Biswas, G. (1997). *Monitoring, prediction and fault isolation in dynamic physical systems*. Proceeding AAAI 1997, pp. 100-105.

Raich, A. and Cinar, A. (1996). *Statistical process monitoring and disturbance diagnosis in multivariable continuous processes*. AIChE Journal, Vol. 42, pp.995~1009.

Russell, E. L. and Chiang, L. H. (2000). *Data-driven methods for fault detection and diagnosis in chemical processes*. Springer-Verlag Incorporated, New York.

Simani, S. (2000). *Model based fault diagnosis in dynamic systems using identification technique*. PhD dissertation, University of Hull, UK.

Upadhyaya, B. R. (1999). *Incipient fault detection and isolation of sensors and field devices*. Research report, Nuclear Engineering Department, The University Of Tennessee, UTNE/BRU/99-02.

Vedam, H. and Venkatasubramanian, V. (1995). *PCA-SDG based Process Monitoring and Fault Diagnosis*. Applications of Artificial Intelligence, Vol. 8 , pp. 689-701.

White, J. D. (1994). *Comparative assessments of nuclear instrumentation and controls in the U.S., Canada, West Europe, Japan and former Soviet Union, Oak Ridge National Laboratory*. (http://itri.loyola.edu/ar93_94/canic.htm).

Wold, S. (1978). *Cross validation estimation of the number of components in factor and principal component analysis*. Technometrics, Vol. 20, pp. 397-406.

Yoon, S. and MacGregor, J. F. (2001*). Fault diagnosis with multivariate statistical models part I: using steady state fault signatures*. Journal of process control, Vol. 11, pp. 387-400.

Zakarian, A. and Kusiak, A. (2000). *Analysis of Process Models. IEEE Transactions on electronics packaging manufacturing*, Vol .23, pp. 137-147.

APPENDIX

Appendix A Matlab Code for PCA fault detection

```
close all;
clear all;
fnn=cell(11);
fnn{1}='E:\kzhao\SGdataNew\PWRrampNew';
fnn{2}='E:\kzhao\SGdataNew\FeedFlowDrift';
fnn{3}='E:\kzhao\SGdataNew\SteamflDrift';
fnn{4}='E:\kzhao\SGdataNew\SteamFeedDrift';
fnn{5}='E:\kzhao\SGdataNew\FeedSGLevelDrift';
fnn{6}='E:\kzhao\SGdataNew\SteamSGLevelDrift';
fnn{7}='E:\kzhao\SGdataNew\SGPrsDrift';
fnn{8}='E:\kzhao\SGdataNew\FeedSGPrsDrift';
fnn{9}='E:\kzhao\SGdataNew\SGLevelPrsDrift';
fnn{10}='E:\kzhao\SGdataNew\SGLevelDrift';
fnn{11}='E:\kzhao\SGdataNew\SteamSGPrsDrift';
fnn1=cell(11);
fnn1{1}='Normal Operation';
fnn1{2}='Feed Water Flow Meter Drift Fault';
fnn1{3}='Steam Flow Meter Drift fault';
fnn1{4}='Steam Flow Meter Feed Flow Meter Drift Faults';
fnn1{5}='Feed Flow Meter Drift Fault and SG Level Sensor Drift Fault';
fnn1{6}='Steam Flow Meter Drift Fault and SG Level Sensor Drift Fault';
fnn1{7}='SG Pressure Sensor Drift Fault';
fnn1{8}='Feed Water Flow Meter Dridt fault & SG Pressure Sensor Drift Fault';
fnn1{9}='SG Level Sensor Dridt Fault & SG Pressure Sensor Drift Fault';
fnn1{10}='SG Level Sensor Dridt Fault';
fnn1{11}='Steam Flow Meter Drift Fault and SG Pressure Sensor Drift Fault';
mmp=length(fnn);
 index=[1,5,9,24,27,29,31,32,33,36,37,39,40,57,70];
 indp=[2,4,7,8,9,11];
 noise=0.003;
 dataNormal='E:\kzhao\SGdataNew\PWRrampNew.dat';
 temp=dlmread(dataNormal,' ');
 A=temp(:,2:end);
 BTP=[A(1:2:end,index)];
 BTP=ran(BTP,noise);
 X_train=BTP;
 BTP=[A(2:2:end,index)];
 BTP=ran(BTP,noise);
 X_test=BTP;
 X=X_train;
 [n,m]=size(X);
 fprintf(' The training set contains %d observations and %d variables\n', n,m);
 [x,meanx,stdr]=zscore1(X);
```

```
  xtest=zscore1(X_test,meanx,stdr);
%%%%%%%%%%%%%%%%%%%%%%%%%%PCA Model%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  [Eigvec, TP, Eigp, TSQUARE] = PRINCOMP(x);
  ttsum=sum(Eigp);
  dps=Eigp/ttsum;
  figure;
  semilogy(dps);
  axis([1,20,0,1]);
  xlabel(' the order of the PCA model');
  ylabel(' the percentage of the total variance explained');
  a=input('the order of your model\n');
% P is the Loading matrix, m rows, a colums;
  P=Eigvec(:,1:a);
% Eig is the eigenvalues of the covariance matrix which are eqaul to the variance of the transformed variables;
  Eigval=Eigp;
  Eig=diag(Eigp(1:a),0);
  Eig1=diag(Eigp,0);
  T=TP(:,1:a);
  figure;
  plot(TP(:,1),TP(:,2),'r*');
  figure;
  plot(P(:,1),P(:,2),'r*');
  [nn,mm]=size(x);
  sse_test=[];
  sse_train=[];
  for aa=1:1:mm
  P=Eigvec(:,1:aa);
  xtrain_pred=x*P*P';
  ss=(xtrain_pred-x)*(xtrain_pred-x)';
  ssp=trace(ss);
  sse_train=[sse_train,ssp];
  xtest_pred=xtest*P*P';
  ss=(xtest_pred-xtest)*(xtest_pred-xtest)';
  ssp=trace(ss);
  sse_test=[sse_test,ssp];
end;
  figure;
  semilogy(sse_train,'b');
  hold on;
  semilogy(sse_test,'r');
  hold off;
  pause;
  a=input('the order of your model\n');
  P=Eigvec(:,1:a);
  Eig=diag(Eigp(1:a),0);
```

```
 Eig1=diag(Eigp,0);
 mv=length(indp);
 [nn,mm]=size(xtest);
 xtest_pred=xtest*P*P';
 for ipp=1:1:mv
    figure;
    plot(xtest_pred(1:20:nn,indp(ipp)),'b+');
    hold on;
    plot(xtest(1:20:nn,indp(ipp)),'ro');
 end;
%%%%fault detection for PCA based on T square and Q statistics %%%
 [T2lim]=Tlim(confidence,n,a);
% Qlim(squared prediction error) is to measure the total sum of variations in the residual space,
 [Qlim]=QFlim(confidence,a,m,Eigval);
for iclass=1:1:mmp
 filem=fnn{iclass};
 for inn=1:1:1
    if iclass==1
       fileName=filem;
       eval(['load ', fileName]);
    BT=PWRrampNew(:,2:end);
    norm00=PWRrampNew(1,2:end);
    norm100=[];
    for ivv=1:1:100
       norm100=[norm100;norm00];
    end;
    else
    fileName=[filem,num2str(inn)];
    eval(['load ', fileName]);
    BT=Faultdata(:,2:end);
    BT=[BT];
end;
 note=fnn1{iclass};
 BTP=[BT(1:end,index)];
 BT=BTP;
 BT=ran(BT,noise);
 fprintf('Detecting Fault:%s\n',fnn1{iclass});
 [TTSQ,QQSQ,miss,miss1,miss2,fal1,fal2] = dtectPCA(BT,meanx,stdr,Eig,P,T2lim,Qlim);
 figure;
 [nn1,mm1]=size(TTSQ);
 plot(TTSQ(1:1:mm1),'b*');
 hold on;
 TTSQ_lim=ones(1,mm1).*T2lim;
 plot(TTSQ_lim(1:1:mm1),'r')
 note1=['detecting ',note,' based on T square statistics'];
```

```
  xlabel('sample');
  ylabel('T square statistics');
  title(note1);
  hold off;
  figure;
  plot(QQSQ(1:1:mm1),'b*');
  hold on;
  [nn1,mm1]=size(QQSQ);
  QQSQ_lim=ones(1,mm1).*Qlim;
  plot(QQSQ_lim(1:1:mm1),'r');
  note2=['detecting ',note,' based on Q statistics'];
  xlabel('sample');
  ylabel('Q statistics');
  title(note2);
  hold off;
end;
end;
pause;
%%%%%%%%%%%%%%%%%%%%%%%%%%Fault Identification%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  ind01=[];ind02=[];ind03=[];ind04=[];
  ind11=[];ind12=[];ind13=[];ind14=[];
  [mn1,mn2]=size(BT);
  pattern=[];
  for iclass=2:1:mmp
      filem=fnn{iclass};
  for inn=1:1:1
     fileName=[filem,num2str(inn)];
     note=fileName;
     eval(['load ', fileName]);
     BT=Faultdata(:,2:end);
     xid=[BT(end,index)];
     xid=ran(xid,noise);
  xid=(xid-meanx);
  [CONT,RES]=ident1(xid,T2lim,Qlim,a,P,Eigvec,Eig1);
  CONT1=sort(abs(CONT));
  ma=length(CONT);
  RES1=sort(abs(RES));
  mb=length(RES);
  if CONT1(ma)~=0.0 indd1=find(abs(CONT)==CONT1(ma)); end;
  ind01=[ind01,indd1];
  if RES1(mb)~=0.0 indd1=find(abs(RES)==RES1(mb)); end;
  ind02=[ind02,indd1];
  end;
  figure;
  bar(1:1:15,CONT);
```

```
  title(note);
  figure;
  bar(1:1:15,RES);
  title(note);
end;
end;
```

Appendix B Matlab Code for PCA fault isolation

```
close all;
clear all;
ytext1=cell(1);
ytext1{1}='Dynamic linear PCA';
fnn=cell(13);
fnn{1}='E:\kzhao\SGdataNew\FeedFlowoffset';
fnn{2}='E:\kzhao\SGdataNew\Steamfloffset';
fnn{3}='E:\kzhao\SGdataNew\SteamFeedoffset';
fnn{4}='E:\kzhao\SGdataNew\FeedSGLevel';
fnn{5}='E:\kzhao\SGdataNew\SteamSGLevel';
fnn{6}='E:\kzhao\SGdataNew\SGPrs';
fnn{7}='E:\kzhao\SGdataNew\FeedSGPrs';
fnn{8}='E:\kzhao\SGdataNew\FeedFCVOffset';
fnn{9}='E:\kzhao\SGdataNew\SteamFCVOffset';
fnn{10}='E:\kzhao\SGdataNew\FCVOffset';
fnn{11}='E:\kzhao\SGdataNew\SGLevelPrs';
fnn{12}='E:\kzhao\SGdataNew\SGLevel';
fnn{13}='E:\kzhao\SGdataNew\SteamSGPrs';
fnns=cell(13);
fnns{1}='Feed flow meter offset';
fnns{2}='Steam flow meter offset';
fnns{3}='Feed flow meter offset and Steam flow meter offset';
fnns{4}='Feed flow meter offset and SG Level sensor offset';
fnns{5}='Steam flow meter offset and SG Level sensor offset';
fnns{6}='SG pressure sensor offset';
fnns{7}='Feed flow meter offset and SG Pressure sensor fault';
fnns{8}='Feed flow meter offset and FCV Offset';
fnns{9}='Steam flow meter offset and FCV Offset';
fnns{10}='FCV Offset';
fnns{11}='SG Level sensor offset and SG Pressure sensor fault';
fnns{12}='SG Level sensor fault';
fnns{13}='Steam flow meter sensor offset and SG Pressure sensor fault';
mpp=length(fnn);
index=[1,5,9,24,27,29,31,32,33,36,37,39,40,57,70];
noi=[0.002,0.002,0.002,0.002,0.002,0.002,0.002,0.002,0.002,0.002,0.002,0.002,0.002,0.002,0.002];
kkk=length(index);
load E:\kzhao\SGdataNew\PWRrampNew;
B1=PWRrampNew(:,2:end);
BTP=[B1(2:1:end,index),B1(1:end-1,index)];
BTT=BTP(1:2:end,:);
load E:\kzhao\SGdataNew\PWRnormalNew;
B11=PWRnormalNew(:,2:end);
BTV=[B11(2:end,index),B11(1:end-1,index)];
```

```
norm100=BTV(1,:);
B11=PWRnormalNew(:,2:end);
BTV=[B11(2:end,index),B11(1:end-1,index)];
norm80=BTV(223,:);
[x,meanx,stdr]=zscore1(BTT);
[Eigvec, TP, Eigp, TSQUARE] = PRINCOMP(x);
 ttsum=sum(Eigp);
 dps=Eigp/ttsum;
 figure;
 plot(dps);
 xlabel(' the order of the PCA model');
 ylabel(' the percentage of the total variance explained');
 a=input('the order of your model\n');
% P is the Loading matrix, m rows, a colums;
 P=Eigvec(:,1:a);
 Eigval=Eigp;
 Eig=diag(Eigp(1:a),0);
 Eig1=diag(Eigp,0);
% T is the score matrix being the corordinates in the new corordinate system spanned by principal components,
T=TP(:,1:a);
figure;
 plot(TP(:,1),TP(:,2),'r*');
 figure;
 plot(P(:,1),P(:,2),'r*');
Ytrue=BTP(2:2:end,:);
xtest=zscore1(BTP(2:2:end,:),meanx,stdr);
testout=xtest*P*P';
pred=unscore(testout,meanx,stdr);
figure;
DDD=(pred-Ytrue)./Ytrue;
plot(DDD(1:1:1810,:)','ro');
xlabel('sample no.');
ylabel([ytext1{1},'normal operation']);
BT1=[];
for iclass=1:1:mpp
filem=fnn{iclass};
Fault(iclass).RX=[];
Fault(iclass).score=[];
for inn=1:1:7
 fileName=[filem,num2str(inn)];
  eval(['load ', fileName]);
BT1=Faultdata(:,2:end);
for itime=700-1:1:700
%BT1=ran(BT1,noise);
BTT=[BT1(itime,index),BT1(itime-1,index)];
```

```
BV=zscore1(BTT,meanx,stdr);
predict = BV*P*P';
Bnorm=zscore1(norm100,meanx,stdr);
normscore=Bnorm*P*P';
PPP=predict-normscore;
QQQ=BV-predict;
Fault(iclass).Case(inn).RX(itime,:)=QQQ/norm(QQQ);
Fault(iclass).Case(inn).score(itime,:)=PPP/norm(PPP);
end;
end;
end;
for iclass=1:1:mpp
 filem=fnn{iclass};
 Fault(iclass).RX=[];
 Fault(iclass).score=[];
for inn=1:1:7
  fileName=[filem,num2str(inn)];
   eval(['load ', fileName]);
BT1=Faultdata(:,2:end);
[nn,mm]=size(BT1);
%BT1=ran(BT1,noise);
for itime=700-1:1:700
noise=[];
for ivv=1:1:length(index)
noise=[noise,noi(ivv).*rand(nn,1)];
end;
BT11=BT1(itime,index);
BT111=BT11+BT11.*noise(itime,:);
BT22=BT1(itime-1,index);
BT222=BT22+BT22.*noise(itime-1,:);
BT=[BT111,BT222];
BTT=BT;
BV=zscore1(BTT,meanx,stdr);
predict = BV*P*P';
if inn==7
   BVV=norm80;
else
   BVV=norm100;
end;
Bnorm=zscore1(BVV,meanx,stdr);
normscore=Bnorm*P*P';
PPP1=(predict-normscore);
PPP1=PPP1/norm(PPP1);
QQQ1=(BV-predict);
QQQ1=QQQ1/norm(QQQ1);
```

```
for idd=1:1:mpp
A1=Fault(idd).Case(1).score(itime,:);
B1=Fault(idd).Case(1).RX(itime,:);
A11=Fault(idd).Case(4).score(itime,:);
B11=Fault(idd).Case(4).RX(itime,:);
if inn==1|inn==2|inn==3|inn==7
Sdd(iclass).Case(inn).CosTheta1(itime,idd)=A1*PPP1';
Sdd(iclass).Case(inn).CosTheta2(itime,idd)=B1*QQQ1';
elseif inn==4|inn==5|inn==6
Sdd(iclass).Case(inn).CosTheta1(itime,idd)=A11*PPP1';
Sdd(iclass).Case(inn).CosTheta2(itime,idd)=B11*QQQ1';
end;
end;
end;
end;
end;

for itime=700:1:700
for iclass=1:1:mpp
   XXX=[];GGG=[];
   for inn=1:1:7
      VVV1=[];
      VVV2=[];
      for idd=1:mpp
         VVV1=[VVV1,Sdd(iclass).Case(inn).CosTheta2(itime,idd)];
         VVV2=[VVV2,Sdd(iclass).Case(inn).CosTheta1(itime,idd)];
      end;
   XXX=[XXX;VVV1];
   GGG=[GGG;VVV2];
    end;
disp(XXX);
disp(GGG);
fprintf('\n');

AM(itime).YYY1(iclass).XXX=XXX;
AM(itime).YYY2(iclass).GGG=GGG;
end;
end;
for iclass=1:1:mpp
   ZZZ2=[];ZZZ1=[];
   for itime=700:1:700
      for inn=2:1:2
      YYYT=AM(itime).YYY2(iclass).GGG(inn,:);
      YYYB=AM(itime).YYY1(iclass).XXX(inn,:);
      ZZZ1=[ZZZ1;YYYB];
```

```
        ZZZ2=[ZZZ2;YYYT];
        figure;
        bar([ZZZ1;ZZZ2]');colormap(cool);
        title(['residual direction in the model space and the residual space']);
        ylabel(fnns{iclass});
        xlabel('Fault number');
    end;
end;
end;
```

Appendix C Matlab Code for ANFIS model based fault isolation

```
clear all;
close all;
  warning off;
  noise=0.01;
indv=[33,31,29,37,24];
ptest_title=cell(5);
ytext=cell(5);
ytext1=cell(5);
fnn{1}='D:\kzhao\SGdataNew\FeedFlowoffset';
fnn{2}='D:\kzhao\SGdataNew\Steamfloffset';
fnn{3}='D:\kzhao\SGdataNew\SteamFeedoffset';
fnn{4}='D:\kzhao\SGdataNew\SGLevel';
fnn{5}='D:\kzhao\SGdataNew\FeedSGLevel';
fnn{6}='D:\kzhao\SGdataNew\SteamSGLevel';
fnn{7}='D:\kzhao\SGdataNew\SGPrs';
fnn{8}='D:\kzhao\SGdataNew\FeedSGPrs';
fnn{9}='D:\kzhao\SGdataNew\SteamSGPrs';
fnn{10}='D:\kzhao\SGdataNew\SGLevelPrs';
fnn{11}='D:\kzhao\SGdataNew\FeedFCVOffset';
fnn{12}='D:\kzhao\SGdataNew\SteamFCVOffset';
fnn{13}='D:\kzhao\SGdataNew\FCVOffset';

fnns=cell(13);
fnns{1}='D:\kzhao\SGdataNew\FeedFlowoffset';
fnns{2}='D:\kzhao\SGdataNew\Steamfloffset';
fnns{3}='D:\kzhao\SGdataNew\SteamFeedoffset';
fnns{4}='D:\kzhao\SGdataNew\SGLevel';
fnns{5}='D:\kzhao\SGdataNew\FeedSGLevel';
fnns{6}='D:\kzhao\SGdataNew\SteamSGLevel';
fnns{7}='D:\kzhao\SGdataNew\SGPrs';
fnns{8}='D:\kzhao\SGdataNew\FeedSGPrs';
fnns{9}='D:\kzhao\SGdataNew\SteamSGPrs';
fnns{10}='D:\kzhao\SGdataNew\SGLevelPrs';
fnns{11}='D:\kzhao\SGdataNew\FeedFCVOffset';
fnns{12}='D:\kzhao\SGdataNew\SteamFCVOffset';
fnns{13}='D:\kzhao\SGdataNew\FCVOffset';

ptest_title{1}='ANFIS model to estimate FCV valve position';
ptest_title{2}='ANFIS model to estimate FCV flow rate';
ptest_title{3}='ANFIS model to estimate SG steam flow rate';
ptest_title{4}='ANFIS model to estimate SG level';
ptest_title{5}='ANFIS model to estimate SG Pressure';

ytext{1}='FCV valve position(%)';
ytext{2}='FCV flow rate(%)';
ytext{3}='SG steam flow rate(%)';
ytext{4}='SG level(%)';
ytext{5}='SG pressure(%)';
BT1=[];
for inn=1:1:7
for iclass=1:1:length(fnn)
filem=fnn{iclass};
fileName=[filem,num2str(inn)];
eval(['load ', fileName]);
BT1=Faultdata;
[nns,mms]=size(BT1);
BBTT=BT1(:,2:mms);
BT1=BT1(:,2:mms);
[nnn,mmm]=size(BT1);
BT1=BT1+0.001.*rand(nnn,mmm).*BT1;
BBTT=BBTT+0.001.*rand(nnn,mmm).*BBTT;

XYY=[];
for i=nns:1:nns
```

```
indexp1=[33,32];
indexp2=[33];
BTT=BT1(i,indexp1);
load ValPos meanx stdx gfis2;
BS1=zscore2(BTT,meanx,stdx);
Valve_position=evalfis(BS1,gfis2);

indexp1=[70,33];
indexp2=[31];
BTT=BT1(i-1,indexp1);
load FCVFlow meanx stdx gfis2;
BS1=zscore2(BTT,meanx,stdx);
FCV_Flowrate=evalfis(BS1,gfis2);

indexp1=[27,70,5,9];
indexp2=[29];
BTT=[BBTT(i-1,[27,70]),BBTT(i,9)-BBTT(i-1,5)];
load SGSteamFlow meanx stdx gfis2;
BS1=zscore2(BTT,meanx,stdx);
Steam_Flowrate=evalfis(BS1,gfis2);



indexp1=[36,27];
indexp2=[37];
BTT=[BT1(i,indexp1)];
load SGLevel meanx stdx gfis2;
BS1=zscore2(BTT,meanx,stdx);
SG_Level=evalfis(BS1,gfis2);


indexp1=[70];
indexp2=[24];
BTT=BT1(i,indexp1);
load SGPressure meanx stdx gfis2;
BS1=zscore2(BTT,meanx,stdx);
SG_Prs=evalfis(BS1,gfis2);
XYY=[XYY;[Valve_position,FCV_Flowrate,Steam_Flowrate,SG_Level,SG_Prs]];

end;

for ipp=1:1:length(indv)
  if ipp==1
   bbb=XYY(:,ipp)-(BBTT(end,indv(ipp))-BBTT(end-1,indv(ipp)));
   if bbb >= 0.05
      bbb=0.05;
   elseif bbb <= -0.05
      bbb=-0.05;
   end;
   Fault(iclass).Variable(ipp).Residual=bbb;
  else
  bbb=(XYY(:,ipp)-BBTT(end,indv(ipp)))./BBTT(end,indv(ipp));
     if bbb >= 0.05
      bbb=0.05;
   elseif bbb <= -0.05
      bbb=-0.05;
   end;
  Fault(iclass).Variable(ipp).Residual=bbb;
  end;
end;
end;

figure;
X=[1:1:length(fnn)];
Y=[];
for iclass=1:1:length(fnn)
  YT=[];
```

```
  for ivar=1:1:length(indv)
      YT=[YT,Fault(iclass).Variable(ivar).Residual(end,1)];
  end;
  Y=[Y;YT];
end;
BAR(Y);
ser=num2str(inn);
title(['Residual Patterns based on ANFIS Local Model Fault Magnitude=Case',ser]);
xlabel('Fault Class(FCV position,FCV Flowrate,Steam Flowrate,SG Level,SG Pressure)');
ylabel('Residual');
end;
```

Appendix D C++ Code for user interface

```
#ifndef MLF_V2
#define MLF_V2 1
#endif
#include "libmatlb.h"
#include "mex.h"
#include "matrix.h"
#include "fault_det_all.h"
#include "convert.h"
#include "convert2.h"
#include "ANNt_PBK.h"
#include "ANNt_PBK1.h"
#include "S1zscore.h"
#include "S2zscore.h"
#include "fun3.h"
#include "fun4.h"
#include "MyBinaryGenerator.h"
#include "libmmfile.h"
#define BOOL_IS_KEYWORD 1
#include <fstream>
#include <afxsock.h>                    // MFC socket extensions
#include <process.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define  BUFSIZE 256
#include <api/api.h>
#include <api/apiAnacronisms.h>
#include "SimulatorInterface.h"



// Simulator Common Blocks

extern "C" struct
{
   float time, Seconds, deltat;
} TIME;

extern "C" struct
{
              int Ntime;
              float Tmax, Dtmin, Dtmax;
              int ITYPEacc;
              int nValvesFailedOpen;
} SIMCONTROL;


extern "C" struct
{
 float aload, bload, Duration;
} BOPLOAD;


extern "C" struct
{
  float VTBV[10], VSDV[10], VADV1, VSRV1[10], VADV2, VSRV2[10],
                VSLbrk, VTCV[4], FlowTBV[10], FlowSDV[10], FlowADV1,
                          FlowSRV1, FlowADV2, FlowSRV2, FlowSLbrk, FlowSL1,
                          FlowSL2, FlowTCV[4], FlowSG1, FlowSG2, hSG1, hSG2;
} BOPFLOW;

extern "C" struct
{
 float DDDtime,DDDQthnew,DDDQrxnew,DDDQtrans,DDDFlowc,
```

```
    DDDFlow1,DDDFlow2,DDDTrxnew,DDDTfuelHot1,DDDTcladMAX,
    DDDTinfnew,DDDTAVEInd,DDDTaveREF,DDDTold16,DDDTold14,DDDTCL1Ind,
    DDDTold18,DDDTCL2Ind,DDDTold5,DDDTHL1Ind,DDDTHL2Ind,DDDTsat,
    DDDMDNBR,DDDWithDrawal,DDDCb,
    DDDRhoTot,DDDDeltaRhoFuel,DDDDeltaRhoMod,DDDDeltaRhoB,DDDRhocr,
        DDDWturb;
} DDD;


extern "C" struct
{
float PPPSeconds,PPPPp,PPPPprz,PPPQhtrKw,PPPGPMspray,
    PPPPrzLvlP,PPPRefLvlP,PPPGPMcharge,PPPGPMletdwn,PPPFlowSIS,
    PPPmporv,PPPFlowPrzSRV,PPPGPMmsrg;
} PPP;

extern "C" struct
{
float SSSSeconds,SSSPs10,SSSPs10Ind,SSSPs20,SSSPs20Ind,
    SSSTsat10,SSSTsat20,SSSFeedTemp,SSSFlowSG1,
    SSSFlowSG1Ind,SSSFlowfd1,SSSFlowfd1Ind,SSSFCV1P,
    SSSFlowEFW1,SSSSGLvl1WR,SSSSGLvl1NR,SSSSGLvl1WRInd,
    SSSSGLvl1NRInd,SSSSG1Mass,SSSSGRefWR,SSSSGRefNR,SSSFlowSG2,
    SSSFlowSG2Ind,SSSFlowfd2,SSSFlowfd2Ind,SSSFCV2P,SSSFlowEFW2,
    SSSSGLvl2WR,SSSSGLvl2NR,SSSSGLvl2WRInd,SSSSGLvl2NRInd,SSSSG2Mass,
    SSSFlowSLbrk;
} SSS;

extern "C" struct
{
float BBBSeconds,BBBFlowTCV1,BBBFlowTCV2,BBBFlowTCV3,BBBFlowTCV4,
    BBBTCVposP1,BBBTCVposP2,BBBTCVposP3,BBBTCVposP4;
} BBB;

extern "C" struct
{
float DeadBand[10][50],Tau[10][50];
} VALVEPROPERTIES;

extern "C" struct
{
        float SensorOffset[20],SensorDrift[20],SensorNoise[20],SensorSpan[20];
} SENSORPROPERTIES;

extern "C" struct
{
float FeedGain[4][2],FeedGainTrip[4],EFWGain[4],G1Feed[2],G2Feed[2];

} FEEDCONTROL;

extern "C" struct
{

 float SGLvl1,SG1IndWRLvl,SG1IndNRLvl,SGLvl2,SG2IndWRLvl,SG2IndNRLvl,
         Ps10,Ps20,hfd1,hfd2,Flowfd1,Flowfd2;
} SGINIT;


extern "C" struct
{
        float Qrx, Trx, RodDepth;
} COREINIT;


extern "C" struct
{
        float   Wturb, Wload, Pcond, Phdr, TCVposition[4], FCV1,
                        FCV2, ADV1Position, ADV2Position, TBVposition;
```

```
} BOPINIT;


extern "C" struct
{
        float   DeltaPcp1, DeltaPcp2;
        int             nfeedpumps;
        float   DeltaPEFW, FeedTempRate, FeedDuration, FeedTempData[100], FeedTempTime[100];
        int             nTimeFeed;
} BOPFEED;


extern "C" struct
{
        float Told[30];
} PRIMTEMPS;
extern "C" struct
{
        float Pprz,PrzLvlP;
} PRZADD;

extern "C" struct
{
        float Qthnew,FeedTemp,SGLvl1NR,SGLvl2NR,SGRefNR;
} TREND;


  static int threadRunning = 0;
  static void GUIThread(void *ptr)
{
  threadRunning = 1;
  PfMainLoop();
  threadRunning = 0;
  return;
}

extern "C" Code();

extern "C"
void whenRtmDisconnects(int status, const char *msg)
{
 printf( "Lost contact with RTM %s\n",applName);

 return;
}
extern "C"
void whenRtmConnects(int status, const char *msg)
{
   if ( status & PfCrtmResume )
   {
    printf("Connection established\n");
        return;
        }


/*Create records, variables and functions */
   if (createRecords() !=OK) quit();
   if (createVariables() !=OK) quit();
   if (registerFunctions() !=OK) quit();

   PfFlush();

/*Create processhandler to be called every time interval if not already created*/

 return ;
}

extern "C"
```

```
int process(int i)
{
SIMCONTROL.Tmax=TIME.time+1.0/3600;
  return OK;
}

extern "C"
int initialize_link()
{
   unsigned long guiThread;
            int error = 0;
            Vardd=0;
   SIMCONTROL.Tmax=(float)(0.1/3600);
// Calls made in the initialize_link function
            printf("\n\n");
   PfInitialize("NERI","NERI","rtm",NULL,0,0,whenRtmConnects,whenRtmDisconnects);

/*Create processhandler to be called every time interval if not already created*/

   if ( processHandlerId == PfBADINDEX)
            { PfSetProcessHandler(process_picasso, 5000);
      if ( apiError !=OK)printf("PfsetProcessHandler failed\n");
                        else
                                    printf("PfsetProcessHandler OK \n");
            }
   guiThread=_beginthread(GUIThread, 0, NULL);
   if (apiError != OK)
   {
                printf("Pfinitialize failed\n");
            }
            else
   {
      initialControllerData();
      initialsensorData();
      initialFCVData();
      initialTCVData();
      initPower();
     return(0);
  PfFlush();
            }
            TRACE( "initialize_link: finished\n" );
            return error;
}

extern "C"
int process_picasso(int i)
{

  int error=0;
  int numcc;
  int inum,ierror,ii,jj;
  float dusy;
  char* FaultTemp;
  FILE * fid50;
  FILE * fid20;
  double *VV;
  double VV1[39];
  float rsd[38]={0.0};
  if (PfIsConnected())
  {
            if  (Vardd == 1)
  {
   VV=getData();
            VV1[0]=Vardd;
            for (int ikk=0;ikk<38;++ikk)
            {
                        VV1[ikk+1]=*(VV+ikk);
            }
```

```
            PushData(1,39,VV1);
            YY = (mxArray *)mlfFault_det_all(XX);
            fid50=fopen("Gmdh_residual.dat","r");
  inum=0;
  do
            {
  ierror=fscanf(fid50,"%g",&dusy);
            if (ierror==EOF) break;
  rsd[inum]=dusy;
            printf(" rsd = %g\n", rsd[inum]);
  inum++;
            }
  while (ierror!=EOF);
  fclose(fid50);

  reds=Class_conversion(rsd);
  ExtractData(YY);
  FaultTemp=FaultType();
  for (numcc=0; *(FaultTemp+numcc) != NULL; ++numcc)
                        FaultEcho[numcc]=*(FaultTemp+numcc);
  Vardd=0;
            }

            if  (Vardd == 2)
 {
  VV=getData();
            VV1[0]=Vardd;
            for (int ikk=0;ikk<38;++ikk)
            {
                        VV1[ikk+1]=*(VV+ikk);
            }
            PushData(1,39,VV1);

            YY1 = (mxArray *)mlfFault_det_all(XX);
  ExtractData(YY1);
  sse_trans=diags[0];
            Vardd=2;
            }
            if  (Vardd == 3)
 {
            PowerChange();
            Vardd=0;
            }
  if (PPP.PPPSeconds<5.0)
            {
            INITFDIData();
            };
   if (PPP.PPPSeconds>5.0)
            {
            FDIFData();
            };
            simulationData();
    FCVData();
    TCVData();
  ControllerData();
  sensorData();

myGlobalTime=PPP.PPPSeconds;
PfSend (controller1_id);
PfSend (controller2_id);
PfSend (FCV1_id);
PfSend (FCV2_id);
PfSend (TCV1_id);
PfSend (TCV2_id);
PfSend (TCV3_id);
PfSend (TCV4_id);
PfSend (SG1_id);
PfSend (SG2_id);
```

```
 PfSend (Flowmeter1_id);
 PfSend (Flowmeter2_id);
 PfSend (simulation_id);
 PfSend (fdi_id);
 PfSend (reds_id);
 PfSend (Var_Id);
 PfSend(a_id);
 PfSend(b_id);
 PfSend(c_id);
 PfSend(d_id);
 PfSend(e11_id);
 PfSend(e12_id);
 PfSend(e13_id);
 PfSend(e14_id);
 PfSend(f11_id);
 PfSend(f12_id);
 PfSend(g11_id);
 PfSend(g12_id);
 PfSend(g13_id);
 PfSend(g14_id);
 PfSend(time_id );
 PfSend(diags_id);
 PfSend(FaultEcho_id);
 PfSend(sse_id);
 PfSend(G_Perload_id);
 PfSend(G_aload_id);
 PfSend(G_bload_id);
 PfSend(G_PerloadMin_id);
 PfSend(G_Refload_id);
 PfSend(G_Duration_id);

 PfFlush();
 printf ( "Transferring Data  to RTM \n");
}

return OK;
}

extern "C"
int terminate_link()
{
int error=0;
if (threadRunning)
{
 PfEndLoop();
 while (threadRunning) { Sleep(1000); }
}
return error;
}

// Send message to log.

extern "C"
int send_log_message(const char *pMsg)
{
        int error = 0;
        return error;
}

extern "C"
float get_sim_time()
{
        return 0.0;
}

/* Standard error macro for reporting API errors */

 #define PERR(bSuccess, api){if(!(bSuccess)) printf("%s:Error %d from %s \
```

```
   on line %d\n", __FILE__, GetLastError(), api, __LINE__);}
extern "C"
void clsgp( )
{
            HANDLE hConsole;
            hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

   COORD coordScreen = { 0, 0 };    /* here's where we'll home the
                                       cursor */
   BOOL bSuccess;
   DWORD cCharsWritten;
   CONSOLE_SCREEN_BUFFER_INFO csbi; /* to get buffer info */
   DWORD dwConSize;                 /* number of character cells in
                                       the current buffer */

   /* get the number of character cells in the current buffer */

   bSuccess = GetConsoleScreenBufferInfo( hConsole, &csbi );
   PERR( bSuccess, "GetConsoleScreenBufferInfo" );
   dwConSize = csbi.dwSize.X * csbi.dwSize.Y;

   /* fill the entire screen with blanks */

   bSuccess = FillConsoleOutputCharacter( hConsole, (TCHAR) ' ',
   dwConSize, coordScreen, &cCharsWritten );
   PERR( bSuccess, "FillConsoleOutputCharacter" );

   /* get the current text attribute */

   bSuccess = GetConsoleScreenBufferInfo( hConsole, &csbi );
   PERR( bSuccess, "ConsoleScreenBufferInfo" );

   /* now set the buffer's attributes accordingly */

   bSuccess = FillConsoleOutputAttribute( hConsole, csbi.wAttributes,
   dwConSize, coordScreen, &cCharsWritten );
   //PERR( bSuccess, "FillConsoleOutputAttribute" );

   /* put the cursor at (0, 0) */

   bSuccess = SetConsoleCursorPosition( hConsole, coordScreen );
   PERR( bSuccess, "SetConsoleCursorPosition" );
   return;
}

extern "C"
void quit()
{
PfEndLoop();
printf ("Picasso is done\n");
SIMCONTROL.Tmax=TIME.time-1.0/3600;
return;
}

extern "C"
int32 KbdHandler(int32 handlerId)
{
char c;
scanf("%c",&c);
if (c=='q' || c=='Q')
quit();
return OK;
}

extern "C"
    int32 registerFunctions()
{
    PfTArg formals[2];
```

```
/*Register a function for terminating the program*/
   PfRegisterFunction("stopApplication",stopApplication,0,NULL);
   if (apiError !=OK)
   printf ("PfRegisFunction failed (%s)\n", applName);
/*Register a function for receiving data points calculated by NCSU for diagnosis*/
   formals[0].dtype=PfCInt;
   formals[0].size=1;
   formals[1].dtype=PfCFloat;
   formals[1].size=1;
   PfRegisterFunction("datMount",datMount,2,formals);
   if (apiError !=OK)
   printf ("PfRegisFunction failed (%s)\n", applName);
   return OK;
}



extern "C"
  int32 createRecords()
{
 int32 numErrors;
 char *Filename="RecordDefs.pdat";
 numErrors=PfReadScript(Filename);//PfReadScript creates records and variables according to specification in NERI.Pdat
 if (apiError!=OK)
 printf ( "pfReadScript reported  errors for %s\n",Filename);
 else
 printf ( "pfReadScript is done\n");
 return apiError;
}

extern "C"
   int32 createVariables()
{
/*PfCreateVar creates the variable locally in api and puts the information into a local buffer
to be used by PfFlushCreateVar.*/

a_id= PfCreateVar("a1",PfCDouble, NULL,0,&a1);
 if(apiError == OK)
 printf("a1 added\n");
 else
 printf("Adding a1 failed\n");
  b_id= PfCreateVar("b1",PfCDouble, NULL,0,&b1);
 if(apiError == OK)
 printf("b1 added\n");
 else
 printf("Adding b1 failed\n");
 c_id= PfCreateVar("c1",PfCDouble, NULL,0,&c1);
 if(apiError == OK)
 printf("c1 added\n");
 else
 printf("Adding c1 failed\n");
 d_id= PfCreateVar("d1",PfCDouble, NULL,0,&d1);
 if(apiError == OK)
 printf("d1 added\n");
 else
 printf("Adding d1 failed\n");
  e11_id= PfCreateVar("e11",PfCDouble, NULL,0,&e11);
 if(apiError == OK)
 printf("e11 added\n");
 else
 printf("Adding e11failed\n");
 e12_id= PfCreateVar("e12",PfCDouble, NULL,0,&e12);
 if(apiError == OK)
 printf("e12 added\n");
 else
 printf("Adding e12 failed\n");
 e13_id= PfCreateVar("e13",PfCDouble, NULL,0,&e13);
 if(apiError == OK)
```

```
printf("e13 added\n");
else
printf("Adding e13 failed\n");
e14_id= PfCreateVar("e14",PfCDouble, NULL,0,&e14);
if(apiError == OK)
printf("e14 added\n");
else
printf("Adding e14 failed\n");
f11_id= PfCreateVar("f11",PfCDouble, NULL,0,&f11);
if(apiError == OK)
printf("f11 added\n");
else
printf("Adding f11failed\n");
f12_id= PfCreateVar("f12",PfCDouble, NULL,0,&f12);
if(apiError == OK)
printf("f12 added\n");
else
printf("Adding f12 failed\n");
g11_id= PfCreateVar("g11",PfCDouble, NULL,0,&g11);
if(apiError == OK)
printf("g11 added\n");
else
printf("Adding e11failed\n");
g12_id= PfCreateVar("g12",PfCDouble, NULL,0,&g12);
if(apiError == OK)
printf("g12 added\n");
else
printf("Adding g12 failed\n");
g13_id= PfCreateVar("g13",PfCDouble, NULL,0,&g13);
if(apiError == OK)
printf("g13 added\n");
else
printf("Adding g13 failed\n");
g14_id= PfCreateVar("g14",PfCDouble, NULL,0,&g14);
if(apiError == OK)
printf("g14 added\n");
else
printf("Adding g14 failed\n");
sse_id= PfCreateVar("sse_trans",PfCDouble, NULL,0,&sse_trans);
if(apiError == OK)
printf("sse_trans added\n");
else
printf("Adding sse_trans failed\n");

G_Perload_id= PfCreateVar("G_Perload",PfCDouble, NULL,0,&G_Perload);
if(apiError == OK)
printf("Perload added\n");
else
printf("Adding Perload failed\n");

G_aload_id= PfCreateVar("G_aload",PfCDouble, NULL,0,&G_aload);
if(apiError == OK)
printf("aload added\n");
else
printf("Adding aload failed\n");

G_bload_id= PfCreateVar("G_bload",PfCDouble, NULL,0,&G_bload);
if(apiError == OK)
printf("bload added\n");
else
printf("Adding bload failed\n");

G_Refload_id= PfCreateVar("G_Refload",PfCDouble, NULL,0,&G_Refload);
if(apiError == OK)
printf("Reference Load added\n");
else
printf("Adding Refload failed\n");
```

```
 G_PerloadMin_id= PfCreateVar("G_PerloadMin",PfCDouble, NULL,0,&G_PerloadMin);
 if(apiError == OK)
 printf("PerloadMin added\n");
 else
 printf("Adding PerloadMin failed\n");

 G_Duration_id= PfCreateVar("G_Duration",PfCDouble, NULL,0,&G_Duration);
 if(apiError == OK)
 printf("Duration added\n");
 else
 printf("Adding Duration failed\n");


 Var_Id= PfCreateVar("Vardd",PfCInt, NULL,0,&Vardd);
 if(apiError == OK)
 printf("Vardd added\n");
 else
 printf("Adding Vardd failed\n");

controller1_id=PfCreateVar("controller1",PfCRecord, "Controller",0,&controller1);
 if(apiError == OK)
 printf("controller1 added\n");
 else
printf("adding controller1 failed\n");
controller2_id=PfCreateVar("controller2",PfCRecord, "Controller",0,&controller2);
 if(apiError == OK)
 printf("controller2 added\n");
 else
 printf("adding controller2 failed\n");
 FCV1_id=PfCreateVar("FCV1",PfCRecord, "Valve",0,&FCV1);
 if(apiError == OK)
 printf("FCV1 added\n");
 else
printf("Adding FCV1 failed\n");
FCV2_id=PfCreateVar("FCV2",PfCRecord, "Valve",0,&FCV2);
 if(apiError == OK)
 printf("FCV2 added\n");
 else
printf("Adding FCV2 failed\n");
TCV1_id= PfCreateVar("TCV1",PfCRecord, "Valve",0,&TCV1);
 if(apiError == OK)
 printf("TCV1 added\n");
 else
printf("Adding TCV1 failed\n");
TCV2_id= PfCreateVar("TCV2",PfCRecord, "Valve",0,&TCV2);
 if(apiError == OK)
 printf("TCV2 added\n");
 else
 printf("Adding TCV2 failed\n");
TCV3_id= PfCreateVar("TCV3",PfCRecord, "Valve",0,&TCV3);
 if(apiError == OK)
 printf("TCV3 added\n");
 else
 printf("Adding TCV3 failed\n");
TCV4_id= PfCreateVar("TCV4",PfCRecord, "Valve",0,&TCV4);
 if(apiError == OK)
 printf("TCV4 added\n");
 else
printf("Adding TCV4 failed\n");
SG1_id= PfCreateVar("SG1",PfCRecord, "Sensor",0,&SG1);
 if(apiError == OK)
 printf("SG1 added\n");
SG2_id= PfCreateVar("SG2",PfCRecord, "Sensor",0,&SG2);
 if(apiError == OK)
 printf("SG2 added\n");
 else
 printf("Adding SG2 failed\n");
Flowmeter1_id= PfCreateVar("Flowmeter1",PfCRecord, "Sensor",0,&Flowmeter1);
```

```
 if(apiError == OK)
 printf("Flow1 added\n");
 else
 printf("Adding Flow1 failed\n");

Flowmeter2_id= PfCreateVar("Flowmeter2",PfCRecord, "Sensor",0,&Flowmeter2);
 if(apiError == OK)
 printf("Flow2 added\n");
 else
 printf("Adding Flow2 failed\n");
simulation_id= PfCreateVar("simulation",PfCRecord, "Simulation",0,&simulation);
 if(apiError == OK)
 printf("simulation added\n");
 else
 printf("adding simulation failed\n");
fdi_id= PfCreateVar("fdi",PfCRecord, "FDIF",0,&fdi);
 if(apiError == OK)
 printf("fdi added\n");
 else
printf("Adding FDIF failed\n");
reds_id= PfCreateVar("reds",PfCRecord, "RESD",0,&reds);
 if(apiError == OK)
 printf("Residual added\n");
 else
printf("Adding Residual failed\n");

diags_id= PfCreateArray("diags",PfCDouble, 20,NULL,true,diags);
 if(apiError == OK)
 printf("diagnostic information added\n");
 else
printf("Adding diagnostic information failed\n");

FaultEcho_id= PfCreateArray("FaultEcho",PfCUnsignedChar, 30,NULL,true,FaultEcho);
 if(apiError == OK)
 printf("Fault Echo information added\n");
 else
printf("Adding Fault Echo information failed\n");

time_id = PfCreateVar("myGlobalTime", PfCInt, NULL, 1, &myGlobalTime);
 if(apiError == OK)
    printf("variable myGlobalTime added\n");
  else
  printf("variable myGlobalTime failed\n");


PfFlushCreateVar();
if ( apiError ==OK)
   printf ( "All variables successively created\n");
  return apiError;
}


extern "C"
RESD  Class_conversion(float* rsd)
{
/*
  reds.level_SG1=-(double)rsd[0]; //SG1 water level;
  reds.level_SG2=-(double)rsd[1]; //SG2 water level;
  reds.flow_FCV1=-(double)rsd[2]; //FCV1 flow rate;
  reds.flow_FCV2=-(double)rsd[3]; //FCV2 flow rate;
  reds.flow_TCV1=-(double)rsd[4]; //TCV1 flow rate;
  reds.flow_TCV2=-(double)rsd[5]; //TCV2 flow rate;
  reds.flow_TCV3=-(double)rsd[6];  //TCV3 flow rate;
  reds.flow_TCV4=-(double)rsd[7]; // TCV4 flow rate;
  reds.T_hl=(double)rsd[8];        // hot leg temperature;
  reds.T_cl=(double)rsd[9];        //cold leg temperature;
  reds.T_FCV1=(double)rsd[10];  //feed water temperature;
  reds.T_FCV2=(double)rsd[11];  //feed water temperature(lumped loop);
```

```
  reds.T_PRZ=(double)rsd[12];   //pressurizer temperature;
  reds.L_PRZ=(double)rsd[13];   //presurizer level;
  reds.set_power=(double)rsd[14]; // power load demand;
  reds.set_level=(double)rsd[15];   //SG water level setpoint;
  reds.ctl_level=(double)rsd[16];  //  SG level controller output
  return reds;
*/
  reds.level_SG1=(double)rsd[0]; //SG1 water level;
  reds.level_SG2=0.0;            //SG2 water level;
  reds.flow_FCV1=(double)rsd[1]; //FCV1 flow rate;
  reds.flow_FCV2=0.0; //FCV2 flow rate;
  reds.flow_steam=(double)rsd[3]; //steam flow rate;
  reds.flow_TCV1=(double)rsd[4]; //TCV1 flow rate;
  reds.flow_TCV2=(double)rsd[5]; //TCV2 flow rate;
  reds.flow_TCV3=(double)rsd[6];  //TCV3 flow rate;
  reds.flow_TCV4=(double)rsd[7]; // TCV4 flow rate;
  reds.T_hl=0.0;        // hot leg temperature;
  reds.T_cl=0.0;        //cold leg temperature;
  reds.FCV1pos=(double)rsd[2];  //FCV1 valve position;
  reds.FCV2pos=0.0;             //FCV2 valve position;
  reds.T_FCV=0.0;  //feed water temperature(lumped loop);
  reds.T_PRZ=0.0;   //pressurizer temperature;
  reds.L_PRZ=0.0;   //presurizer level;
  reds.set_power=0.0; // power load demand;
  reds.set_level=0.0;   //SG water level setpoint;
  reds.ctl_level=0.0;  //  SG level controller output
  return reds;
}

extern "C"
 void FDIFData()
{
  a1=SSS.SSSSGLvl1NR;
  b1=SSS.SSSSGLvl2NR;
  c1=SSS.SSSFlowfd1;
  d1=SSS.SSSFlowfd2;
  e11=BBB.BBBFlowTCV1;
  e12=BBB.BBBFlowTCV2;
  e13=BBB.BBBFlowTCV3;
  e14=BBB.BBBFlowTCV4;
  f11=simulation.power=DDD.DDDQthnew; //reactor power
  f12=simulation.load=DDD.DDDWturb; //reactor power output
  g11=DDD.DDDTold5;        // hot leg temperature;?
  g12=DDD.DDDTold14; //cold leg temperature;?
  g13=TREND.FeedTemp;//feed water temperature;
  g14=TREND.FeedTemp;//feed water temperature(lumped loop);
  fdi.set_power=BOPLOAD.aload; // power load demand;
  fdi.set_level=SSS.SSSSGRefNR;   //SG water level setpoint;
  fdi.ctl_level=0.0;   //  SG level controller

  fdi.level_SG1=(SSS.SSSSGLvl1NR-aa[0])/aa[0];
  fdi.level_SG2=(SSS.SSSSGLvl2NR-aa[1])/aa[1];
  fdi.flow_FCV1=(SSS.SSSFlowfd1-aa[2])/aa[2];
  fdi.flow_FCV2=(SSS.SSSFlowfd2-aa[3])/aa[3];
  fdi.flow_TCV1=(BBB.BBBFlowTCV1-aa[4])/aa[4];
  fdi.flow_TCV2=(BBB.BBBFlowTCV2-aa[5])/aa[5];
  fdi.flow_TCV3=(BBB.BBBFlowTCV3-aa[6])/aa[6];
  fdi.flow_TCV4=(BBB.BBBFlowTCV4-aa[7])/aa[7];
  fdi.T_hl=(DDD.DDDTold5-aa[9])/aa[9];
  fdi.T_cl=(DDD.DDDTold14-aa[8])/aa[8];
  fdi.T_PRZ=(PPP.PPPPprz-aa[10])/aa[10];
  fdi.L_PRZ=(PPP.PPPPrzLvlP-aa[11])/aa[11];
  fdi.T_FCV1=0.0;
  fdi.T_FCV2=0.0;
}
extern "C"
 void simulationData()
{
```

```
  simulation.power=DDD.DDDQthnew; //reactor power
  simulation.T_hl=DDD.DDDTold5; // hot leg temperature
  simulation.T_cl=DDD.DDDTold14;// cold leg temperature
  simulation.P_PRZ=PPP.PPPPprz;//  pressure in the pressurizer
  simulation.L_PRZ=PPP.PPPPrzLvlP; //level in the pressurizer
  simulation.L_SG=SSS.SSSSGLvl1NR; //steam generator water level
  simulation.flow_FCV=SSS.SSSFlowfd1; //feed water flow rate to SG1
  simulation.T_FCV=TREND.FeedTemp; //main feed water temperature
  simulation.flow_TCV=SSS.SSSFlowSG1;//steam flow rate from SG1
  simulation.speed_Turbine=0.0;
//  simulation.load=BOPINIT.Wturb;// turbine output
}
extern "C"
   void FCVData()
{
 BOPINIT.FCV1=FCV1.stuck;  //valves stuck position
 VALVEPROPERTIES.DeadBand[0][13]=FCV1.offset;
 VALVEPROPERTIES.Tau[0][13]=FCV1.timeconst; //time constant
 BOPINIT.FCV2=FCV2.stuck;  //valves stuck position
 VALVEPROPERTIES.DeadBand[1][13]=FCV2.offset;  // offset fault
 VALVEPROPERTIES.Tau[1][13]=FCV2.timeconst; //time constant
}

   void initialFCVData()
{
 FCV1.stuck=BOPINIT.FCV1;  //valves stuck position
 FCV1.offset=VALVEPROPERTIES.DeadBand[0][13];
 FCV1.timeconst=VALVEPROPERTIES.Tau[0][13]; //time constant
 FCV2.stuck=BOPINIT.FCV2;  //valves stuck position
 FCV2.offset=VALVEPROPERTIES.DeadBand[1][13];  // offset fault
 FCV2.timeconst=VALVEPROPERTIES.Tau[1][13]; //time constant
}
   void INITFDIData()
{

 aa[0]=SSS.SSSSGLvl1NRInd;
 aa[1]=SSS.SSSSGLvl2NRInd;
 aa[2]=SSS.SSSFlowSG1Ind;
 aa[3]=SSS.SSSFlowSG2Ind;
 aa[4]=BBB.BBBFlowTCV1;
 aa[5]=BBB.BBBFlowTCV2;
 aa[6]=BBB.BBBFlowTCV3;
 aa[7]=BBB.BBBFlowTCV4;
 aa[8]=DDD.DDDTCL1Ind;
 aa[9]=DDD.DDDTHL1Ind;
 aa[10]=PPP.PPPPprz;;
 aa[11]=PPP.PPPPrzLvlP;
 return;
}

extern "C"
    void TCVData()
{
VALVEPROPERTIES.DeadBand[0][2]=TCV1.offset;  // offset fault
 VALVEPROPERTIES.Tau[0][2]=TCV1.timeconst; //time constant
VALVEPROPERTIES.DeadBand[1][2]=TCV2.offset;  // offset fault
 VALVEPROPERTIES.Tau[1][2]=TCV2.timeconst; //time constant
VALVEPROPERTIES.DeadBand[2][2]=TCV3.offset;  // offset fault
 VALVEPROPERTIES.Tau[2][2]=TCV3.timeconst; //time constant
VALVEPROPERTIES.DeadBand[3][2]=TCV4.offset;  // offset fault
 VALVEPROPERTIES.Tau[3][2]=TCV4.timeconst; //time constant
}
extern "C"
    void initialTCVData()
{
TCV1.offset=VALVEPROPERTIES.DeadBand[0][2];  // offset fault
 TCV1.timeconst=VALVEPROPERTIES.Tau[0][2]; //time constant
TCV2.offset=VALVEPROPERTIES.DeadBand[1][2];  // offset fault
```

```
 TCV2.timeconst=VALVEPROPERTIES.Tau[1][2]; //time constant
TCV3.offset=VALVEPROPERTIES.DeadBand[2][2];  // offset fault
 TCV3.timeconst=VALVEPROPERTIES.Tau[2][2]; //time constant
TCV4.offset=VALVEPROPERTIES.DeadBand[3][2];  // offset fault
 TCV4.timeconst=VALVEPROPERTIES.Tau[3][2]; //time constant
}

extern "C"
        void ControllerData()
{
FEEDCONTROL.FeedGain[0][0]=controller1.offset;       // main feed water control valve 1 controller offset fault
 FEEDCONTROL.FeedGain[1][0]=controller1.Kp;  //main feed water control valve 1   controller proportional gain
 FEEDCONTROL.FeedGain[2][0]=controller1.Ki;       //main feed water control valve 1  controller integral gain fault
 FEEDCONTROL.FeedGain[0][1]=controller2.offset;       // main feed water control valve 1 controller offset fault
 FEEDCONTROL.FeedGain[1][1]=controller2.Kp;  //main feed water control valve 1   controller proportional gain
 FEEDCONTROL.FeedGain[2][1]=controller2.Ki;       //main feed water control valve 1  controller integral gain fault
}

extern "C"
        void initialControllerData()
{
 fdi.set_power=BOPLOAD.aload;
 controller1.offset=FEEDCONTROL.FeedGain[0][0];       // main feed water control valve 1 controller offset fault
 controller1.Kp=FEEDCONTROL.FeedGain[1][0];  //main feed water control valve 1   controller proportional gain
 controller1.Ki=FEEDCONTROL.FeedGain[2][0];       //main feed water control valve 1  controller integral gain fault
 controller2.offset=FEEDCONTROL.FeedGain[0][1];       // main feed water control valve 1 controller offset fault
 controller2.Kp=FEEDCONTROL.FeedGain[1][1];  //main feed water control valve 1   controller proportional gain
 controller2.Ki=FEEDCONTROL.FeedGain[2][1];       //main feed water control valve 1  controller integral gain fault
}

extern "C"
        void initPower()
{
G_aload=BOPLOAD.aload;
 G_bload=BOPLOAD.bload;
 G_Duration=BOPLOAD.Duration;
}

extern "C"
        void PowerChange()
{
BOPLOAD.aload=G_aload;
 BOPLOAD.bload=G_bload;
 BOPLOAD.Duration=G_Duration;
}

extern "C"
  void sensorData()
{
SENSORPROPERTIES.SensorDrift[1]=SG1.drift;
SENSORPROPERTIES.SensorDrift[6]=SG2.drift;
SENSORPROPERTIES.SensorDrift[2]=Flowmeter1.drift;
SENSORPROPERTIES.SensorDrift[7]=Flowmeter2.drift;
}

extern "C"
  void initialsensorData()
{
SG1.drift=SENSORPROPERTIES.SensorDrift[1];
SG2.drift=SENSORPROPERTIES.SensorDrift[6];
Flowmeter1.drift=SENSORPROPERTIES.SensorDrift[2];
Flowmeter2.drift=SENSORPROPERTIES.SensorDrift[7];
}
/*Function to be called from an RTM*/
extern "C"
   int32 stopApplication(int32 numArgs,void* args)
 {
   quit();
```

```
  return OK;
}

/*Function to be called from an RTM*/
extern "C"
  int32 datMount(int32 numArgs, void* args)
{

  void* data;
  int32 type, size;
  if (numArgs !=2)
  return !OK;
  data=PfGetFuncArg(&args,&type,&size);
  if (type!=PfCInt||size!=1)
  return !OK;
  datPoint=*(int32*)data;
  data=PfGetFuncArg(&args,&type,&size);
  if (type!=PfCFloat||size!=1)
  return !OK;
  timeInterval=*(float*)data;
  printf("data Points =%d,timeInterval=%5.2f\n",datPoint,timeInterval);
  return OK;
}

extern "C"
   void PushData(int rows,int cols,double pr_data[])
// This is a small program to push data into mxArray Data Structure;
{
     double  *start_of_pr;
//      mxArray *array_ptr;

    /* Create a 2-by-4 real double matrix named "B". */
    XX = mxCreateDoubleMatrix(rows, cols, mxREAL);
    mxSetName(XX, "B");

    /* Populate the real part of the created array. */
    start_of_pr = (double *)mxGetPr(XX);
    memcpy(start_of_pr, pr_data, rows * cols * sizeof(double) );
}

//   void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
  extern "C"
  void ExtractData(const mxArray * XY)
// This is a small program to push data into mxArray Data Structure);
  {
    int c, total_num_of_elements;
    double *real_data_ptr;

      if (mxIsDouble(XY))  {
        /* Get starting address of real data in input array. */
          real_data_ptr = (double *)mxGetPr(XY);

        /* Using pointer auto-increment, display every element in
           the array. */
          total_num_of_elements = mxGetM(XY) * mxGetN(XY);

        /* Display the contents of every real value. */
          for (c = 0; c < total_num_of_elements; c++)
                                        {
                                          diags[c]=*real_data_ptr;
                                                printf("%g\n", *real_data_ptr++);

                                        }
      }
      else
        printf("First argument must be a double array.");
}
```

```
extern "C"
double * getData()
{
double * VV1;
double VV[38];
VV[0]=DDD.DDDQthnew;
VV[1]=DDD.DDDTaveREF;
VV[2]=DDD.DDDTold16;
VV[3]=DDD.DDDTCL1Ind;
VV[4]=DDD.DDDTCL2Ind;
VV[5]=DDD.DDDTHL1Ind;
VV[6]=DDD.DDDTsat;
VV[7]=PPP.PPPQhtrKw;
VV[8]=SSS.SSSPs10Ind;
VV[9]=SSS.SSSPs20Ind;
VV[10]=SSS.SSSTsat10;
VV[11]=SSS.SSSTsat20;
VV[12]=SSS.SSSFeedTemp;
VV[13]=SSS.SSSFlowSG1Ind;
VV[14]=SSS.SSSFlowfd1Ind;
VV[15]=SSS.SSSFCV1P;
VV[16]=SSS.SSSSGLvl1WR;
VV[17]=SSS.SSSSGLvl1NR;
VV[18]=SSS.SSSSGLvl1WRInd;
VV[19]=SSS.SSSSGLvl1NRInd;
VV[20]=SSS.SSSSG1Mass;
VV[21]=SSS.SSSSGRefWR;
VV[22]=SSS.SSSSGRefNR;
VV[23]=SSS.SSSFlowSG2Ind;
VV[24]=SSS.SSSFlowfd2Ind;
VV[25]=SSS.SSSFCV2P;
VV[26]=SSS.SSSSGLvl2WR;
VV[27]=SSS.SSSSGLvl2NR;
VV[28]=SSS.SSSSGLvl2WRInd;
VV[29]=SSS.SSSSGLvl2NRInd;
VV[30]=BBB.BBBTCVposP1;
VV[31]=BBB.BBBFlowTCV1;
VV[32]=BBB.BBBTCVposP2;
VV[33]=BBB.BBBFlowTCV2;
VV[34]=BBB.BBBTCVposP3;
VV[35]=BBB.BBBFlowTCV3;
VV[36]=BBB.BBBTCVposP4;
VV[37]=BBB.BBBFlowTCV4;
VV1=VV;
return VV1;
}
extern "C"
char* FaultType()
{
char* faultDDD;
char* FaultTable[9];
FaultTable[0]="No Error";
FaultTable[1]="NR drifting";
FaultTable[2]="NR deadband fault";
FaultTable[3]="FCV deadband";
FaultTable[4]="FCV stuck";
FaultTable[5]="Flowmeter drifting";
FaultTable[6]="Bypass valve error";
FaultTable[7]="TCV degradation";
FaultTable[8]="Unknown Fault";
if (diags[0]==0.0) faultDDD=FaultTable[0];
if (diags[0]==1.0) faultDDD=FaultTable[1];
if (diags[0]==2.0) faultDDD=FaultTable[2];
if (diags[0]==3.0) faultDDD=FaultTable[3];
if (diags[0]==4.0) faultDDD=FaultTable[4];
if (diags[0]==5.0) faultDDD=FaultTable[5];
if (diags[0]==6.0) faultDDD=FaultTable[6];
```

```
if (diags[0]==7.0) faultDDD=FaultTable[7];
if (diags[0]==8.0) faultDDD=FaultTable[8];
return faultDDD;
}
```

VITA

Ke Zhao was born in Sichuan, China on May 3,1969. He received the Bachelor of Science degree from the Physics Department of Sichuan University, China, in July 1989.

In July 1989, Ke Zhao was employed by Nuclear Power Institute of China. He became an engineer in radiation safety and environmental influence assessment in 1994. He was promoted to be a senior engineer in reactor safety analysis and probabilistic risk assessment in 1998. He did research on failed fuel behavior analysis in Atomic Energy Commission, Cadarache, France, from July 1998 to February 1999.

Ke Zhao came to the United States and entered The University of Tennessee as a graduate student in Nuclear Engineering Department in August 2000. He is advised by Dr. B.R. Upadhyaya in the field of reactor simulation, control, and fault diagnosis. Ke Zhao has coauthored the following papers during his M.S. degree program at The University of Tennessee.

(1) B.R. Upadhyaya, K. Zhao, B. Lu, and M. Doster, *Fault Detection and Isolation of Sensors and Actuators in a Nuclear Plant Steam Generator,* Transactions of the American Nuclear Society, Vol. 85, pp. 350-351, November 2001.

(2) B.R. Upadhyaya, B. Lu, K. Zhao, and J.M. Doster, *Equipment Monitoring During Process Transients and Multiple Fault Conditions,* Proceedings of MARCON 2002, Knoxville, TN, May 2002.

(3) B.R. Upadhyaya, K. Zhao, and B. Lu, *Fault Monitoring of Nuclear Power Plant Sensors and Field Devices,* Proceedings of SMORN-8, Symposium on Nuclear Power Plant Surveillance and Diagnostics, Goteborg, Sweden, May 2002.

(4) B.R. Upadhyaya, K. Zhao, B. Lu, J.M. Doster, M.G. Na, Y.R. Sim, and K.H. Park, *Nuclear Plant System Monitoring Under Process Transients and Multiple Fault Conditions,* Transactions of the American Nuclear Society, Vol. 86, pp. 482-484, June 2002.

(5) B.R. Upadhyaya, B. Lu, K. Zhao, J.A. Mullens, *Data driven prediction of process variables,* ORNL/TM-2002/196, 2002.