

To the Graduate Council:

I am submitting herewith a dissertation written by Paulo Brasko Ferreira entitled "Incipient Fault Detection and Isolation of Sensors and Field Devices." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Belle R Upadhyaya
Belle R. Upadhyaya, Major Professor

We have read this dissertation
and recommend its acceptance:

Laurence F. Miller

Laurence F. Miller

Arthur E. Ruggles

Arthur E. Ruggles

Jack Wasserman

Jack Wasserman

Accepted for the Council:

Leuridan
Associate Vice Chancellor and
Dean of The Graduate School

INCIPIENT FAULT DETECTION AND ISOLATION OF SENSORS AND FIELD DEVICES

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Paulo Brasko Ferreira
May 1999

ACKNOWLEDGMENTS

I would like to express my most sincere gratitude to my major advisor, Dr. B. R. Upadhyaya. His support throughout this research has always been encouraging and invaluable. I would also like to thank my dissertation committee members, Dr. L. F. Miller, Dr. A. E. Ruggles of the Nuclear Engineering Department, and Dr. J. F. Wasserman of Mechanical and Aerospace Engineering and Engineering Science Department. Their comments and suggestions contributed to this work significantly. Many thanks to Dr. H. L. Dodds, Head of the Nuclear Engineering Department, and Dr. T. W. Kerlin, the former Department Head for their comments and guidance.

Throughout my dissertation studies, I was funded by Brazilian National Council of Research and Scientific Development, CNPq, and by Fisher-Rosemount, Inc. I would like to thank Dr. Evren Eryurek and Dr. Kadir Kavaklioglu of Fisher-Rosemount Performance Technologies for making my Ph.D. studies possible.

In the pursuit of this degree, I benefited from technical discussions with my fellow students. In particular, I would like to thank Ali S. Erbay, Joao Pinto, Ibrahim K. Attieh, and Luiz da Silva. In addition, the help of several of the support staff including, Richard Bailey and Gary Graves is appreciated.

Finally, I wish to thank my wife Maria, my daughter Thamirys, and my parents Geraldo and Annita, for their love, compassion, kindness, and patience, during my graduate studies.

ABSTRACT

The purpose of this research is to develop a robust fault detection and isolation method, for detecting faults in process sensors, actuators, controllers and other field devices. The approach to the solution to this problem is summarized below.

A novel approach for the validation of control system components and sensors was developed in this research. The process is composed of detecting a system anomaly, isolating the faulty component (such as sensors, actuators, and controllers), computing its deviation from expected value for a given system's normal condition, and finally reconstructing its output when applicable.

A variant of the Group Method of Data Handling (GMDH) was developed in this research for generating analytical redundancy from relationships among different system components. A rational function approximation was used for the data-driven modeling scheme. This analytical redundancy is necessary for detecting system anomalies and isolating faulty components.

A rule-base expert system was developed in order to isolate the faulty component. The rule-based was established from model-simulated data. A fuzzy-logic estimator was implemented to compute the magnitude of the loop component fault so that the operator or the controller might take corrective actions. This latter engine allows the system to be operated in a normal condition until the next scheduled shutdown, even if a critical component were detected as degrading.

The effectiveness of the method developed in this research was demonstrated through simulation and by implementation to an experimental control loop. The test loop consisted of a level control system, flow, pressure, level and temperature measuring sensors, motor-operated valves, and a pump. Commonly observed device faults were imposed in different system components such as pressure transmitters, pumps, and motor-operated valves.

This research has resulted in a framework for system component failure detection and isolation, allowing easy implementation of this method in any process control system (power plants, chemical industry, and other manufacturing industry). The technique would also aid the plant personnel in defining the minimal number of sensors to be installed in a process system, necessary for reliable component validation.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1. Background and Motivation	1
1.2. Statement of the Problem	8
1.2.1. Single Sensor Validation	8
1.2.2. Process Component Validation	14
1.3. Solution Methods	16
1.4. Review of Previous Work	21
1.4.1. Single Sensor Validation	22
1.4.2. Hardware Redundancy	24
1.4.3. Analytical Redundancy	25
1.5. Contribution of the Dissertation	31
1.6. Organization of the Dissertation	33
2. STEPS IN THE DEVELOPMENT OF A FAULT DETECTION AND ISOLATION ALGORITHM	35
2.1. Introduction	35
2.2. General Overview	35
2.3. Development of the Fault Detection and Isolation Algorithm	39

3. AN ALGEBRAIC APPROACH FOR FAULT DETECTION AND ISOLATION OF SENSORS AND FIELD DEVICES USING THE GROUP METHOD OF DATA HANDLING	42
3.1. Introduction	42
3.2. Using GMDH for State and Control Function Estimation and Residual Generation	43
3.3. Overview of the Group Method of Data Handling (GMDH)	44
3.4. General Description of the GMDH	45
3.5. The GMDH Algorithm	47
3.6. Enhancement of the GMDH Using Rational Function Approximation	51
3.7. Applying GMDH Method in an FDI Algorithm	56
4. DEVELOPMENT OF A MATHEMATICAL MODEL OF A LOW-PRESSURE WATER LOOP SYSTEM AND TEST FACILITY FOR FDI FEASIBILITY STUDIES	59
4.1. Introduction	59
4.2. Development of a Low-Pressure Water Loop System Model	60
4.2.1. Model Components	60
4.2.2. Model Equations	63
4.3. Development of a Low-Pressure Experimental Control Loop System	67
4.3.1. Description of the Low-Pressure Experimental Control Loop System	67

5. SINGLE SENSOR VALIDATION IN PROCESS CONTROL SYSTEMS	 82
5.1. Introduction	 82
5.2. A Brief Overview	 83
5.3. Problem Statement	 84
5.4. Signal Decomposition Using Wavelet Transforms and Validation	
Algorithm	 85
5.4.1. Multi-Resolution Analysis and Signal Decomposition	 85
5.4.2. Edge Distortion	 86
5.4.3. Application to Temperature Data from RTD Experiments	 88
5.4.4. New Signal Decomposition Algorithm	 91
5.5. Sensor Drift Recovery in a Highly Oscillatory Process	 93
5.6. Single Sensor Validation in a Process with Feedback	 95
5.7. Single Sensor Validation for the Case of a Laboratory Control System	100
5.8. Concluding Remarks	 102
6. DEVELOPMENT OF A FAULT DETECTION AND ISOLATION	
ALGORITHM FOR A COMPLEX SYSTEM	 106
6.1. Introduction	 106
6.2. The Overall Procedure	 106
6.2.1. The Development of an FDI Algorithm	 107
6.2.2. Implementation of an FDI Algorithm	 110
6.3. Development of an FDI Algorithm using a Theoretical Model	 111

6.3.1. Generation of the Fault-Free Database	112
6.3.2. Generating Qualitative Relationships among Loop Components	112
6.3.3. Generating Quantitative Relationships among Loop Components	115
6.3.4. Interpreting the Results of the GMDH Modeling Process	119
6.4. Fault Types Studied in this Research	123
6.5. Development of the Rule-based Decision Module for Fault Isolation	125
6.6. Minimum Requirements for an FDI system	132
6.7. Implementation of the FDI using the Theoretical Model	133
6.8. Results of Simulations of Some Fault Detection and Isolation Cases	136
6.8.1. Results of the FDI Algorithm for Changes in System Set Points	137
6.8.2. Results of the FDI Algorithm for Faulty Cases	140
6.9. Fault Compensation	147
7. DEVELOPMENT OF A FUZZY-LOGIC ENGINE FOR	
FAULTY DEVICE RECONSTRUCTION	148
7.1. Introduction	148
7.2. Fuzzy Logic – A Brief Introduction	149
7.3. Control Loop Device Reconstruction using ANFIS	156
7.4. Implementation of the Faulty Device Reconstruction in the	
FDI Algorithm	163
7.5. Fuzzy Logic Applied to Faulty Device Reconstruction – Results	165
7.6. Next Step	165

8. DEVELOPMENT AND IMPLEMENTATION OF THE FDI ALGORITHM	
FOR AN EXPERIMENTAL CONTROL LOOP	167
8.1. Introduction	167
8.2. Generation of Fault-free Database	168
8.3. Generation of Qualitative Relationships among the Control Loop	
Components	169
8.4. Development of the Rule-based Decision Module	169
8.5. Implementation of the FDI Algorithm	171
8.6. Experimental Results	175
8.6.1. Changes in the System Set Points	180
8.6.2. Artificially Imposed Faults	183
8.7. Fault Compensation using a Fuzzy Logic System	190
9. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS	
FOR FUTURE WORK	191
9.1. Summary	191
9.2. Conclusions	194
9.3. Recommendations for Future Work	196
LIST OF REFERENCES	198
APPENDICES	218
Appendix A	219
A.1. Procedure for using the Group Method of Data Handling Software ...	220

A.1.1. Observations	221
A.2. Subroutines Called by the GMDH Algorithm	222
A.3. An example of Obtaining the GMDH Model from a Data Set	228
A.4. Some Simulated Cases Studied	240
A.4.1. Changes in System Set Points	241
A.4.2. Some Fault Cases	245
A.5. Rule-based Expert System	252
VITA	263

LIST OF FIGURES

Figure 1.1. Pressure transmitter sensor signal outputs	11
Figure 1.2. Threshold limit problems when applied to oscillatory processes	13
Figure 1.3. Selective filters for separating sensor and process components of sensor measurements in an oscillatory process	15
Figure 2.1. An overall procedure for developing a fault detection and isolation (FDI) algorithm	40
Figure 2.2. An overall procedure for implementing a fault detection and isolation algorithm in an industrial process	41
Figure 3.1. Self-organizing GMDH structure with m inputs and k layers	46
Figure 3.2. A node of the GMDH structure	46
Figure 3.3. Input to the GMDH algorithm	48
Figure 3.4. GMDH Training stopping criterion	50
Figure 3.5. One of many combinations of regression equation	52
Figure 3.6. Residuals for different regression equations	55
Figure 3.7. One of the best combinations of GMDH regression equation	57
Figure 4.1. A schematic of the modeled low-pressure water loop system	61
Figure 4.2. Low-pressure water loop system theoretical model in Matlab-Simulink	64
Figure 4.3. Worcester series 75 motor-operated valve	68
Figure 4.4. Circuit board for valve position control	68
Figure 4.5. K_p as a function of set point error	71

Figure 4.6. K_i as a function of set point error	71
Figure 4.7. Changes in the actual water level for different set point values	71
Figure 4.8. Motor-operated valves	72
Figure 4.9. Constant speed centrifugal pump	72
Figure 4.10. Rosemount multivariate pressure transmitter model 3095 MV for mass flow measurements	73
Figure 4.11. A view of the water tank	73
Figure 4.12. Turbine type water velocity meter	74
Figure 4.13. A view of the turbine flow meter internal parts	74
Figure 4.14. Rosemount pressure transmitter for water level measurements	76
Figure 4.15. Water heater element	76
Figure 4.16. A view of the entire low-pressure water loop experimental setup	77
Figure 4.17. Side view of the low-pressure water loop experimental setup	77
Figure 4.18. Power supply and signal conditioners panel	78
Figure 4.19. Patch panel for data acquisition and experiment control	78
Figure 4.20. Computer used for data acquisition, data analysis and experiment control	79
Figure 4.21. Labview data acquisition and experiment control program	81
Figure 5.1. Signal decomposition into low and high frequency bands	87
Figure 5.2. Successive decomposition of a signal into different pass-bands using wavelet transform	87

Figure 5.3. Illustration of combining actual sensor output with process fluctuation to provide the composite signal	89
Figure 5.4. Multi-resolution analysis of RTD signal	90
Figure 5.5. Appended one-month data window with one week of data and quadratic extrapolation on each side	92
Figure 5.6. Results of multi-resolution analysis	92
Figure 5.7. Comparison of the actual RTD #A-44 signal and the processed signal using multi-resolution wavelet analysis	94
Figure 5.8. Recovered average signal	96
Figure 5.9. Plot of recovered average signals from three RTDs	96
Figure 5.10. Block diagram of the procedure used for the case of sensor validation in a closed-loop system	98
Figure 5.11. Sensor output. Water level is kept at 5 inches	99
Figure 5.12. PID controller output drift as a function of time	99
Figure 5.13. Noisy sensor drift obtained from the relationship between sensor drift and PID controller drift	101
Figure 5.14. Comparison between sensor drift recovered from the controller output and the original drift data	101
Figure 5.15. Water level signal which is kept constant during the entire experiment even though the sensor is drifting	103
Figure 5.16. Controller changes through time due to the sensor drift	103

Figure 5.17. Sensor drift recovered from the controller output and the conversion table between the controller output and sensor drift	104
Figure 5.18. Comparison between the experimental sensor drift and the original drift imposed on the sensor	104
Figure 6.1. The GMDH predicted values against the training output values for the Inlet Flow rate as a function of Bypass MOV and Inlet MOV positions	120
Figure 6.2. The GMDH predicted values against the test output values for the Inlet Flow rate as a function of Bypass MOV and Inlet MOV positions	121
Figure 6.3. The error between the predicted values and the measurements for the inlet flow rate as a function of Bypass MOV and Inlet MOV positions	122
Figure 6.4. Plots of different loop components for the case of a drift fault in the water level sensor due to aging process	127
Figure 6.5. Control loop layout	134
Figure 7.1. Triangular membership functions of a given input variable	152
Figure 7.2. The overall fuzzy logic inference process	153
Figure 7.3. ANFIS graphical user interface	155
Figure 7.4. Fuzzy logic structure for fault device reconstruction	159
Figure 7.5. Gaussian-type membership functions of the Fuzzy system	159
Figure 7.6. Training phase in the ANFIS subroutine	160
Figure 7.7. Test Data versus Fuzzy output	160

Figure 7.8. Training data versus fuzzy output	161
Figure 7.9. Performance tests of the Fuzzy logic drift recovering system	162
Figure 7.10. An overall procedure for implementing a Fault Detection and Isolation algorithm in an industrial process	164
Figure 8.1. Plot of GMDH algorithm output	170
Figure 8.2. Absolute error between GMDH predicted values and the test data ...	170
Figure 8.3. Drift profile as a function of time for the sensor water level	173
Figure 8.4. Main screen of the data acquisition and experiment control LabView program for the low-pressure water loop experimental system ...	174
Figure 8.5. Schematic structure of fault detection and isolation system	176
Figure 8.6. Low frequency low magnitude system oscillation due to inlet MOV intrinsic features	178
Figure 8.7. An example of the PID output signal during a failure in the water level sensor	179
Figure A.1. The GMDH predicted values against the training output values for the inlet flow rate as a function of Bypass MOV and Inlet MOV positions	225
Figure A.2. The GMDH predicted values against the testing output values for the inlet flow rate as a function of Bypass MOV and Inlet MOV positions	226

Figure A.3. The error between the predicted values against the measurements	
for the inlet Flow rate as a function of Bypass MOV and Inlet MOV	
positions	227

LIST OF TABLES

Table 4.1. System variables for the LPWL model	66
Table 5.1. Residuals between decomposition levels	94
Table 6.1. Representation of correlation coefficients of the loop variables	113
Table 6.2. Qualitative error between GMDH prediction values and the measured values for seven fault cases	129
Table 6.3. Minimum requirement table for different monitored faults	135

LIST OF ACRONYMS

ARMA:	Auto-Regression Moving Average model
DAQ:	Data Acquisition
DC:	Direct Component
FL:	Fuzzy Logic
GMDH:	Group Method of Data Handling
LPWL:	Low-Pressure Water Loop
MOV:	Motor-Operated Valve
MRWA:	Multi-Resolution Wavelet Analysis
PC:	Personal Computer
PID:	Proportional-Integral-Derivative
PWR:	Pressurized Water Reactor
RMT:	Rosemount
RTD:	Resistance Temperature Detector
TFM:	Turbine Flow Meter
WL:	Water Level

CHAPTER 1

INTRODUCTION

1.1. BACKGROUND AND MOTIVATION

In large process systems, such as power plants, chemical plants, pulp and paper plants, metals and other manufacturing systems, the operating state is monitored by various sensors and controlled by using such sensor outputs. Changes in some sensor values directly affect the system controller outputs or the system operators' decisions regarding initiation of necessary control actions.

A failure in an instrument (sensor) used to provide a feedback signal in a control system can cause a serious deterioration in the dynamic performance of the system. If the failure occurs gradually and is detected in its early stages, the faulty instrument can be replaced before serious trouble develops. Otherwise, proper control actions can be undermined by invalid sensor values. The resulting improper control actions may often lead to total system shutdown [39,63]. Both situations can create economic losses and can also threaten the safety of human operators. Economic losses and safety issues make it desirable to implement reliable fault-tolerant control systems [28,41,48].

A fault is defined as any kind of malfunction in the actual dynamic system, which leads to an unacceptable anomaly in the overall system performance. Such malfunctions may occur either in the sensors (instruments), actuators, or in process components.

The measurement system is of great importance for any complex dynamic process. It provides information for both manual operation and automatic control in the form of sensor signals. The validity of these signals is necessary for successful process operation. To enhance system safety and improve its performance, it is important that failures be promptly detected and identified so that appropriate remedies can be applied [15, 63].

Sensors are crucial for automation and process control, both for achieving performance and for guaranteeing plant safety. The technological drive towards automated plants and high energy-intensive processes demands greater intelligence, availability, and reliability of instrumentation. Even when a process is intrinsically safe, loss of a sensor would lead to loss of process control and hence a sustained period of poor quality production.

“Work on sensor fault detection began in the late 1950s. Automatic control systems and their control algorithms are becoming increasingly complex and sophisticated. Consequently, there is a growing demand for increased fault tolerance, which can be achieved not only by improving the reliability of functional units, but also by an efficient fault detection, isolation, and accommodation (FDIA) concept” [49, 51].

The tasks of a failure detection and isolation (FDI) system include the detection of a process anomaly and the isolation of the component responsible for the observed system irregularity. The FDI system should monitor both the basic equipment (actuators) and its control instrumentation (sensors). The applications of FDI are found in the

process industries (chemical plants, oil refineries, power stations), aerospace industry (high-performance airplanes, space-vehicles and stations), and most recently in some mass-produced consumer equipment sector (automobiles, appliances).

Advances in mathematical modeling and modern control theory, as well as the application of on-line computers, allow for process state estimation and parameter identification to be implemented in real time. This feasibility has led to the development of a wide variety of FDI systems [12].

In order to design a reliable fault-tolerant control system, or to maintain a high level of performance for complex processes, it is crucial that process changes are detected and diagnosed promptly. As soon as these changes are observed, a corrective action can be taken to reconfigure the control action and accommodate the system alteration [31,37,45,61,83,84].

As an example of the importance of an FDI system, consider a feed condenser with an optimum exit temperature of 100 °C. If the measurement drifts 2.2 °C, the optimum energy tradeoff inside the plant will not be achieved and there are associated penalties. In an application that generates benefits of \$3,000 per day, assuming a penalty function the profit is given by

$$\text{Profit} = \$3,000 - 60(T - 100)^2$$

then 10% of the original profit is lost due to the wrong measurement, that is, \$ 9,000/month or \$ 109,500/year.

As another example, reference [81] describes how a system component fault can jeopardize a whole process: In 1980, a ground test of the Space Shuttle Main Engine (SSME) experienced an erroneous value of the Main Combustion Chamber Pressure. The failed sensor reading resulted in a severely abnormal operating pressure. An internal fire and subsequent explosion occurred as a result of the sensor failure. The engine was virtually destroyed.

“In order to improve the reliability of an operation, it is necessary to validate the measured sensor data, isolate any failed sensor, and recover the failed sensor measurements before the controller uses them. The trend in many dynamic systems is to move towards more complexity to meet higher levels of performance. Such systems require equally increased levels of reliability. This reliability can only be obtained through the use of advanced monitoring systems.

A traditional scheme for protecting a system against failures takes advantage of hardware redundancy as a primary checking procedure. In a hardware redundancy system, two or more similar sensors or devices are present. A voting logic scheme may then be used to identify a faulty instrument output, provided one of the redundant counterparts of the faulty sensor does not fail at the same time. Sensor validation basically consists of cross-checking sensor's output for unexpected behavior. Abnormalities can then be detected by threshold tests, moving averages, filtering, sequential probability ratio tests, and numerous other tests both in the time and frequency domain” [7].

Hardware redundancy offers reliable data validation. However, there are a number of drawbacks.

1. The cost of installing and maintaining multiple sensors increases the cost of the overall system.
2. Common fault failures still occur, especially in cases of severe operation failures.
3. There are systems, such as a chemical pressure vessel, where physical limitations do not permit the installation of redundant sensors.

During the last two decades, the so called “model-based fault diagnosis” approach has received increasing attention in both research and application [31,37,61,83,102,108]. This philosophy for the FDI methodology has emerged and is increasingly discussed. It is based on the use of analytical rather than physical (hardware) redundancy. This implies that the inherent redundancy contained in the static and dynamic relationships among the system inputs and measured outputs is exploited for FDI. This approach makes use of mathematical models of the system or parts of it. Analytical redundancy uses on-line data processing techniques to generate redundant signals from a single set of instruments. These schemes are intended to be superior to the conventional “hardware redundant” methods. While analytical redundancy offers a wealth of instrument information, this information is not helpful unless it can be processed and interpreted.

The major advantage of the analytical redundancy approach compared to the hardware redundancy is that no additional hardware components are needed in order to realize an FDI algorithm. The FDI algorithm only requires input-output data processing

for implementation. A model-based FDI algorithm can be implemented on the process control computer, often without additional hardware requirements. Moreover, the measurements necessary to control the process are, in many cases, also sufficient for the FDI algorithm so that no additional sensors have to be installed. Under these circumstances, only additional storage capacity and possibly greater computer power are needed for the implementation of a model-based FDI algorithm.

The analytical redundancy approach also overcomes the problem of diagnosing anomalies when common failures happen for sensors of the same type. The analytical redundant sensor measurement is obtained from mathematical relationships among different types of sensors.

Analytical redundancy relies upon information that is obtained from a mathematical model. This model is formulated on the basis of physical relationships among the system variables. These variables can be a combination of direct and other analytical measurements. Thus, analytical measurements may supplement sensor redundancy for plant variables of interest. Furthermore, analytical measurements allow detection of (non-sensor) plant component failures such as actuators and controllers. They also allow detection of common-mode failures, that is, simultaneous and identical failures of two similar devices possibly due to a common cause. The failure decisions can be made either solely on the basis of current values of the available measurements or they can be made by sequential tests that use past observations in addition to the current measurements.

Primarily, a sensor validation scheme should fulfill the following two tasks: detection and diagnosis. The former involves discovering a malfunction in a sensor, while the latter can be sub-divided into three stages: localization (establishing which sensor has failed), identification (determining the type of failure), and estimation (indicating the effect and extent of the failure).

Fault detection systems must actively produce information before any failure decisions can be made. In order for analytical redundancy systems to report an indication of a potential fault, it is necessary to first derive the normal expected values of the sensors under the current operational condition. The expected values of the sensors can be derived from existing relationships among the measured parameters. These relationships are derived from either system theoretical models or from data-driven modeling techniques. Any significant mismatch between the actual sensor measurement and its estimated value from analytical redundancy indicates a system anomaly. Either a fault in a sensor or a fault in a system component that disrupts the relationship used for estimating the process variable may cause this anomaly.

Intuitively, the residuals represent the difference between various functions of the observed sensor outputs and the expected values of these functions in the normal (no-fail) mode. In the absence of a failure, residuals should be unbiased, showing agreement between observed and expected normal behavior of the system; a failure signature typically takes the form of residual biases that are characteristic of the failure. Thus,

residual generation is based on knowledge of the normal behavior of the system. The actual process of residual generation can vary in complexity.

1.2. STATEMENT OF THE PROBLEM

In general, a fault in a process component may be due to sensor fault, actuator fault, controller fault, and others. The purpose of this research is to develop and demonstrate a generic approach for fault detection and isolation of these components in a process system using available measurements, and to establish minimum requirements to achieve this goal.

Most of the algorithms developed for performing system component fault detection and isolation can be categorized into one of two groups: single sensor validation or distributed sensor validation.

1.2.1. Single Sensor Validation

In single sensor validation, the FDI algorithm performs sensor validation basically at the sensor level, i.e., only the sensor output signal is monitored. This kind of analysis implies that the overall system condition does not change during operation and any observed change in the sensor output signal is attributed to the sensor fault.

In single sensor validation, all sensor values are assigned a confidence value. This confidence value depends on the specific sensor characteristics, the predicted value, and

the physical limitations of the sensor value. A sensor signal is validated by evaluating the properties of the sensor measurements as follow [54]:

- Range checking, which checks whether the sensor's output signal is outside its operational bandwidth.
- Trend analysis, which monitors any significant trend in the data throughout regression analysis.
- Mean value.
- Standard deviation, which checks if the sensor measurements follow a Gaussian distribution.
- Moving averages.
- Frequency analysis.
- Sequential probability ratio test.

If no anomalies are observed during sensor validation, its measurements are fused into a higher level of a hierarchical central control system, where its values can then be displayed or used as input to a system controller.

Drawbacks to the use of FDI algorithm exist when applying to single sensor validation. Different forms of sensor behavior can be observed when a system device has a fault. The specific behavior depends not only on the type of faulty device (pressure transmitter, temperature sensor, etc.), but also, and most important, the sensor's primary task in the system. The sensor output is often used either for condition monitoring, or as an input to a process controller. For condition monitoring, the sensor's output is

normally displayed on a panel or in the computer's screen. The sensor's output is not used in any other way. The sensor information is not used as an input parameter to any system control. When the sensor's output is used as input to a controller, this information is used for giving necessary information to the controller to take actions to keep the system at normal conditions.

If the faulty sensor is the one used as a monitoring device, then the effect of its fault will be isolated, that is, the fault will be observed in its own signature and nowhere else. The faulty device will affect no other sensors. For example, imagine a case where two water level sensors are used to monitor the level of water in a tank. The system operation condition is kept constant with the water level set point at 10 inches. Observing both sensors' outputs, one could conclude that pressure sensor #2 has a fault, while sensor #1 can be considered healthy (See Figure 1.1).

The above situation is not observed when the sensor's output is used as input to a level control system. As soon as the sensor drifts, the controller notices the sensor signal variations and actuates in the opposite direction, in an attempt to correctly match the set point. Assuming that sensor #1 is used for water control and is drifting positively, then the level of water indicated by the sensor is lower than the actual value. This kind of fault is normally caused due to an aging process. Even though the actual water level in the tank is constant, the sensor indicates an increase in the water level. The controller acts in the opposite direction, for example slightly closing an inlet pipeline valve. The controller closes the valve until the measurement of the water level matches the set point.

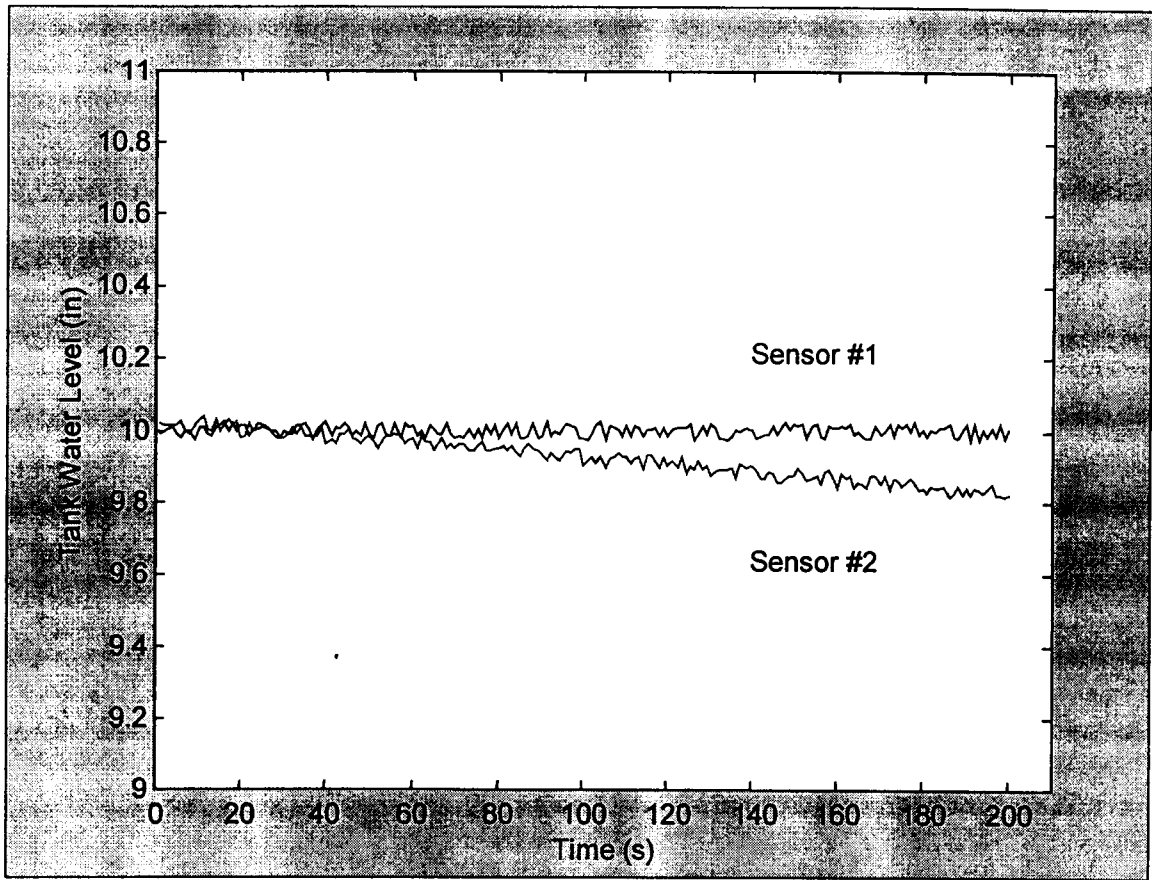


Figure 1.1. Pressure transmitter sensor signal outputs.

However, the actual water level in the tank is lower than the set point (such as sensor #2, assuming that it is not drifting). This is a case where the drifting sensor output is masked or compensated by the controller action.

The two examples discussed above show how misleading single sensor validation could be. Even though the system's operational conditions are kept constant in both cases, the faulty sensor can be either the one that has drifting measurements or the one that has its measurements at a constant level.

Another drawback when using single sensor validation occurs when changes in the sensor's output are caused by changes in the process conditions. Changes in the system steady-state condition are normally caused by a change in one of the controlled system variable set points. An example of this is the change in the water level set point from 20 inches to 12 inches. The observed decrease in the water level sensor measurements is not related to a sensor problem, but due to a set point change. Therefore, a robust fault and isolation diagnostic system must differentiate from changes in a sensor output signal caused by a failure in the sensor and from one caused by changes in the process state.

Careful attention must also be given when an actuator or a controller itself has failed. In this case some sensors may change their output values, while others may not. It will depend on how related are these sensors with the actuator/controller in question.

Drawbacks to the use of FDI algorithm also exist when it is applied in a process that has an oscillatory behavior as shown in Figure 1.2. In this case, the sensor

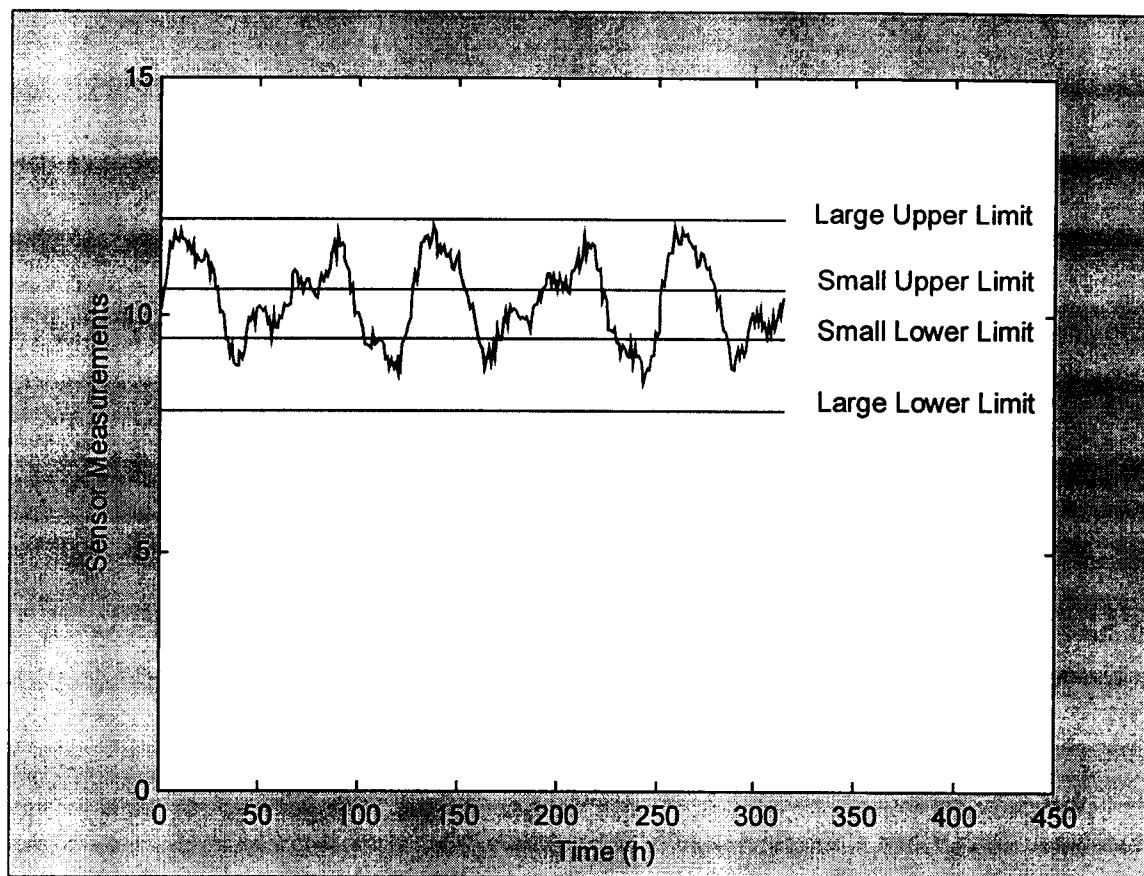


Figure 1.2. Threshold limit problems when applied to oscillatory processes.

measurements have a natural oscillation. The use of threshold limits to monitor the sensor's healthy state in this case is not viable. If the threshold limits were set to small values it would lead often to false alarms. If the threshold limits were set to larger values than the process oscillations, FDI would not be sensitive to incipient sensor drifts. In cases where the system has an oscillatory behavior, selective filters should be applied to the sensor's measurements. Selective filters may be able to separate the process oscillation term from the sensor's signal output. After filtering, the FDI algorithm could use the term related to the sensor's signal to monitor its health (Figure 1.3).

1.2.2. Process Component Validation

The second kind of FDI algorithm takes advantage of a set of related sensors installed in the system. This algorithm generates a redundant measurement analytically through physical or empirical relationships among the sensors.

Although fault detection based on analytical redundancy has attracted considerable academic interest, it has rarely been applied in process industries. The reasons are as follows.

- It has proved difficult to develop plant models that are reliable (that is, having a high probability of detecting faults and a low probability of false alarms) and robust to plant disturbances and modifications.
- Detailed mathematical or knowledge-based models of the process are required. These are generally complicated to construct and, even then, may have gross errors.

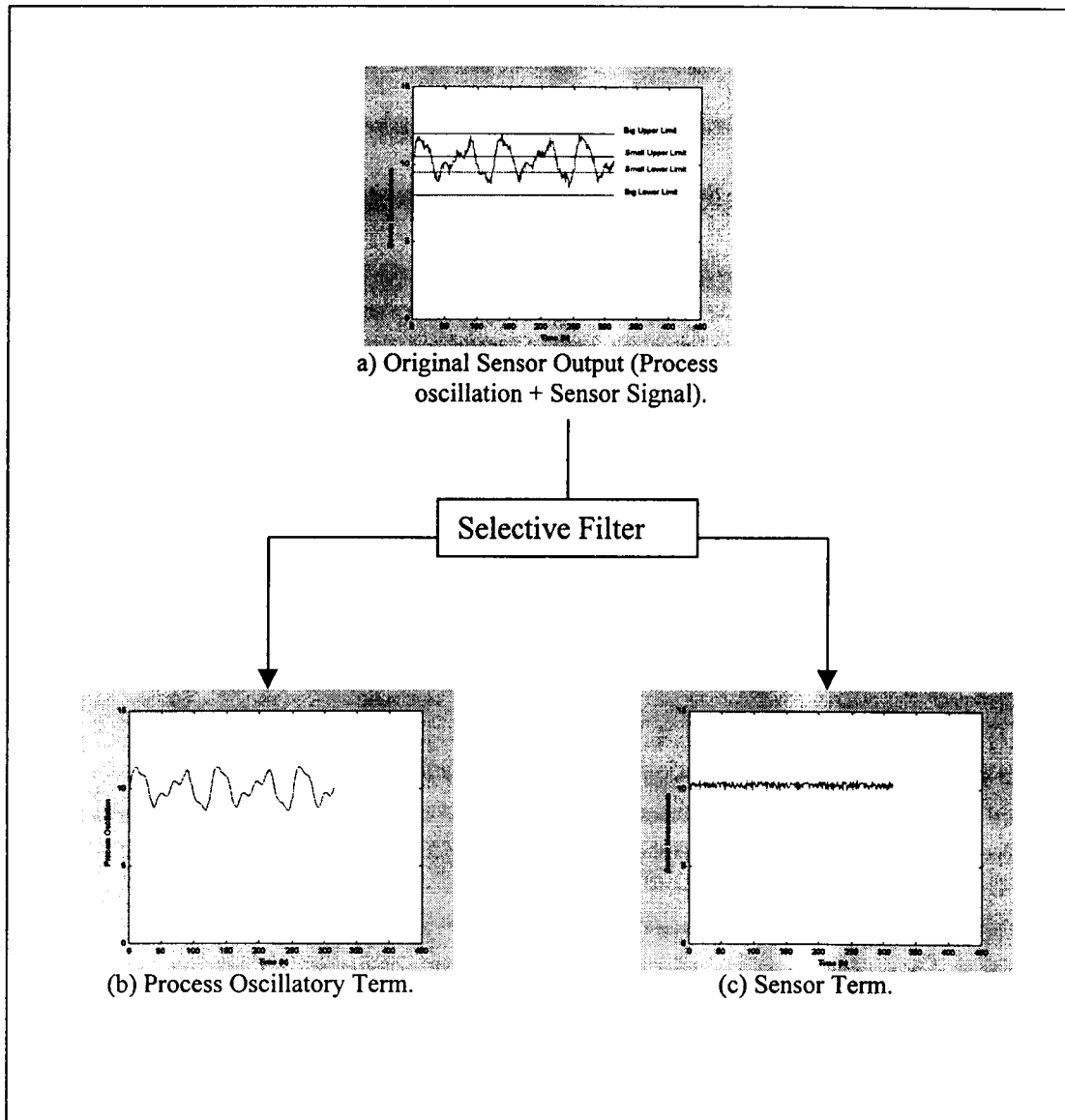


Figure 1.3. Selective filters for separating sensor and process components of sensor measurements in an oscillatory process.

- Such schemes are expensive and must be implemented and maintained by highly skilled personnel.
- Since the models are 'tailor-made' and the validation algorithms are 'well-tuned' for each process, a slight modification in the process characteristics or the use of the same model/algorithm to a new application requires costly adjustments.

1.3. SOLUTION METHODS

Normally, a fault detection and isolation algorithm is a four-step procedure. In the first step, an analytical loop device output must be generated using a mathematical model. This information is compared to the actual related device. A residual value is obtained, where residual is the difference between the analytical and actual loop device values. In the second step, the residual value information is processed. The FDI algorithm determines whether there is a fault present in the loop. Changes in the loop variable set points must be considered at this point to avoid false alarms. In case of a fault being observed, the third step is used, where the FDI algorithm concentrates its efforts in isolating the faulty device. The last step of an FDI procedure attempts to adjust the device signal to compensate for the fault. Normally this compensation can be done only in cases where the fault is not catastrophic, such as a pump has stopped or a sensor has an open circuit. It also depends on which device has failed and the importance of this device in the loop system. These adjustments compensate for the signal drift and let the

system operate properly even in the presence of a faulty device. The process could run uninterruptedly until its next scheduled maintenance.

As discussed in the previous section, developing an FDI algorithm is normally a difficult task due to the following reasons.

- Model development: aspects of reliability, complexity, development costs, and difficulty in making any modifications due to small changes in the actual system must be addressed.
- Signal analysis: dependency of the type of component and its functionality in the process.
- Influence of natural process oscillations in the device output signal.

The purpose of this research project is to develop a robust fault detection and isolation method, which would detect faults in process sensors, actuators, controllers, and other field devices. The approach to the solution to the above problem, as developed in this dissertation, is summarized below. The effectiveness of the method developed in this research was demonstrated through simulation and by implementation to an experimental control loop.

If a faulty sensor is used for system monitoring, then the effect of its fault will be isolated, i.e., the fault will be observed in its own signal characteristics, and it will have no influence on any other system component. Therefore, to detect a fault in such a sensor requires analysis of only that sensor's signal. In this case, if the system process is kept roughly constant, the use of threshold limit values could be enough for monitoring the

sensor's health. However, noisy signals and even small natural process oscillations could lead to false alarms. For these cases, the use of digital filters is normally required. While digital filters are more precise, they have their drawbacks. Digital filters are stationary transforms and may not work effectively for multi-resolution analysis of non-stationary data. Also, edge effects are significant at the beginning and at the end of finite-length data sets when using digital filters.

An approach using wavelet transforms for addressing the signal reconstruction, especially during transient events, was developed in this research. Signal reconstruction with wavelet transforms has a high accuracy, as compared with digital filters. Also, two important modifications were introduced into the standard multi-resolution wavelet analysis technique in this work: one that takes care of the edge effects observed when analyzing small sets of data, and the other that automatically chooses the best wavelet decomposition sub-level.

When a sensor's output is used in a process controller, the drift in the measured (or controlled) signal will not be seen because the control output is constantly adjusted to provide sufficient actuation to maintain the set point value of the measured (controlled) variable. However, one would see a drift in the controller output (such as a valve opening or an increase in the heater output). Therefore, to obtain the sensor drift for post-fault analysis, it is necessary to extract the sensor's malfunction information from the controller output signal. The correlation between sensor and controller devices was

established during the development of an FDI algorithm through a mathematical model. This model was used subsequently to map the controller output back to the sensor drift.

In most cases, however, a simple mapping between the sensor output and the controller is not sufficient. Many factors must be considered in the relationship, such as water density, pump discharges, bypass valves, and other system components. For these cases, a more complex model must be developed. In this dissertation, a data-driven modeling method, called the Group Method of Data Handling (GMDH) was developed and successfully implemented. The GMDH generalizes the multi-dimensional mapping among sensors and device-specific inputs and outputs. It is desirable to achieve this objective by algebraic methods in order to facilitate real-time applications. The generalization is accomplished using a rational function approximation. This approach also overcomes the problems of false alarms due to changes in the system variable set points and developing expensive and sometimes unreliable physical models for a specific system. The method is general so that it can be applied to most of the present industry processes without any modification. The use of this approach requires the knowledge of sub-system loop variables. In this research, this knowledge base is obtained by correlation analysis of the various state and control function variables.

After generating the predicted loop device value through mathematical models, this is compared with the actual device output measurements. The difference between the measurement and the predicted device value is what is called the residual value. If the residual value is close to zero, this indicates that the predicted value is close to the

measured value. The FDI system checks for any significant residual discrepancy (large values). This determines whether there is an anomaly in the system or not. Because the components in the sensor signal related to noise, process oscillations, and influence of other devices were already filtered out or taken into consideration, a threshold limit can be applied at this point to evaluate the device residual.

If a fault is detected, then another module of the FDI algorithm is executed. This module attempts to isolate the fault component. A rule-based expert system was developed for this purpose. This expert system consists of a set of rules, each of them based on a specific device fault scenario. Every known fault type influences the system behavior in a specific way. A database was established containing all studied characteristic behavior of faults. This database is consulted by an expert system every time a system abnormal behavior is observed for retrieving the most probable faulty device.

Three ways of obtaining a set of characteristic behavior of faults for different fault scenarios in a specific process can be considered.

- a) Analyzing what would happen through human expertise.
- b) Using (simple) mathematical models.
- c) Using the actual system.

In the last two cases above, artificially generated drifts are introduced into each loop device and the system response is observed.

If the observed characteristic system behavior due to a specific fault is not unique, that is, if two different faults give the same set of device output behavior, a hardware redundant may be necessary. In some situations, implementing a redundant sensor, is the only way an ambiguous situation can be avoided.

The last step in an FDI monitoring system should be the attempt to reconfigure the faulty device. For cases where loop devices are experiencing drift fault, one can compute the amount of drift and adjust the faulty device to compensate for it. This device reconfiguration would allow the continuation of the process until the next loop maintenance without losing performance. In this work, a fuzzy logic engine is proposed to perform this task.

1.4. REVIEW OF PREVIOUS WORK

“The work on sensor fault detection began in the late 1950s when it became clear that automated control based on faulty data could lead to system failures. Interest was stimulated by the drive for even greater efficiency and increasing penalties for incidents causing environmental damage. This development was (and still is) mainly stimulated by the trend of automation towards more complexity and the corresponding growing demand for higher availability and security of control systems.

With the proliferation of on-line computers, a wide variety of FDI approaches have been developed. These include simple limit-checking, redundant sensors, special sensors, frequency analysis, and artificial intelligence techniques. A strong impetus also

comes from the side of modern control theory that has brought forth powerful techniques of mathematical modeling, state estimation and parameter identification that have been made feasible by the accelerated progress of modern computer technology” [31-34].

Traditional approaches to fault diagnosis can be categorized into three distinct groups: single sensor validation, hardware redundancy validation, and analytical redundancy. Each of them has its good points and drawbacks.

1.4.1. Single Sensor Validation

Traditional approaches to fault diagnosis, pre-dating the use of analytical redundancy (and hence the model-based approach), are based on the use of signal processing techniques such as frequency spectrum analysis and Measurement Aberration Detection (MAD) [36-39, 83-84]. These processing techniques are still very much in use, even though new powerful techniques are being developed.

In the frequency spectrum analysis technique, plant measurements are considered to have a standard frequency spectrum under normal operating conditions. Any deviation from this is an indication of abnormality. Certain fault types may even have a characteristic signature in the spectrum that can be used for fault isolation. A spectrum can be compared with a standard spectrum to diagnose the nature and type of fault and even predict the outcome that the fault will have upon the system. Spectral analysis is especially useful when plant model information is unavailable or difficult to obtain.

Spectral analysis such as the fast Fourier transform (FFT) is the most commonly used signal processing technique in machine tool condition monitoring systems. A disadvantage is that it provides a good solution only in the frequency domain and loses some signal information in the time domain: it is used only to process stable stochastic signals. Wavelet transforms have recently been proposed as a significant new tool in signal processing and information extraction. It has been used to analyze tool failure monitoring signals. The wavelet transform has a good solution in the frequency domain and in the time domain; it can extract more information in the time domain at different frequency bands [109].

In the Measurement Aberration Detection (MAD) technique, the output of the single sensor only is examined. When applying MAD, it is assumed that the true measurement signal has certain time and/or frequency domain properties, and that faults may occur randomly at any time [49-51]. Normally, the components of algorithmic sensor validation techniques being addressed in the design of an MAD include the following:

- Range checking.
- Trend analysis.
- Performing a "running test" on data sets (consistency of trend).
- Performing a "hugging test" on data sets (closeness to target, or set point).
- Calculation of expected values and regression coefficients and comparison with measured values.

- Calculation of uncertainties.
- Calculation of uncertainty propagation.
- Probabilistic analysis on combined algorithmic/heuristic uncertainties [54].

The limit checking has a serious drawback in that process variables may vary in accordance with different operating states of the process. The limit thus has to be dependent on the operating state of the process. The advantages of the limit-checking-based methods are simplicity and reliability. However, they are only able to react after a relatively large change in the feature has occurred. That means that either a large sudden fault has occurred, or a series of soft or slow drifting faults has been present for a long time.

1.4.2. Hardware Redundancy

A traditional scheme for protecting a system against failures in its feedback sensors is to provide the system with three (or more) sets of sensors, so that there is redundancy in the feedback- information. A voting logic may then be used to identify a faulty instrument output, provided one of the redundant counterparts of the faulty instrument does not fail at the same time. This approach to the problem works well in those systems where the redundant instrument sets do not impose an undue burden of cost, weight, or size [17-19, 112-113].

Parallel (hardware) redundancy: An important prerequisite for achieving fault-tolerance in safety-critical control systems is by means of multiple lanes of identical

hardware. Two principle disadvantages of this approach are the weight penalty (particularly important in flight control) and the possibility of a common-mode fault of the system remaining undetected. Hardware redundancy can be combined with model-based methods to maximize the effectiveness of fault isolation.

However, it is too expensive and often impractical to install secondary measurement devices throughout a process plant. Instead, operators have turned to software tools for data validation [93].

The actual process of residual generation can vary in complexity. For example, in voting systems the residuals are simply the difference of the outputs of the various like sensors, whereas in the likelihood ratio testing, the residuals are innovations generated by the Kalman filter technique [16].

1.4.3. Analytical Redundancy

During the last two decades, the so-called 'model-based fault diagnosis' approach has received increasing attention in both research and application [16, 31-34, 36-39, 60-61, 71, 83-84, 108]. This approach is based on the concept of 'analytical redundancy', as opposed to 'physical (hardware or parallel) redundancy', which uses measurements from redundant sensors for fault diagnosis purposes. Analytical redundancy makes use of signals generated by the mathematical model of the system being considered. These signals are compared with the actual measurements obtained from the system. The

comparison is done using the residual quantities, which give the difference between the measured signals and signals generated by the mathematical model [83-84].

Looking at the literature one can see that there is a variety of different approaches to the problem of FDI using analytical redundancy. The great variety of proposed methods can, however, be brought down to a few basic concepts such as

- The parity space approach.
- The dedicated observer approach and innovation-based approach.
- The fault detection filter approach.
- The parameter identification approach.
- Soft computation approach (Neural networks, Fuzzy Logic)

Parity space approach:

The early contribution to the parity space approach were made by Potter and Suman, Desai and Ray, and Willsky [16,71,108]. The key idea is to check the parity (consistency) of the mathematical equations of the system (analytical) redundancy relations by using the actual measurements. A fault is declared to have occurred once pre-assigned error bounds are exceeded.

Dedicated observer approach

Many authors have approached the FDI problem by directly starting with single or banks of Luenberger observers or Kalman filters. Papers such as Clark et al. [17-18], Willsky [108], Frank and Keller [32-34] are good examples of this technique.

The basic idea of the observer approach is to reconstruct the outputs of the system from the measurements or subsets of the measurements with the aid of observers or Kalman filters using the estimation error or innovation, respectively, as a residual for the detection and isolation of the faults. It is well known from observer theory that for state estimation one can use linear or nonlinear, full or reduced order state observers in the deterministic case or Kalman filters in the stochastic case when noise has to be considered.

Guo [44,73] has used a specially designed Kalman Filter to detect and accommodate the sensor failures in the jet engine control. In general, the Kalman filter approach utilizes the engine model to detect and isolate sensor failures. Thus, it is strongly dependent upon a hi-fidelity engine model, which may be a drawback in the implementation for a complex system.

Guo has implemented a single sensor validation through validation gates. The validation of a given sensor's data through this validation gate is obtained by using a Kalman filtering estimate applied to an appropriate model of the studied system. Readings that lie outside the validation gate are classified as faulty.

Parameter identification approach

This is an alternative approach to the above-described methods based on state estimation. It makes use of the fact that faults of a dynamic system are reflected in the physical parameters as, for example, friction, mass, viscosity, resistance, capacitance,

inductance, etc. The idea of the parameter identification approach is to detect the faults via estimation of the parameters of the mathematical model [60-61].

“Although fault detection theory has attracted considerable academic interest, it has rarely been applied in the process industry. The reasons are both technical and economic. Technically, it has proved difficult to develop plant models that are reliable (that is, having a high probability of detecting faults and a low probability of false alarms) and robust to plant disturbances and modifications. Economically, such schemes are expensive (typically, \$1 million for a moderate section of plant), and must be implemented and maintained by highly skilled personnel) [49-51]. A large investment of modeling expertise is needed for each plant, as there is little carry-over of effort between applications, and key personnel must be available to maintain and update the system” [54].

Several principles may be used to generate residuals. These include Kalman filtering, observers and the straightforward use of an input-output model.

Artificial Neural Networks

Other emerging methodologies for performing fault detection are artificial intelligence-based techniques such as expert systems and neural networks, pattern recognition, and process hypercube. Many of these methods are designed for the detection of a single failure mode and, therefore, lack the robustness required for practical, large-scale applications. For this reason, there has been research into combining

diverse signal processing techniques in a parallel architecture to perform comprehensive signal validation.

Guo [44] describes a method of sensor failure detection, isolation, and accommodation using a neural network approach. The proposed technique utilizes dimensionality reduction to develop the neural network for sensor failure detection and data recovery. In this work, the sensor validation scheme is applied in a closed-loop for detection and recovery of sensor faults in a simulation of the Space Shuttle Main Engine (SSME) control.

Signal validation and process monitoring problems in many cases require the prediction of one or more process variables in a system. The feasibility of using neural networks to characterize one variable as a function of other related variables is studied. The Backpropagation Network (BPN) is used to develop "models" of signals from both a commercial power plant and the Experimental Breeder Reactor-II (EBR-II). Several innovations are made in the algorithm, the most significant of which is the progressive adjustment of the sigmoidal threshold function and weight updating terms, thus leading to the designation adaptive backpropagation neural network [98].

Neural networks offer great potential for both sensor validation and diagnostics. The current lack of a comprehensive training set, however, prevents the use of neural networks [68].

Parity-Space

The basic principle of parity relation approaches for FDI is to check the inconsistency between inputs and outputs of the monitored system. The unbalance term is used as a residual signal [16, 31, 36].

One of the drawbacks to the parity-space technique is the rigid boundaries used with the residual during the fault isolation stage. The residual is nonzero because of inherent noise in the signals. In addition, for analytical redundancy, the residual becomes even larger because of the modeling error and the propagation of input signal errors through the model. This results in the need for fault detection thresholds. Generally, these thresholds are based on using the signal tolerance for the measurement. This tolerance is usually defined as “3-sigma” of the measurement error. Such an approach provides a useful comparison point; however, the large comparison band can actually allow a biased/drifted signal to be routinely declared as valid when, in fact, it is nearly “6-sigma” from the other signal(s) [56].

Patton and Chen [83] provide a good review of the parity-space approach. The decision making is based on various tests, including the generalized likelihood ratio the sequential probability ratio test and innovation properties (tests for whiteness, mean, covariance, Chi Square, etc.), applied to the residuals [56].

The parity-space approach is a very popular fault detection and isolation technique for use with redundant sensors. The parity-space representations are especially useful where possible discrepancies among several sensors and analytically redundant

measurements must be resolved. The parity-space technique models the redundant measurements as the true value combined with a measurement error term. Measurement inconsistencies are obtained by eliminating the underlying measured quantity from the data by creating linearly independent relationships between the measurements known as the parity equations. The magnitude of the parity vector represents the inconsistency between the redundant measurements, and the direction of the parity vector indicates the faulty signal(s). This particular method has been simplified and expanded by others for scalar measurements.

1.5. CONTRIBUTION OF THE DISSERTATION

The following is a summary of the original contributions of this dissertation.

1. Implementation of the multi-resolution wavelet analysis technique for signal reconstruction for a higher accuracy. Sub-levels of decomposition can be used for filtering not only process noise but also natural process oscillations that could lead to false alarms. Multi-resolution wavelet analysis works effectively for non-stationary data, the type of data observed in drift failures.
2. Development of an approach for automatic selection of the best decomposition level for filtering purposes using wavelet transform. This approach makes use of a residual technique. The residual is computed as the difference between subsequent levels of decomposition. For all cases tested, the residuals between the initial decomposition levels are very high, but they have a decreasing trend as the level increases.

However, this trend reverses after a few decomposition levels. At this point the recovered signal starts to lose the component of the signal that carries the information about the device drift. This technique was successfully applied to all cases of device drift fault that were studied in this research.

3. Development of innovations in the Group Method of Data Handling approach. Two innovations were introduced to make this method versatile. The first was the inclusion of rational functions in the set of basic functions used in the GMDH algorithm. The second innovation was the selection of a best set of basic functions during run time. A binary algorithm was created with the task of making all possible combinations of basic functions. This procedure facilitates the choice of the best set of basic functions for each case studied. This procedure demonstrated an improved way of finding mathematical models for residual generation.
4. Utilization of the GMDH method for implementing the FDI algorithm. A variant of the GMDH method was proposed in this research to avoid the high costs associated with the development of physical models and the high cost of their maintenance (normally required when small changes are implemented in the real process). Furthermore this technique can be easily implemented in a variety of industrial processes without the need of experts to develop a new model for each situation.
5. Development of a fault detection and isolation (FDI) system that integrates optimal data-driven models for process estimation, a knowledge base of the relationship

among the variables, and a fuzzy-logic based signal reconstruction. This system is capable of detecting faults in sensors, actuators, controllers and other field devices.

6. Development of a theoretical model and an experimental control loop that represents a typical industrial process control. The implementation of the new FDI system was demonstrated using simulated and real data from these systems, including various failure cases.

1.6. ORGANIZATION OF THE DISSERTATION

Chapter 2 of the dissertation presents a procedure for development of a fault detection and isolation algorithm. The main steps are described in detail. A variant of the Group Method of Data Handling (GMDH) [30,59] is derived in Chapter 3. This variant is critical for the generation of relationships among related loop components and for the generation of analytical redundancy. Chapter 4 describes the development of a theoretical and experimental setup of a low-pressure water loop (LPWL) system. The theoretical model is used for feasibility study, while the experimental setup is used for practical FDI implementation. In Chapters 5, a new technique has been developed for validating isolated sensors in process control systems. This technique is based on multi-resolution signal decomposition using wavelet transforms and mapping of control variables and actuator dynamics. Cases of open-loop (sensors used for monitoring) and feedback dynamics (sensors used as a controller input) are considered.

Chapter 6 describes the entire process involved in developing and implementing the fault detection and isolation algorithm using the theoretical model. Simulations of normal and faulty conditions are shown and discussed. Chapter 7 develops and implements an algorithm for reconstructing the desired state of a faulty device in such a way that the effect of the fault in the system is cancelled. The mathematical tool used for device reconstruction was the fuzzy logic. Tuning the fuzzy logic membership functions was accomplished by using the Adaptive Neuro-Fuzzy Inference System (ANFIS) methodology. In Chapter 8, the performance of the proposed Fault Detection and Isolation algorithm was tested in a real process. The low-pressure water loop (LPWL) experimental system was used for this task. All the steps for development and implementation of an FDI algorithm used exclusively data generated from this experimental facility. Chapter 9 summarizes all the work performed in this dissertation, and presents conclusions and recommendations for future work.

CHAPTER 2

STEPS IN THE DEVELOPMENT OF A FAULT DETECTION AND ISOLATION ALGORITHM

2.1. INTRODUCTION

This chapter presents the basic procedure developed in this research for the implementation of a distributed loop component fault detection and isolation (FDI) algorithm. The FDI algorithm monitors the system health conditions by looking at the loop component outputs. If the FDI detects any system anomaly, a second step is performed, which tries to isolate the most probable faulty loop-component. If the type of fault is a drift, then the FDI algorithm will attempt to make adjustments in the faulty device to compensate for the fault. If the adjustments are successful, the process can be kept running without loss of performance until its next maintenance schedule. A description of the required steps for developing the FDI algorithm for a general system is presented in the following sections.

2.2. GENERAL OVERVIEW

When developing a fault detection and isolation algorithm for system components, many important points must be addressed. The first consideration is about the type of system components themselves, such as pumps, actuators, controllers, and sensors. Each of these components has a different impact in the system behavior when it

suffers a fault. Another important factor is related to the component's function in the system. As mentioned in Chapter 1, a sensor can be either used for condition monitoring, or as an input to a process controller. For condition monitoring, the sensor's output is normally displayed on a panel or on a computer screen. The sensor's output is not used in any other purpose. The sensor information is not used as an input parameter of any system control. The sensor's output is used as an input signal to provide necessary information for a controller to take actions to keep the system in normal conditions. If the faulty sensor is the one used as a monitoring device, then the effect of its fault will be isolated, that is, the fault will be observed in its own signature and nowhere else. The faulty device would affect no other sensors. This behavior is not observed when the sensor's output is used as input to a level control system. As soon as the sensor drifts, the controller notices the sensor signal variations and actuates in the opposite direction, in an attempt to correctly match the set point.

Another point that must be addressed is the normal changes in the process conditions. These changes may be caused by changes in one or more process set points. A robust fault detection and isolation system must differentiate between changes in a sensor's output signal due to its failure and changes due to normal variations in the process operational conditions.

Similar to the case of changes in a system variable set point is the situation when the process has an oscillatory behavior. In this case, the sensor measurements also have a natural oscillation. Applying threshold limits to monitor the sensor's normal state in this

case is not recommended. If the threshold limits were set to small values, it would often lead to false alarms. If the threshold limits were set to large values compared to the process oscillations, it would not be sensitive to incipient sensor drifts. In cases where the system has an oscillatory behavior, selective filters should be applied to the sensor's measurements prior to any FDI analysis. Selective filters may be able to separate the process oscillation term from the sensor's signal output. After the filtering, the FDI algorithm could be used to monitor the sensor's health.

For more complex systems, analytical redundancy can be used to improve the FDI robustness. In analytical redundancy, a loop component (such as a sensor) output is estimated through a mathematical relationship among different (but related) loop components. For example, the equation $PV=nRT$ may be used to estimate the pressure inside a gas tank and an actual pressure measurement health condition can be verified using this information. However, in order to use analytical redundancy, mathematical relationships must be previously obtained. The first step in obtaining these relationships is to identify related components. This information can be obtained through the expertise of system engineers or through certain mathematical tools such as correlation coefficients.

After identifying related loop components, it is necessary to characterize these relationships through mathematical models. Mathematical models may be derived in several different forms, the most expensive being the development of an accurate physics model. Another approach is to use data-driven models, such as the autoregressive

moving average (ARMA) or the GMDH models. These models generate necessary redundant sensor outputs for the FDI algorithm. The models must consider different normal system conditions. Changes in the measurements of one or more loop components can be caused by changes in a loop variable set point. This is an important aspect in avoiding false alarms.

The FDI algorithm basically consists of three steps: step one monitors the residuals of all available system variables. The residual is the difference between a loop variable (such as a temperature sensor) measurement and its model predicted (analytical) value. In a normal system operational condition, each individual residual should be close to zero, indicating a good agreement between the system-predicted condition and the system actual condition. Normally threshold limits are used to delimit the region in which the residuals are considered satisfactory. If any residual goes beyond the threshold limits, it indicates a probability of a system anomaly (fault detection). If a fault has been detected, the second step of FDI is activated, in order to isolate the faulty component. Normally, many system variables are influenced by the faulty system component. This makes it difficult to know which component is the cause and which one is the consequence. In this step, the FDI algorithm isolates the faulty component through an expert rule-based system. This expert system is based on the particular characteristics of each individual fault. A faulty component disturbs the system behavior in different ways not only for different types of sensors, but also for same types of sensors that are in different system locations. A system engineer can generate the rules of the expert

system, by small system perturbation experiments or by computer simulations. After the faulty component is identified, the third step of the FDI algorithm is executed. In this step the FDI algorithm will attempt to compensate the signal used by the faulty component. If it succeeds, the process can be kept in operation without losing the performance until the next programmed maintenance schedule. In this dissertation, the compensation is performed by a fuzzy logic engine, which is developed using faulty experimental data.

Figure 2.1 shows the overall procedure for developing a fault detection and isolation algorithm. Figure 2.2 shows the procedure for implementing it in a given system.

2.3. DEVELOPMENT OF THE FAULT DETECTION AND ISOLATION ALGORITHM

The development of a fault detection and isolation algorithm consists of performing all the tasks described in the last section. The next chapters describe the work realized in this dissertation for developing and implementing an FDI algorithm for a small process control loop. Theoretical and experimental studies were performed on both single sensor validation, for the cases when the sensor's output is used for monitoring or as a controller's input, and on a more complex system. An experimental low-pressure water loop (LPWL) was developed and used for these tasks.

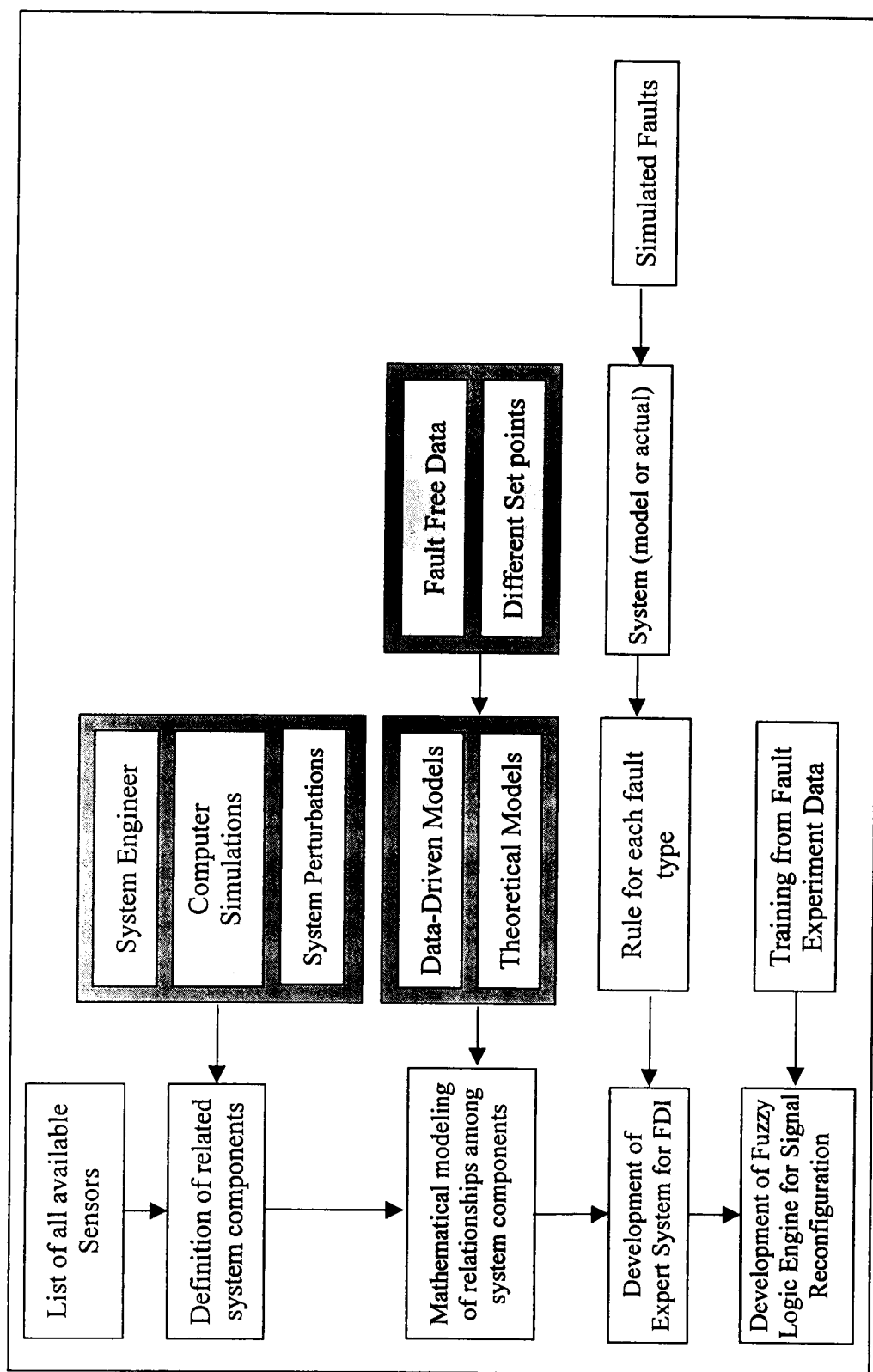


Figure 2.1. An overall procedure for developing a Fault Detection and Isolation (FDI) algorithm.

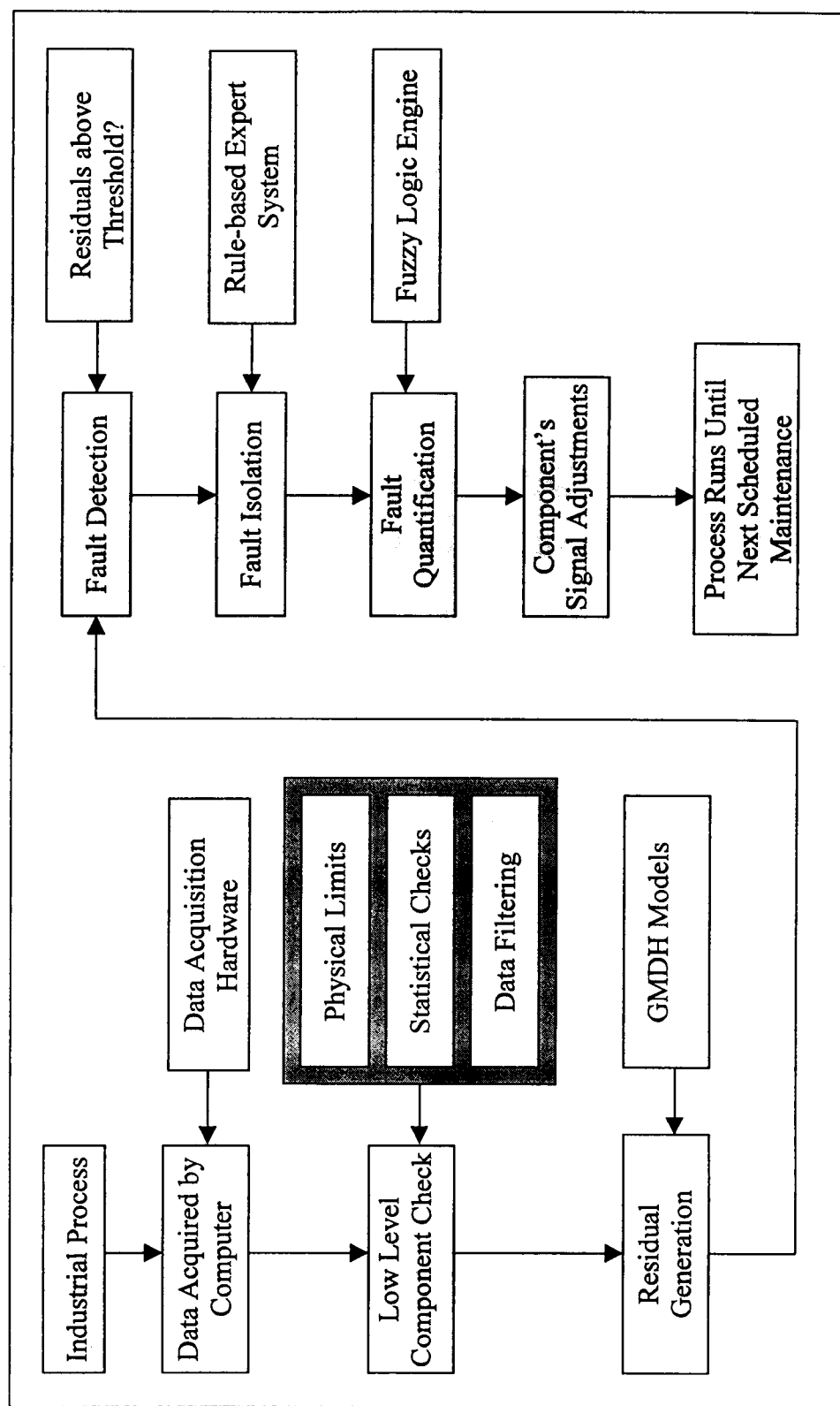


Figure 2.2. An overall procedure for implementing a Fault Detection and Isolation algorithm in an industrial process.

CHAPTER 3

AN ALGEBRAIC APPROACH FOR FAULT DETECTION AND ISOLATION OF SENSORS AND FIELD DEVICES USING THE GROUP METHOD OF DATA HANDLING

3.1. INTRODUCTION

This chapter describes an approach that has been developed based on the Group Method of Data Handling (GMDH), and implemented as part of an on-line monitoring, fault detection and isolation (FDI) system. The GMDH provides a general framework for characterizing the relationship among a set of state variables of a process and is used for generating estimates of critical variables in an optimal data-driven model form. The GMDH is used to provide the necessary analytical redundancy to the FDI algorithm. As mentioned in Chapter 2, analytical redundancy is the predicted value of a loop component variable based on other related state variables. The analytical redundancy is used to monitor the component's health. A general description of the GMDH method and the procedure for applying it to a system's data are also presented. For feasibility studies of using this method as part of an FDI, a simulation model of a process control loop has been developed and includes sensors, controllers and actuators. Known faults are imposed on different devices such as pressure transmitters, motor-operated valves, control elements, and a centrifugal pump.

3.2. USING GMDH FOR STATE AND CONTROL FUNCTION ESTIMATION AND RESIDUAL GENERATION

Major industry sectors, including chemical, hydro-carbon processing, pulp and paper, food and beverage, pharmaceuticals, rubber/plastics, metals, automotive and power, have economic and quality concerns as addressed by overall plant performance, minimizing unscheduled downtime, and the long-term management of critical assets. The key to achieving these needs in the manufacturing industry is to develop an integrated approach for monitoring, control, fault detection and diagnosis of process components such as sensors, actuators, control devices, machinery and other equipment. Several techniques, developed by industry and academia, for monitoring isolated sensors and system components have been reported [43,45,63,74,99,100,103]. Model-based local sensor validation and fault diagnosis methods [27,28] were developed for specific applications. Theoretical models used in FDI are important for system state estimation and residual generation. A residual is the difference between a predicted sensor output value and its actual measurement for a given process condition. The residuals play an important role in an FDI algorithm. However, developing theoretical models are generally expensive, sometimes not completely reliable, and are system specific.

This chapter presents a novel approach for automatically developing physical models for related control loop devices. The approach, called the Group Method of Data Handling (GMDH), is a general method that can be applied to most of the present industry processes without any modification. The use of this approach requires the

knowledge of the sets of related process variables. This knowledge is obtained in this work by a correlation analysis procedure. The GMDH generalizes the multi-dimensional mapping among sensors and device-specific inputs and outputs. The GMDH was chosen to avoid the development of expensive theoretical models. It is desirable to achieve this objective by algebraic methods in order to facilitate real-time applications. The generalization is accomplished by using a rational function approximation.

An overview of the theory behind the GMDH is presented in Section 3.3. A description of the procedure involved in implementing this method and the algorithm developed for the GMDH computations are given in Sections 3.4 and 3.5. Section 3.6 describes the modifications made in the original method for enhancing its performance. Section 3.7 describes how this methodology is used in this dissertation.

3.3. OVERVIEW OF THE GROUP METHOD OF DATA HANDLING (GMDH)

The GMDH is a self-organizing algebraic model. It is based on sequential learning networks, which are networks of mathematical functions that characterize complex, nonlinear relationships in a compact and rapidly executable form. Such networks subdivide a problem into manageable sub-problems and then automatically apply advanced mapping techniques to solve each of these simpler problems [83].

The network constructed using the GMDH algorithm is an adaptive, supervised learning model. The architecture of a polynomial network is formed during the training

process (non-parametric model). The node activation function is based on elementary polynomials of arbitrary order. A polynomial network is shown in Figure 3.1.

Each node output can be expressed by a second order polynomial function as shown in Figure 3.2, where x_i and x_j are inputs and A, B, C, D, E and F are the polynomial coefficients, which are equivalent to the network weights, and Y is the node output.

The GMDH training algorithm can be used to adjust the polynomial coefficients and to find the network architecture. Polynomial nets can also be used for function approximation.

3.4. GENERAL DESCRIPTION OF THE GMDH

The GMDH can be considered as a further propagation of inductive self-organizing method to the solution of more complex practical problems. It solves the problem of handling data, which are given in real noisy or short samples of observations. The goal is to get a mathematical model of an object (the problem of identification and pattern recognition) or to describe the process.

GMDH solves by a sorting-out procedure, by choosing a model from a candidate set of models in accordance with a given criterion. Most GMDH algorithms use the polynomial reference functions. General relation between input and output variables can be expressed by the Volterra functional series [12]:

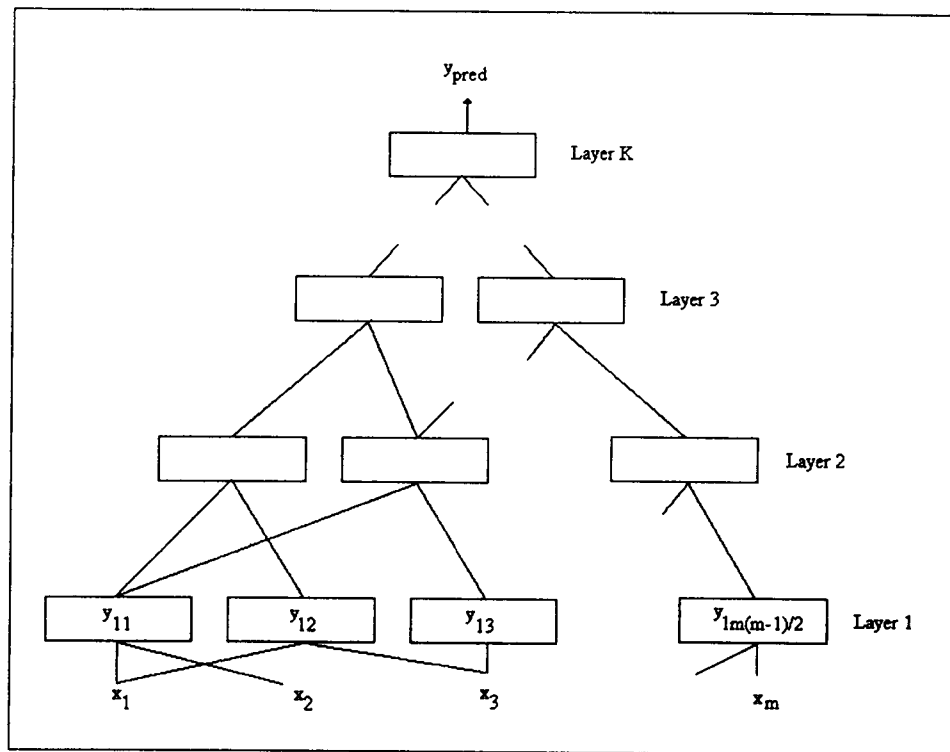


Figure 3.1. Self-organizing GMDH structure with m inputs and k layers.

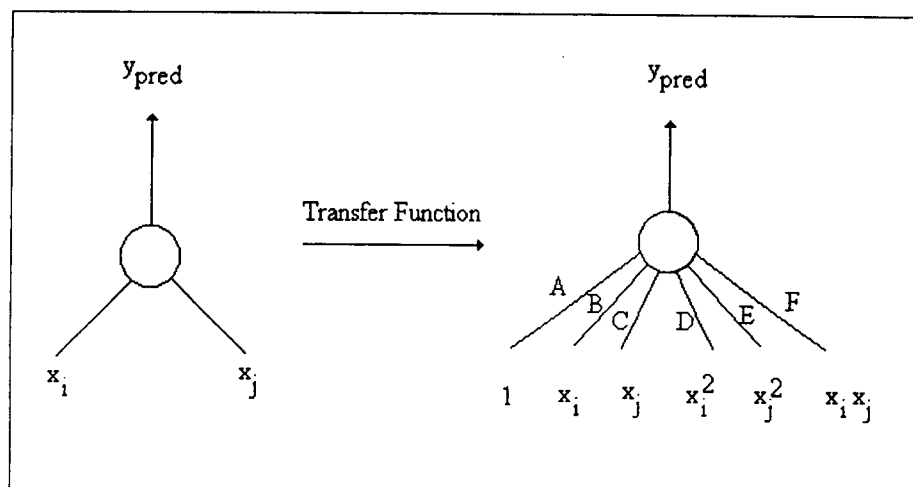


Figure 3.2. A node of the GMDH structure. This node uses a polynomial transfer function.

$$y = a + \sum_{i=1}^m b_i x_i + \sum_{i=1}^m \sum_{j=1}^m c_{ij} x_i x_j + \sum_{i=1}^m \sum_{j=1}^m \sum_{k=1}^m d_{ijk} x_i x_j x_k + \dots \quad (3.1)$$

where $\{x_1, x_2, \dots, x_m\}$ is the input variable vector and $\{a, b_i, c_{ij}, \dots\}$ is the vector of coefficients or weights.

Components of the input vector can be independent variables, functional forms or finite difference terms. Other nonlinear reference functions, such as difference, probabilistic, harmonic, and logistic values can also be used. The method allows finding simultaneously the structure of the model and the dependence of modeled system output on the values of the most significant inputs of the system.

3.5. THE GMDH ALGORITHM

This section describes the basic GMDH algorithm implementation [30]. The following procedure is used for a given set of n observations of the m independent variables $\{x_1, x_2, \dots, x_m\}$ and their associated matrix of dependent values $\{y_1, y_2, \dots, y_n\}$ (a multiple input single output system).

- i) Subdivide the data into two distinct subsets. One for training and other for testing (Figure 3.3).
- ii) For each pair of input variables x_i and x_j and the associated output y of the training set, compute the regression polynomial

$$y = A + Bx_i + Cx_j + Dx_i^2 + Ex_j^2 + Fx_i x_j \quad (3.2)$$

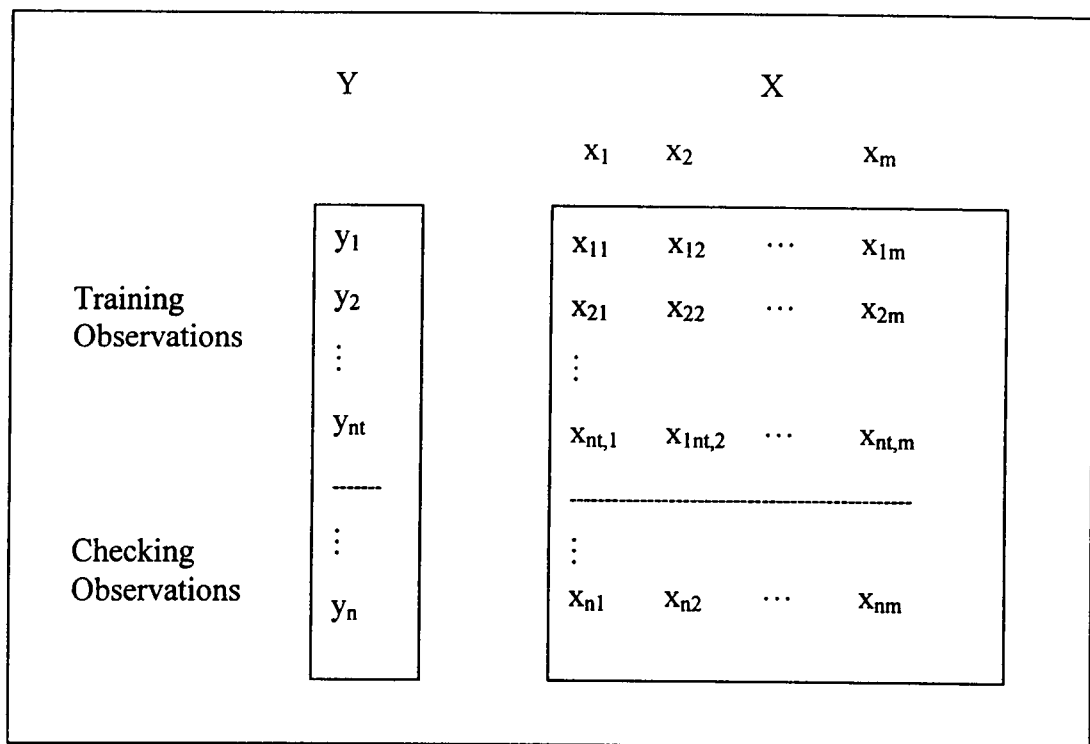


Figure 3.3. Input to the GMDH algorithm. The data is split into two sets. The first is used to fit the GMDH model and the second is used to evaluate the polynomial to avoid model overfitting.

that best fits the dependent observations y in the training set. $m(m-1)/2$ regression polynomials will be computed from the observations.

- iii) For each regression, evaluate the polynomial for all n observations. Store these n new observations into a new matrix Z . The other columns of Z are computed in a similar manner. These variables can be interpreted as new improved variables that have better predictability than those of the original generation $x_1, x_2, \dots, x_m$.
- iv) Screening out the least effective variables: For each column of Z matrix, the algorithm computes the root-mean-square value (also called the regularity criterion) r_j over the test data set and is given by

$$r_j^2 = \frac{\sum_{i=1}^{nt} (y_i - z_{ij})^2}{\sum_{i=1}^{nt} y_i^2} \quad (3.3)$$

- v) Order the columns of Z according to increasing r_j , and then pick those columns of Z satisfying $r_j < R$ (where R is some prescribed value chosen by the user) to replace the original columns of X .
- vi) Testing for optimality: The above process is repeated and new generations are obtained until the method starts overfitting the data set. One can plot the smallest of the r_j 's computed in each generation and compare it with the smallest r_j 's of the previous generation. The process should stop when the r_j 's of the most recent generation start to have an increasing trend (Figure 3.4).

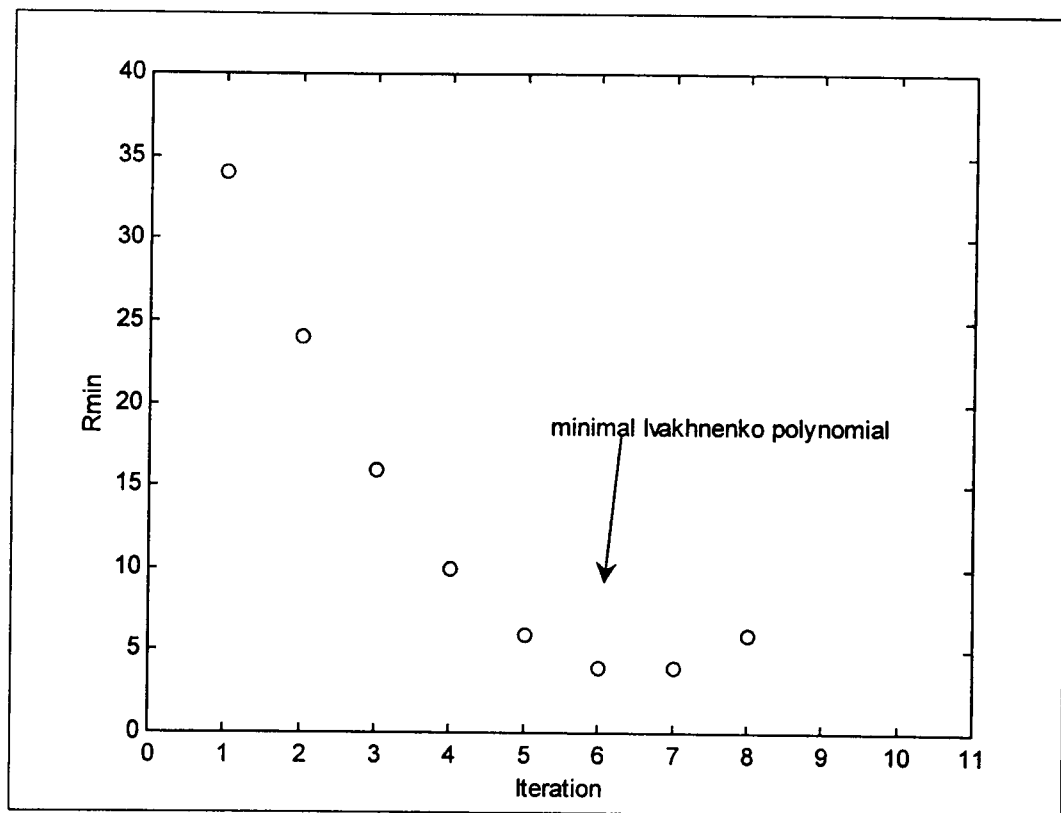


Figure 3.4. GMDH Training stopping criterion.

Once all quadratic regression polynomials are stored in the computer, it is possible to compute the prediction $\hat{y}$ of the dependent variable y from these regressions.

3.6. ENHANCEMENT OF THE GMDH USING RATIONAL FUNCTION APPROXIMATION

The GMDH approach proposed by Ivankhnenko [30,59] uses polynomial functions. This set of polynomial functions is normally sufficient in finding some of the relationships of interest in this study. However, some results were not satisfactory for other relationships. For cases where the relationship is of the form of a rational function approximation, the polynomial regression equation is not adequate. An example of this kind of rational function approximation is the relationship between tank water level and all motor-operated valve positions. Basically the relationship is of the form

$$\text{WaterLevel} = \frac{\text{InletMOVPosition}}{(\text{InletMOVPosition} + \text{BypassMOVposition}) * \text{OutletMOVPosition}^2}$$

The use of polynomial regression models resulted in errors of the order of 20% between simulated data and predicted values. As an example, Figure 3.5 shows the predicted values of the outlet flow rate variable obtained from the GMDH method versus its simulated value obtained from a low-pressure water loop theoretical model (this theoretical model is discussed in Chapter 4). The data obtained from GMDH method is

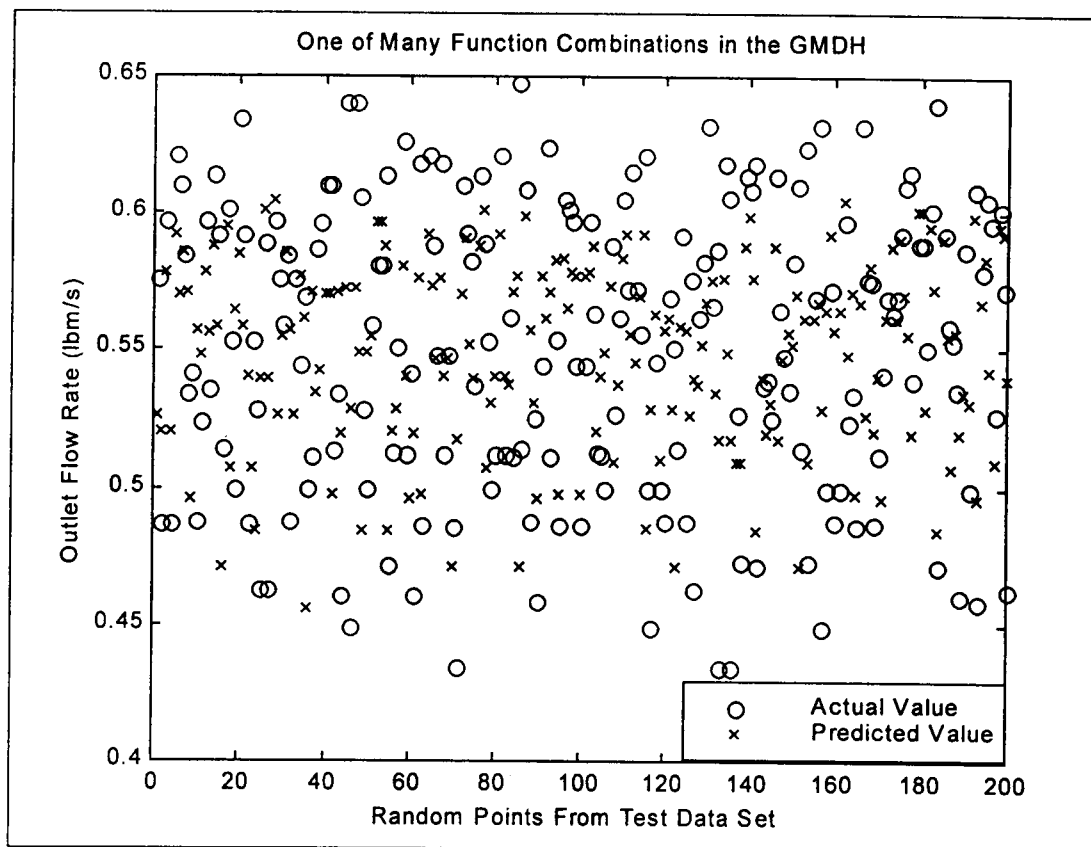


Figure 3.5. One of many combinations of regression equation.

represented by the symbol “x”, and the data from the model prediction is represented by “o”. The error between the prediction and actual data is as high as 20%.

To overcome this problem a new set of rational functions was introduced into the regression equation (Equation (3.2)). The complete set of terms used by the GMDH in this work has the following form. There are eleven different terms used in this model.

$$\left\{ 1, (x_1, x_2), (x_1^2, x_2^2), (x_1 x_2), \left(\frac{1}{x_1}, \frac{1}{x_2}\right), \left(\frac{1}{x_1^2}, \frac{1}{x_2^2}\right), \left(\frac{1}{x_1 + x_2}, \frac{1}{x_1 x_2}\right), \left(\frac{x_1}{x_2}, \frac{x_2}{x_1}\right), \left(\frac{x_1}{x_1 + x_2}, \frac{x_2}{x_1 + x_2}\right), \left(\frac{x_1 + x_2}{x_1}, \frac{x_1 + x_2}{x_2}\right), (\sin(x_1), \sin(x_2), \cos(x_1), \cos(x_2)) \right\} \quad (3.4)$$

These new terms facilitated in achieving a better prediction. Parsimony in the choice of these terms must be exercised in order to maintain computational efficiency.

In this work an algorithm was developed to select different combinations of basic functions (Equation 3.4). The selected combination of functions is then used by the GMDH regression equation in a systematic manner. All possible combinations of terms are tested, such as one by one, two by two, three by three, etc. A binary number generator was used to make this selection automatically. For the case of a maximum of k candidate terms, this binary number goes from 1 to $2^k - 1$. For example, assuming that $k = 8$, we have a binary number from 1 to 256. When the binary number is 173, that is

$$[1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1],$$

just the following terms would be used by the GMDH in the regression equation.

$$\left\{ 1, (x_1^2, x_2^2), \left(\frac{1}{x_1}, \frac{1}{x_2}\right), \left(\frac{1}{x_1^2}, \frac{1}{x_2^2}\right), \left(\frac{x_1}{x_2}, \frac{x_2}{x_1}\right) \right\} . \quad (3.5)$$

For each possible combination of the basic functions, the GMDH algorithm estimates the value of the dependent variable. Then an overall residual value between the data points and the predicted value of the best regression equation found by the GMDH is computed. This overall residual is compared with residuals from other models. Figure 3.6 shows the plot of the residuals of each model as a function of the binary number. From this plot we can see that the 10 lowest residuals correspond to the models whose terms are generated from the following binary numbers

$$[255 \ 191 \ 251 \ 247 \ 187 \ 179 \ 183 \ 243 \ 254 \ 127].$$

The binary representation of the above integers (8-bit binary numbers) is given as

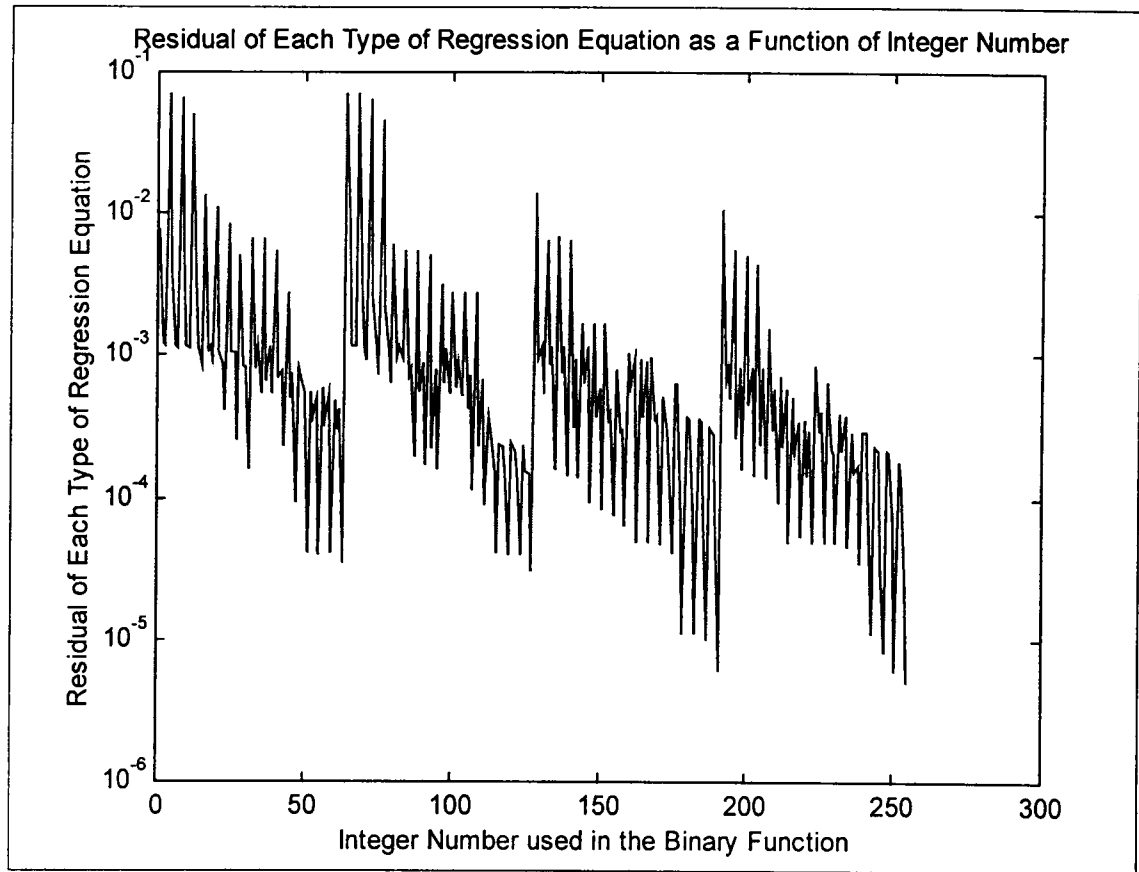


Figure 3.6. Residuals for different regression equations.

1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1
1	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1
1	0	1	1	1	0	1	1
1	0	1	1	0	0	1	1
1	0	1	1	0	1	1	1
1	1	1	1	0	0	1	1
1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1

Assuming that all these regression models give a satisfactory residual, then one could choose the model that uses the least number of terms, in this case the model number 179. The plot of the predicted and actual dependent variable obtained using the GMDH as a function of test data index for this specific model (model 179) is shown in Figure 3.7. The predicted and actual values are very close after the introduction of rational terms in the set of basic functions.

3.7. APPLYING GMDH METHOD IN AN FDI ALGORITHM

The GMDH method finds a mathematical expression which maps a given dependent variable y to a set of independent variables $\{x_1, x_2, \dots, x_k\}$. The user qualitatively defines which independent variables are physically related to the dependent variable y . This definition can come from the knowledge of a system engineer, from theoretical simulations, or even from mathematical tools such as correlation coefficients. In this work, the latter technique was used to obtain the set of (inputs, output) variables.

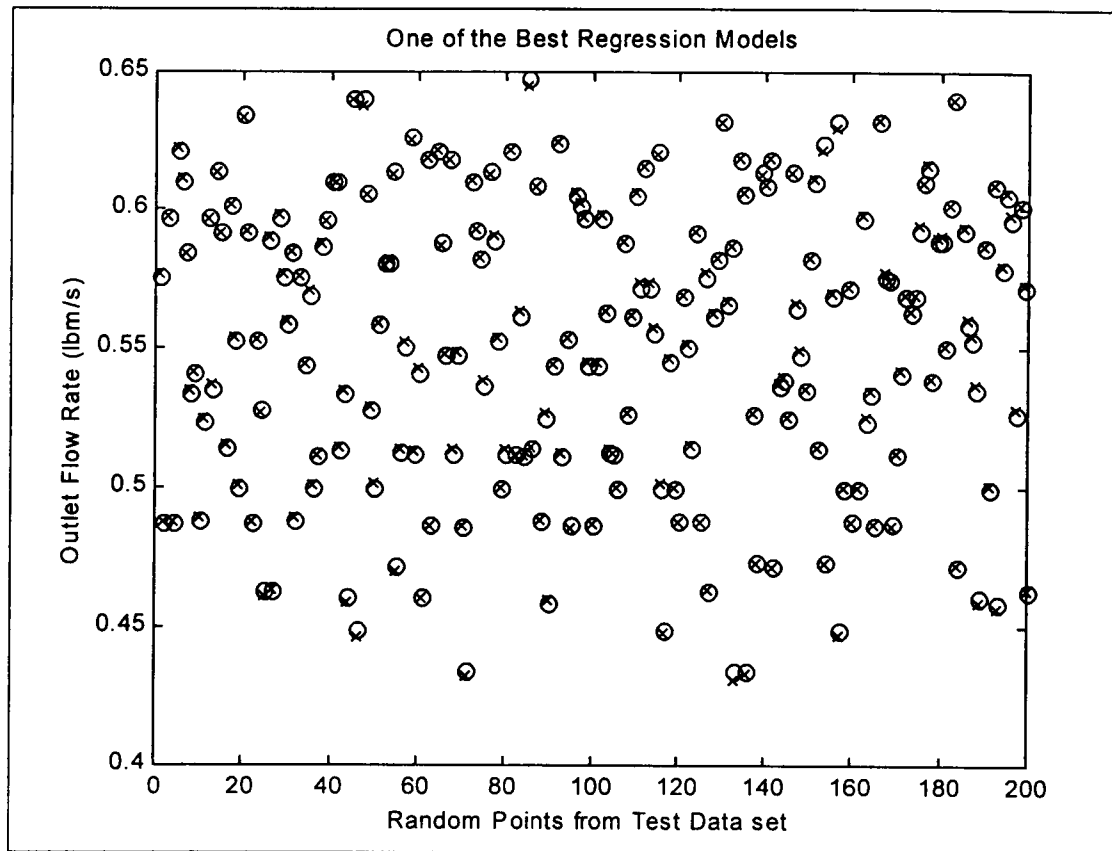


Figure 3.7. One of the best combinations of GMDH regression equation.

The mathematical relationships generated by the GMDH are used for predicting the output value of each loop component. This predicted value (also called the analytical measurement) is compared to actual measurements. If the difference between predicted and actual measurements is greater than a pre-defined threshold, it is concluded that a fault has occurred in the system (fault detection). As discussed in Chapter 1, the fault can be either in the device itself or in other loop components. A rule-based expert system is used in this work for isolating the probable faulty devices.

CHAPTER 4

DEVELOPMENT OF A MATHEMATICAL MODEL OF A LOW-PRESSURE WATER LOOP SYSTEM AND TEST FACILITY FOR FDI FEASIBILITY STUDIES

4.1. INTRODUCTION

Theoretical and experimental studies were performed for feasibility studies of the Fault Detection and Isolation (FDI) method proposed in this work. For the theoretical study, a simulation model of a process control loop, including sensors, controllers, and actuators, was developed. This model was implemented in Matlab-Simulink™ [90] program environment. For experimental studies a low-pressure water loop (LPWL) system was designed and built in the Nuclear Engineering Department. A LabView [65] program was developed to acquire loop measurements and to control the experiment remotely. For both systems, known faults were imposed on different devices, such as pressure transmitters, motor-operated valves, control elements, and a centrifugal pump. The purpose of the model and the test system was to provide useful data and an environment for developing and testing the proposed FDI algorithm described in Chapter 2. Data from all available sensors for normal loop operation and for the case with faulty devices were used to build a database. This database was used for the following tasks:

- To obtain mathematical relationships among loop devices by applying the GMDH method.

- To develop a rule-based expert system for fault detection and isolation.
- To develop a fuzzy-logic engine for possible on-line loop component reconstruction.

In the following sections, a description of both the theoretical model and the experimental system is presented along with results of simulation and control loop tests. Chapters 6 and 8 present the analysis of the data collected in both systems and the development of the FDI algorithm.

4.2. DEVELOPMENT OF A LOW-PRESSURE WATER LOOP SYSTEM

MODEL

A low-pressure water loop (LPWL) system model was developed in a Matlab-Simulink [90] environment and was used to provide data for FDI feasibility studies. This theoretical model simulates an experimental water loop system fabricated at The University of Tennessee. A schematic of this water loop is shown in Figure 4.1. The actual experimental loop setup was used for performing the same tests in a real system, where noisy data and other technical problems are present.

4.2.1. Model Components

The LPWL system has the following components:

1. Inlet water piping with:
 - A Worcester series 75 motor-operated valve (identified as Inlet MOV).
 - Rosemount mass flow meter model 3095 MV (a multivariate pressure transmitter).

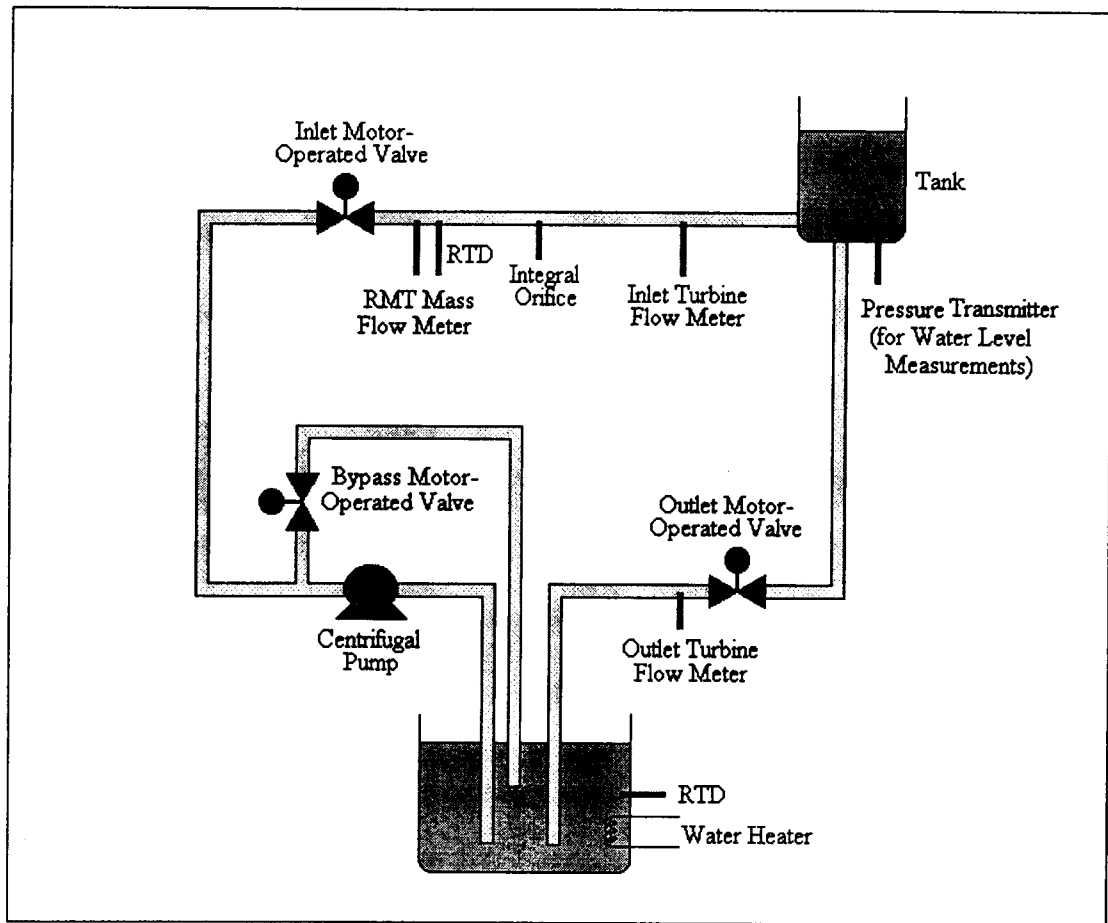


Figure 4.1. A schematic of the modeled low-pressure water loop system.

- An integral orifice device used as one of the 3095 MV inputs.
 - A resistance temperature detector (RTD) used as the second 3095 MV input. The water temperature is used for compensating the value of water density in the mass flow rate calculations.
 - A turbine-type water velocity meter. Using the water temperature from the RTD, a mass flow rate value can be obtained from this device. This turbine is identified in this work as Inlet Turbine Flow meter.
2. Bypass water line with another Worcester series 75 motor-operated valve. This valve is used to simulate faults due to aging of the constant speed centrifugal pump. Changing the valve position simulates pump discharge changes due to the aging process.
 3. Water tank. The LPWL system has a water level controller. A Rosemount pressure transmitter model 3051 is installed at the bottom of the water tank. The pressure of the column of water in the tank is linearly proportional to the water height. The water level controller uses the pressure transmitter output value.
 4. Outlet water piping line with the following devices.
 - An MOV which is used to simulate either obstruction in the outlet piping due to material deposits or a fault in the MOV itself. This MOV is identified as Outlet MOV.

- A turbine-type water velocity meter, which is identified as Outlet Turbine Flow meter.
5. Water reservoir with a water heater element. This heater element is used to keep the loop water at a constant temperature. This temperature is around 110 °F. If no heat is provided, the pump heat will increase the loop water temperature up to 95 °F over a period of 6 hours of loop operation.

4.2.2. Model Equations

A theoretical model was developed in Simulink [90] for a detailed simulation study. Simulink is a powerful programming environment used for modeling, simulation, and analysis of dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. Simulink includes a comprehensive block library of sinks, sources, and other components. Model analysis tools include linearization and trimming tools, which can be accessed from the Matlab command line. The user can revise the models either in Matlab or in Simulink due to their integrated feature.

The theoretical model developed in this work has all the components described earlier. Figure 4.2 shows the main screen of this model in the Simulink environment. Some of the model components were developed using pre-recorded experimental data. Figure 4.2 does not show subsystems associated with some of the model blocks, such as

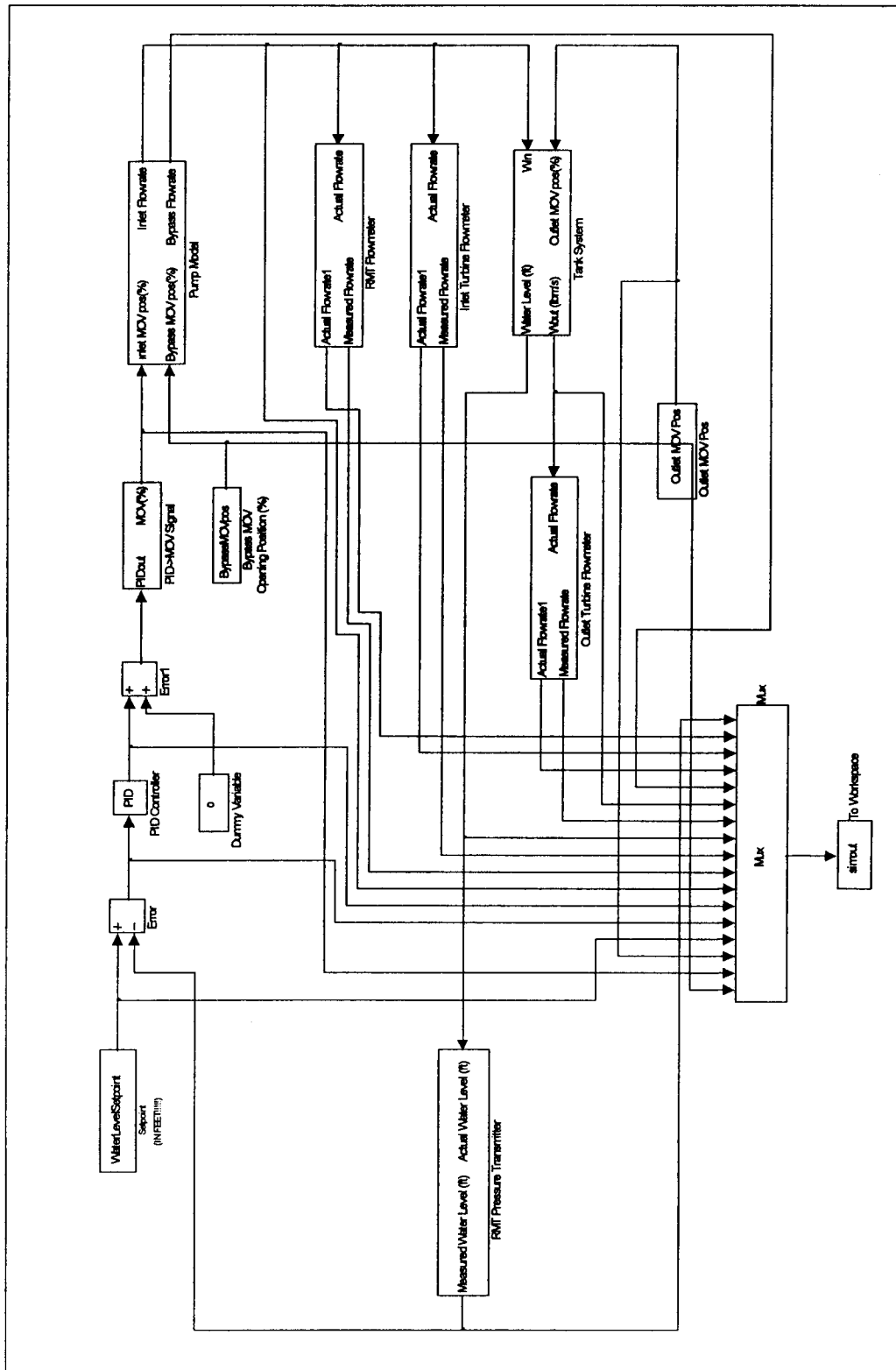


Figure 4.2. Low-Pressure Water Loop (LPWL) system theoretical model in Matlab-Simulink

the pump model or the water tank model. Table 4.1 shows all system variables available for possible analysis.

Water Tank Model:
$$\frac{dh}{dt} = \frac{W_{in}}{a_T \rho} - \frac{a_p}{2a_T} \sqrt{\frac{2g}{h_0}} h(t)$$

Where $a_T = \pi r^2$ is the area of the tank

$a_p = \pi * \text{OMOVP} * C$ is the cross-sectional area of the tank outlet piping. This term has a dependency on the outlet MOV position (OMOVP). Changes in the MOV position creates changes in the piping area by a factor C (C was obtained experimentally).

$\frac{W_{in}}{a_T \rho}$ is the forcing function of the above differential equation

$\frac{a_p}{2a_T} \sqrt{\frac{2g}{h_0}} h(t)$ is the tank water loss term

Pump Model:
$$\text{IFR} = \frac{\text{IMOVP}}{\text{IMOVP} + \text{BMOVP}} * \text{PD} \quad \text{and}$$

$$\text{BFR} = \frac{\text{BMOVP}}{\text{IMOVP} + \text{BMOVP}} * \text{PD}$$

where: IFR = Inlet Flow Rate going through the tank inlet piping

BFR = Bypass Flow Rate going through the bypass piping of the system

IMOVP = Inlet MOV position

BMOVP = Bypass MOV position

PD = Pump discharge.

Table 4.1. System variables for the LPWL model.

	System Variables	Signal Type
1	Water Level Set Point	Internal
2	Water Level in the Tank	Input
3	Bypass MOV Position	Output
4	Inlet MOV Position	Output
5	Outlet MOV Position	Output
6	Inlet Turbine Flow Velocity	Input
7	RMT Mass Flow Rate Meter	Input
8	Outlet Turbine Flow Velocity	Input
9	Bypass MOV Feedback Position	Input
10	Inlet MOV Feedback Position	Input
11	Outlet MOV Feedback Position	Input
12	Inlet MOV PID Controller Action	Output
13	Reservoir Tank Water Temperature	Input
14	Water Temperature in the Tank	Input
15	Heater Element PID Controller Action	Output

The models for the existing sensors in the loop consist basically of first order linear equations, where the slope and the interception parameters were obtained from experiment. The "RMT pressure transmitter", "RMT flow meter", "Turbine-type flow meter (inlet)", and "Turbine-type flow meter (outlet)" are the sensor models used in this simulation study. Also, inside each of these models, a second component can be added to the model output. This term is used for simulating a drift component, when studying faults in the sensor.

4.3. DEVELOPMENT OF A LOW-PRESSURE EXPERIMENTAL CONTROL LOOP SYSTEM

The LPWL experimental system setup was designed and built in the Nuclear Engineering Department at The University of Tennessee. This loop was also used to generate data and for checking the FDI algorithm performance in a real system. The loop components are described in Section 4.2. A description of the experimental setup and its components is given below, along with some observed technical problems during either system commissioning or experiments.

4.3.1. Description of the Low-Pressure Experimental Control Loop System

Figure 4.3 shows the Worcester series 75 motor-operated valve (MOV) with its yellow upper cap in place. To control the MOV position through a computational interface, special circuit board was installed in all three MOV bodies (as shown in Figure 4.4). Many technical problems were observed when commissioning these valves:

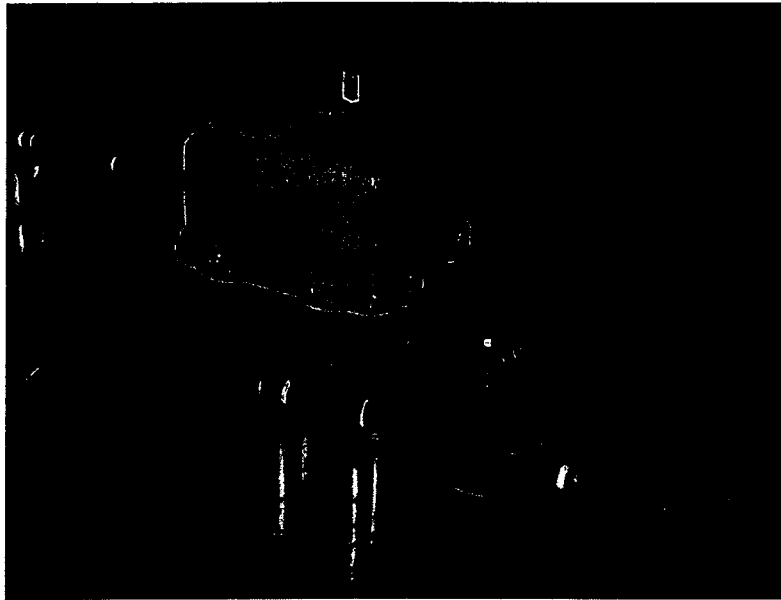


Figure 4.3. Worcester series 75 motor-operated valve.



Figure 4.4. Circuit board for valve position control.

- There was approximately a 10-degree (in angle) of valve free motion. This freedom was more significant in case of changes of valve movement direction such as open-to-close to close-to-open and vice-versa. In these cases, the motor would rotate, with its positioning sensor indicating valve rotation, although in reality the valve itself would not change its position at least until the motor gear rotated about 10 degrees. Significant oscillations in the valve position were observed due to the valve controller (PID), which was trying to match the tank water level set point without success. A few hardware modifications in the valve structure were made for reducing the valve movement freedom.
- The valve positioning sensor device was losing its calibration every other day of use. After a long inspection in the internal valve parts, a design flaw was discovered: one of its moving parts was physically touching the sensor potentiometer not allowing it to move, even though the valve was moving. This flaw caused the observed calibration problems. Necessary adjustments were made to solve this problem.
- Due to some actuator response delay and due to 0.5% inherent movement dead band in the MOV, the ordinary PID controller was having difficulties in keeping the water level at the desired set point. An adaptive PID controller was developed and implemented to overcome this problem. The PID proportional and integral constants were modified according to the set point error. The proportional constant (K_p) was set to large values for large set point errors and set to small values for small set point errors. The integral constant (K_i) was set to zero for large error values and set to

large values for small set point errors. Figures 4.5 and 4.6 show K_p and K_i values as a function of the set point error.

- The implementation of this adaptive PID controller significantly improved the water level control. The overshoot traditionally seen in the PID controllers was also minimized. Figure 4.7 shows the actual water level for different set points.

Three Motor-Operated Valves (MOV) were used in this system. They were identified as “Bypass MOV,” “Inlet MOV,” and “Outlet MOV” (Figure 4.8). The Bypass MOV was used to redirect part of the pump discharge to the reservoir to simulate aging faults in the constant speed centrifugal pump. The constant speed pump used in the experimental setup is shown in Figure 4.9.

Figure 4.10 shows the Rosemount mass flow meter model 3095 MV (a multivariate pressure transmitter). It has an integral orifice device used as one of the 3095 MV inputs (also shown in Figure 4.10). A resistance temperature detector (RTD) was used as the second 3095 MV input. The water temperature is used for compensating the value of water density in the mass flow rate calculations. The RTD was placed inside the water tank. It can be partially seen in Figure 4.11.

Two other flow meters were placed in the system. One was placed in the inlet piping and the other in the outlet piping, as seen in Figures 4.8 and 4.12. These devices are turbine-type water velocity meters (TFM). Figure 4.13 shows its internal turbine part. The TFM output is water velocity and not mass flow rate. To convert it to a mass flow

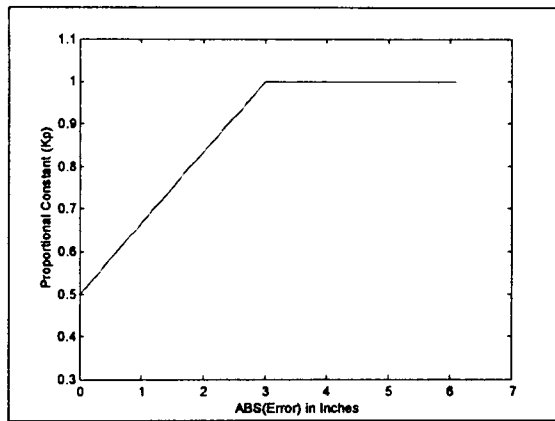


Figure 4.5. K_p as a function of set point error.

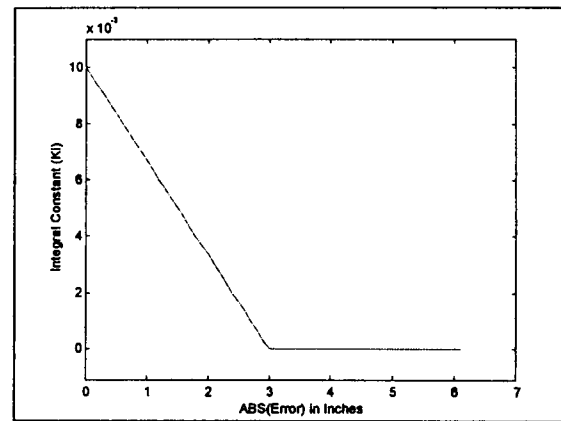


Figure 4.6. K_i as a function of set point error.

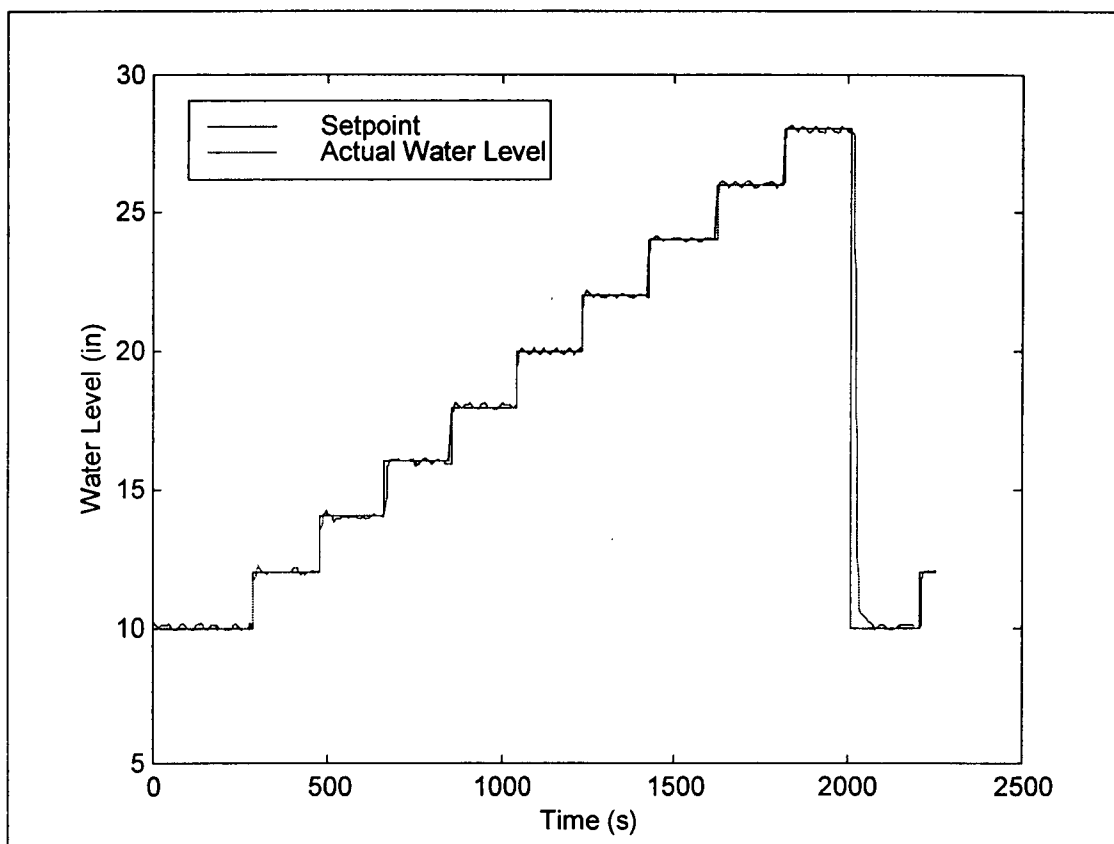


Figure 4.7. Changes in the actual water level for different set point values.

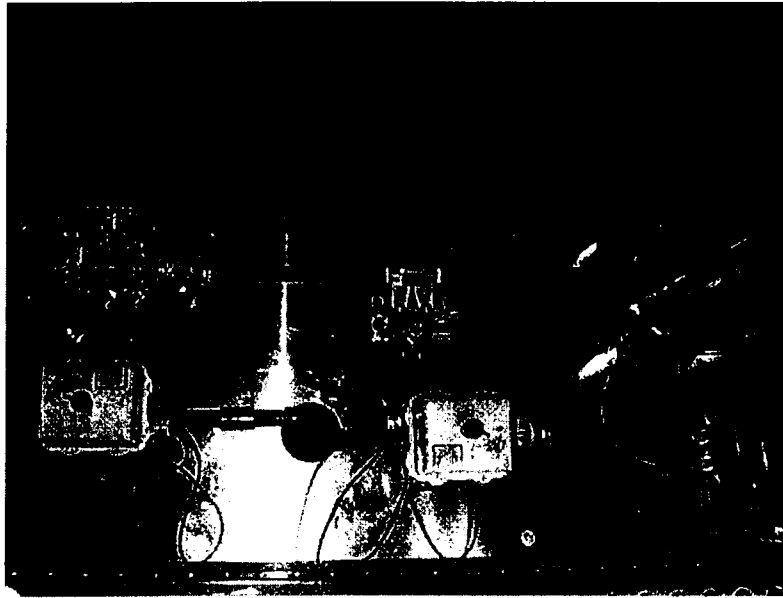


Figure 4.8. Motor-operated valves. From top to bottom, left to right: inlet MOV, bypass MOV, and outlet MOV.

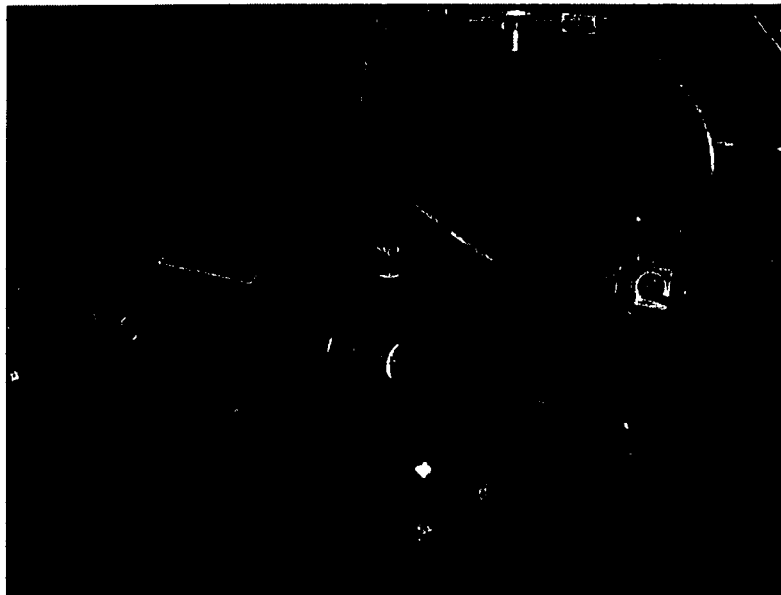


Figure 4.9. Constant speed centrifugal pump.



Figure 4.10. Rosemount multivariate pressure transmitter model 3095 MV for mass flow measurements.



Figure 4.11. A view of the water tank. A pressure sensing line (white piping) and a RTD (red wires) are inserted in the tank from its bottom part.



Figure 4.12. Turbine type water velocity meter.

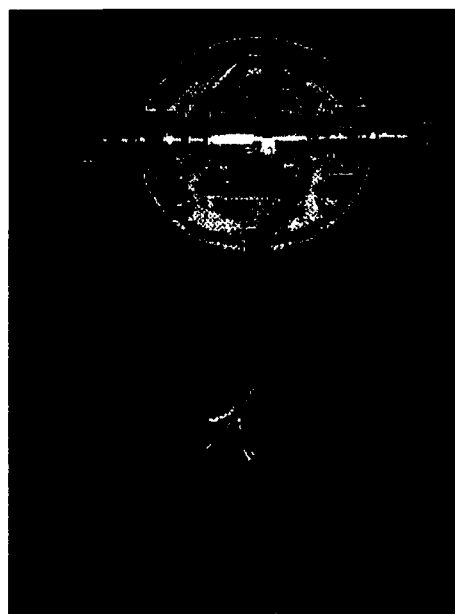


Figure 4.13. A view of the turbine flow meter (TFM) internal parts.

rate, correction for the water density as a function of temperature was necessary. An extra RTD was placed in the water reservoir and its signal was used to compute these corrections.

The LPWL system has basically two controllers acting at the same time: one controlling the water level and the other controlling the water loop temperature. For the water level controller, a Rosemount pressure transmitter model 3051 was installed in the system, with its sensing line connected at the bottom of the water tank (Figure 4.14). The pressure of the column of water in the tank is linearly proportional to the water height.

For the water temperature control, a Rosemount temperature transmitter was installed. This transmitter requires a temperature signal from an RTD, which was placed in the water reservoir for this purpose. The temperature set point was maintained using electrical heater elements (Figure 4.15).

The loop was built on top of an aluminum table placed on a mobile frame structure as shown in Figures 4.16 and 4.17. The power supplies for the loop components were centralized in one single panel as shown in Figure 4.18. TFM signal conditioners were also placed in that panel. The input/output signals of the various loop devices were transmitted from/to a second patch panel as shown in Figure 4.19. This patch panel was connected to three National Instruments data acquisition boards. These boards were plugged in the available slots inside a personal computer (Figure 4.20). The main purpose of these boards was to acquire data from the loop devices and to send signals to the ones that require an input signal, such as the MOVs and heater elements. A LabView

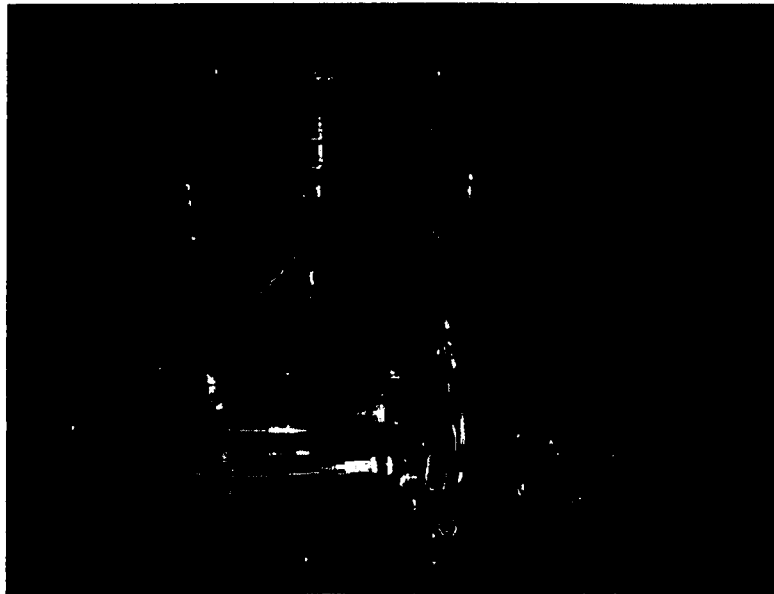


Figure 4.14. Rosemount pressure transmitter for water level measurements.



Figure 4.15. Water heater element (in red).

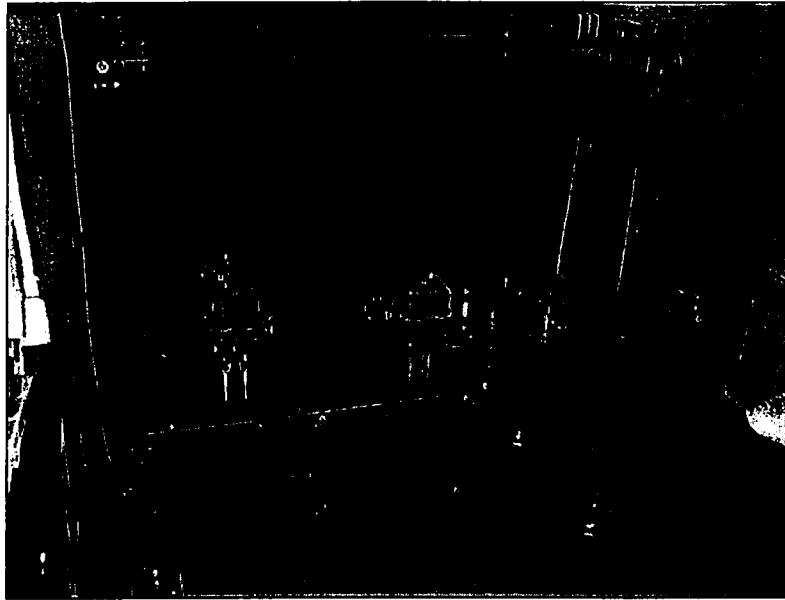


Figure 4.16. A View of the entire low-pressure water loop experimental setup.

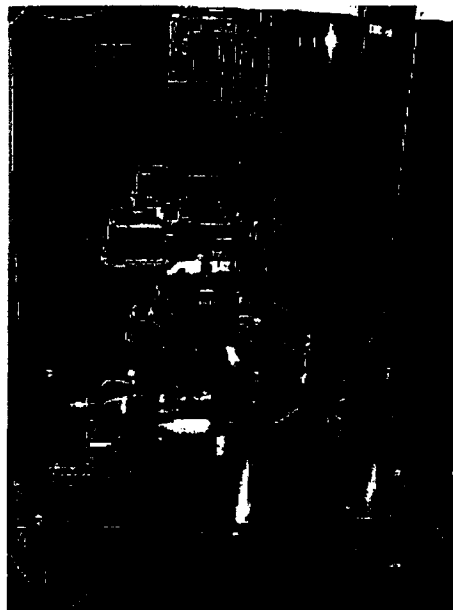


Figure 4.17. Side view of the low-pressure water loop experimental setup.

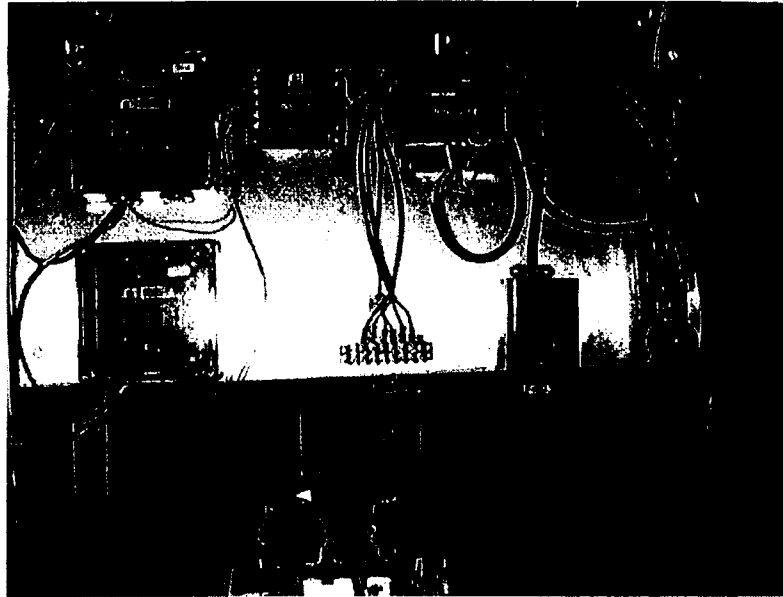


Figure 4.18. Power supply and signal conditioners panel.

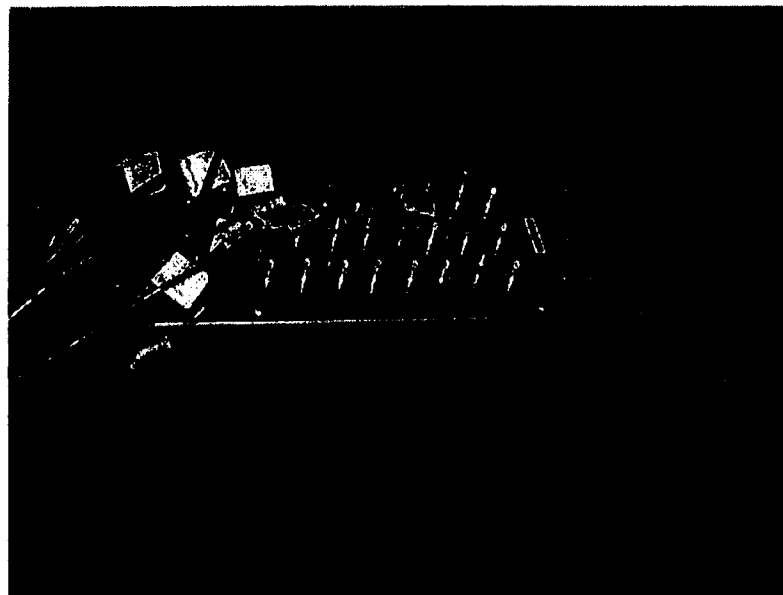


Figure 4.19. Patch panel for data acquisition and experiment control.

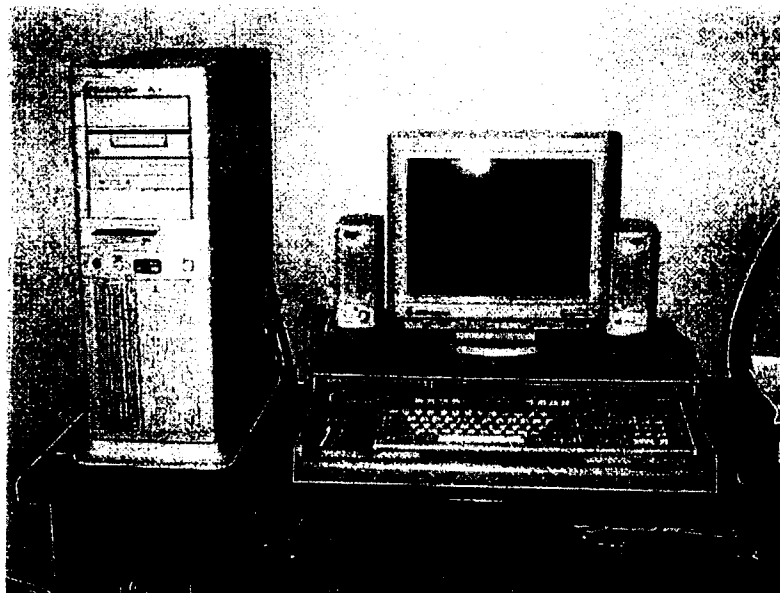


Figure 4.20. Computer used for data acquisition, data analysis and experiment control.

program developed in this work performed all the signal transmissions and the experiment control. The main screen of this program is shown in Figure 4.21.

LabView is a program development application software. It uses a graphical programming language to create programs in a block diagram form. LabView is a general-purpose programming system with extensive libraries of functions for any programming task. It includes a library for data acquisition, data analysis, data representation, and data storage. These libraries are very important for interfacing computer and external devices, controlling experiments, and displaying results.

Results of simulations and experiments performed in both theoretical model and experimental setup are presented in Chapter 7. The fault detection and isolation algorithm, developed in this dissertation, is also presented in that chapter.

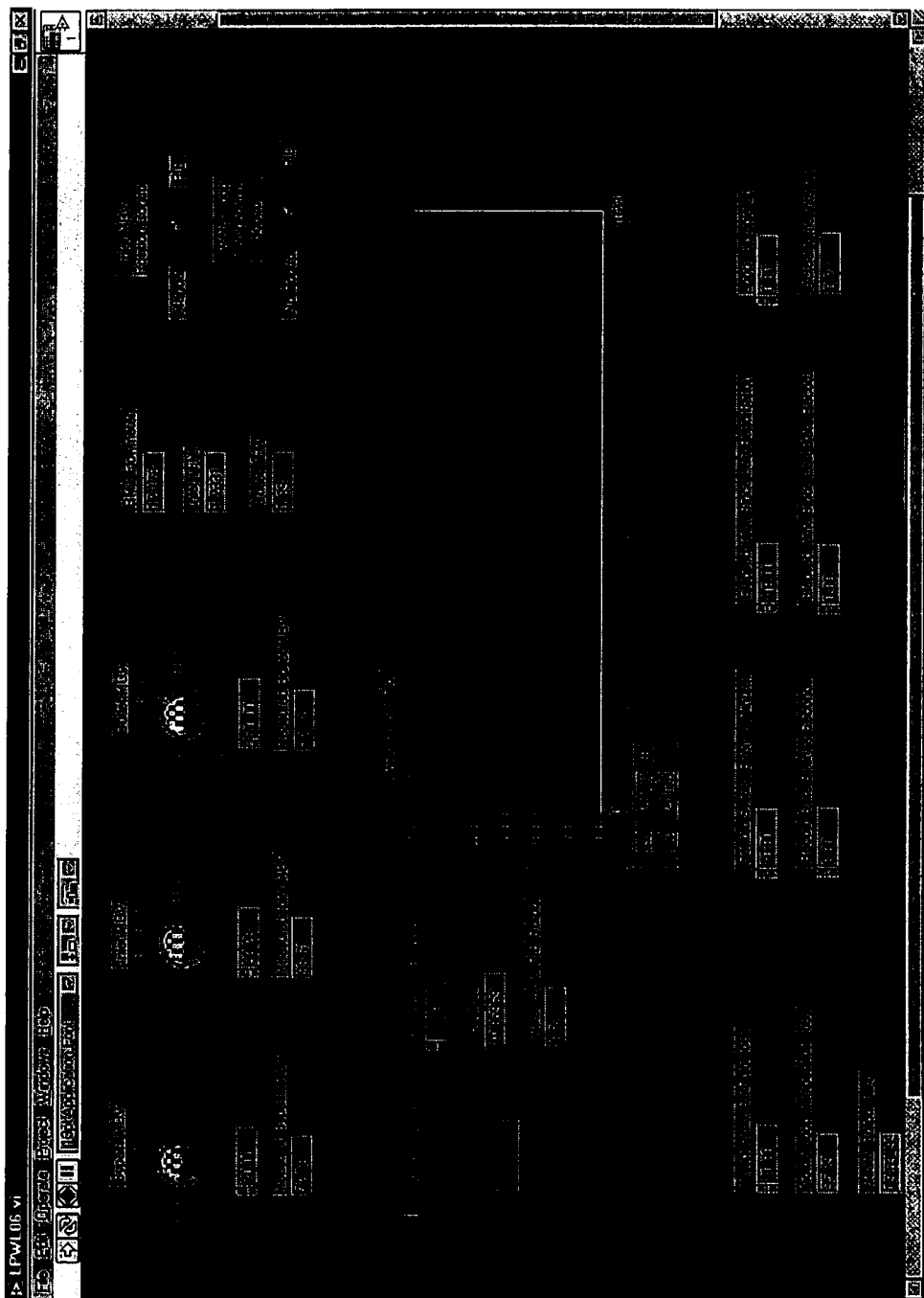


Figure 4.21. LabView data acquisition and experiment control program.

CHAPTER 5

SINGLE SENSOR VALIDATION IN PROCESS CONTROL SYSTEMS

5.1. INTRODUCTION

A new technique, developed for validating isolated sensors in process control systems, is described in this chapter. This technique is based on multi-resolution signal decomposition using wavelet transforms and on mapping of control variables and actuator dynamics. The cases of open-loop (sensors used for monitoring) and feedback dynamics (sensors used as a controller input) were considered. No redundant measurements were assumed. The primary objective was to separate process dynamics from sensor dynamics in operating distributed control systems. An algorithm, which uses multi-resolution capability of wavelet transforms, was developed to update the sensor's output. In situations where the control feedback may conceal sensor faults, a mapping between the controlled variable and the corresponding actuator system was developed. The proposed technique was successfully tested for validating sensors in two different scenarios. In the first, a temperature detector was monitored using experimental data under strong noise contamination and in the second, a level detector in a feedback system was monitored.

5.2. A BRIEF OVERVIEW

Most signal validation techniques require measurements from multiple process sensors and a prediction approach based on physics or data-driven models [1,2]. This chapter presents a new approach for validating single, isolated sensors. Local sensor validation is designed to be at the lowest level of a hierarchical fault detection system. This validation must be fast and accurate, and must be performed at the transmitter level without the use of a separate processor [3-6].

In this research, the nature of the process and sensor bandwidths under normal conditions is assumed to be known. The goal is to separate sensor dynamics from process dynamics. This requires selective filtering of the signals, often referred to as sub-band analysis. An algorithm that uses wavelet transforms and multi-resolution analysis was developed for detecting sensor faults and for reconstructing the sensor output. The new algorithm automatically selects the optimal signal decomposition level and updates the reconstructed sensor output in an on-line form.

The cases of open loop and closed loop (with feedback) situations were considered. In the case of a feedback control system, a mapping to relate the controlled variable and the corresponding process actuator was developed in order to cancel the effect of feedback on sensor output. In this chapter, the following cases were studied with applications to actual data:

1. Recovery of sensor drift from highly oscillatory data.
2. Reconstruction of sensor drift in feedback systems.

3. Validation of single sensor in an experimental feedback system.

Section 5.3 presents the problem statement. Signal decomposition using wavelet transforms and the new signal validation algorithms are presented in Section 5.4. Section 5.5 describes the application of this technique to sensor drift recovery in a highly oscillatory process. Single sensor validation in a process with feedback is described in Section 5.6. Finally, an application to a laboratory feedback system is presented in Section 5.7. Concluding remarks are given in Section 5.8.

5.3. PROBLEM STATEMENT

The local sensor validation is designed to be at the lowest level of a hierarchical fault detection system. This validation must be fast and accurate, and must be performed at the transmitter level without having to use a separate processor [3-6]. The output of a sensor may be represented as

$$Y(t) = Y_p(t) + Y_s(t) + n(t) \quad (5.1)$$

where $Y(t)$ = measured sensor output

$Y_p(t)$ = part of $Y(t)$ due to process changes

$Y_s(t)$ = part of $Y(t)$ due to intrinsic changes in sensor characteristics

$n(t)$ = measurement noise.

The various sensor signal components are assumed to be additive in nature. During plant operation it is expected that both process and sensor variations occur.

In most processes the component $Y_p(t)$ is band-limited. The measurement noise $n(t)$ generally has a higher frequency range. The process and sensor variations may be in the same bandwidth. We want to identify the portion of the signal that is outside this range, at a frequency lower than the process bandwidth. These are issues to be addressed in actual applications. Thus the learning of the process and sensor characteristics is very important and must be part of the knowledge base for each type of process plant and the transmitters used for measurements.

5.4. SIGNAL DECOMPOSITION USING WAVELET TRANSFORMS AND VALIDATION ALGORITHM

In this section, sensor signal decomposition using wavelet transforms for sensor validation is proposed. Problems of edge effects when applying multi-resolution wavelet analysis to a finite length of data is discussed and solutions to that problem are presented. Finally, this methodology is applied to a faulty temperature sensor and the results are discussed in detail.

5.4.1. Multi-Resolution Analysis and Signal Decomposition

From the observation made above, one approach for pre-processing the sensor data is to use digital filters. However, a digital filter has certain drawbacks. A digital

filter is a stationary transform and may not work effectively for multi-resolution analysis of nonstationary data [13,115]. The edge effects are more pronounced while using digital filters. These issues can be easily addressed with wavelet transforms. Furthermore, signal reconstruction with wavelet transforms has a high accuracy, as compared with digital filters [115].

In wavelet analysis the original signal (with a Nyquist folding frequency of N_f) passes through two filter banks and emerges as low-frequency ($0-N_f/2$) and as high-frequency ($N_f/2-N_f$) components. As shown in Figure 5.1, these are referred to as approximation (A) and detail (D) components, respectively. This procedure may be repeated (see Figure 5.2) to achieve decomposition of the signal at any pass-band. This is called the wavelet decomposition tree. The components cA_i 's and cD_i 's may be combined to reconstruct the original signal with a minimum loss of information.

5.4.2. Edge Distortion

When convolution is performed on finite-length signals, edge effects often appear, in the form of signal distortion near the edges of the data window. Various methods have been suggested to deal with this problem [115]. Often simple schemes have been used based on signal extension or padding of the signal beyond the active data window. These and other methods were tried for solving this problem, the most effective being the padding of the data window, at the start and the end, by actual process data. This procedure requires the availability of small portions of the signal beyond the active data window; but this is not a major constraint.

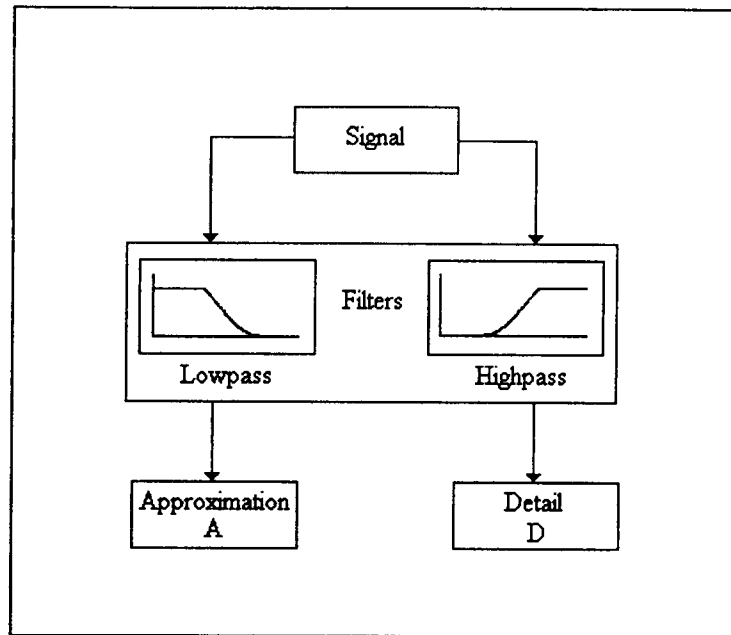


Figure 5.1. Signal decomposition into low and high frequency bands.

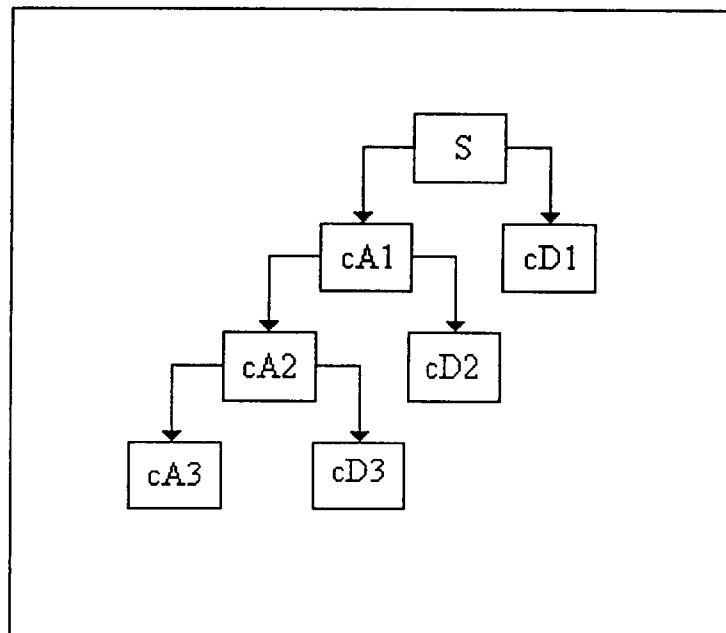


Figure 5.2. Successive decomposition of a signal into different pass-bands using wavelet transform.

5.4.3. Application to Temperature Data from RTD Experiments

Our current objective is to verify the feasibility of using multi-resolution wavelet analysis (MRWA) as applied to a non-stationary signal to remove the process fluctuations and reconstruct the drift in one Resistance Temperature Detector (RTD), identified here as RTD #A-44. This sensor was introduced for thermal aging into an industrial furnace in the laboratory. The RTD experienced an aging process due to the high temperatures it is exposed to. In this specific case, the aging process observed was a constant slow drift as seen in the upper plot of Figure 5.3. An artificial process fluctuation was introduced (middle plot).

The composite signal is shown in the lower plot of Figure 5.3. About eight months of data are shown in these plots. The artificial fluctuations correspond to a maximum temperature variation of 5 °C. A data window of one-month length was used in recovering the RTD drift signal. Figure 5.4 shows the results of MRWA using the Matlab Wavelet Toolbox [115]. Figure 5.4 shows that the best signal decomposition level is the one labeled as *a6*. Even with the best available Matlab routines, significant edge effects have been observed. Only about 35% of the original signal could be recovered from this analysis.

A second approach was then tried to resolve this problem:

1. Get one month of data.
2. Append one week of real data at the start and end of the active window.
3. Fit a quadratic polynomial to the data and extrapolate beyond the appended data.

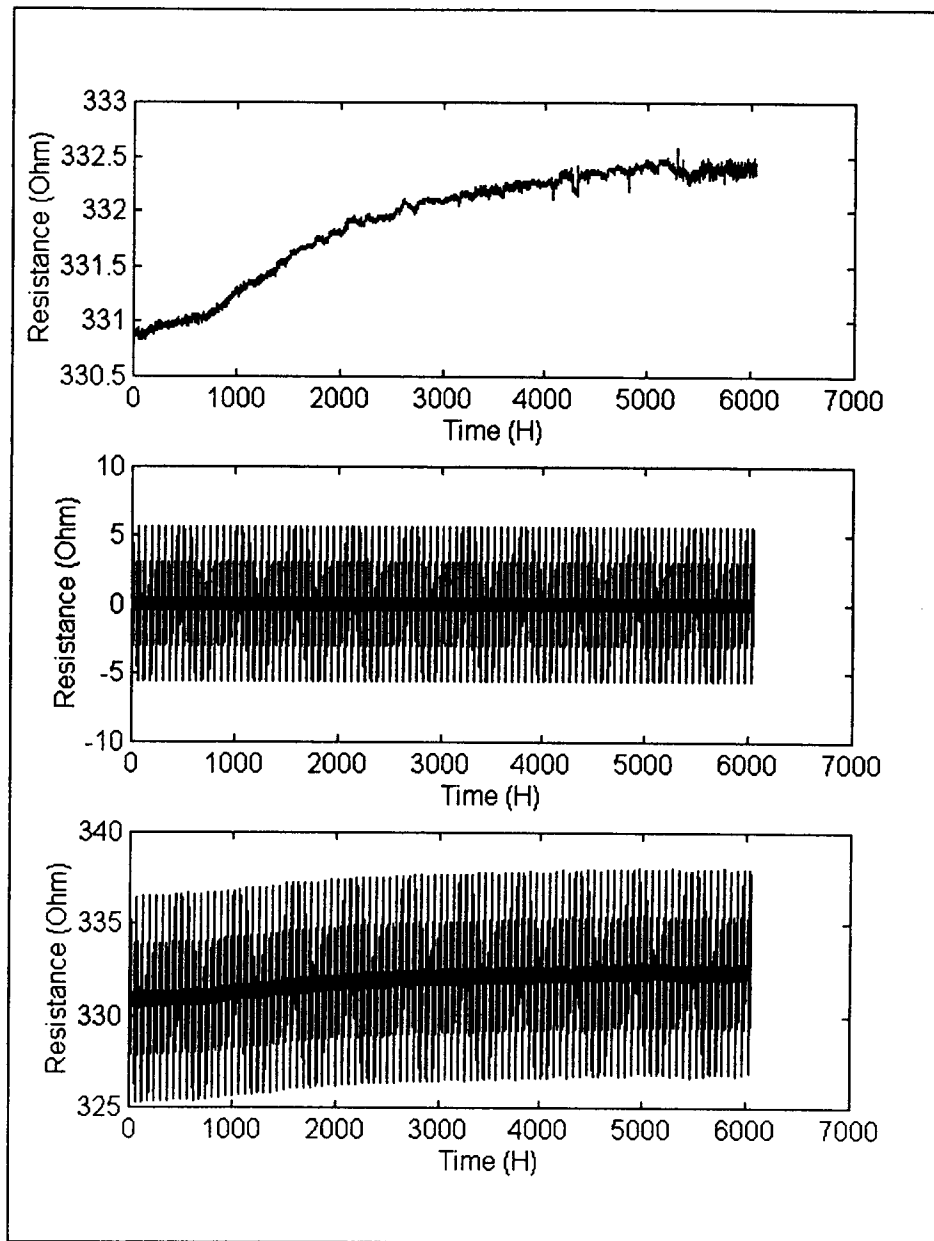


Figure 5.3. Illustration of combining actual sensor output (a) with process fluctuation (b) to provide the composite signal (c). The original signal (a) is from RTD #A-44.

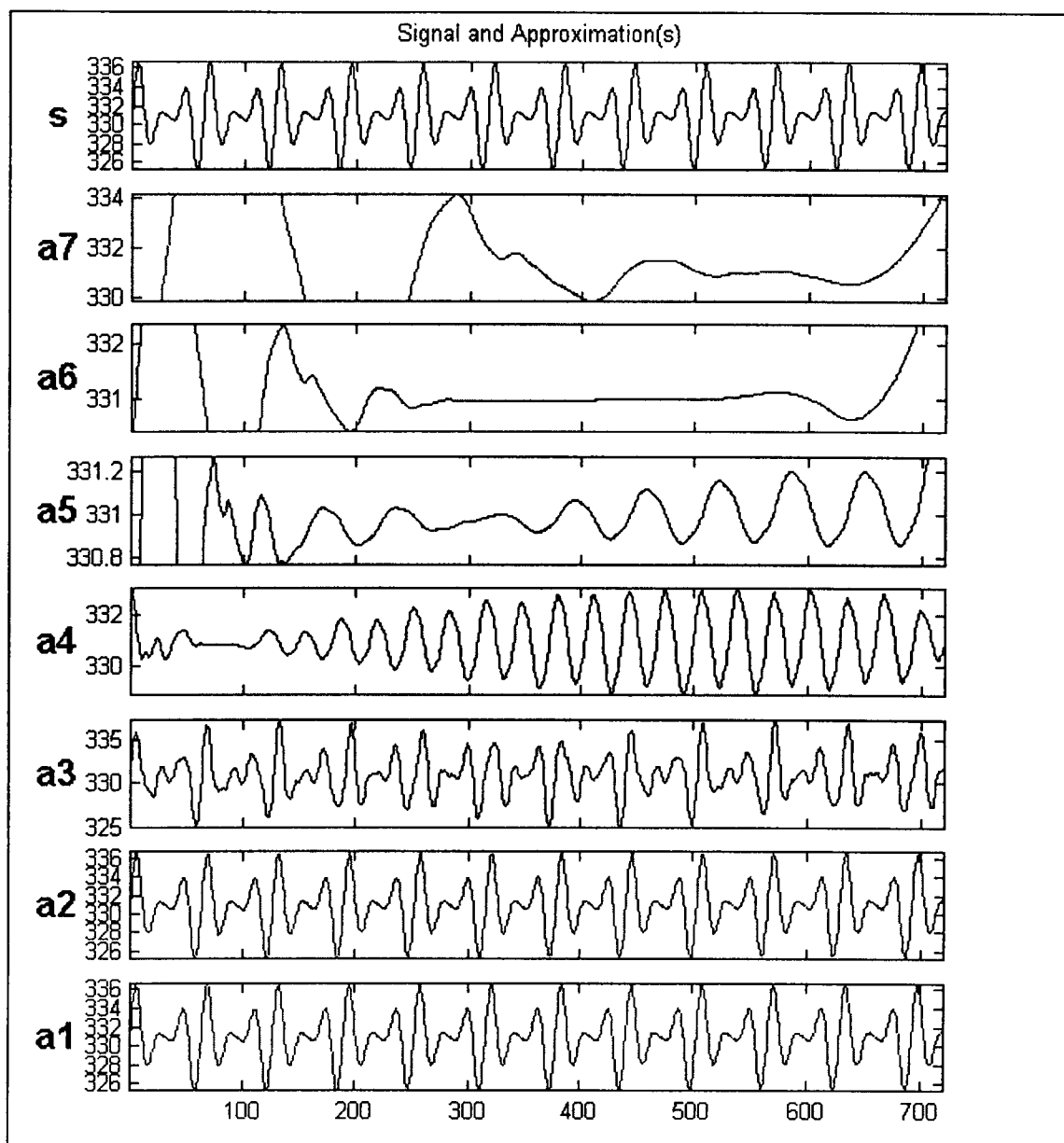


Figure 5.4. Multi-resolution analysis of RTD signal.

The new signal, derived from the updated procedure, is shown in Figure 5.5. The plot in region I represents the polynomial extrapolation, and the plot in region II shows the extra week of data appended to the active window (one month long) shown in region III. The result of MRWTA is shown in Figure 5.6. The points in region I are part of the extrapolated data, and the points in region II correspond to one-week appended data. Region III shows the filtered signal and the original signal from RTD #A-44. The signal in the active window is exactly reproduced.

5.4.4. New Signal Decomposition Algorithm

The above algorithm was used to develop a Matlab-based routine for signal decomposition. The new procedure is presented below. Again RTD #A-44 is used as the example signal.

1. Get one month of data.
2. Append one week of data at the start and end of active data window.
3. Make a quadratic fit to the total signal.
4. Extrapolate 25% beyond the data window on either side.
5. Apply MRWA and make decomposition from levels 4 through 9.
6. Establish the best decomposition level by computing the total absolute error between successive levels and determine the minimum error to identify the best signal decomposition.
7. The best monthly data windows are then concatenated to establish the total signal.
8. The procedure is repeated continuously in an on-line fashion.

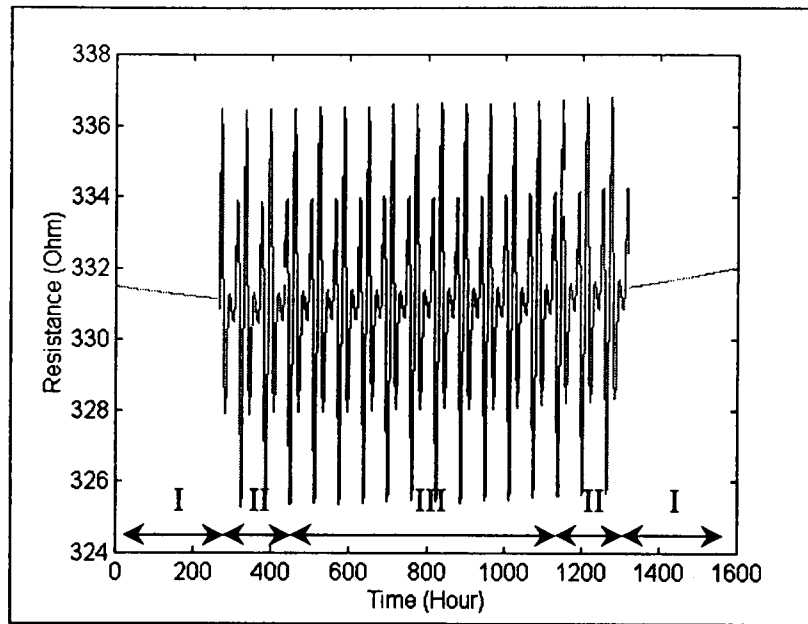


Figure 5.5. Appended one-month data window with one week of data and quadratic extrapolation on each side.

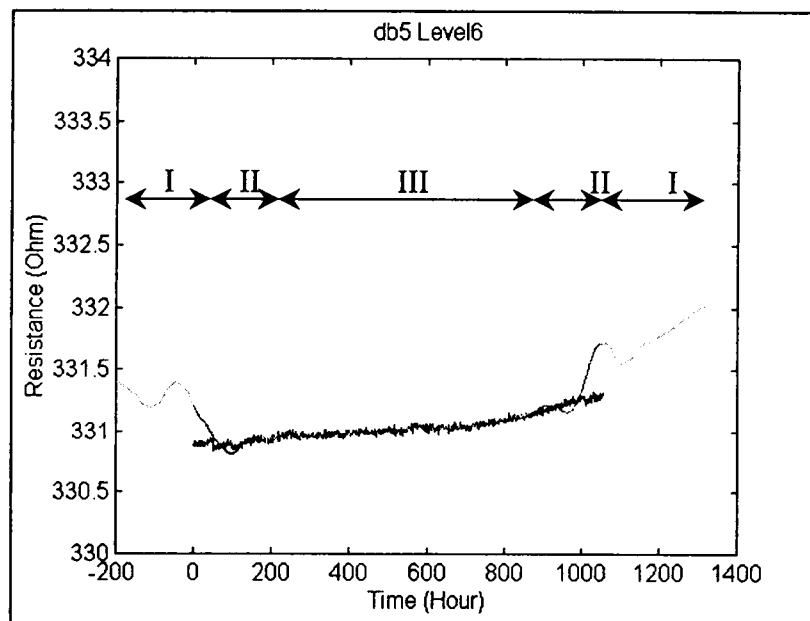


Figure 5.6. Results of multi-resolution analysis. The important portion of the filtered signal is within region III.

A comparison of the original and the processed signal from RTD #A-44 is shown in Figure 5.7. The results indicate excellent performance of the new signal processor, which could be used for single-sensor validation. It is important to note that the process variable used in this example is not affected by any feedback or control actions.

Tests were performed using different RTD drift trends and different process oscillations, and it was observed that the best decomposition level could be obtained from applying a residual comparison technique. Of course, residuals between the recovered signal and the RTD drift signal could not be computed because the latter is not available in practice. (This is what we pursue in this work). The residuals were computed between subsequent levels of decomposition. For all cases tested, the residuals between the first decomposition levels are very high, but they have a negative trend as the level increases. However, this trend changes sign after a few more decomposition levels. At this point the recovered signal starts to lose the information about the sensor drift. Table 5.1 shows the residuals between decomposition levels for a given set of data.

5.5. SENSOR DRIFT RECOVERY IN A HIGHLY OSCILLATORY PROCESS

Two of the RTD resistance measurements from a second industrial furnace were used to demonstrate the aspect of recovering average values of signals under highly oscillatory process conditions. This second furnace was operating in a temperature cycling mode. The temperature range is from 550°C to 750°C with a period of 2 cycles per week. The two RTDs used here were the ones labeled as RTDs #A-74 and #A-87.

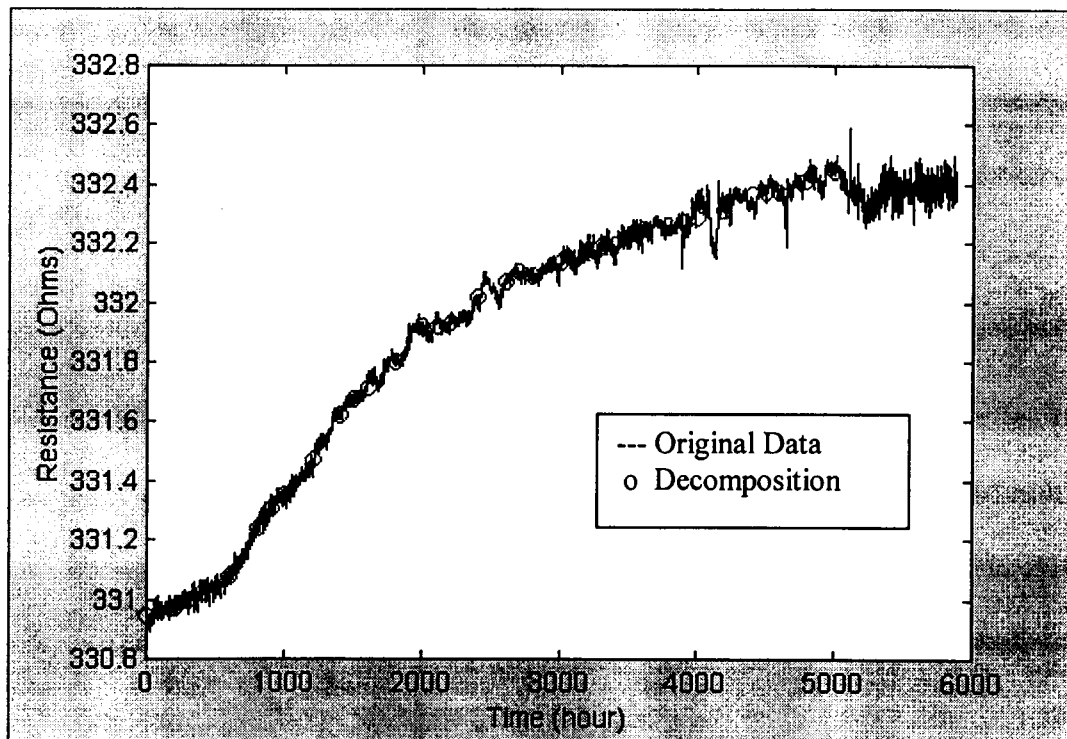


Figure 5.7. Comparison of the actual RTD #A-44 signal and the processed signal using multi-resolution wavelet analysis.

Table 5.1. Residuals between decomposition levels.

Decomposition Levels Used	Residuals Between Levels
Level 4 and Level 5	514.1533
Level 5 and Level 6	49.8176
Level 6 and Level 7	10.3949
Level 7 and Level 8	22.3630
Level 8 and Level 9	70.4206

Figure 5.8 shows a portion of the measurements from RTD #A-74. The multi-resolution analysis (sub-band analysis) was performed using wavelet transforms. The recovered average signal is also shown in Figure 5.8 as a smoother line. Figure 5.9 shows the recovered averaged signals RTD #A-74 and RTD #A-87, along with a third RTD (RTD #A-82) that has remained at an approximately constant level.

This approach of recovering signals from high-background noise is recommended for monitoring detectors for calibration and for on-line adjustments in the measurements given by such drifting RTDs. Furthermore, the recovered signal can be used for RTD life prediction.

5.6. SINGLE SENSOR VALIDATION IN A PROCESS WITH FEEDBACK

In the first phase of this task we demonstrated the use of multi-resolution analysis for a sensor measurement without any process feedback. The next logical step is to consider the case where the sensor output is fed into a process controller. In the following sections, a simple system is studied. In this system, the only two variables are the sensor and the controller outputs. This system is used for feasibility studies. A more general approach is presented in chapter 6, where a complex system is used.

In the case of a sensor-controller system, the drift in the sensor (or controlled variable) signal will not be seen because the controller is constantly adjusting the system to provide sufficient actuation to maintain the value of the measured variable at the desired set point. Therefore, one would see drift in the controller's output. It is then

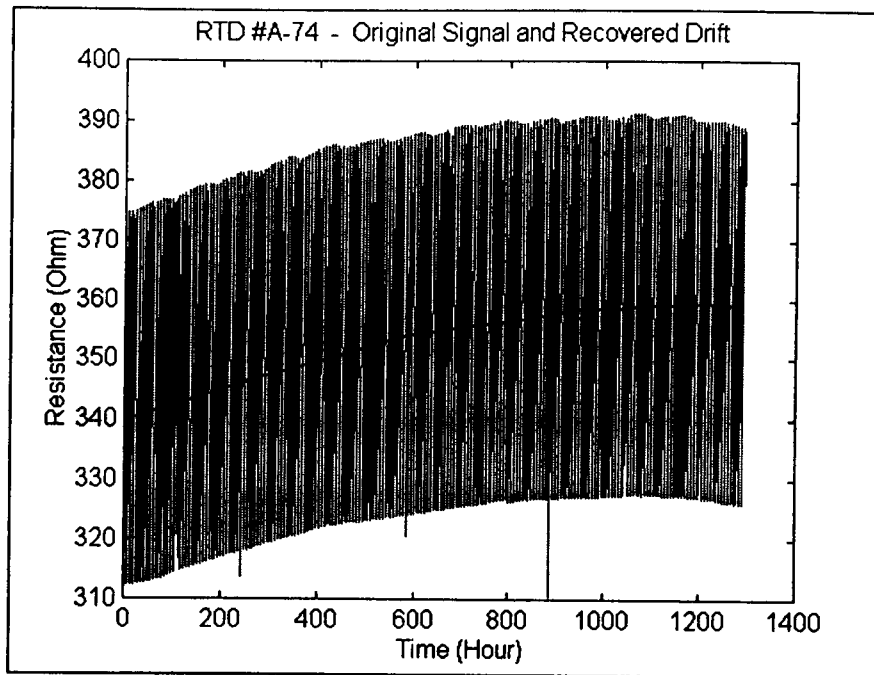


Figure 5.8. Recovered average signal.

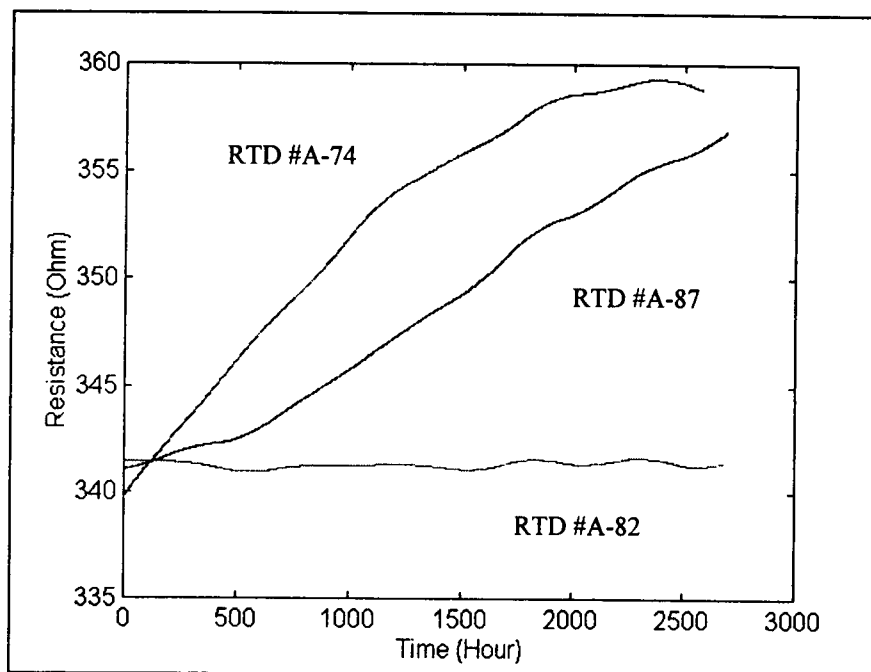


Figure 5.9. Plot of recovered average signals from three RTDs.

necessary to correlate the controller drift with the sensor drift during the “learning phase.” During the “application phase” the controller output is mapped back to obtain the sensor drift.

First a closed-loop system was simulated, where an artificial drift was added to the original sensor signal. Later, a real experimental setup was used to verify the performance of the developed technique over real conditions. In both cases, the following steps are followed for recovering the sensor’s drift:

- a) A relationship between the controller output and the sensor drift is required.
- b) The P-I controller output is recorded for analysis.
- c) The changes in the controller output are due to sensor drift.
- d) The sensor drift is used as input to the single sensor validation algorithm.

Figure 5.10 shows the block diagram of the algorithm used to accomplish this task. A mapping between the P-I controller output changes due to known sensor drifts was developed from the simulation. Incorporating successive step changes in the sensor output generated the sensor anomaly. A quadratic fit was generated between the sensor drift and the corresponding controller output.

The experimental drift shape observed in RTD #A-44 was used as the sensor drift in this simulation. The sensor in question is a pressure transmitter that gives the water level in the tank. Figure 5.11 shows the sensor output during the simulations. This signal remained approximately constant because of the controller action, which is plotted in Figure 5.12. Using the previously established functional relationship between the sensor

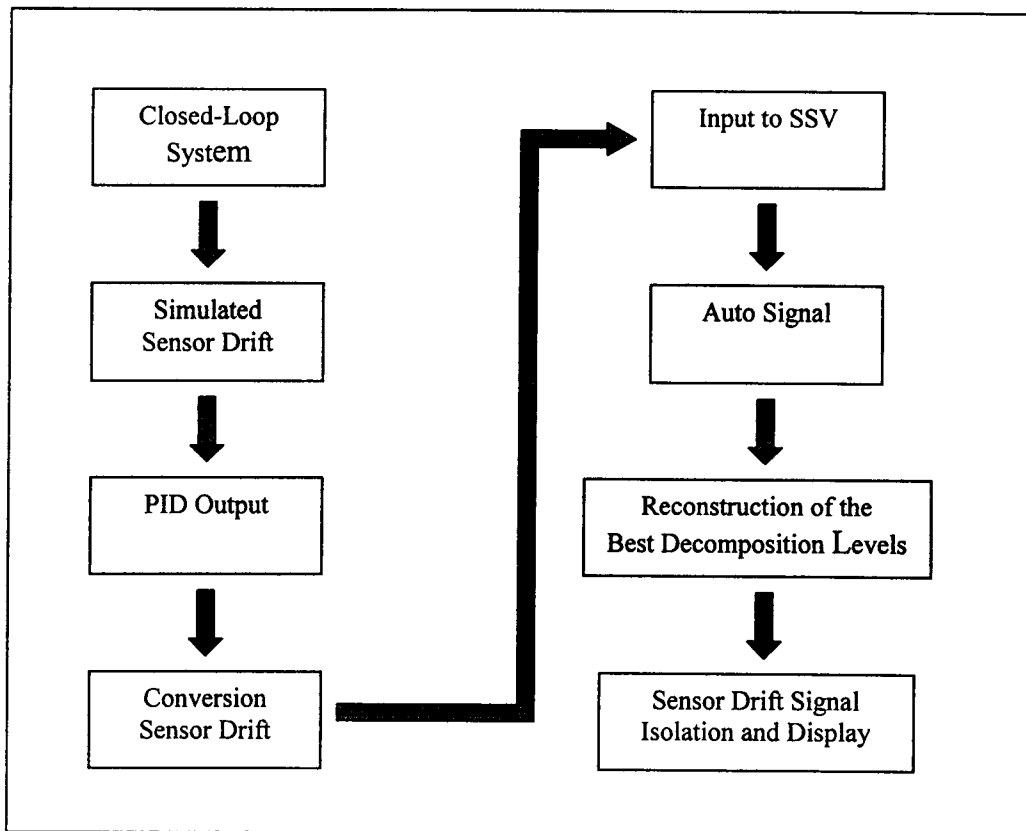


Figure 5.10. Block diagram of the procedure used for the case of sensor validation in a closed-loop system.

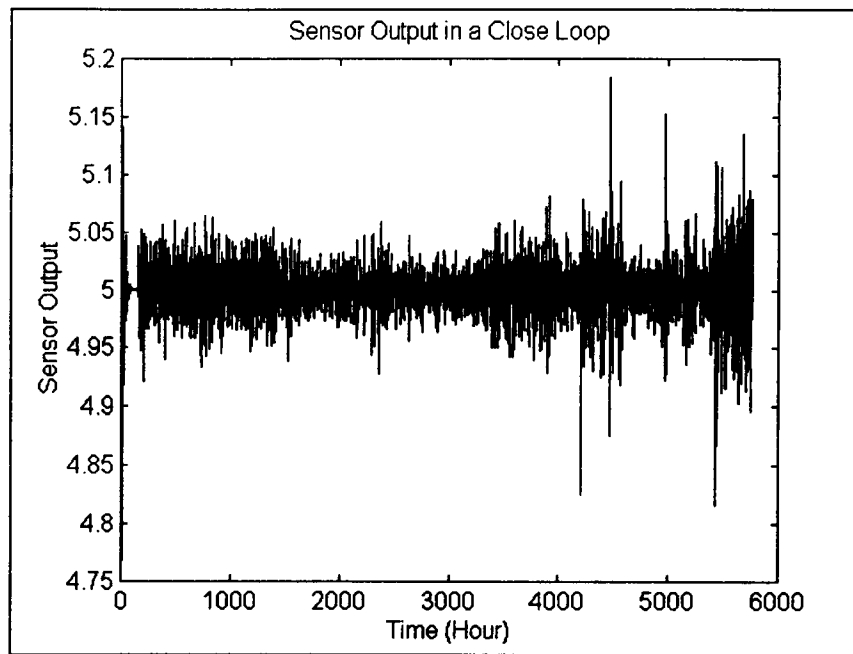


Figure 5.11. Sensor output. Water level is kept at 5 inches. Noise comes from the real sensor drift data used in this simulation.

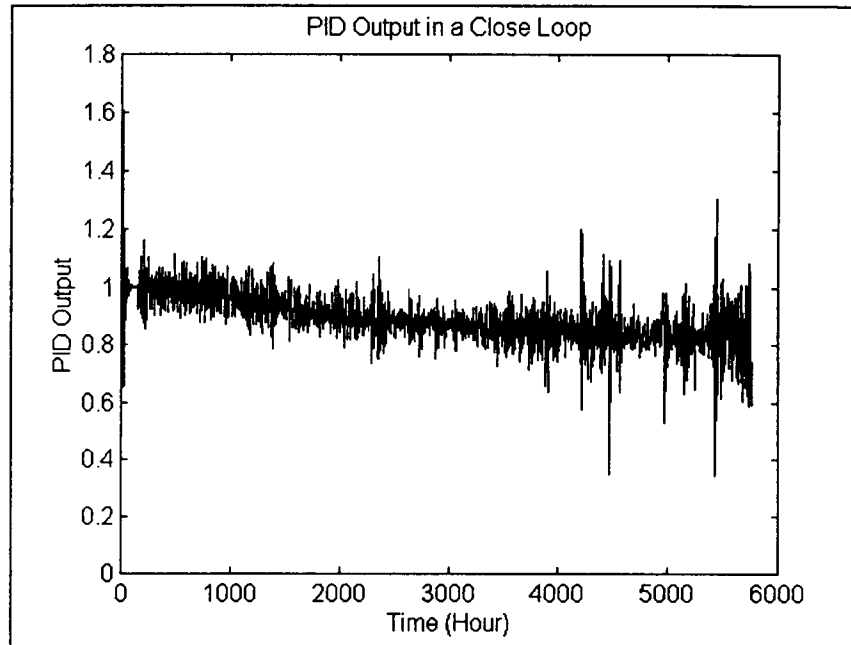


Figure 5.12. PID controller output drift as a function of time.

measurement and the controller output, the actual sensor output is recovered as shown in Figure 5.13. As a final step in the algorithm, the multi-resolution analysis is applied to recover the sensor drift. This step is shown in Figure 5.14. The imposed sensor drift is also shown in this figure. The technique indicates that the separation of process and sensor dynamics could be performed when the measurements are affected by process feedback.

5.7. SINGLE SENSOR VALIDATION FOR THE CASE OF A LABORATORY CONTROL SYSTEM

The application of the sensor validation method described in Section 5.5 is now illustrated for a laboratory water level control system. The measurements in the system include the water level in the tank and the controller output signal that was used for valve actuation. The procedure used for characterizing the sensor-controller relationship and for separating the process and sensor bandwidths is similar to the one employed in Section 5.5.

The following steps are executed in this procedure:

- Establishment of a quadratic functional form between the level sensor drift and the controller output.
- Insertion of an artificial drift in the sensor signal.
- Archiving of the controller output during the experiment.
- Calculation of the sensor drift using the controller output and the functional relationship previously obtained.

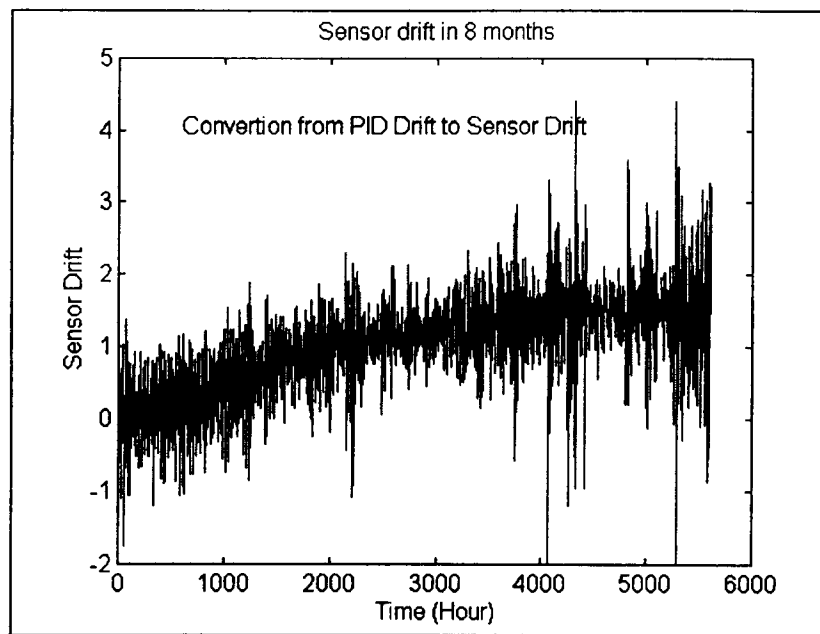


Figure 5.13. Noisy sensor drift obtained from the relationship between sensor drift and PID controller drift.

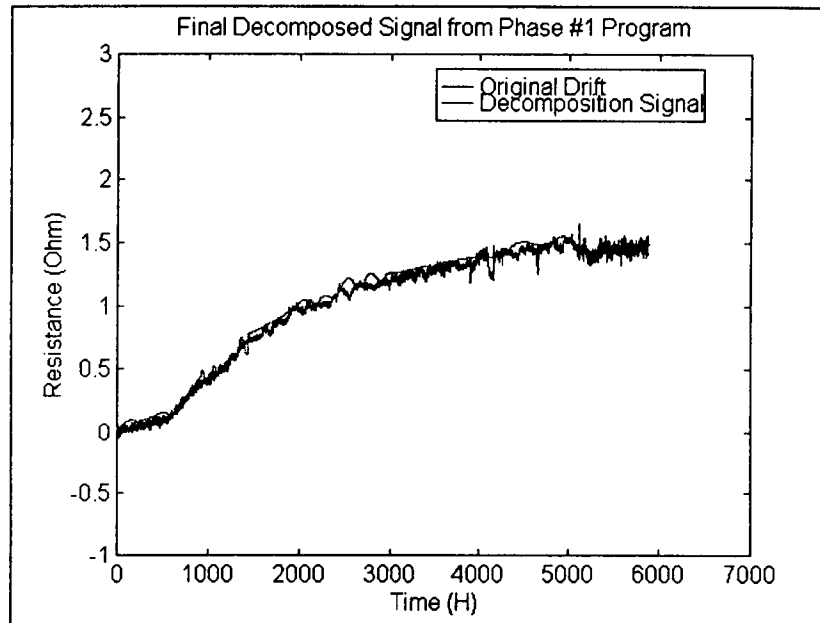


Figure 5.14. Comparison between sensor drift recovered from the controller output and the original drift data.

- Implementation of the multi-resolution analysis to recover the sensor drift from the noisy data.

The one-tank water level control experimental setup in the Nuclear Engineering Department was used for this demonstration. A LabView program was developed for data acquisition and to implement a P-I controller. A drift component similar to the $\tan(x)$ function was added to the level measurement.

Figure 5.15 shows the actual level measurement, and Figure 5.16 shows the controller output. The recovered sensor drift computed from the mapping between the controller output and the sensor output is shown in Figure 5.17. The recovered sensor drift and the artificially imposed drift are compared in Figure 5.18. The results indicate excellent performance of the single-sensor validation method for the experimental case study with feedback.

It may be concluded that this new method for distributed sensor validation can be a powerful tool for application to process industry instruments. Chapters 6 and 8 focus on generalizing this approach to the fault detection and fault isolation of sensors, actuators, controllers, and other devices in a more complex process system.

5.8. CONCLUDING REMARKS

A new method, based on the integration of signal decomposition using multi-resolution wavelet analysis and the mapping of feedback control variables, was developed for fault detection and isolation of distributed sensors in process control

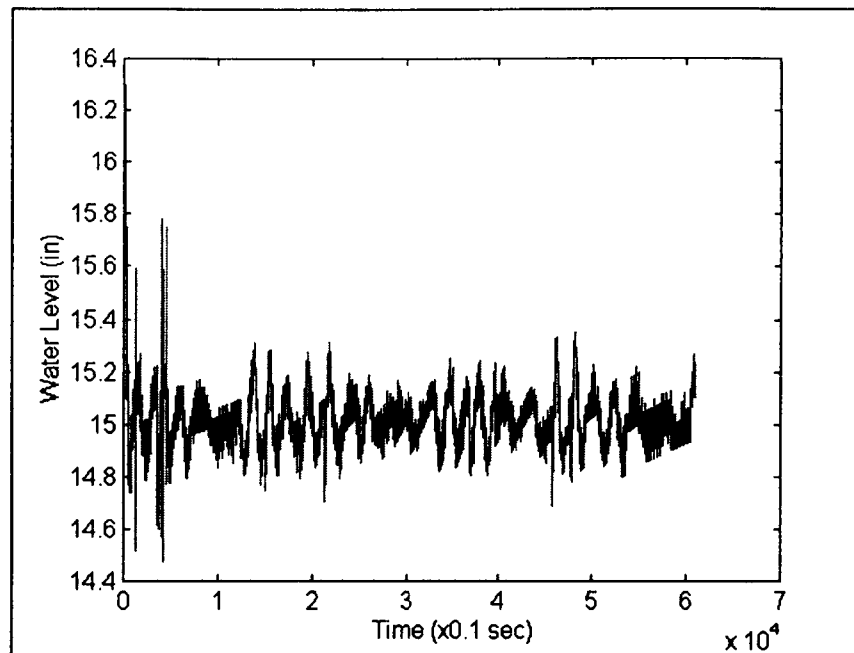


Figure 5.15. Water level signal which is kept constant during the entire experiment even though the sensor is drifting.

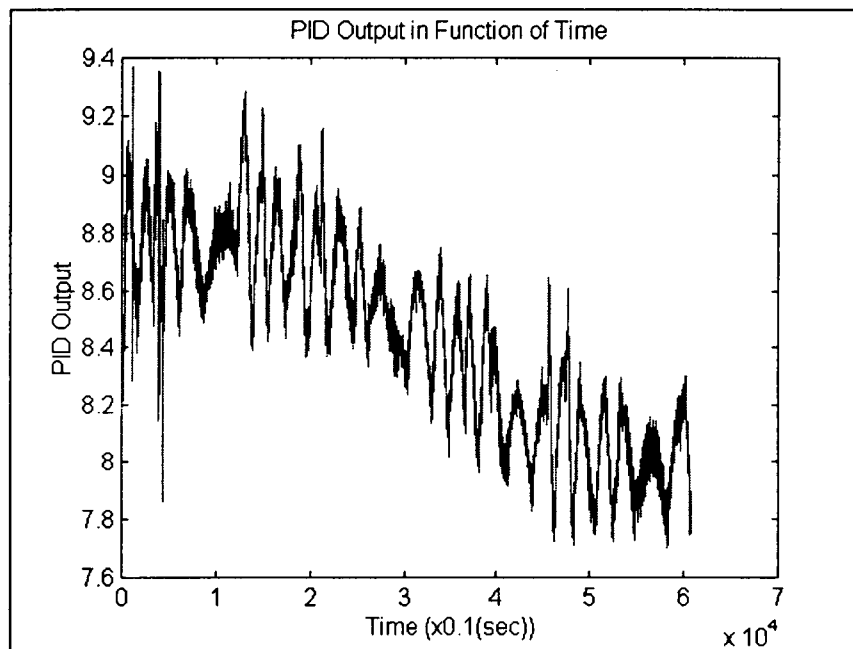


Figure 5.16. Controller changes through time due to the sensor drift.

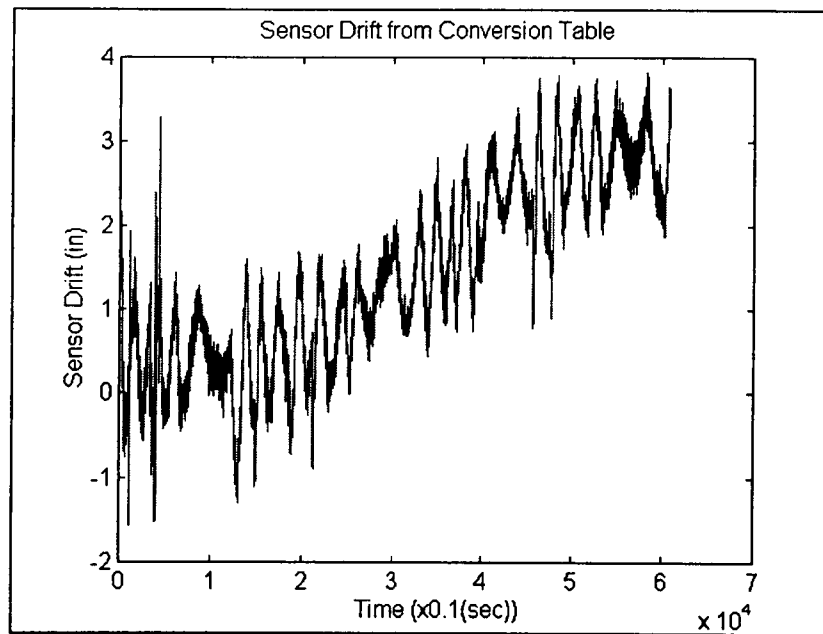


Figure 5.17. Sensor drift recovered from the controller output and the conversion table between the controller output and sensor drift.

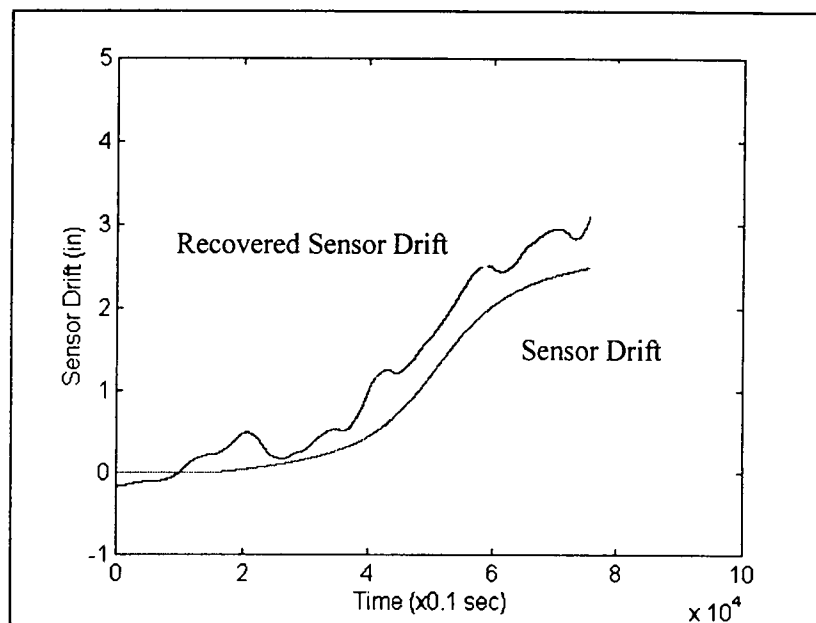


Figure 5.18. Comparison between the experimental sensor drift (after MRWA) and the original drift imposed on the sensor.

systems. The feasibility of this technique was evaluated using both open loop and closed-loop systems. The algorithm can be implemented on-line and is capable of retrieving sensor malfunction, even when the process has an oscillatory behavior. The use of wavelet transforms eliminates filter leakage effects and allows the on-line processing of non-stationary measurements.

CHAPTER 6

DEVELOPMENT OF A FAULT DETECTION AND ISOLATION ALGORITHM FOR A COMPLEX SYSTEM

6.1. INTRODUCTION

In the last chapter, the procedure for developing and implementing an FDI algorithm in a simple sensor-controller system was presented. For that case, it was assumed that the overall system condition was constant over time and that any change in the controller output was caused by a fault in the sensor. If any other component in this loop has a fault, such as the pump, the valve or even the controller itself, the above method would not give reliable results. Nevertheless, if the system is very simple and all other components condition can be verified by other means, the technique presented in Chapter 5 could still be applied.

In this chapter, the steps for developing and implementing an FDI algorithm for a more complex system is presented. Here, no assumptions are made either about the system operating condition, or about the degradation status of any loop component.

6.2. THE OVERALL PROCEDURE

As mentioned in earlier chapters, a Fault Detection and Isolation (FDI) algorithm is constructed in two steps: development and implementation. In the development phase,

mathematical models for analytical redundancy are derived, rules for expert system for fault isolation are defined, and a fuzzy logic algorithm for signal reconfiguration is created. Step two consists of implementing the FDI in a real system. In this second phase, the FDI algorithm tasks are:

- a) Retrieving all available sensor data from the loop.
- b) Computing the analytical measurements (predicted value).
- c) Cross-checking them with the actual measurements.
- d) Detecting and isolating possible faults.
- e) When appropriate, attempting to adjust the signal from the faulty sensor, so the system can run until the next scheduled maintenance without loss of performance.

A more detailed explanation is given in the following sections along with the development and implementation of an FDI algorithm, using the theoretical model of a low-pressure water loop system. In Chapter 8 the implementation of the FDI algorithm in a real system is described.

6.2.1. The Development of an FDI algorithm

As mentioned in earlier chapters, many important points must be addressed when developing a fault detection and isolation algorithm for a complex system. Faults in different types of components normally have different influence on a system's dynamic behavior. For example, one would expect that a fault in a temperature detector would influence the system differently from a fault in a water flow valve.

Other important consideration is the fact that although the system variables may change their values through time, it does not imply that a fault has occurred: the system may be operating at a different but normal condition. Therefore, a robust fault detection and isolation diagnostic system must not only be able to detect and isolate faults, but also be able to differentiate system changes due to a sensor's failure from system changes caused by changes in the process normal operational conditions.

Along with these aspects, an analytical redundancy should be used to improve the FDI robustness for more complex systems. In an analytical redundancy, a loop component's output is estimated through a mathematical relationship among different (but related) loop components. Normally these loop components are of different types. Two redundant devices, such as two pressure sensors, may degrade with a common-mode failure. Using different types of sensors for predicting a variable can significantly decrease the possibility of common failures. For example, one can estimate the pressurizer water level in a PWR using hot leg temperature, cold leg temperature, reactor power and primary pressure..

The first step in obtaining mathematical relationships for analytical redundancy is to identify related loop components. This first step may be accomplished through some mathematical tools such as correlation analysis. This work used the latter technique.

After identifying related loop components, the characterization of these relationships through mathematical models are required. The models generate the necessary redundant sensor outputs for the FDI algorithm. As mentioned in earlier

chapters, the models are obtained by applying the Group Method of Data Handling (GMDH), which uses data generated from the loop for different normal set point conditions. These data are referenced to in this work as “fault-free database.” All possible (or available) different normal system conditions must be considered when developing these relationships. This is an important aspect in avoiding future false alarms by the FDI algorithm.

The set of analytical models for loop component redundancy is then used to predict each loop component’s state variables and control functions. During a normal operating condition each prediction error should be close to zero, that is, there should be a good agreement between the predicted value and the actual measurement from the system component. If a given component residual value is above some predefined threshold limit, then a fault has been detected.

If a fault is detected, the isolation of the fault component is performed by an expert system. The expert system proposed in this work is based on rules, which are developed from experiments, simulations, or from the expert knowledge of system engineers.

The last step in the FDI analysis is the reconstruction of the faulty component signal output. In this step the FDI algorithm attempts to compensate the signal used by the faulty component. If it succeeds, the process can be kept in operation without losing performance until next programmed maintenance schedule. In this research the

compensation is performed by a fuzzy logic system, which is built during the development of the FDI algorithm.

In summary, the required steps for developing an FDI algorithm are as follow:

1. Generation of a fault-free database. Many different system operational conditions must be considered here.
2. Determination of a qualitative relationship among different loop components through correlation coefficient technique.
3. Determination of quantitative relationships among different loop components through the GMDH technique.
4. Development of a rule-based decision module for fault detection and isolation. This is accomplished by simulating and characterizing individual faults in each important loop component.
5. Development of a fuzzy logic system for signal compensation. This is also accomplished by analyzing the data obtained in item 4.

Figure 4.1 in Chapter 4 shows a schematic of the overall procedure for developing a fault detection and isolation algorithm. The next sections will cover in more detail each of these steps for developing an FDI algorithm.

6.2.2. Implementation of an FDI algorithm

The implementation of an FDI algorithm basically consists of three steps: step one monitors the residuals of all available system variables. Residual is the difference

between a loop variable (such as a flow sensor) measurement and its predicted (analytical) value. Under a normal operating condition, each individual residual should be close to zero, that is, there should be a good agreement between the system predicted condition and the system actual condition. Normally, threshold limits are used to delimit the region in which the residuals are considered satisfactory. If any residual goes beyond the threshold limits, it indicates a probability of a system anomaly (fault detection).

If a fault is detected, the second step of the FDI is activated. The FDI algorithm isolates the faulty component throughout a rule-based decision module. In the case of a minor fault, the third step of the FDI is executed. In this step the FDI algorithm attempts to compensate the signal used by the faulty component. If it succeeds, the process can be kept in operation without losing performance until the next scheduled maintenance. As previously mentioned, the compensation is performed by a fuzzy logic engine, which is developed from measured data when known faults are imposed.

Figure 4.2 in Chapter 4 shows the procedure for implementing the FDI algorithm in a given system.

6.3. DEVELOPMENT OF AN FDI ALGORITHM USING A THEORETICAL MODEL

For the theoretical study, a simulation model of a process control loop was developed that included sensors, controllers and actuators. This model was implemented in Matlab-Simulink [90] program environment. Chapter 4 gives a more detailed

description of this model. This theoretical model simulates an experimental water loop system fabricated at The University of Tennessee. A schematic of this water loop is shown in Figure 4.1. Table 4.1 shows all available system variables.

6.3.1. Generation of the Fault-Free Database

A fault-free database was generated using the theoretical model for different device configurations. Bypass and Outlet MOV positions were systematically changed one at a time. Water level set point was also changed. The Inlet MOV being a part of the water level control system, its position can not be set manually; its position is set directly by the water level controller. No faulty devices were allowed in this phase. About 1,235 cases were simulated and the data generated were stored in a database. This database is used to obtain both qualitative and quantitative relationships among the loop devices.

6.3.2. Generating Qualitative Relationships Among Loop Components

The database generated from the fault-free simulations consists of a matrix of order 1,235 by 15, where 1235 represents the number of simulations (or observations) and 15 represents the number of available variables. The list of available variables is given in Table 4.1. To obtain a qualitative relationship among the loop components, the correlation coefficient method was applied. From this analysis, sets of related variables were defined, although the characterization of these relationships through mathematical expressions is not obtained in this step. Table 6.1 indicates the correlation coefficient of

Table 6.1. Representation of correlation coefficients of the loop variables.
Bypass and Outlet MOV positions were kept constant in this specific case.

	Inlet MOV position	Tank Water Level Set	PID Output	RMT Measured Flow Rate	Inlet Turbine Flow Rate	Tank Water Level	Outlet Turbine Flow Rate
Inlet MOV position							
Outlet MOV position							
Tank Water Level Set Point							
WL Set Point Error							
PID Output							
RMT Measured Flow Rate							
Inlet Turbine Flow Rate							
Tank Water Level							
Outlet Turbine Flow Rate							

Darker cells in the above table indicate high correlation coefficients (close to unity).

the most important signals. Darker cells in the table represent high correlation coefficients (close to unity).

Even though sometimes a large set of variables with high correlation existed, groups of small number of variables were selected to be modeled by the GMDH. The variables selected in each group were the ones from the components that are physically close to each other. For example the bypass and inlet MOV positions determine the flow that goes through the inlet piping. One could also use the flow rate that goes through the outlet piping in the above correlation set, however this variable would not bring new information to this relationship. Creating a small number of models with local variables makes the fault detection very efficient, that is, it is easy to isolate a faulty component. This efficiency is reflected in the rule-based expert system. The simpler the expert system, the easier it is to develop and maintain it.

Four relationships were defined for this particular loop system. With this relationships the FDI algorithm is be able to isolate basically all possible faults that may happen in that loop. The relationships are

- Inlet flow rate as a function of bypass and inlet MOV positions.
- Outlet flow rate as a function of tank water level and outlet MOV position.
- Tank water level as a function of inlet flow rate and outlet MOV position.
- PID output value as a function of bypass MOV position, tank water level and outlet flow rate.

For characterization of these four relationships through mathematical equations, the GMDH method was applied to the same fault-free database. A coefficient matrix representing each model is generated by the GMDH (see Chapter 3 for details). This matrix is used to obtain the predicted (or analytical) component value. This predicted value is then later used for fault detection. A detailed description of how to use the software developed for the Group Method of Data Handling is presented in the following sections.

6.3.3. Generating Quantitative Relationships among Loop Components

The quantitative relationships are obtained by running the GMDH algorithm developed in this research. This section describes how the data should be input to the GMDH algorithm and how to obtain the final models and the prediction values from it.

The first step is to collect data from the system for many different normal operating conditions. The data must be stored in a matrix. Each column represents a system variable and each row represents an observation. As an example, if there were 300 observations of a system with 15 available variables, the matrix size would be 300 by 15.

For better efficiency in the development of GMDH models, these data should be normalized. The user defines the normalization. Normalization such as -1 to 1 , 0 to 1 , Uniform, and Gaussian, can be applied. The GMDH does not normalize the data

internally. This data pre-processing must be done prior to the use of the GMDH algorithm.

Create a Matlab script file similar to the one shown in Appendix A. This script sets initial values necessary for internal calculations and also it starts a for-loop where the GMDH computation is executed. The for-loop is responsible for choosing different basic functions to be used by the GMDH algorithm (see Chapter 4 for a discussion of basic functions). For each chosen set of basic functions, the GMDH algorithm is processed. The main task of the GMDH algorithm is to find the model that best maps the input/output set for given basic functions. After finding the best model, a residual value is computed between the predicted value and the target value. This residual is stored in memory (RESIDUAL variable). This variable is used later for comparison purposes among all residuals from other runs. Each run with a different set of basic functions is tagged with a number stored in a variable called JobCounter. This variable is used later to recover the best of the GMDH models. The loop runs 2^K times, where K is the number of basic functions to be tested. As an example, $K = 11$ for the set of basic functions proposed in Equation (3.4) in Chapter 3. In this case, the number of possible combinations is $2^{11} = 2048$.

At the end of this process, the variable called RESIDUAL contains the residuals of all possible combinations of basic functions (2^K). Using the Matlab function called SORT ([y, index]=sort(residual)) the residuals are sorted from smallest to the highest values in the variable Y. The INDEX variable contains the JobCounter case numbers

associated with the residuals in Y . Normally the first JobCounter (index(1)) should be used because it represents the best set of basic functions for the GMDH model structure.

As discussed in Chapter 4, a GMDH model is basically defined by matrices of coefficients, which represent the relationship between input/output data. These matrices are not stored in the memory during the above process of finding the best set of basic functions, due to memory limitations. However, after finding the best model, the user can rerun the same Matlab script replacing the for-loop by the JobCounter single value (index(1)). By doing so, the script generates the matrices of the best GMDH model in the computers. The matrices are called *Layer*, *LayerOrder*, and *Coef*. Using the command

SAVE FileName Layer LayerOrder Coef

these matrices are saved into a data file (FileName.mat).

After obtaining the mathematical relationship among a set of loop variables, the model can be used for generating an analytical redundant measurement. This redundant measurement is the prediction of the GMDH model. This prediction is made by calling the function named ANNt (source code in Appendix A) developed also in this research. This function takes four arguments as input: a data array which contains the independent variables of this particular model, the matrices, *Layer*, *LayerOrder*, and *Coef* (previously saved).

The Fault Detection and Isolation algorithm uses the predicted values to compare with the measured values. If the error (residual) between the predicted and measured

values is higher than a pre-defined threshold, the algorithm assumes that a fault exists. In this case, a second module is executed; namely, the fault isolation module. This module is based on a rule-based expert system. This expert system is described in the next section.

Some observations must be made at this point about the constraints related to the use of the models developed by the GMDH algorithm.

- a) The number of nodes in a new layer of a GMDH model is given by $m(m-1)/2$, where m is the number of independent variables for each layer, where for the first layer m is equal to the number of independent variables.. If, for example, the number of independent variables is 4, then for the first layer $m=4$. For the second layer $m=6$, and for the third $m=15$, and so on. To avoid explosion of the number of nodes in a higher GMDH layer, the program always uses the best m nodes for the succeeding layers. That is, all the layers will always have m nodes.
- b) If the number of independent variables is equal to 2, the GMDH would stop at the second layer ($m(m-1)/2 = 1$). However, when only two independent variables are available, the Matlab script file called TrainTest.m (used in the training phase) creates a third variable which is the ratio between the first and the second inputs (x_1/x_2). This allows the GMDH to go beyond the second layer.
- c) If the number of independent variables is one, one should come up with some variants of this variable, such as $\sin(x)$, in order to increase the number of input variables for

the GMDH training phase. Another option is the use of polynomial mapping instead of the GMDH mapping for these situations.

- d) To avoid long training times, the maximum number of layers for a single model is limited to 30, i.e., if during training phase a model exceeds 30 layers the process is stopped. It has been observed that after this value, the model does not increase its performance significantly.

6.3.4. Interpreting the Results of the GMDH Modeling Process

As mentioned earlier in this chapter, the GMDH algorithm uses the fault-free database to obtain the best model that maps the input variables (independent variable) to the output variable (dependent variable). The algorithm splits the input data into two sets of data. The first set is used to find the best model, while the second monitors for model over fitting. Figure 6.1 plots the training output data along with the value predicted by the model for the best set of basic functions. Figure 6.2 plots the test data set with the predicted values. Finally, Figure 6.3 plots the error (in percentage) between the predicted and the expected output results of the best GMDH model for predicting the flow rate in the tank inlet piping as a function of Bypass and Inlet MOV positions. The plots indicate that the GMDH maps the input-output variables satisfactorily for both the training data and the test data.

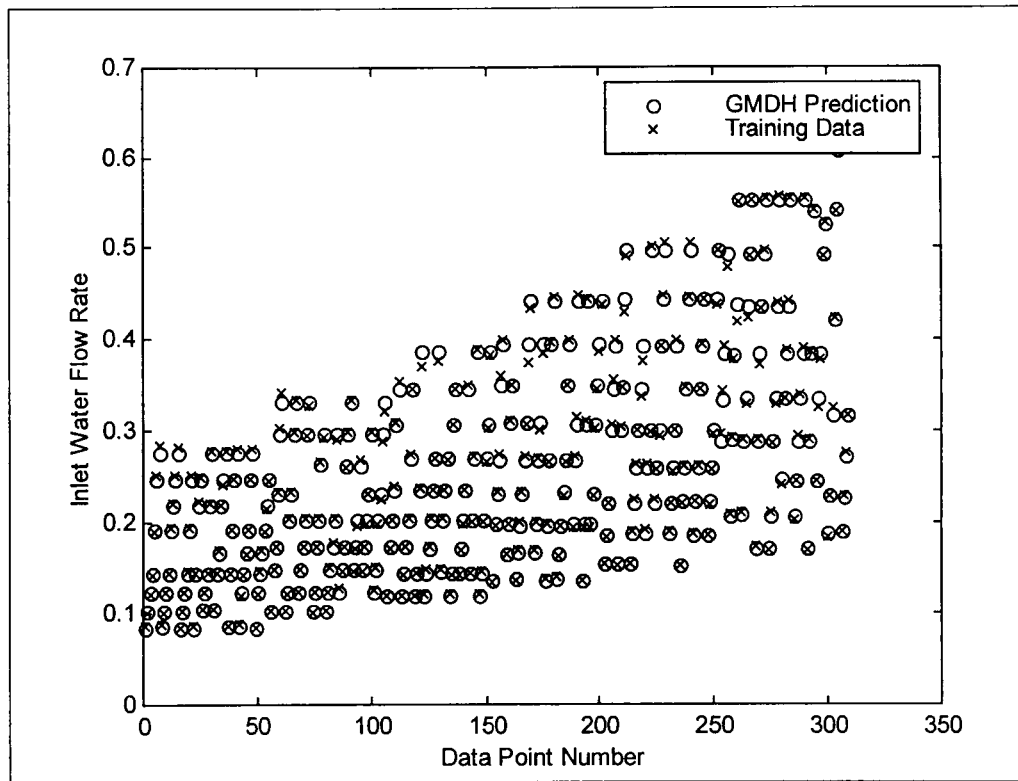


Figure 6.1. The GMDH predicted values (“o”) against the training output values (“x”) for the Inlet Flow rate as a function of Bypass MOV and Inlet MOV positions.

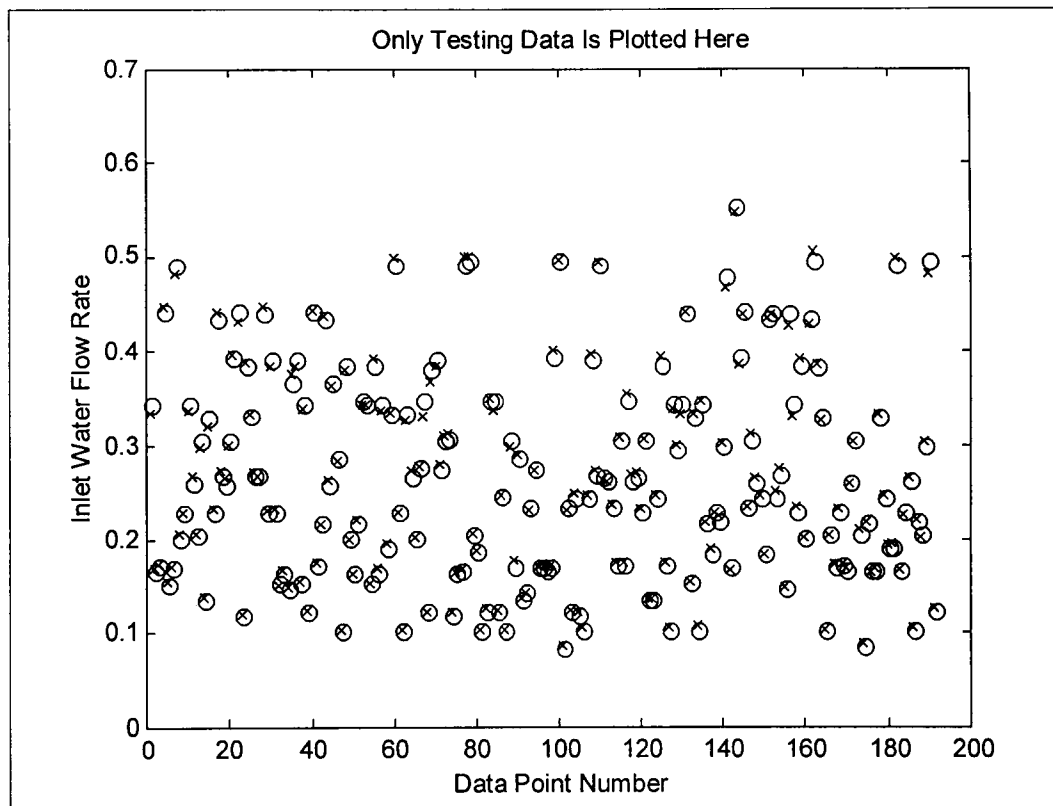


Figure 6.2. The GMDH predicted values ("o") against the test output values ("x") for the Inlet Flow rate as a function of Bypass MOV and Inlet MOV positions.

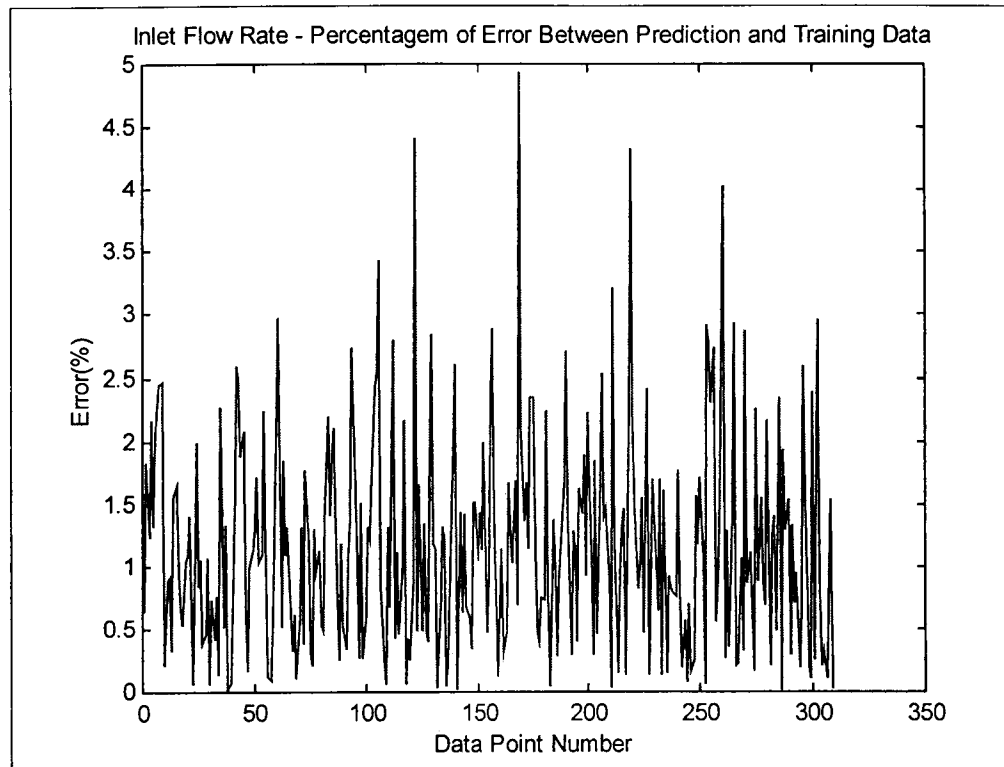


Figure 6.3. The error (%) between the predicted values and the measurements for the Inlet Flow rate as a function of Bypass MOV and Inlet MOV positions.

6.4. FAULT TYPES STUDIED IN THIS RESEARCH

Before discussing the development and implementation of the rule-based decision module for fault isolation, it is necessary to define the types of faults this expert system would be asked to recognize. The choice of the set of faults to be monitored will define not only the number of rules in the expert system but also how each rule is designed to recognize the faults.

Many types of faults can occur in the water loop system such as sensor faults, actuator faults, controller faults, pump-stop type of fault, leaks in pipings, etc. This study limits itself to those faults that can lead to significant error in the GMDH prediction models. The components that contribute to a specific GMDH model are its inputs and its output variables. As discussed in Section 6.3.2, the following GMDH models are proposed in this work:

- Inlet Flow Rate = $f(\text{Bypass MOV position, Inlet MOV Position})$.
- Outlet Flow Rate = $f(\text{Tank Water Level, Outlet MOV Position})$.
- Tank Water Level = $f(\text{Inlet Flow Rate, Outlet MOV Position})$.
- Controller Output = $f(\text{Bypass MOV position, Tank Water Level, Outlet Flow Rate})$.

If the predicted Inlet Flow Rate has a significant discrepancy compared with the measured inlet flow, three possible faults may be responsible for this discrepancy: a) the inlet flow meter itself, b) the bypass MOV valve, and c) the Inlet MOV valve. If any of these devices is faulty, the predicted model will not match closely with the measured inlet

flow rate. Using the above argument to the other GMDH models, the following faults were selected for on-line monitoring in this dissertation.

- Water level sensor is drifting (sensor fault).
- Outlet turbine flow meter is drifting (sensor fault).
- Outlet MOV positioning device is drifting (actuator).
- Bypass MOV position (actuator fault).
- Inlet MOV positioning device is drifting (actuator fault).
- RMT flow meter is drifting (sensor fault).
- Water Level Controller (controller fault).

It is worth to comment that the Bypass MOV is used in the low-pressure water loop system to simulate changes in the pump speed. Closing this MOV has the same effect in the flow that goes to the system as if the pump had decreased its speed (due to aging, for example). Therefore, for systems that do not have a similar bypass MOV, the results of the fault simulated here can be generalized as a fault in the pump itself. A similar argument can be used for the Outlet MOV fault. In systems that do not have an Outlet MOV, the faults could be related to piping leakage or piping obstruction. In summary, this study can be used for FDI not only to monitor faults in sensors, actuators and controllers, but also to monitor pumps and leakage/obstruction of pipings.

6.5. DEVELOPMENT OF THE RULE-BASED DECISION MODULE FOR FAULT ISOLATION

During normal system operation, the predicted values obtained by the GMDH methods are very close to those obtained by sensor measurements. However, when a fault occurs in one of the system components, one or more of these analytical redundant values will not match with their measurements.

During the experiments, the faults mentioned in the previous section were simulated. The basic type of fault imposed on the loop devices was a drift type. In all the studied fault cases, the imposed drift was about 10% of the device nominal value. For example, if the water level sensor is drifting, instead of its output being proportional to 20 inches of water, it would be proportional to 18 inches.

The rule-based expert system was based on the system behavior characteristics for each simulated fault. For most of the cases, the system dynamic behavior responds differently for each loop component fault. For these cases, each multidimensional residual vector (vector whose components are made of each individually generated residual) is unique. In cases where there are two or more different faults that produce the same system behavior, other system variables (such as water level set point) was supplied to the expert system. When this action is still not enough for differentiating two similar system-behavior faults, hardware redundancy is suggested. However, for the control loop used in this dissertation, redundant hardware was not necessary.

Figure 6.4 shows plots of different loop components for the case of a drift fault in the water level sensor. As shown in these plots, some variables decreased from their expected values, some increased, and the others remained at their nominal values. This example also illustrates that even though the water level sensor has an output roughly constant, it is the device that is drifting and causing the other components to drift.

Conversely, even though most of the loop components have changed their output values, some of the GMDH relationships still give very close predicted values. This happens because each GMDH model uses measurements of locally near sensors. This approach helps in identifying the faulty device.

Using the same example of drift fault in the water level sensor and keeping in mind the four GMDH models, one can infer the following.

- a) Even though the inlet MOV position has changed its position and the inlet mass flow has decreased, the estimated inlet flow rate gives good prediction because the relationship used to obtain the inlet flow does not use data from the faulty device for its calculations.
- b) For the same reason, the estimated controller output also gives a good prediction.
- c) Conversely, the predicted outlet flow rate has a significantly different value from its measurement. This happens because this parameter is a function of the water level (WL) and the outlet MOV position. Even though the WL sensor is drifting, the

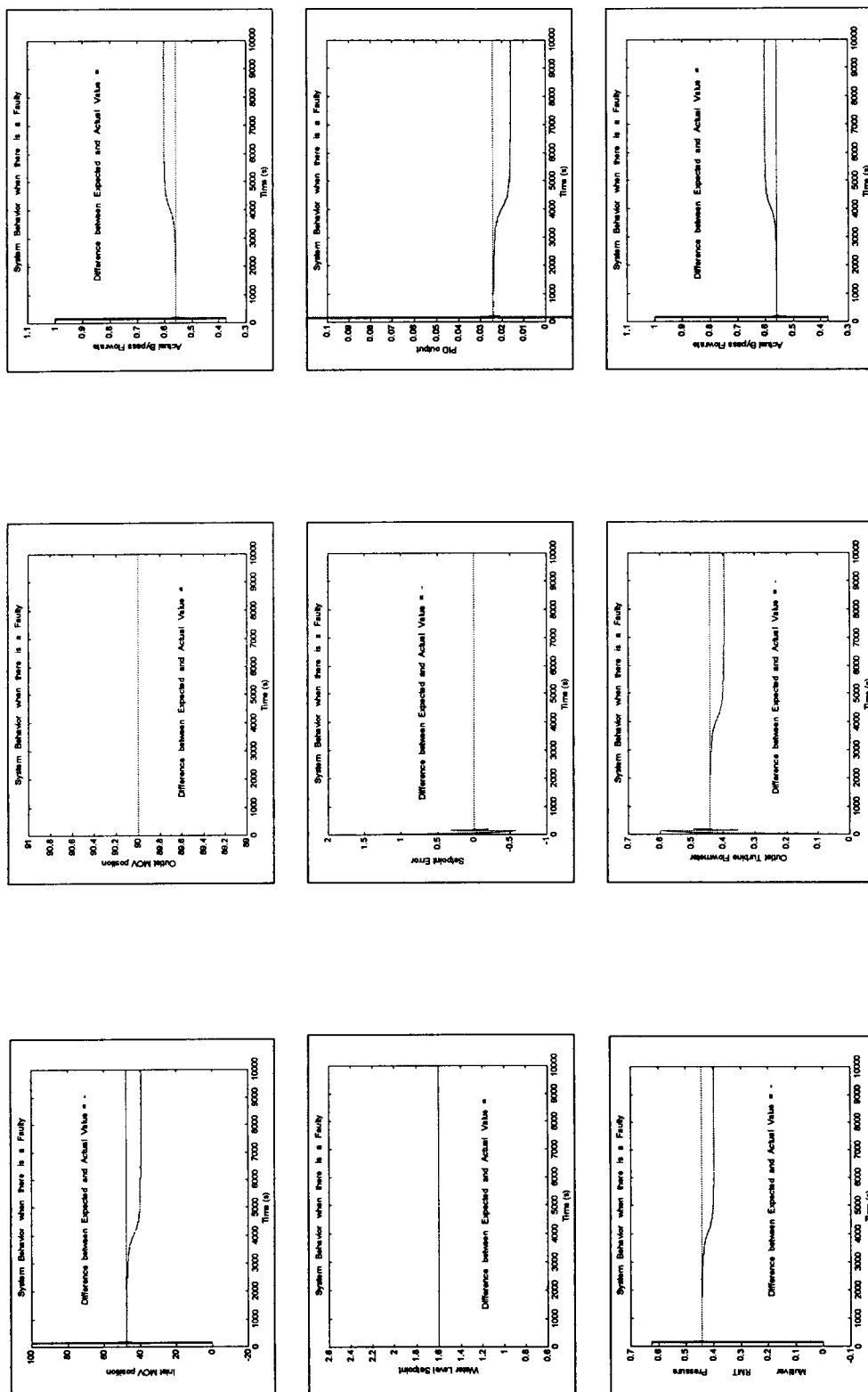


Figure 6.4. Plots of different loop components for the case of a drift fault in the water level sensor due to aging process.

controller closes the inlet MOV valve so that the WL sensor outputs a value close to its set point. However, the inlet flow rate and consequently the outlet flow rate are decreased). For this GMDH model, both input values (WL and Outlet MOV position) are constants prior to and after the fault. Therefore, the predicted outlet flow rate is also constant prior to and after the fault. However, the measured flow rate had decreased due the controller action. This causes a mismatch between the prediction and measured values of this variable.

- d) The same mismatch happens with the predicted water level. The water level is a function of inlet flow rate and outlet MOV position. The actual inlet flow rate has decreased, while the outlet MOV position has not. Even though one of the GMDH model input variables has changed its value, the measured WL is constant, thus predicting a change in the water level. However, the water level measurements are constant before and after the fault (due to controller actions), causing a significant error between the predicted water level and its measurement.

The seven faults in the sensors, actuators, and the controller described above were studied using the simulation model developed in Matlab-Simulink environment, as discussed in Chapter 4. For each simulation, the behavior of the loop variables was recorded and the GMDH prediction variables calculated. Table 6.2 shows the errors between the predicted and measured values of the GMDH models for each imposed fault. Cells with “+” sign indicate that the measured value of the specific loop variable is greater than the predicted value. Conversely, cells with “-” sign indicate that the

Table 6.2. Qualitative error between GMDH prediction values and the measured values for seven fault cases.

GMDH Prediction		Results						
		Inlet MOV Position	Water Level Sensor	Inlet Flowrate	Outlet MOV Position	Bypass MOV Position	Outlet Flowmeter	Controller Output
Inlet Flowrate		+		+		+		
Outlet Flowrate			-		-		-	
Controller Output		-				-	+	
Water Level			+	-	+			

measured value is less than the predicted value. Cells with no sign indicate that the predicted value is within 2% tolerance close to the measured value.

The following observations are made from the results presented in Table 6.2.

- a) The errors in the GMDH predictions are the same for the faults in the Inlet MOV set point position and the Bypass MOV set point position. In this case, a redundant measurement is required such as Bypass MOV measured position.
- b) The same situation happens with faults in the water level sensor and Outlet MOV. Again, a redundant measurement is required such as the Outlet MOV measured position.
- c) For the controller fault, even though the controller is not able to match the water level to the requested set point, all the GMDH predicted values match with the measurement values. In this case, a simple comparison between the water level set point and its measured (or predicted) value makes the isolation of this fault possible. Therefore, the water level set point must also be given to the expert system if controller faults are to be monitored.

With the above discussion and the results shown in Table 6.2, an expert system for fault isolation was developed. The rules consist of if-then statements based on the characteristic behavior of each studied fault. As discussed earlier, the expert system algorithm needs not only the value of the four GMDH models, but also the WL set point, and the Bypass and Outlet MOV measured positions. A description of each rule is given below.

- 1) IF (Inlet Flow rate Error > IF_Threshold)
 AND (Controller Output Error > CO_Threshold)
 AND (Bypass MOV Position Error < BMOV_P_Threshold)

 THEN Loop Faulty Component = Inlet MOV
- 2) IF (Outlet Flow rate Error > OF_Threshold)
 AND (Water Level Error > WL_Threshold)
 AND (Outlet MOV Position Error < BMOV_P_Threshold)

 THEN Loop Faulty Component = Water Level Sensor
- 3) IF (Inlet Flow rate Error > IF_Threshold)
 AND (Water Level Error > WL_Threshold)

 THEN Loop Faulty Component = Inlet Flow rate
- 4) IF (Outlet Flow rate Error > OF_Threshold)
 AND (Water Level Error > WL_Threshold)
 AND (Outlet MOV Position Error > OL_MOVp_Threshold)

 THEN Loop Faulty Component = Outlet MOV
- 5) IF (Inlet Flow rate Error > IF_Threshold)
 AND (Controller Output Error > CO_Threshold)
 AND (Bypass MOV Position Error > B_MOVp_Threshold)

 THEN Loop Faulty Component = Bypass MOV
- 6) IF (Outlet Flow rate Error > OF_Threshold)
 AND (Controller Output Error > CO_Threshold)

 THEN Loop Faulty Component = Outlet Flowmeter
- 7) IF (Water Level Set point Error > WLSp_Error_Threshold)

 THEN Loop Faulty Component = Water Level Controller

where IF_Threshold, CO_Threshold, BMOVP_Threshold, OF_Threshold, WL_Threshold, OL_MOVP_Threshold, B_MOVP_Threshold, and WLSp_Error_Threshold are thresholds for each variable in question. Also, the variable “Water Level Set point Error” is the error between the Water Level set point and the measured water level.

The FDI rule-based algorithm is also included in Appendix A.

6.6. MINIMUM REQUIREMENTS FOR AN FDI SYSTEM

In the previous section, a rule-based expert system was developed based on possible types of faults that a water loop system could experience. The number of rules and input variables for this expert system depend strongly on the faults to be monitored. If, for example, we assume that the bypass MOV is a very reliable device, then rule number 7 could be deleted from the list of rules. Also, neither the input variable named “Bypass MOV Position Error” nor the redundant signal that measures the Bypass MOV position would be necessary.

The above discussion illustrates the aspect of *minimum requirements* for an FDI system: the number of monitored signals in a system for fault detection and isolation strongly depends on the number of faults to be monitored. Therefore, when developing a FDI algorithm for a specific industrial control loop, the engineer should observe the number of available sensors in that system and with this information define the types of faults that could be monitored or vice-versa.

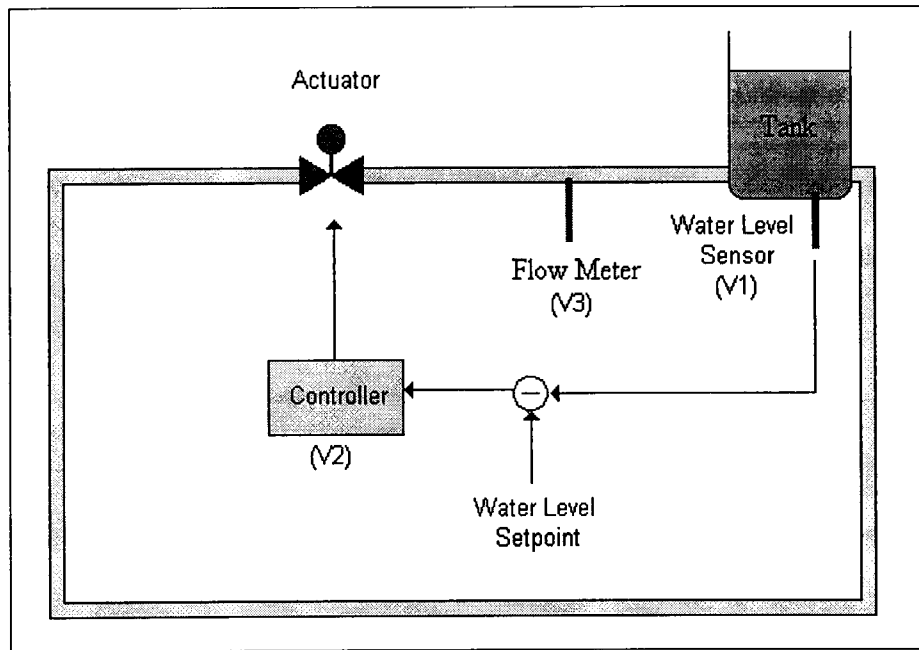
As a further example, Figure 6.5 shows a schematic of a control loop. The faults being monitored include (a) actuator fault, (b) controller fault, and (c) sensor fault. The minimum measurement set required to detect and isolate any one of these faults at any time includes: water level (set point, V1), PI controller output (V2), and mass flow rate (process variable, V3).

Table 6.3 summarizes the fault types and the minimum measurement requirements for this control loop. Note that only three measurements are used to detect and isolate these faults. Both normal operational data and data corresponding to different imposed faults are available for this system. The definition of minimum requirements is important in applications to operating process systems. The number of measurements available is often limited and the decision has to be made while the system is in operation.

6.7. IMPLEMENTATION OF THE FDI USING THE THEORETICAL MODEL

Step two consists of implementing the FDI in a real system. In this phase, the FDI algorithm performs the following tasks:

- a) Retrieves all available sensor data from the loop.
- b) Computes analytical measurements (predicted values).
- c) Crosschecks them with the actual measurements.
- d) Detects and isolates a possible fault.



Types of faults being monitored:

- Motor-operated valve Fault (actuator fault – Fault #1)
- Water Level Sensor Fault (process variable fault – Fault #2)
- MOV PID Controller Fault (controller fault - Fault #3)

Variable List:

- V1: Measured water level (set point to be controlled)
- V2: Water level PID controller output (process controller)
- V3: Mass flow rate measurement (process variable)

Figure 6.5. Control loop layout.

Table 6.3. Minimum Requirement Table for different monitored faults.

Fault Type	Minimum Variable Required	Comments
Actuator Fault	V2 or V3	If no other fault is being monitored
	V3	If fault #2 is also being monitored
	(V2 or V3) and V1	If fault #3 is also being monitored
	V1 and V3	If all faults are being monitored
Process Sensor Fault	V2	If no other fault is being monitored
	V2 and V3	If fault #1 is also being monitored
	V2 and (V1 or V3)	If fault #3 is also being monitored
	V2 and V3	If all faults are being monitored
Controller Fault	V1 or V2 or V3	If no other fault is being monitored
	V1	If fault #1 is also being monitored
	V1	If fault #2 is also being monitored
	V1	If all faults are being monitored

The implementation of the FDI for the case of simulation of the theoretical model consists of using the GMDH models obtained in the first step of this work to compute the expected value for some loop variables. If any of these errors are larger than their respective thresholds, these errors are sent to the rule-based expert system. All necessary modules used by FDI algorithm were developed in the Matlab script language.

6.8. RESULTS OF SIMULATIONS OF SOME FAULT DETECTION AND ISOLATION CASES

This section presents results of some simulations performed in Matlab-Simulink using the theoretical model described in Chapter 4. As discussed in the previous chapters, the first module of the FDI algorithm should be able to distinguish between two cases: changes in the system due to changes in one of its set points (Bypass MOV position, Outlet MOV position, or water level) and changes in the system due to the presence of a faulty device. In the first case, the GMDH predictions should be very close to the actual measurements, while in the second case some discrepancies should be observed. The FDI algorithm diagnostics for the first case would be of the form “no fault detected,” while in the second case it would not only detect the fault but would also isolate it.

Next, a brief description of each fault imposed on the system is given and the results obtained from the implemented FDI algorithm are shown.

6.8.1. Results of the FDI Algorithm for Changes in System Set Points

The first cases studied were the ones where changes in the loop variables were caused by changes in the system set points. The low-pressure water loop system has three types of set points: the Bypass MOV position, the Outlet MOV position, and the water level. The FDI algorithm should not give false alarms when one or more of them have been changed to a new value. The results for the various normal cases are shown and indicate a high performance of the FDI system.

A – Change in the water level set point (from 20 inches to 18 inches)

Predicted Inlet Flow rate: 0.009427 (← from GMDH Model)

Model Inlet Flow rate: 0.009228

Percentage of Variation: -2.149558 (%)

Predicted Outlet Flow rate: 0.074271 (← from GMDH Model)

Model Outlet Flow rate: 0.073825

Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.490487 (← from GMDH Model)

Model PID Output: 0.497338

Percentage of Variation: 1.377414 (%)

Predicted Water Level: 0.042790 (← from GMDH Model)

Model Water Level: 0.042857

Percentage of Variation: 0.156170 (%)

Faulty Component: No fault has been detected (← FDI output)

B – Changes in the Bypass MOV position (from 80% to 90% open)

Predicted Inlet Flow rate: 0.009432

Model Inlet Flow rate: 0.009228

Percentage of Variation: -2.205635 (%)

Predicted Outlet Flow rate: 0.074271

Model Outlet Flow rate: 0.073825

Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.478500

Model PID Output: 0.485636

Percentage of Variation: 1.469425 (%)

Predicted Water Level: 0.042790

Model Water Level: 0.042857

Percentage of Variation: 0.156170 (%)

Faulty Component: No fault has been detected

C – Changes in the Outlet MOV position (from 80% to 70% open)

Predicted Inlet Flow rate: 0.008754

Model Inlet Flow rate: 0.008588

Percentage of Variation: -1.930489 (%)

Predicted Outlet Flow rate: 0.069062

Model Outlet Flow rate: 0.068706

Percentage of Variation: -0.518361 (%)

Predicted PID Output: 0.437957
Model PID Output: 0.444804
Percentage of Variation: 1.539406 (%)

Predicted Water Level: 0.042815
Model Water Level: 0.042857
Percentage of Variation: 0.099292 (%)

Faulty Component: No fault has been detected

D – Simultaneous changes in all the above three set points

Predicted Inlet Flow rate: 0.009355
Model Inlet Flow rate: 0.009161
Percentage of Variation: -2.120764 (%)

Predicted Outlet Flow rate: 0.073629
Model Outlet Flow rate: 0.073286
Percentage of Variation: -0.467559 (%)

Predicted PID Output: 0.466817
Model PID Output: 0.474257
Percentage of Variation: 1.568847 (%)

Predicted Water Level: 0.045649
Model Water Level: 0.045714
Percentage of Variation: 0.142301 (%)

Faulty Component: No fault has been detected

6.8.2. Results of the FDI Algorithm for Faulty Cases

As mentioned in the earlier section, the following faults were studied in this work.

- Water level sensor is drifting (sensor fault).
- Outlet turbine flow meter is drifting (sensor fault).
- Outlet MOV positioning device is drifting (actuator).
- Bypass MOV position (actuator fault).
- Inlet MOV positioning device is drifting (actuator fault).
- RMT flow meter is drifting (sensor fault).
- Water Level Controller (controller fault).

The basic type of fault imposed on the loop devices was a drift. In all the fault cases studied, the imposed drift was about 10% of the device nominal value.

A - Inlet MOV Fault (open by 10%)

Predicted Inlet Flow rate: 0.008791

Model Inlet Flow rate: 0.009228

Percentage of Variation: 4.742529 (%)

Predicted Outlet Flow rate: 0.074271

Model Outlet Flow rate: 0.073825

Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.490487

Model PID Output: 0.447438

Percentage of Variation: -9.621309 (%)

Predicted Water Level: 0.042790
Model Water Level: 0.042857
Percentage of Variation: 0.156170 (%)

Faulty Component: Inlet MOV

B – Water Level Sensor Fault (drifts up)

Predicted Inlet Flow rate: 0.008456
Model Inlet Flow rate: 0.008306
Percentage of Variation: -1.805978 (%)

Predicted Outlet Flow rate: 0.074271
Model Outlet Flow rate: 0.066445
Percentage of Variation: -11.777340 (%)

Predicted PID Output: 0.416372
Model PID Output: 0.422887
Percentage of Variation: 1.540606 (%)

Predicted Water Level: 0.038575
Model Water Level: 0.042857
Percentage of Variation: 9.990798 (%)

Faulty Component: Water Level Sensor

C – Rosemount Inlet Flow meter Fault

Predicted Inlet Flow rate: 0.009427
Model Inlet Flow rate: 0.010144

Percentage of Variation: 7.068562 (%)

Predicted Outlet Flow rate: 0.074271

Model Outlet Flow rate: 0.073825

Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.490487

Model PID Output: 0.497338

Percentage of Variation: 1.377414 (%)

Predicted Water Level: 0.047030

Model Water Level: 0.042857

Percentage of Variation: -9.737045 (%)

Faulty Component: Inlet Flow meter

D – Outlet MOV Fault

Predicted Inlet Flow rate: 0.007561

Model Inlet Flow rate: 0.007438

Percentage of Variation: -1.654478 (%)

Predicted Outlet Flow rate: 0.074271

Model Outlet Flow rate: 0.059505

Percentage of Variation: -24.814871 (%)

Predicted PID Output: 0.355361

Model PID Output: 0.360006

Percentage of Variation: 1.290273 (%)

Predicted Water Level: 0.034594
Model Water Level: 0.042857
Percentage of Variation: 19.280208 (%)

Water Level Set point: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Set point Position: 0.850000
Measured Position: 0.763120
Percentage of Variation: 10.221227 (%)

Bypass MOV Set point Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Faulty Component: Outlet MOV

E – Bypass MOV position

Predicted Inlet Flow rate: 0.009968
Model Inlet Flow rate: 0.009228
Percentage of Variation: -8.013686 (%)

Predicted Outlet Flow rate: 0.074271
Model Outlet Flow rate: 0.073825
Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.490487
Model PID Output: 0.548853

Percentage of Variation: 10.634087 (%)

Predicted Water Level: 0.042790

Model Water Level: 0.042857

Percentage of Variation: 0.156170 (%)

Water Level Set point: 0.042857

Measured Water Level: 0.042857

Percentage of Variation: 0.000000 (%)

Outlet MOV Set point Position: 0.850000

Measured Position: 0.850000

Percentage of Variation: 0.000000 (%)

Bypass MOV Set point Position: 0.850000

Measured Position: 0.938044

Percentage of Variation: -10.358170 (%)

Faulty Component: Bypass MOV

F – Outlet Flowmeter

Predicted Inlet Flow rate: 0.009427

Model Inlet Flow rate: 0.009228

Percentage of Variation: -2.149558 (%)

Predicted Outlet Flow rate: 0.074271

Model Outlet Flow rate: 0.066244

Percentage of Variation: -12.117199 (%)

Predicted PID Output: 0.414494
Model PID Output: 0.497338
Percentage of Variation: 16.657494 (%)

Predicted Water Level: 0.042790
Model Water Level: 0.042857
Percentage of Variation: 0.156170 (%)

Water Level Set point: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Set point Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Bypass MOV Set point Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Faulty Component: Outlet flow meter

G – Water Level Controller

Predicted Inlet Flow rate: 0.005949
Model Inlet Flow rate: 0.005957
Percentage of Variation: 0.135395 (%)

Predicted Outlet Flow rate: 0.048103
Model Outlet Flow rate: 0.047654

Percentage of Variation: -0.941647 (%)

Predicted PID Output: 0.264213

Model PID Output: 0.265879

Percentage of Variation: 0.626770 (%)

Predicted Water Level: 0.027751

Model Water Level: 0.027664

Percentage of Variation: -0.314502 (%)

Water Level Set point: 0.042857

Measured Water Level: 0.027664

Percentage of Variation: 35.450573 (%)

Outlet MOV Set point Position: 0.850000

Measured Position: 0.850000

Percentage of Variation: 0.000000 (%)

Bypass MOV Set point Position: 0.850000

Measured Position: 0.850000

Percentage of Variation: 0.000000 (%)

Faulty Component: Water level controller

From the results presented above, it is clear that the performance of the FDI algorithm in detecting set point changes and faulty devices is excellent. No false alarms were observed for these and other previously performed tests.

6.9. FAULT COMPENSATION

The last step in the FDI implementation is the reconstruction of the faulty component signal output. In this step the FDI algorithm attempts to compensate the signal used by the faulty component. If it succeeds, the process can be kept in operation without losing performance until the next scheduled maintenance. In this research the compensation was performed by a fuzzy-logic system, which was developed as part of the FDI algorithm.

CHAPTER 7

DEVELOPMENT OF A FUZZY-LOGIC ENGINE FOR FAULTY DEVICE RECONSTRUCTION

7.1. INTRODUCTION

When detecting a faulty component in a system several questions arise: Is the system safe for continuing its operation? Has the system performance significantly decreased? Is the quality and economy of the final product compromised? The best way to overcome all these questions would be to promptly replace the faulty component. However, in some situations it is not practical and implementing an algorithm for reconstructing the faulty device in a way to minimize the effect of the fault in the system could be an optimal optimization problem.

Reconfiguring a faulty device can be accomplished by introducing an extra term into the original device signal. As an example, if a valve has a positioning offset of +5%, the signal sent by the controller should add a -5% offset signal into its output so that the final valve position would be very close to where it should be without the fault. If this reconfiguration succeeds, the process could be kept in operation without losing performance or endangering human lives until the next system programmed maintenance schedule. Of course, not all system faults can be cancelled by this process. Catastrophic faults, such as a broken pump or a stuck valve, cannot be reconfigured and a shutdown

must be performed to fix the problem. This chapter focuses its attention to the types of faults, such as drift faults, which can be cancelled by reconfiguring the faulty device in most situations.

7.2. FUZZY LOGIC – A BRIEF INTRODUCTION

The mathematical tool proposed in this work for device reconstruction is the fuzzy logic [116] approach. Fuzzy logic (FL) is a convenient way to map an input space to an output space. Some reasons for using the fuzzy logic approach for device reconstruction are the following:

- Fuzzy logic is conceptually easy to understand. The mathematical concepts behind fuzzy reasoning are very simple.
- Fuzzy logic is flexible.
- Fuzzy logic is tolerant of imprecise data.
- Fuzzy logic can model nonlinear functions of arbitrary complexity.
- Fuzzy logic can be built using the experience of experts.

The concept of FL was conceived by Lotfi Zadeh, and presented not as a control methodology, but as a way of processing data by allowing partial set membership rather than crisp set membership or non-membership. This approach to set theory was not applied to control systems until the 1970's due to limited capabilities of small computers. Professor Zadeh reasoned that human beings do not require precise, numerical

information input, and yet they are capable of making very good decisions. If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement. Unfortunately, U.S. manufacturers have not been so quick to embrace this technology while the Europeans and Japanese have been aggressively building real products around it.

The fuzzy logic is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software, or a combination of both. The fuzzy logic provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. This approach to control problems mimics how a human being would make decisions, only much faster.

The fuzzy logic incorporates a simple, rule-based IF X AND Y THEN Z approach to solving a control problem rather than attempting to model a system mathematically. The FL model is empirically-based, relying on an operator's experience rather than their technical understanding of the system. For example, rather than dealing with temperature control in terms such as "SP =500F", "T <1000F", or "210C <TEMP <220C", terms like "IF (process is too cool) AND (process is getting colder) THEN (add heat to the process)" or "IF (process is too hot) AND (process is heating rapidly) THEN (cool the process quickly)" are used. These terms are imprecise and yet very descriptive of what must actually happen. Consider what you do in the shower if the temperature is too cold:

you will make the water comfortable very quickly with little trouble. The fuzzy logic is capable of mimicking this type of behavior but at very high rate. More sophisticated rules can be generated by detailed simulation studies.

The FL rules use the concept of Membership Functions. Membership function is a graphical representation of the magnitude of participation of each input. Figure 7.1 shows three membership functions: two trapezoidal type and one triangular type. These functions are characterized by a set of numbers, which define their position and width. These parameters can be defined by an expert engineer, or can be adjusted based on the data.

The membership function associates a weighting with each of the inputs that are processed, defines functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. Once the functions are inferred, scaled, and combined, they are defuzzified into a crisp output which drives the system. Figure 7.2 shows the overall FL inference process.

The basic structure of the type of fuzzy inference system is a model that maps input characteristics to input membership functions, input membership function to rules, rules to a set of output characteristics, output characteristics to output membership functions, and the output membership function to a single-valued output or a decision associated with the output.

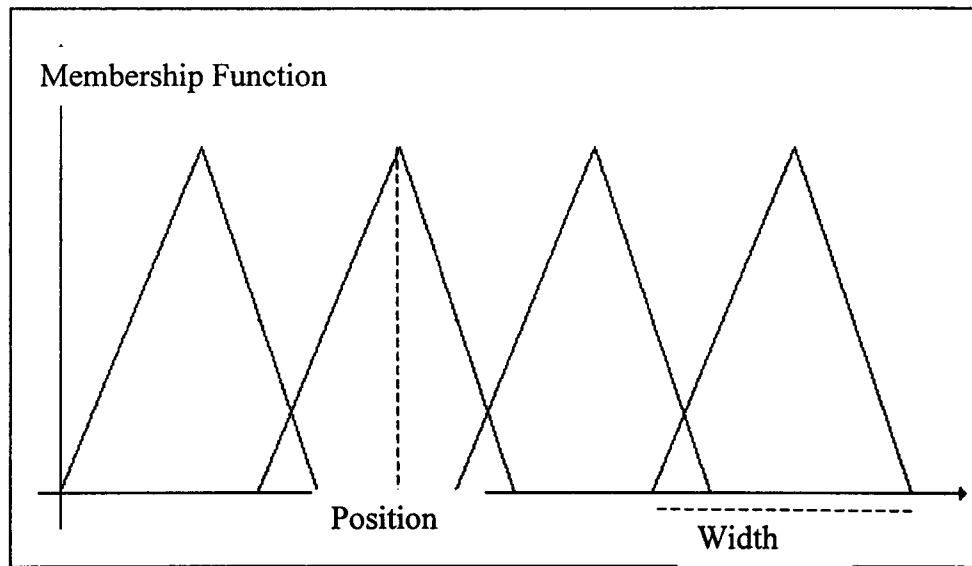


Figure 7.1. Triangular membership functions of a given input variable. Each membership function is characterized by a set of numbers which defines its position and width.

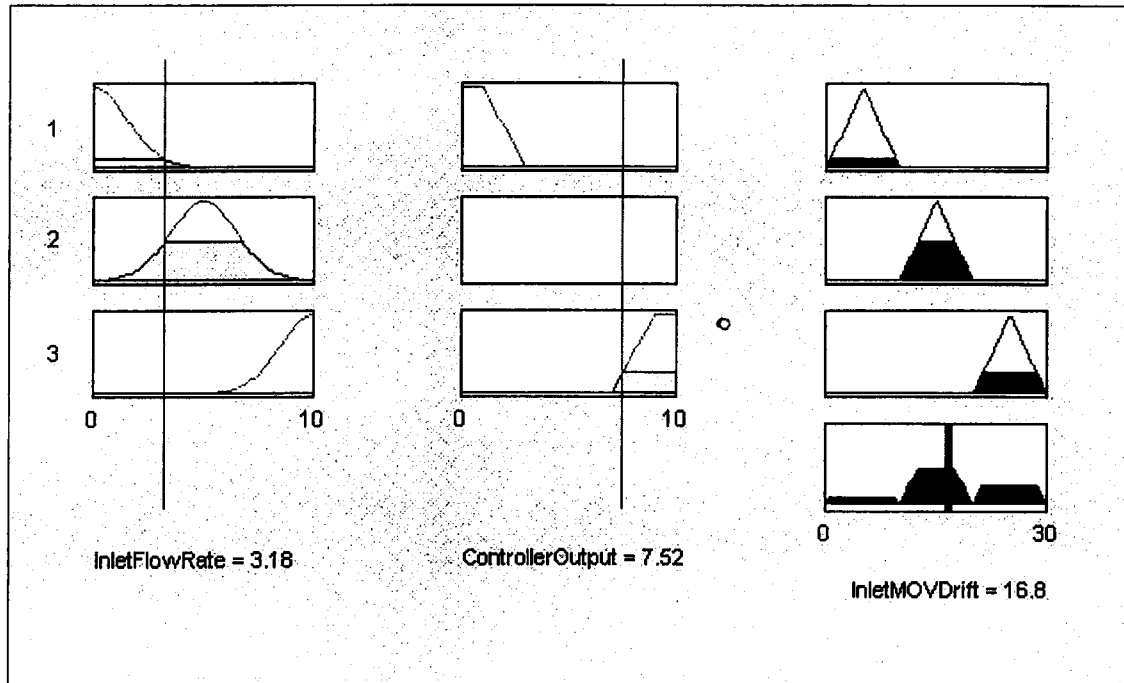


Figure 7.2. The overall Fuzzy logic inference process. In this example, there are two inputs (Inlet Flow Rate and Controller Output), three rules, and one output (Inlet MOV Drift Value).

We have only considered membership functions that have been fixed, and somewhat arbitrarily chosen. Also, we've only applied fuzzy inference to modeling systems whose rule structure is essentially predetermined by the user's interpretation of the characteristics of the variables in the model. The shape of the membership functions depends on parameters, such as center value and width. Changing these parameters will change the shape of the membership function. Membership function parameters can also be chosen automatically using a technique called Adaptive Neuro-Fuzzy Inference System (ANFIS).

The basic idea behind these neuro-adaptive learning techniques is very simple. These techniques provide a method for the fuzzy modeling procedure to learn information about a data set, in order to compute the membership function parameters that best allow the associated fuzzy inference system to track the given input/output data. This learning method works similar to that of neural networks.

In this research, the Matlab Fuzzy Logic toolbox was used. This toolbox provides an ANFIS subroutine, which allows the fuzzy system to learn from the data. Using a given input/output data set, the toolbox function ANFIS constructs a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using a back-propagation algorithm along with a least squares type of method. The Matlab implementation of ANFIS system is performed by a Graphical User Interface (GUI). This interface is shown in Figure 7.3.

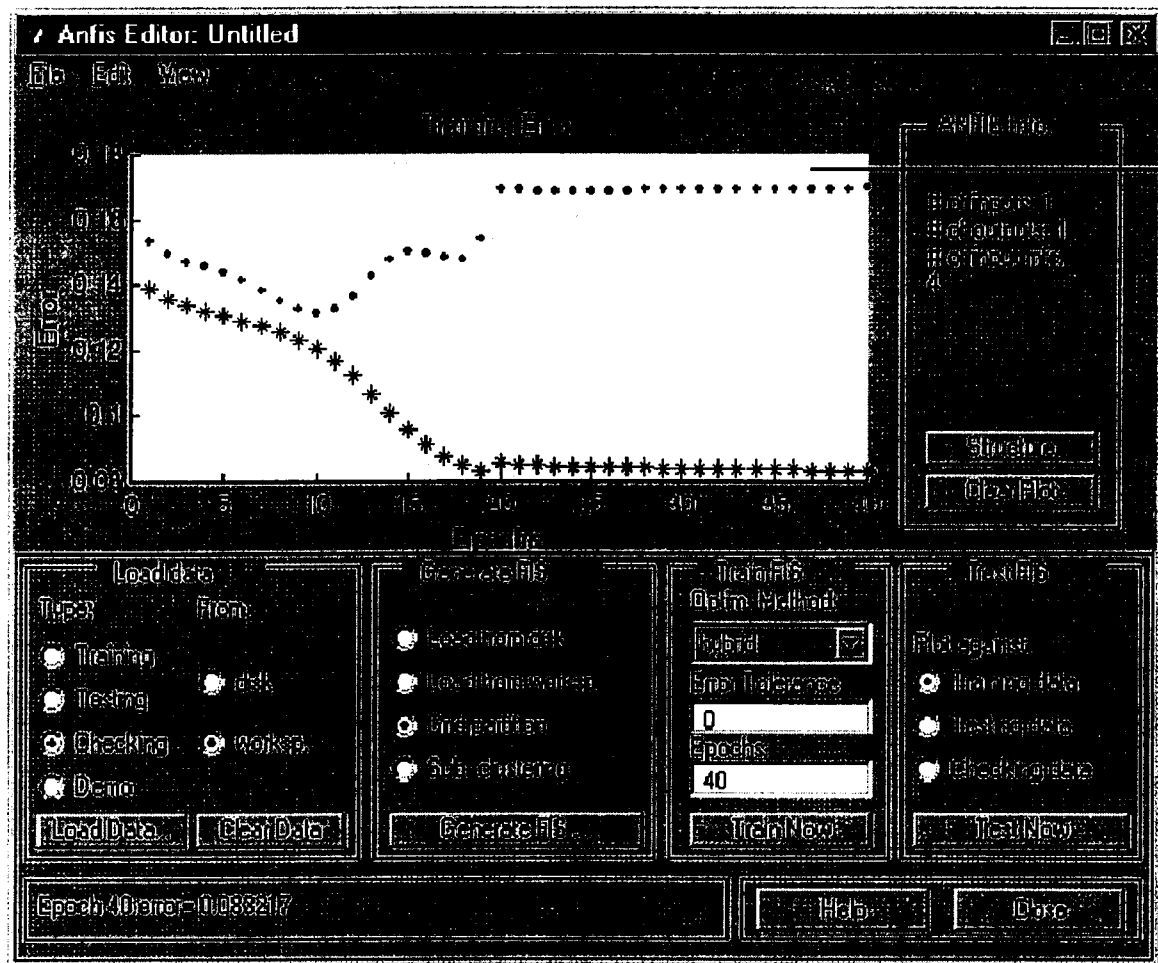


Figure 7.3. ANFIS graphical user interface. During training the ANFIS engine adjusts the membership functions of the fuzzy logic system using the training data set (*), while it also checks for model overfitting using the test data set (•) [116].

During the learning process, the parameters associated with the membership functions change. The computation of these parameters (or their adjustment) is facilitated by a gradient vector, which provides a measure of how well the fuzzy inference system is modeling the input/output data for a given set of parameters. Once the gradient vector is obtained, any of several optimization routines could be applied in order to adjust the parameters so as to reduce some error measure (usually defined by the sum of the squared difference between actual and desired outputs).

To start training an fuzzy inference system using the ANFIS Editor GUI, first the user needs to have a training data set that contains desired input/output data pairs of the target system to be modeled. The user should also supply a test data set that can be used to check the generalization capability of the resulting fuzzy inference system, and to check for model overfitting during training.

7.3. CONTROL LOOP DEVICE RECONSTRUCTION USING ANFIS

As mentioned earlier in this chapter, reconfiguring a faulty device can be accomplished by introducing an extra signal component into the original faulty device signal. Seven types of faults were studied in this work, as follows

- Water level sensor is drifting (sensor fault).
- Outlet turbine flow meter is drifting (sensor fault).
- Outlet MOV positioning device is drifting (actuator).
- Bypass MOV position (actuator fault).

- Inlet MOV positioning device is drifting (actuator fault).
- RMT flow meter is drifting (sensor fault).
- Water Level Controller (controller fault).

These faults have a direct influence in the four GMDH models used for fault detection and isolation algorithm. The GMDH models are

- Inlet Flow Rate = $f(\text{Bypass MOV position, Inlet MOV Position})$.
- Outlet Flow Rate = $f(\text{Tank Water Level, Outlet MOV Position})$.
- Tank Water Level = $f(\text{Inlet Flow Rate, Outlet MOV Position})$.
- Controller Output = $f(\text{Bypass MOV position, Tank Water Level, Outlet Flow Rate})$.

For the cases where the faulty devices are the four variables predicted by the GMDH models (inlet flow rate, outlet flow rate, tank water level, and controller output), the reconstruction signal sent to the faulty device can be computed directly from the difference between the prediction and the measured values. In these cases, there is no necessity of using a fuzzy logic system. The fuzzy logic system is used for the cases where the faulty devices are not the ones predicted by the GMDH models, i.e., Bypass MOV, Inlet MOV, and Outlet MOV.

A fuzzy logic engine was created for each MOV fault. For adjustments of the membership functions of each fuzzy engine, the Matlab ANFIS subroutine was used based on faulty data. This data came from the theoretical model (model described in Chapter 4).

Each fuzzy engine is composed of two inputs and one output. The inputs were defined based on Table 6.2. Each fault affects only two of the four GMDH predicted variables. These variables were used as input to the fuzzy logic system. For example, a fault in the Inlet MOV position will cause a mismatch in the inlet flow rate and the controller output. Therefore, these variables are the inputs of the fuzzy logic engine for this specific device (inlet MOV). The output is the drift value imposed on the faulty device. The drift value is known from simulation. 231 different drift levels were imposed on a single MOV. The data from these simulations were used to train the FL engine. 139 cases were used for training, while the other 92 were used to monitor model overfitting.

The ANFIS structure for the faulty device reconstruction used in this research has two inputs, each input represented by five Gaussian-type membership functions, and one output. Figure 7.4 shows the ANFIS structure, while Figure 7.5 shows the membership functions. Figure 7.6 shows the training data for the Inlet MOV fault case. Figure 7.7 shows the testing data set. Figure 7.8 shows the learning process where the plot represented by “*” is the overall training error and the plot represented by “•” is the overall testing error. The model overfitting starts between the iterations 7 and 8. The fuzzy engine chosen is the one preceding these points. The predicted values from the fuzzy engine are close to the test data points, meaning that the learning is satisfactory. Figure 7.9 shows the plot of the predicted drift and the simulated drift for the training data. It is also observed a good agreement between them.

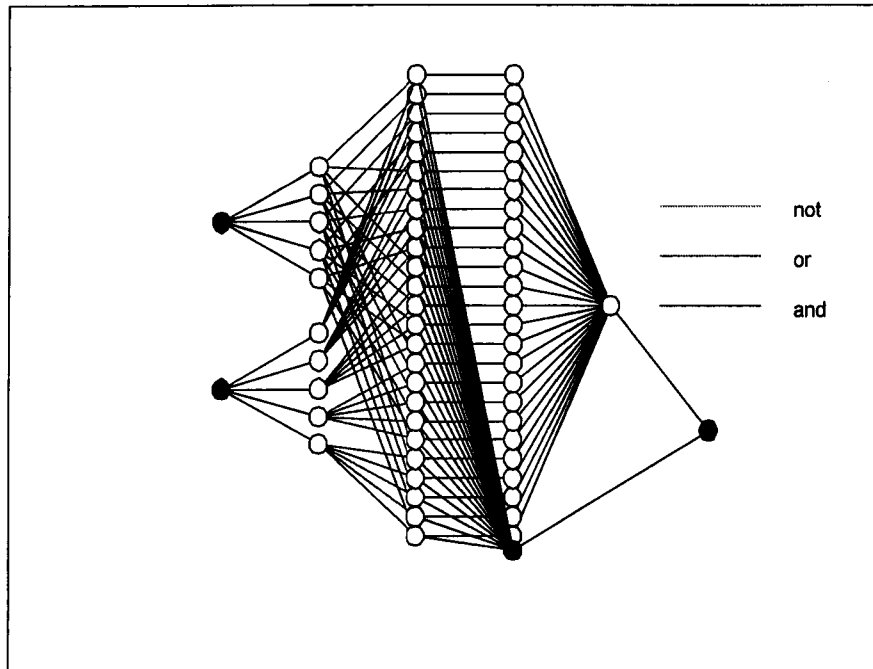


Figure 7.4. Fuzzy logic structure for fault device reconstruction.

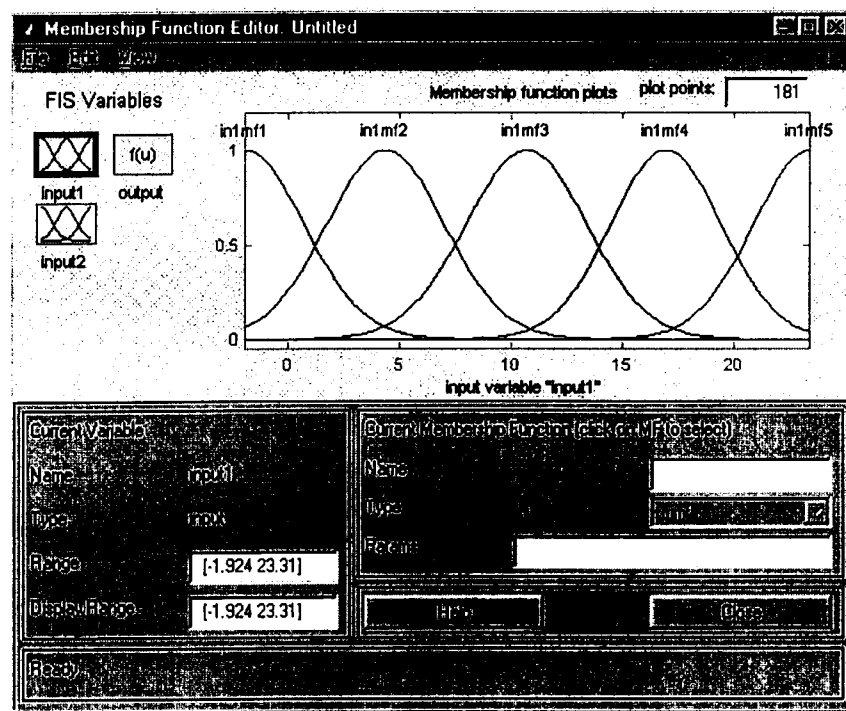


Figure 7.5. Gaussian-type membership functions of the fuzzy system.

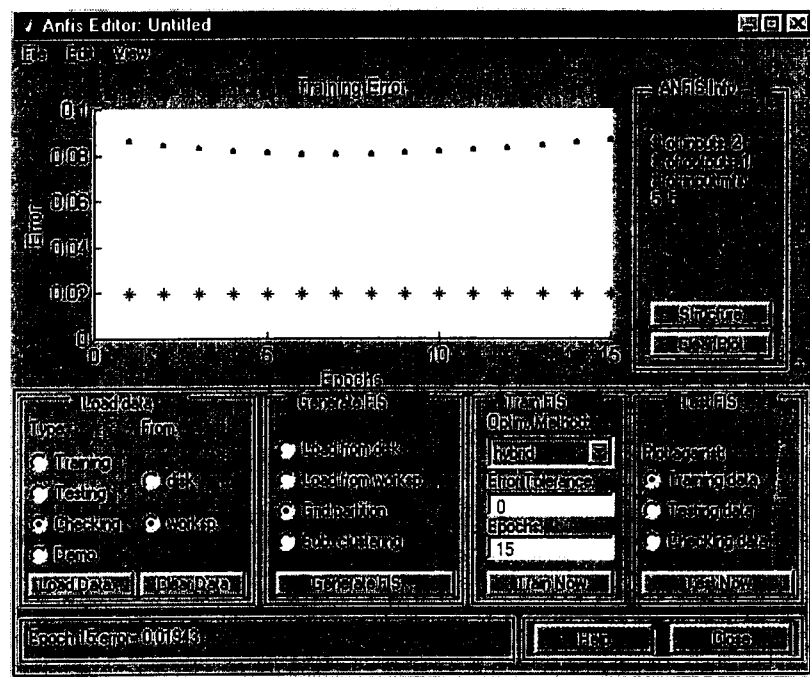


Figure 7.6. Training phase in the ANFIS subroutine. “*” represents the overall training error and “•” represents the overall testing error.

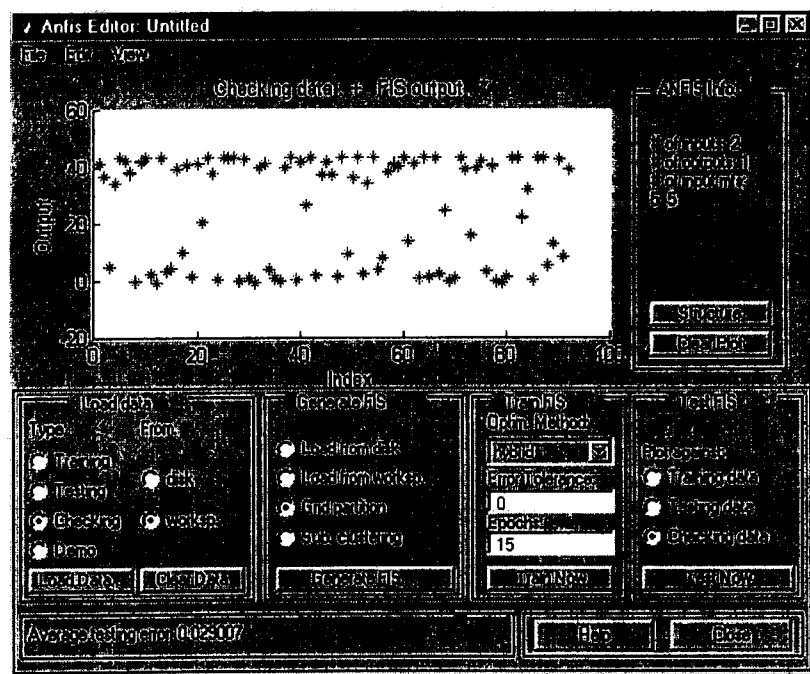


Figure 7.7. Test data versus fuzzy output.

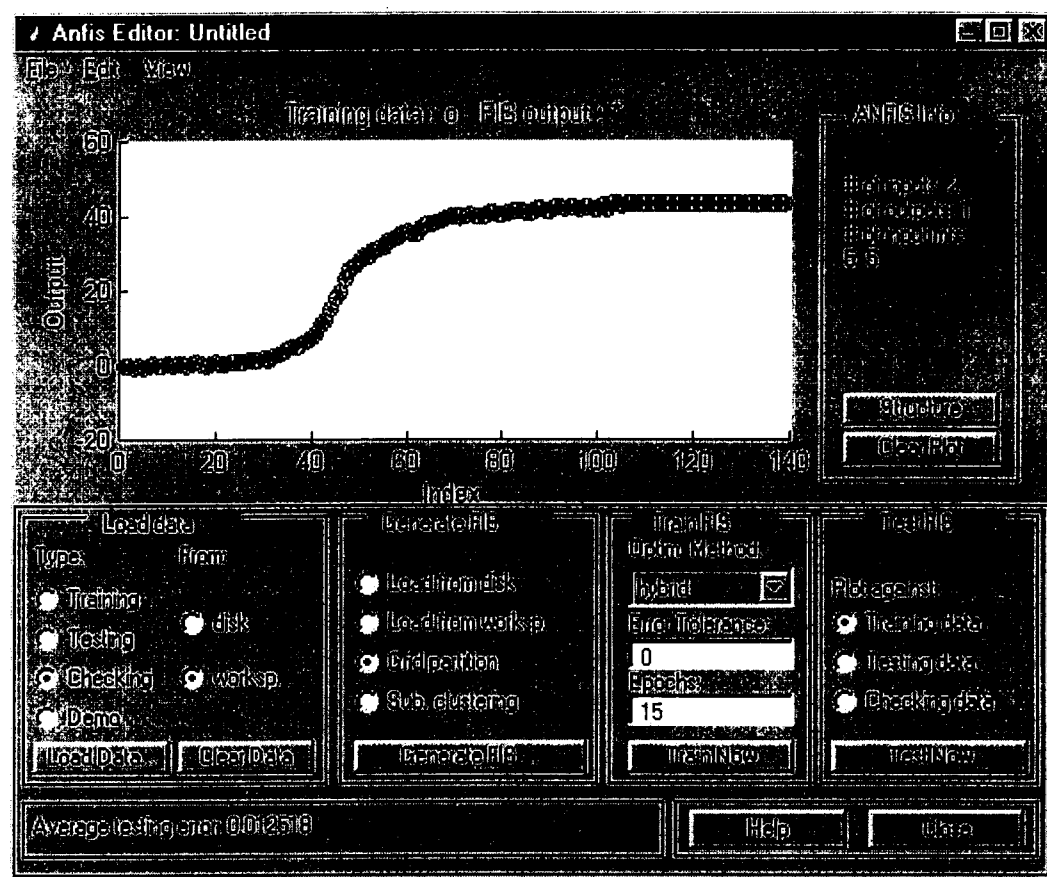


Figure 7.8. Training data versus fuzzy output.

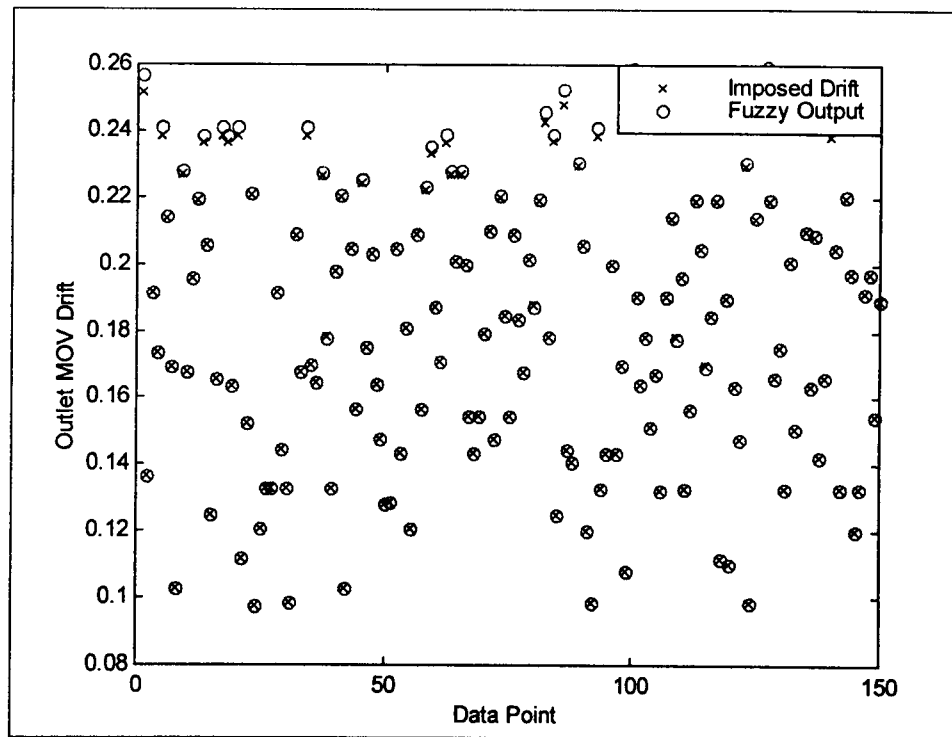


Figure 7.9. Performance tests of the Fuzzy logic Drift Recovering system. The recovered drift value along with the initial drift value imposed on the Outlet MOV are plotted.

7.4. IMPLEMENTATION OF THE FAULTY DEVICE RECONSTRUCTION IN THE FDI ALGORITHM

After adjusting the membership functions of each fuzzy engine (one fuzzy engine for each MOV), the parameters that define the engine are saved into a file with extension FIS. Later, to obtain a reconstruction signal value for a faulty MOV, the following commands in Matlab environment must be given to perform the fuzzy inference calculations.

```
fismat = readfis('filename.fis');  
reconf = evalfis([x1, x2],fismat)
```

where: filename.fis is an FL structure to be evaluated, x_1 and x_2 are the input values (for the case of fault in the inlet MOV, $x_1 = \text{inlet flow rate}$ and $x_2 = \text{controller output}$), and reconf is the magnitude of the drift observed in the loop device.

These lines of code were inserted into the FDI algorithm as an independent module, which is executed in the case a fault is present, more specifically if the fault is in one of the motor-operated valves. The output of the fuzzy engine is then added to the measured signal corresponding to the faulty device, compensating the fault and keeping the process running without loss of performance until the next scheduled maintenance shutdown. Figure 7.10 shows all modules involved in the Fault Detection and Isolation (and reconfiguration) algorithm developed in this work.

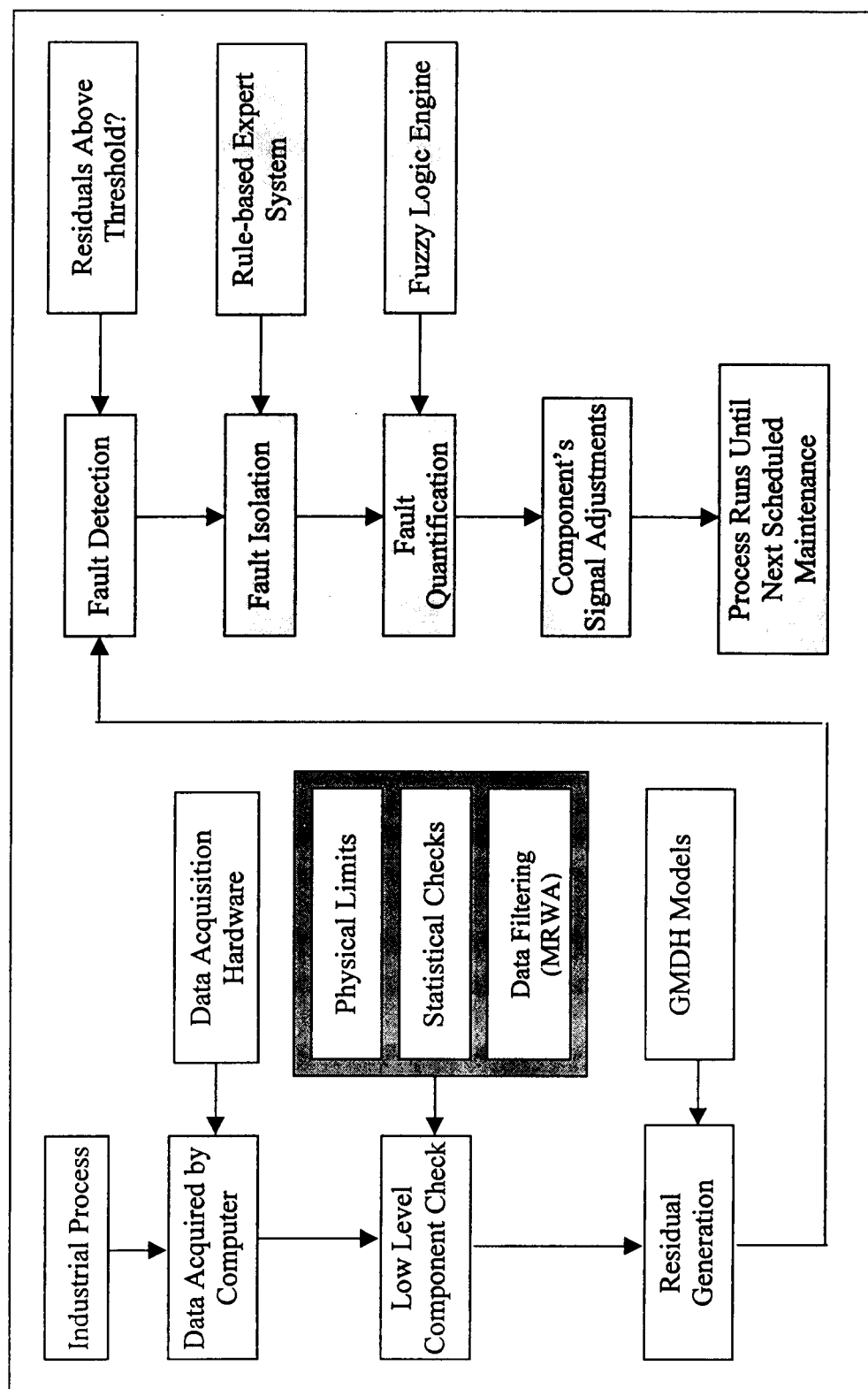


Figure 7.10. An overall procedure for implementing a Fault Detection and Isolation algorithm in an industrial process.

7.5. FUZZY LOGIC APPLIED TO FAULTY DEVICE RECONSTRUCTION - RESULTS

The fuzzy logic engines trained to perform device reconstruction were implemented into the Fault Detection and Isolation algorithm. As mentioned before, this reconfiguration takes place when the faulty device is one of the three motor-operated valves (MOV) in the control loop used in this research.

To perform fuzzy inference calculations, the following commands were introduced into the Matlab FDI algorithm.

```
fismat = readfis('filename.fis');  
reconf = evalfis([x1, x2],fismat);
```

where filename.fis was replaced by each individual MOV FL structure, error values of the GMDH predicted variables that are changed when a fault in the specific MOV is imposed. Reconf is the magnitude of the drift observed in the loop device (such as the inlet MOV).

Many simulations were used to check the performance of the Fuzzy Logic system in recovering the drift value imposed initially. This recovered value was then applied to the loop faulty device.

7.6. NEXT STEP

Chapters 6 and 7 described the development and implementation of the Fault Detection and Isolation algorithm and device reconstruction of the faulty device, with

application to the theoretical model developed in this research. The results of this FDI algorithm were excellent.

The next step is to apply all these algorithms to a real system. However, even though the focus will be now on a real system, all the concepts of acquiring data, defining qualitative and quantitative relationships, building GMDH models, defining rules for the expert system, and training an FL structure for device reconfiguration described in Chapters 6 and 7 can be directly applied. Chapter 8 describes the development and implementation of this technology in the low-pressure water loop system designed and built at The University of Tennessee.

CHAPTER 8

DEVELOPMENT AND IMPLEMENTATION OF THE FDI ALGORITHM FOR AN EXPERIMENTAL CONTROL LOOP

8.1. INTRODUCTION

A low-pressure water loop (LPWL) experimental system was designed, built, and commissioned for the purpose of checking the performance of the fault detection algorithm in an operating control loop. The loop components are described in Chapter 4, Section 4.2. A description of the experimental setup and its components is given in Section 4.3.1.

All the steps for the development and implementation of the FDI algorithm performed in Chapter 7 were repeated for this experimental setup. The steps are:

- Generation of fault free database.
- Definition of qualitative relationships among loop devices.
- Quantitative definition of those relationships by applying the GMDH method.
- Development of a rule-based expert system based on results of externally imposed faults in the system.
- Development of Fuzzy logic engines for faulty device reconfiguration.
- On-line implementation of the FDI algorithm.

A LabView program was developed for experiment control, data acquisition and system perturbations (imposed faults). A more detailed description of how this program interacts with the FDI algorithm is given later in this chapter.

8.2. GENERATION OF FAULT-FREE DATABASE

As described in Chapter 6, obtaining a fault-free database is important for the development of a FDI algorithm. This database should contain as many normal system conditions as possible. This database was used for generating prediction models and allowing the FDI algorithm to determine whether changes in the sensors' outputs are due to changes in the system set point or due to a loop component malfunction.

For generating the fault-free database, a LabView program was developed to automatically change the system normal operational conditions every fifteen minutes. The changes were made by changing the bypass and outlet motor-operated valve positions from 70% to 90% open (in steps of 5%). Water level set point was also changed from 10 to 30 inches (in steps of 2 inches). A total of 250 cases were performed. All 15 available variables were collected every three seconds for each perturbation. The experiment was run continuously for about 48 hours. To check the system reproducibility, the complete set of 250 cases was run three times.

8.3. GENERATION OF QUALITATIVE RELATIONSHIPS AMONG THE CONTROL LOOP COMPONENTS

The qualitative relationship among loop components were obtained by the same process used in Chapter 7, that is, by applying correlation coefficients method to the data in the fault-free database. Because the theoretical model was based on this experimental loop, the same qualitative relationships among components were obtained. However, we could not use the same theoretical quantitative relationships (models) for the variable prediction, due to the simplicity of the theoretical model. The GMDH method was applied to the experimental data to get the mathematical models for variable prediction. Half of the data ($3 \times 250 / 2 = 375$ cases) were used for training and the other half for performance evaluation.

Figures 8.1 shows both the measured and the predicted inlet flow rate for the training and testing data set. The predicted values are very close to the measured ones. The error between these values (in percentage) is shown in Figure 8.2. For this specific GMDH model, errors less than 1% were observed. However, the models for the other loop variables presented a little bit higher errors (up to 5%). But even with this level of errors, the models were considered satisfactory.

8.4. DEVELOPMENT OF THE RULE-BASED DECISION MODULE

Basically, the behavior of the experimental system is very similar to the behavior observed in the theoretical model in terms of its response to normal and faulty conditions.

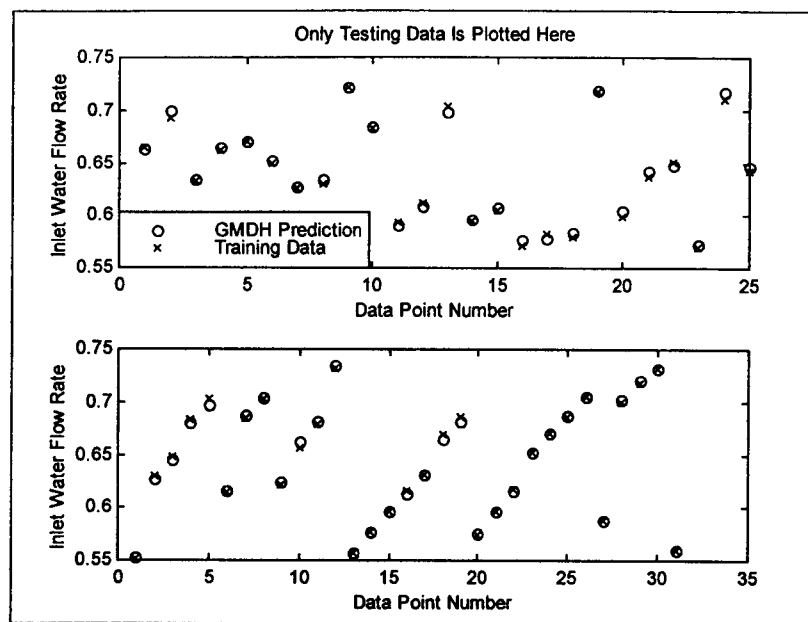


Figure 8.1. Plot of GMDH algorithm output. The upper plot shows the predicted value for the training data. The lower plot shows the predicted value for the test data.

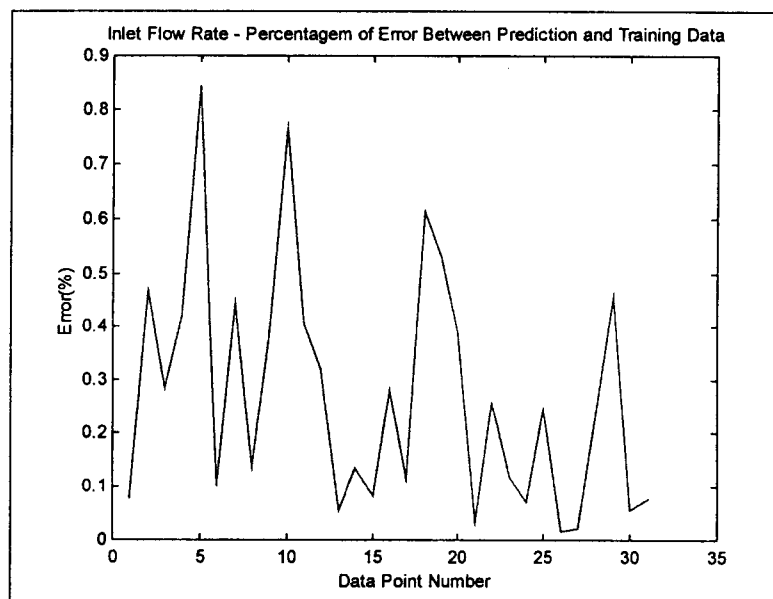


Figure 8.2. Absolute error between GMDH predicted values and the test data.

Therefore, it is expected that the rule-base generated in Chapter 6 can be applied as it is.

The rule-based expert system looks at changes in the residuals between different loop variables. When the residuals are greater than the user's pre-defined thresholds, the FDI algorithm considers that there is an anomaly in the system (Fault Detection). The rule-based expert system then uses these residuals to isolate the most probable faulty device.

8.5. IMPLEMENTATION OF THE FDI ALGORITHM

Initially, some work had been performed to implement the Fault Detection and Isolation algorithm directly inside the data acquisition and experiment control LabView program. The whole FDI algorithm would be implemented as a Dynamic Link Library (DLL). As mentioned before, the FDI algorithm was written in Matlab due to its high performance in manipulating matrix calculations. Using the proprietary C Compiler from Matlab, the FDI algorithm was converted to C++ and later using Microsoft Visual C++, an attempt was made to create a DLL from this C++ source code. However, it was discovered that the version of Matlab C Compiler available at that time could not handle 3-Dimensional matrices. These matrices were used extensively in the GMDH algorithm. Therefore this kind of approach was discarded.

Due to the above limitations, the fault detection and isolation algorithm was implemented as an independent program. This program runs in parallel with the LabView

program. It reads the available loop variables such as set points, controller output, and measured parameters from a data file stored in the hard disk by the LabView program. Subsequently, it normalizes the data from 0 to 1, computes the predicted values of some of the loop components, generates residuals and runs the expert system for detection and isolation of any possible fault in the system. For the cases where a faulty device is isolated, the device reconfiguration module is processed. As discussed in Chapter 7, this module uses the fuzzy logic engine for reconstruction MOVs, and it uses the predictions for the variables that are computed directly by the GMDH models (Inlet and Outlet flow meters, Water Level Sensor, and Controller).

The LabView program developed in this dissertation performs not only data acquisition and experiment control, but also artificially creates the faults in the different loop components. An *ATAN*-type function was used as drift. The drift was introduced into each loop component signal one at a time. A maximum drift of 10 % of the sensor's nominal value prior to the fault was added in a fifteen-minute time interval. For example, if the water level sensor output is sending a signal representing 20 inches of water before the fault is imposed, a drift of 2 inches was added at the end of the fifteen minutes of perturbation. Figure 8.3 shows the fault profile as a function of time for this case.

The selection of the loop component to introduce the drift is made by the user in the main screen of the LabView program. A list box at the lower left corner of the main screen makes the selection. Figure 8.4 shows the LabView main screen.

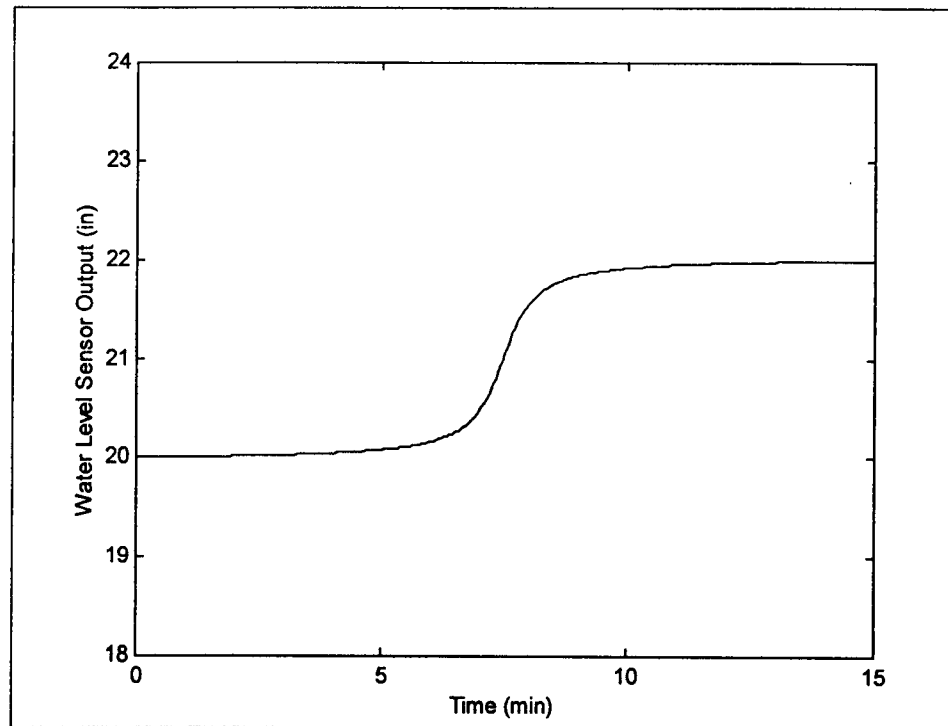


Figure 8.3. Drift profile as a function of time for the sensor water level.

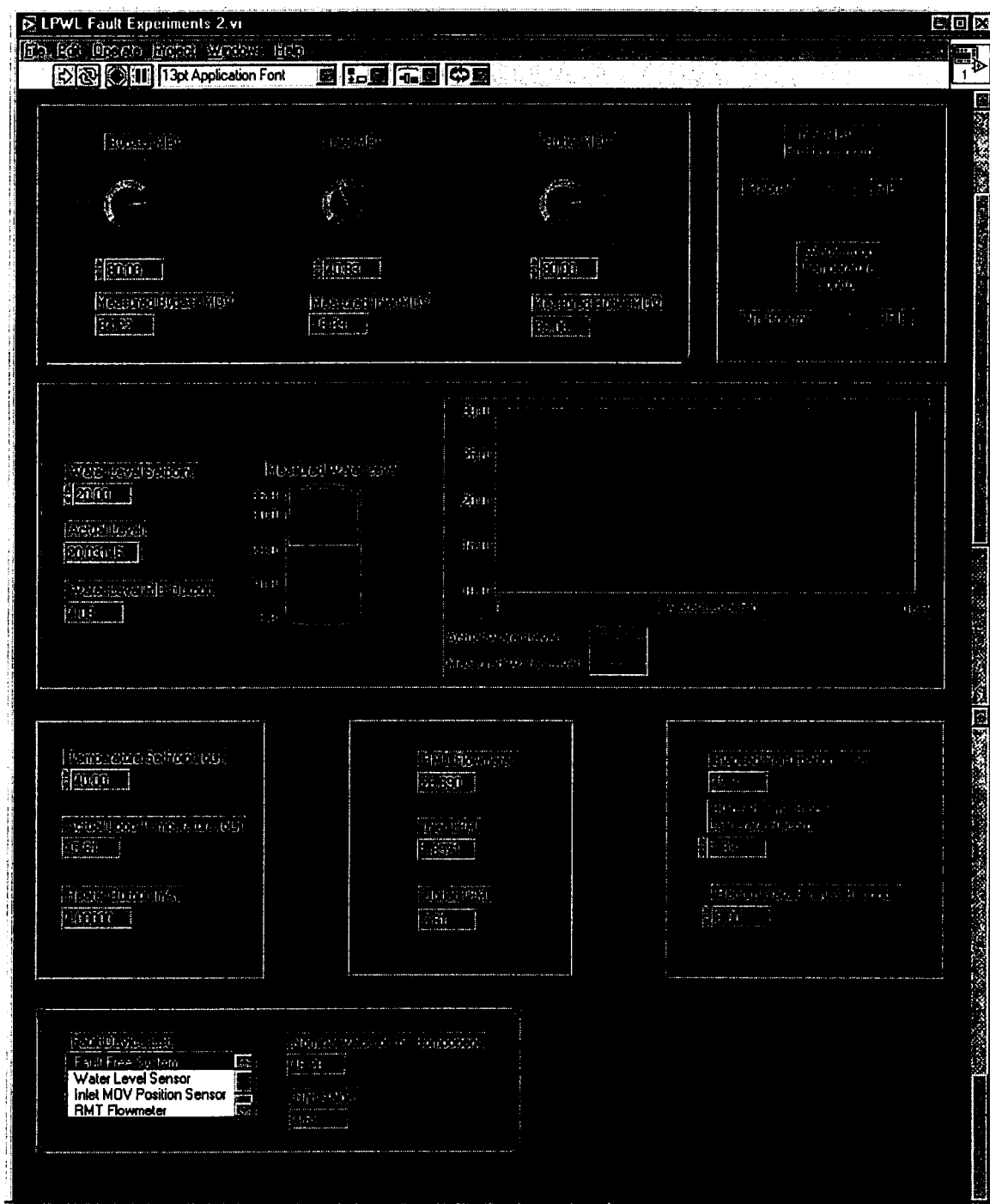


Figure 8.4. Main screen of the data acquisition and experiment control LabView program for the low-pressure water loop experimental system.

As discussed in earlier chapters, the FDI algorithm developed in this research must be able to distinguish changes in sensor's output due to normal set point changes (fault-free perturbations) and due to the existence of a faulty device. Figure 8.4 also shows the possibility for the user to change set point values for the motor-operated valves (at the top of the screen) and the water level set point (middle left part of the screen).

Inside the LabView program, a circular data buffer was developed. During the program's execution, this circular buffer contains five minutes of previous data. A static data file, saved in the hard disk performs the data transfer between this program and the Matlab FDI program. Every minute, on the minute, the LabView program writes the average value of the data buffer into the static data file. Also, every minute the FDI algorithm reads the data from this static data file and performs the fault detection and isolation (if a fault is detected). Figure 8.5 shows the process as a whole.

8.6. EXPERIMENTAL RESULTS

Several experiments were performed in the low-pressure water loop system for checking the performance of the fault detection and isolation algorithm developed in this research. Two categories of experiments were performed:

- Experiments where the MOV position set points and the water level set point were changed. This type of experiment was performed to check if the FDI algorithm would not give false fault alarms, that is, it would recognize that the observed changes in the system were not caused by a faulty component.

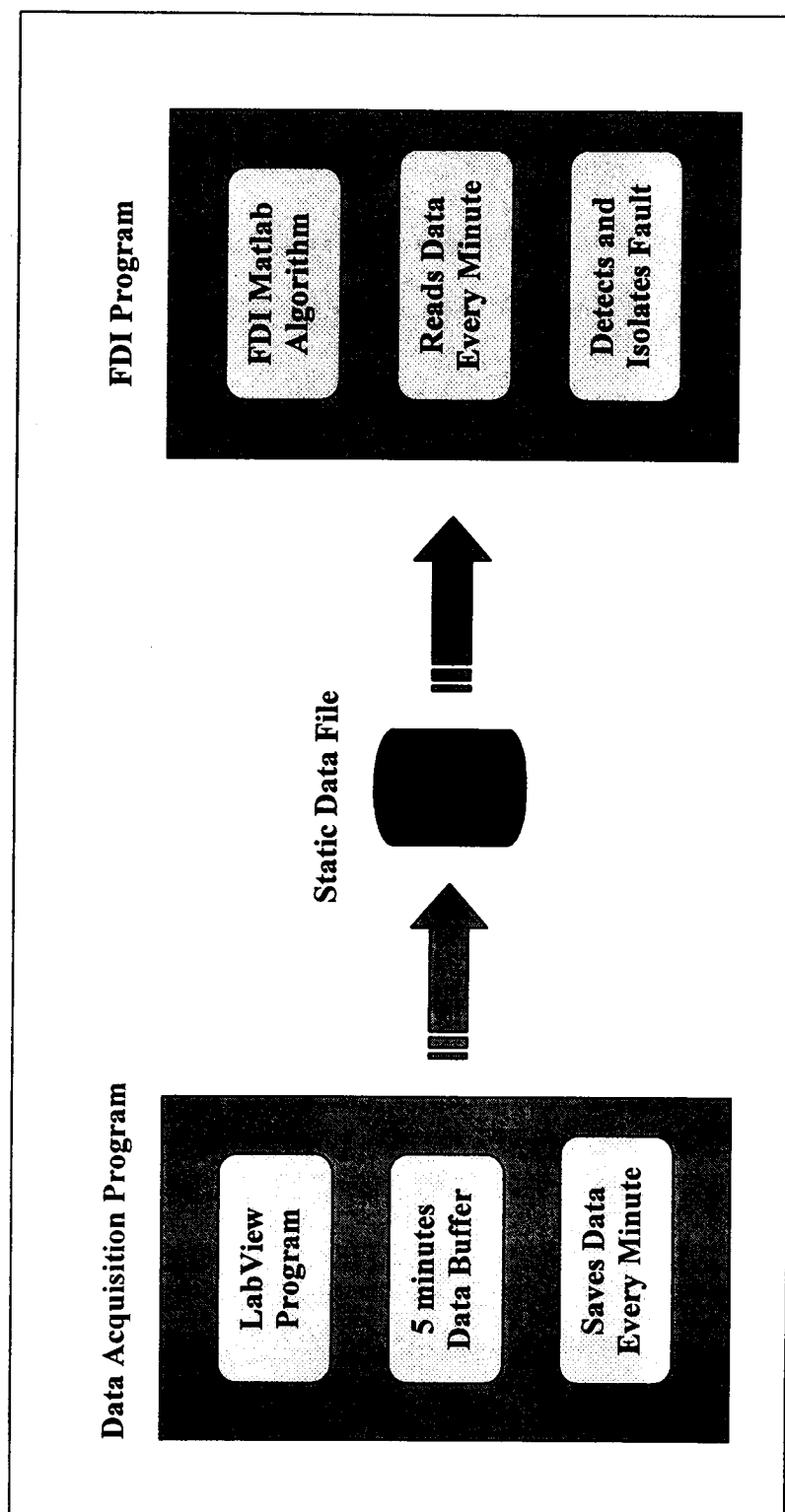


Figure 8.5. Schematic structure of Fault Detection and Isolation system. The LabView program uses a circular data buffer which contains 5 minutes of previous data. The data transfer between this program and the Matlab FDI program is performed by a static data file. Every minute the FDI algorithm reads the data from this static data file and performs the fault detection and isolation.

- Experiments where different faults were imposed on different loop components. This type of experiments was performed to check if the FDI algorithm could first detect system anomaly, and second if it could isolate the faulty component.

An interesting fact observed during the realization of the experiments in the loop is that even when the experimental water loop system was operated at steady-state conditions, a low frequency oscillation was observed all the time. This oscillation is due to two factors, both coming from the Inlet motor-operated valve. The first factor is that the valve's actuator has an internal dead-band, that is, even though the water level controller output is increasing, the valve stays steady in its current position until the controller's output reaches a threshold. Reaching this threshold, the motor actuator moves the valve to its new position. The second factor is that the valve has a free motion of approximated five-degree in angle. These two factors cause the system to oscillate at a low frequency around the set point values. Figure 8.6 shows this oscillation.

The Multi-Resolution signal decomposition using wavelet transform, described in Chapter 5, was used to resolve this problem. By applying this technique, oscillatory and noisy components can be separated from the original signal, leaving the DC component for the FDI analysis. Therefore, this technique was applied to the data coming from the low-pressure water loop system. Preprocessing the loop variables with the multi-resolution wavelet signal decomposition, a smoother signal can be presented to the FDI algorithm, helping to avoid false failure alarms. Figure 8.7 shows an example of the controller output signal during a failure in water level sensor. Along with the original

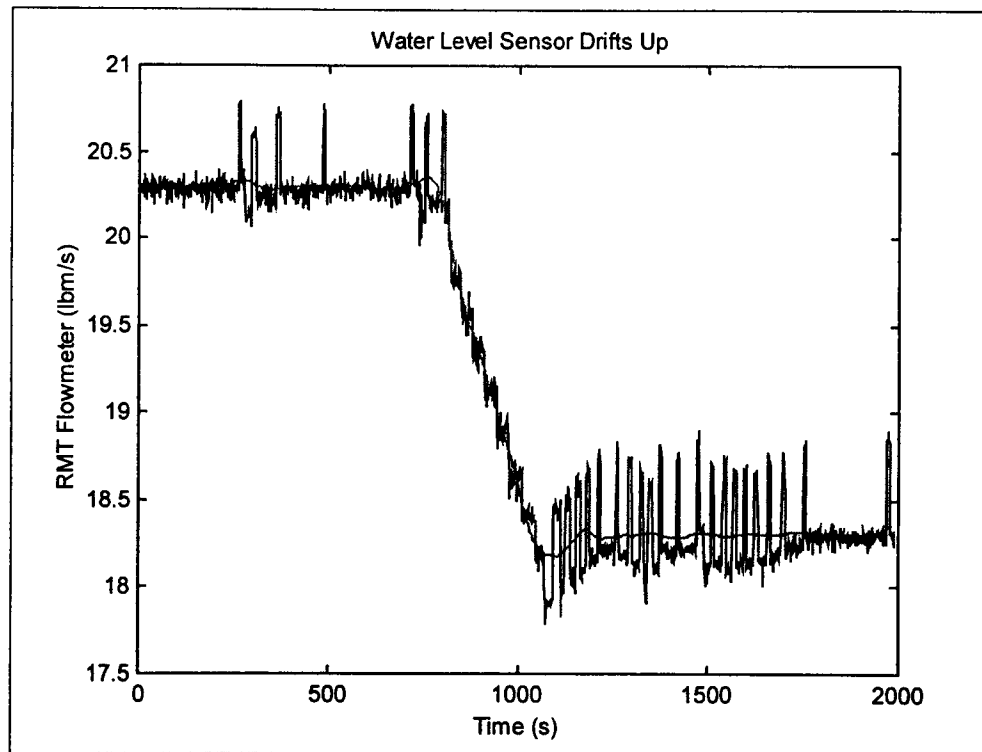


Figure 8.6. Low frequency low magnitude system oscillation due to Inlet MOV intrinsic features.

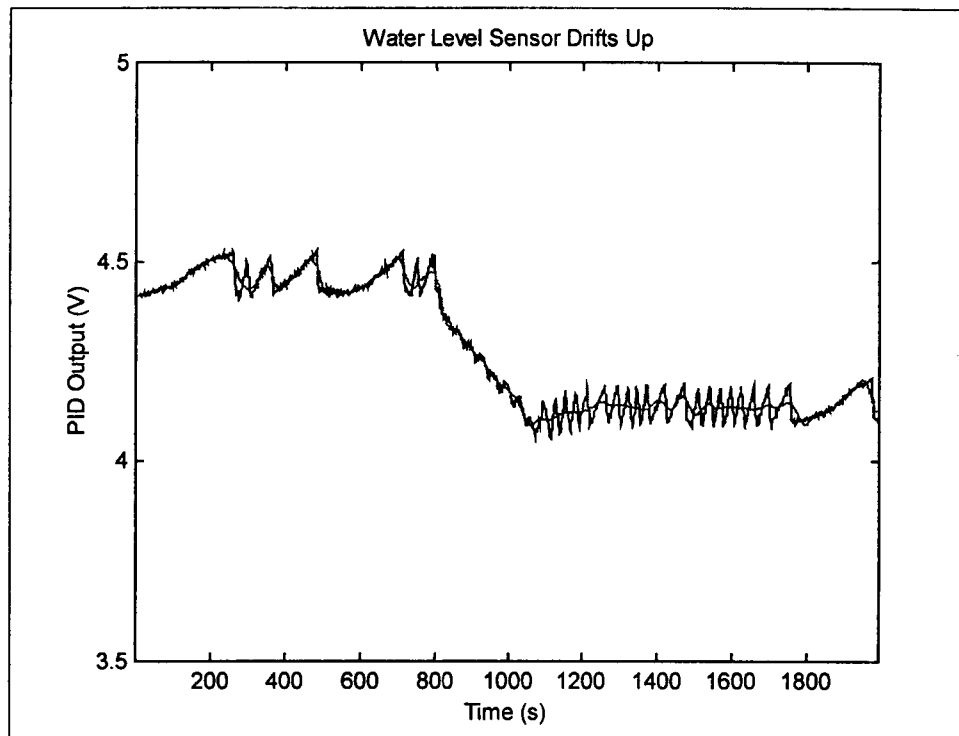


Figure 8.7. An example of the PID output signal during a failure in the water level sensor. Along with the original sensor signal, the multi-resolution wavelet decomposed signal is also plotted (seen as a smoother line).

sensor signal, the multi-resolution wavelet signal is also plotted (seen as a smoother line). As before, the best decomposition level is chosen automatically by the algorithm developed in this research. Also, an extrapolation process also developed in this work diminished the edge effects.

8.6.1. Changes in the System Set Points

As discussed earlier, even when the system variable values have changed from one condition to another, it does not imply that a fault has occurred in the system: the system may be operating at a different but normal condition. A fault detection and isolation diagnostic system must differentiate between changes in a sensor output signal due to a sensor's failure and one caused by changes in the process normal operational condition.

The low-pressure water loop system used in this work has three types of set points: the Bypass MOV position, the Outlet MOV position, and the water level. The FDI algorithm should not give false alarms when one has been or all of them have been changed to a new value(s). The next results show the FDI performance for these cases.

A - Change in the Water Level Set point

Predicted Inlet Flow rate: 0.008299	(← from GMDH Model)
Model Inlet Flow rate: 0.008174	
Percentage of Variation: -1.518967 (%)	

Predicted Outlet Flow rate: 0.065678 (← from GMDH Model)
Model Outlet Flow rate: 0.065395
Percentage of Variation: -0.432238 (%)

Predicted PID Output: 0.384712 (← from GMDH Model)
Model PID Output: 0.388666
Percentage of Variation: 1.017533 (%)

Predicted Water Level: 0.042803 (← from GMDH Model)
Model Water Level: 0.042857
Percentage of Variation: 0.127335 (%)

FaultType = No fault detected (← from FDI algorithm)

B - Change in the Bypass MOV position set point

Predicted Inlet Flow rate: 0.009461
Model Inlet Flow rate: 0.009264
Percentage of Variation: -2.119921 (%)

Predicted Outlet Flow rate: 0.074374
Model Outlet Flow rate: 0.074115
Percentage of Variation: -0.350177 (%)

Predicted PID Output: 0.492883
Model PID Output: 0.500436
Percentage of Variation: 1.509285 (%)

Predicted Water Level: 0.048501
Model Water Level: 0.048571

Percentage of Variation: 0.145451 (%)

FaultType = No fault detected

C - Change in the Outlet MOV position set point

Predicted Inlet Flow rate: 0.010638

Model Inlet Flow rate: 0.010459

Percentage of Variation: -1.717527 (%)

Predicted Outlet Flow rate: 0.083897

Model Outlet Flow rate: 0.083669

Percentage of Variation: -0.272773 (%)

Predicted PID Output: 0.568221

Model PID Output: 0.575381

Percentage of Variation: 1.244373 (%)

Predicted Water Level: 0.048509

Model Water Level: 0.048571

Percentage of Variation: 0.127548 (%)

FaultType = No fault detected

D - Change in all set points simultaneously

Predicted Inlet Flow rate: 0.009427

Model Inlet Flow rate: 0.009228

Percentage of Variation: -2.149558 (%)

Predicted Outlet Flow rate: 0.074271
Model Outlet Flow rate: 0.073825
Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.490487
Model PID Output: 0.497338
Percentage of Variation: 1.377414 (%)

Predicted Water Level: 0.042790
Model Water Level: 0.042857
Percentage of Variation: 0.156170 (%)

FaultType = No fault detected

8.6.2. Artificially Imposed Faults

A - Fault in the Inlet MOV

Predicted Inlet Flow rate: 0.008795
Model Inlet Flow rate: 0.009264
Percentage of Variation: 5.060935 (%)

Predicted Outlet Flow rate: 0.074374
Model Outlet Flow rate: 0.074115
Percentage of Variation: -0.350177 (%)

Predicted PID Output: 0.463829
Model PID Output: 0.424345
Percentage of Variation: -9.304644 (%)

Predicted Water Level: 0.048501
Model Water Level: 0.048571
Percentage of Variation: 0.145451 (%)

FaultType = Inlet MOV Fault

B - Fault in the Water Level Sensor

Predicted Inlet Flow rate: 0.008453
Model Inlet Flow rate: 0.008325
Percentage of Variation: -1.531720 (%)

Predicted Outlet Flow rate: 0.074374
Model Outlet Flow rate: 0.066604
Percentage of Variation: -11.666963 (%)

Predicted PID Output: 0.394609
Model PID Output: 0.399433
Percentage of Variation: 1.207848 (%)

Predicted Water Level: 0.043585
Model Water Level: 0.048571
Percentage of Variation: 10.265974 (%)

FaultType = Water Level Sensor Fault

C - Fault in the Inlet Flowmeter

Predicted Inlet Flow rate: 0.009451
Model Inlet Flow rate: 0.010198

Percentage of Variation: 7.327647 (%)

Predicted Outlet Flow rate: 0.074374

Model Outlet Flow rate: 0.074115

Percentage of Variation: -0.350177 (%)

Predicted PID Output: 0.463829

Model PID Output: 0.470999

Percentage of Variation: 1.522300 (%)

Predicted Water Level: 0.053477

Model Water Level: 0.048571

Percentage of Variation: -10.100387 (%)

FaultType = RMT Flow meter Fault

D - Fault in the Outlet MOV

Predicted Inlet Flow rate: 0.007568

Model Inlet Flow rate: 0.007519

Percentage of Variation: -0.651970 (%)

Predicted Outlet Flow rate: 0.074374

Model Outlet Flow rate: 0.060150

Percentage of Variation: -23.647898 (%)

Predicted PID Output: 0.342292

Model PID Output: 0.344083

Percentage of Variation: 0.520683 (%)

Predicted Water Level: 0.039417
Model Water Level: 0.048571
Percentage of Variation: 18.847738 (%)

Water Level Set point: 0.048571
Measured Water Level: 0.048571
Percentage of Variation: -0.000000 (%)

Outlet MOV Set point Position: 0.800000
Measured Position: 0.720702
Percentage of Variation: 9.912258 (%)

Bypass MOV Set point Position: 0.800000
Measured Position: 0.800000
Percentage of Variation: 0.000000 (%)

FaultType = Outlet MOV position

E -Fault in the Bypass MOV

Predicted Inlet Flow rate: 0.010016
Model Inlet Flow rate: 0.009264
Percentage of Variation: -8.115071 (%)

Predicted Outlet Flow rate: 0.074374
Model Outlet Flow rate: 0.074115
Percentage of Variation: -0.350177 (%)

Predicted PID Output: 0.463829
Model PID Output: 0.519231

Percentage of Variation: 10.670006 (%)

Predicted Water Level: 0.048501

Model Water Level: 0.048571

Percentage of Variation: 0.145451 (%)

Water Level Set point: 0.048571

Measured Water Level: 0.048571

Percentage of Variation: 0.000000 (%)

Outlet MOV Set point Position: 0.800000

Measured Position: 0.800000

Percentage of Variation: 0.000000 (%)

Bypass MOV Set point Position: 0.800000

Measured Position: 0.881923

Percentage of Variation: -10.240353 (%)

FaultType = Bypass MOV Position

F - Fault in the Outlet Flowmeter

Predicted Inlet Flow rate: 0.009451

Model Inlet Flow rate: 0.009264

Percentage of Variation: -2.015160 (%)

Predicted Outlet Flow rate: 0.074374

Model Outlet Flow rate: 0.066437

Percentage of Variation: -11.947919 (%)

Predicted PID Output: 0.393178
Model PID Output: 0.470999
Percentage of Variation: 16.522492 (%)

Predicted Water Level: 0.048501
Model Water Level: 0.048571
Percentage of Variation: 0.145451 (%)

Water Level Set point: 0.048571
Measured Water Level: 0.048571
Percentage of Variation: 0.000000 (%)

Outlet MOV Set point Position: 0.800000
Measured Position: 0.800000
Percentage of Variation: 0.000000 (%)

Bypass MOV Set point Position: 0.800000
Measured Position: 0.800000
Percentage of Variation: 0.000000 (%)

FaultType = Outlet Turbine Flowmeter

G - Fault in the WL Controller

Predicted Inlet Flow rate: 0.006250
Model Inlet Flow rate: 0.006317
Percentage of Variation: 1.060405 (%)

Predicted Outlet Flow rate: 0.050612
Model Outlet Flow rate: 0.050532

Percentage of Variation: -0.159051 (%)

Predicted PID Output: 0.272895

Model PID Output: 0.270463

Percentage of Variation: -0.898978 (%)

Predicted Water Level: 0.033142

Model Water Level: 0.033116

Percentage of Variation: -0.078217 (%)

Water Level Set point: 0.048571

Measured Water Level: 0.033116

Percentage of Variation: 31.819221 (%)

Outlet MOV Set point Position: 0.800000

Measured Position: 0.800000

Percentage of Variation: 0.000000 (%)

Bypass MOV Set point Position: 0.800000

Measured Position: 0.800000

Percentage of Variation: 0.000000 (%)

FaultType = PID controller Fault

The results of the experimental studies showed that the performance of the FDI algorithm in detecting set point changes and faulty devices was excellent. No false alarms were observed. The FDI algorithm correctly isolated the faulty devices.

8.7. FAULT COMPENSATION USING A FUZZY LOGIC SYSTEM

The last step in the FDI algorithm is the reconstruction of a fault variable. In this step, the FDI algorithm attempts to compensate the signal used by the faulty component. If it succeeds, the process can be kept in operation without losing performance until the next scheduled maintenance. As described in Chapter 7, this dissertation uses a fuzzy logic system for computing the amount of compensation required for reconfiguring the faulty device. The ANFIS algorithm was used for adjustments in the fuzzy membership functions. Each MOV fuzzy engine has two inputs and one output, each of them represented by five Gaussian-type membership functions.

All the steps for creating the fuzzy logic engines, described in Chapter 7, were repeated using actual faulty data from the low-pressure water loop. Results similar to the training data analysis were also achieved for the experimental cases. However, the output of this module is currently used as an advisory information, that is, the module just suggests the value of the compensated variable. This was suggested in order to avoid making any changes in the system without alerting the engineer responsible for the control loop maintenance.

CHAPTER 9

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

9.1. SUMMARY

A new approach for the fault detection and isolation of sensors and field devices in process control loops was developed and demonstrated in this research. The method is composed of detecting a system anomaly, isolating the faulty component, computing its deviation from expected value for a given system's normal condition, and finally adjusting its output when applicable. To accomplish these tasks, many different data analysis and modeling techniques were used. The FDI system integrates multi-resolution wavelet analysis (MRWA), Group Method of Data Handling (GMDH), rule-based expert system, and fuzzy logic inference engines. In the first two techniques, novel approaches were developed for improving their performance, such as automatic selection of the best decomposition level and minimization of edge effects for the MRWA, and automatic selection of basic functions and inclusion of rational terms in the GMDH.

For the feasibility studies of the proposed FDI algorithm, a theoretical model of a low-pressure water loop system was developed. To demonstrate applications to an operating system, an experimental control flow loop, with automated data acquisition, was established.

A general framework for system component failure detection and isolation was also proposed, allowing easy implementation of this methodology in any process control loop. This technique can also help the system operator or engineer in defining the minimal number of sensors to be installed in the system, necessary for reliable component validation. This feature helps in avoiding installation of unnecessary redundant sensors.

Chapter 2 presents the basic procedure proposed in this work for the development of a distributed loop component fault detection and isolation (FDI) algorithm. The modules for system monitoring, detection of anomalies, isolation of the most probable faulty loop-component, and device reconstruction are described.

Chapter 3 describes an approach that was developed based on the Group Method of Data Handling (GMDH), and was implemented as part of an on-line monitoring, fault detection and isolation (FDI) system. The GMDH provides a general framework for characterizing the relationship among a set of state variables of a process and is used for generating estimates of critical variables in an optimal data-driven model form. The GMDH is used to provide the necessary analytical redundancy to the FDI algorithm. The analytical redundancy is used to monitor the component's health.

In Chapter 4, a simulation model of a process control loop, including sensors, controllers, and actuators, was developed. This model was implemented in Matlab-Simulink [90] program environment. Also, a low-pressure water loop (LPWL) system was designed and built. A LabView [65] program was developed to acquire loop

measurements and to control the experiment remotely. For both the systems, known faults may be imposed on different devices such as pressure transmitters, motor-operated valves, control elements, and a centrifugal pump. The main purpose of each system was to provide useful data and an environment for developing and testing the proposed FDI algorithm.

In Chapter 5 a new technique was developed for validating isolated sensors in process control systems. This technique is based on multi-resolution signal decomposition using wavelet transforms and mapping of control variables and actuator dynamics. The cases of open-loop (sensors used for monitoring) and feedback dynamics (sensors used as a controller input) were considered. The primary objective was to separate process dynamics from sensor dynamics in operating distributed control systems. The technique was successfully applied in two different experiments: an open-loop system, where a drifting sensor under strong noise contamination was analyzed, and a closed-loop system, where a drifting water level sensor used in a feedback system was analyzed.

In Chapter 6, a PC-based Fault Detection and Isolation (FDI) algorithm was developed and implemented using the theoretical model developed in Chapter 4. Mathematical models for analytical redundancy were derived, and rules for expert system for fault isolation were defined. The FDI algorithm performs data acquisition, data-driven modeling, fault detection and isolation of a faulty sensor or field device.

In Chapter 7, the development of an algorithm for device reconstruction was presented. The algorithm was implemented into the FDI program as a separate module. A fuzzy-logic estimator was used to accomplish this task. The membership function parameters were chosen automatically by using the technique called ANFIS (Adaptive Neuro-Fuzzy Inference System).

In Chapter 8, all steps performed in Chapter 6 for development and implementation of the FDI algorithm were demonstrated for the experimental setup. A LabView program was developed to control the experiments, data acquisition and system perturbations (imposed faults). A detailed description of how this program interacts with the FDI algorithm was presented.

9.2. CONCLUSIONS

The results obtained throughout this dissertation have shown the feasibility of using the new Fault Detection and Isolation system for monitoring the operability of system components. The performance of the FDI algorithm was excellent with applications to both the process control simulation models and the experimental flow control loop.

Some of the success was directly attributed to the use of the enhanced Group Method of Data Handling approach for process modeling developed in this research, which gave accurate values for the analytical measurements. These redundant estimators

were extensively used throughout the FDI, and due their good prediction, the FDI performance was highly robust.

Other factors that contributed to the success of the FDI algorithm were the innovations implemented in the multi-resolution wavelet analysis and in the GMDH training algorithm. The rule-based expert system was able to isolate faults in sensors, controllers, actuators and other field devices with a small false alarm rate.

The procedure for developing and implementing the FDI algorithm in an arbitrary system (Chapter 2) is basically generic. It can be applied to most of the industrial control loops in existence. Also, the set of basic functions defined in the GMDH algorithm in Chapter 4 is quite sufficient for representing the dynamics of a large number of process control loops.

The FDI algorithm was developed with the focus on achieving modularity. The tasks of generating analytical redundancy, fault detection, fault isolation, and device reconstruction, were created as separate modules. If modifications in one of these steps are desired, it could be achieved with minimum interference with the other modules.

This dissertation has successfully integrated a number of emerging techniques in order to accomplish the goal of developing and implementing a reliable FDI system. Application to theoretical and experimental systems demonstrated a high performance of the approach for not only detecting and isolating faulty devices, but also in minimizing false alarms for cases where changes in the loop variables were caused by normal changes in the system set points.

9.3. RECOMMENDATIONS FOR FUTURE WORK

The theoretical and experimental work carried out for testing the performance of the Fault Detection and Isolation of sensors and loop devices had given excellent results. All the imposed faults were properly detected and isolated. Also, changes in set points did not cause false alarms. However, this study was realized in a very controlled environment for both the theoretical model and the experimental setup. The system should be evaluated by implementation in an industrial control loop, such as in a chemical plant or a power plant. Some practical issues not studied in this research may present potential difficulties in implementing the FDI algorithm.

Another research area is to develop rule-bases for the common process control loops. There are about fifteen generic control loops that cover most of the industrial systems.

Another important research aspect is the development of a system that could automatically generate rules for fault isolation based on a system description. This program could use extensive graphical user interface, such that the user would just drag and drop pre-defined loop components into a window, connect them by lines and the system would come up with the basic rules for the fault isolation module. This program could also specify the minimum requirements for a robust FDI algorithm.

This research addressed the detection of isolated faults. In some instances, degradation may occur in more than one plant device simultaneously. The detection and

isolation of simultaneous multiple faults is a challenging and practically important research topic that should be pursued in the future.

LIST OF REFERENCES

REFERENCES

- [1] M. C. Aaron and F. H. Guy, "Accurate Predictive Emissions Monitoring System (PEMS) for an Ethylene Furnace," *Environmental Progress*, Vol. 15 No.1, pp. 19-27, Spring 1996.
- [2] S. Alag, K. Goebel, and A. M. Agogino, "Methodology for Intelligent Sensor Validation and Fusion used in Tracking and Avoidance of Objects for Automated Vehicles," *Proceedings of the American Control Conference*, Vol. 5, 95CH35736, pp. 3647-3653, part 5 (of 6), 1995.
- [3] S. M. Alexander; C. M. Vaidya, and J.H. Graham, "Model for the Diagnosis of CIM Equipment," *Computers and Electrical Engineering*, Vol. 19 No. 2, pp. 175-183, March 1993.
- [4] J. W. Allen and L. M. Oesterling, "Loose Part Monitor Sensor Validation, Impact Detection, and Characterization," *Nuclear Plant Journal*, Vol.7 No. 3 May-June 1989.
- [5] J. W. Allen, "Loose Part Monitor Sensor Validation, Impact Detection, and Characterization," *Nuclear Plant Journal*, Vol. 7 No. 3 May-June 1989.

- [6] E. J. E. Amadi, "Detecting Changes in the Condition of Process Instruments," Proceedings of the 1993 IEEE Instrumentation and Measurement Technology Conference, pp. 515-518, Irvine, CA, USA, 1993.
- [7] E. J. E. Amadi and H. Zhu, "Detecting Changes in the Condition of Process Instruments," IEEE Transactions on Instrumentation and Measurement, Vol. 43 No. 2, pp. 355-358, April 1994.
- [8] M. Aynsley, D. Peel; and A. J. Morris, "Real-time Knowledge-based System for Fermentation Control," Proceedings of the American Control Conference. Pittsburgh, PA, USA, Vol. 3 (of 3), pp. 2239-2244, 1989.
- [9] P. Balle et al, "Integrated Control, Diagnosis and Reconfiguration of a Heat Exchanger," IEEE Control Systems, pp. 52-63, June 1998.
- [10] D. M. Braunstein, A. S. Brown, and D. L. Drew, "Knowledge-based System for Process Monitoring and Control in Propellant Manufacturing," Proceedings of the ISA/89 International Conference and Exhibition: Advances in Instrumentation and Control (Part 2 of 4). Philadelphia, USA, pp. 457-464, 1989.
- [11] R. P. Champagne, "Accurate Valve Diagnosis Depends on Accurate Data Interpretation," InTech, pp. 48-50, January 1998.
- [12] B. Chandrasekaran and W. F. Punch, "Hierarchical Classification: its Usefulness for Diagnosis and Sensor Validation," IEEE-J-Sel-Areas-Commun. Vol. 6 No 5, pp. 884-891, June 1988.

- [13] P. Soo-Chang and Y. Min-Hung, "An Introduction to Discrete Finite Frames – Wavelet Theory," IEEE Signal Processing Magazine, Vol. 1 pp. 84-96, November 1997.
- [14] G. H Chisholm et al, "Toward Formal Analysis of Ultra-reliable Computers – a Total System Approach," Proceedings - IEEE/AIAA 7th Digital Avionics Systems Conference. Fort Worth, TX, USA, pp. 678-686, 1997.
- [15] H. P. Chou, J. Prock, and J. P. Bonfert, "Fault Diagnosis in Nuclear Power Plants using an Artificial Neural Network Technique," Transactions of the 1991 American Nuclear Society Winter Meeting, San Francisco, pp. 149-150, 1993.
- [16] E. Y. Chow and A. S. Willsky, "Analytical Redundancy and the Design of Robust Failure Detection Systems," IEEE Transactions on Automatic Control, Vol. AC-29, No. 7, pp. 603-614, 1984.
- [17] R. N. Clark, "Detecting Instrument Malfunctions in Control Systems," IEEE Transactions on Aerospace and Electronic Systems, AES-14, pp. 558-563, 1975.
- [18] R. N. Clark, "A Simplified Instrument Failure Detection Scheme," IEEE Transactions on Aerospace and Electronic Systems, AES-14, pp. 456-465, 1978.
- [19] R. N. Clark and W. Setzer, "Sensor Fault Detection in a System with Random Disturbances," IEEE Transactions on Aerospace and Electronic Systems, AES-16, pp. 468-473, 1980.

- [20] M. Cornell, "KATE shell – An Implementation of Model-based Control, Monitor and Diagnosis," First Annual Workshop on Space Operations Automation and Robotics (SOAR '87). Houston, TX, USA, NASA Conference Publication. No. 2491, pp. 355-360, 1987.
- [21] C. V. Dahanwada and E. B. Bartlett, "A New Method for Nuclear Plant Diagnostics Using Neural Networks," Transactions of the 1992 American Nuclear Society International Conference on Fifty Years of Controlled Nuclear Chain Reaction: Past, Present, and Future, Chicago, Illinois, pp. 114-125, 1992.
- [22] B. M. Dawant et al, "SIMON: a Distributed Computer Architecture for Intelligent Patient Monitoring," Expert-Systems-with-Applications. Vol. 6 No. 4, pp. 411-420, October-December 1993.
- [23] J. C. Deckert et al, "F-8 Digital Fly-by-wire Sensor Failure Identification Using Analytic Redundancy," IEEE Transactions on Automatic Control, Vol. AC-22, No. 5, October 1977.
- [24] M. Desai and A. Ray, "A Fault Detection and Isolation Methodology," Proceedings of 20th Conference on Decision and Control, pp. 1363-1369, 1981.
- [25] D. M. Himmelblau, M. Bhalodia, "On-line Sensor Validation of Single Sensors using Artificial Neural Networks," Proceedings of the American Control Conference, Vol. 1, 95CH35736, pp. 766-770, part 1 (of 6). Seattle, WA, USA, 1995.

- [26] D. Dong; T. J. McAvoy, and J. L. Chang, "Emission Monitoring using Multivariate Soft Sensors," Proceedings of the American Control Conference, Vol. 1, 95CH35736, pp. 761-765, part 1 (of 6). Seattle, WA, USA, 1995.
- [27] A. S. Erbay and B. R. Upadhyaya, "A Personal Computer-Based On-line Signal Validation Systems for Nuclear Power Plants." Nuclear Technology, Vol.119, No.1, pp 63-75, July 1997.
- [28] E. Eryurek and B. R. Upadhyaya, "Fault-Tolerant Control and Diagnostics for Large-scale Systems," IEEE Control Systems, Vol.15, No.5, pp 34-42, October 1995.
- [29] E. Eryurek and B. R. Upadhyaya, "Sensor Validation for Power Plants using Adaptive Backpropagation Neural Network," IEEE Transactions on Nuclear Science, Vol. 37 No. 1 part 2, pp. 1040-1047. 1989 Nuclear Science Symposium, April 1990.
- [30] S. J. Farlow, Self-organizing Methods in Modeling: GMDH-type Algorithms, New York : M. Dekker, 1984.
- [31] P. M. Frank, "Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy – A Survey and Some New Results," Automatica, Vol. 26, No. 3, pp. 459-474, 1990.

- [32] P. M. Frank, "Fault Detection in Dynamic Systems via State Estimation – A Survey," System Fault Diagnostics, Reliability and Related Knowledge-Based Approaches, Reidel, Dordrecht, Vol. 1, pp. 35-98, 1987.
- [33] P. M. Frank, "Advances in Observer-based Fault Diagnosis," International Conference on Fault Diagnosis, France, pp. 817-836, 1993.
- [34] P. M. Frank and J. Wunnenberg, "Robust Fault Diagnosis Using Unknown Input Observer Schemes," Fault Diagnosis Systems, Prentice Hall, pp. 47-98, 1989.
- [35] W. Ge and Z. Fang, "Detection of Faulty Components via Robust Observation," International Journal of Control, Vol. 47, pp. 581-599, 1988.
- [36] J. Gertler and D. Singer, "A New Structural Framework for Parity Equation-based Failure Detection and Isolation," Automatica, Vol. 26, No. 2, pp. 381-388, 1990.
- [37] J. Gertler, Fault Detection and Diagnosis in Engineering Systems, Marcel Dekker, Inc., 1998.
- [38] J. Gertler, "Analytical Redundancy Methods in Fault Detection and Isolation," IFAC/IMACS Safe process Conference, Baden-Baden, Germany, pp. 9-21, 1991.
- [39] J. Gertler, "Survey of Model Based Failure Detection and Isolation in Complex Plants," IEEE Control Systems Magazine, pp. 3-11, 1988.
- [40] O. Glockler, "Fault Detection in Nuclear Power Plants Applying the Sequential Probability Ratio Test of MAR-Based Residual Time Series," Proceedings of the

Artificial Intelligence - Frontiers in Innovative Computing for the Nuclear Industry, Jackson, Wyoming, pp. 859-868, 1991.

- [41] K. Goebel and A. Agogino, "An Architecture for Fuzzy Sensor Validation and Fusion for Vehicle Following in Automated Highways," 29th international Symposium on Automotive (ISATA) in Florence, Italy, June 4, 1996.
- [42] R. E. Goldschmidt and P. C. Miller, "Means for Achieving Accuracy and Validity in Gas Turbine Temperature Measurement," Instrumentation in the Aerospace Industry: Proceedings of the 40th International Instrumentation Symposium, pp. 221-231, Baltimore, MD, USA, 1994.
- [43] W. Gough, "Predictive Adaptive Control Aids Pulp Digestion," Control Engineering, p 97, November 1997.
- [44] T. H. Guo and J. Musgrave, "Neural Network Based Sensor Validation for Reusable Rocket Engines," Proceedings of the American Control Conference. Vol. 2, 95CH35736, pp. 1367-1372, part 2 (of 6). Seattle, WA, USA, 1995.
- [45] C. Hawman, "Intelligent Alarm Management Saves Industry," InTech, Vol.44, No.10, pp 34-36, October 1997.
- [46] B. K. Hajek et al, "Generic Task Approach to a Real Time Nuclear Power Plant Fault Diagnosis and Advisory System," Proceedings of the International Workshop on Artificial Intelligence for Industrial Applications, Hitachi, Japan, pp. 154-160, 1995.

- [47] F. U. Henderson, "Real-time Advisory Control Applications in the Control Room," Proceedings-of-the-1991-ISA. Marketing & Sales, Scientific Instrumentation & Research, Process Measurement & Control, Robotics & Expert Systems, Maintenance, & Water & Wastewater Division Symposia Edmonton 91 Spring Symposium, pp. 161-168, 1991.
- [48] S. A. Heger, E. K. Holbert, and M. A. Ishaque, "Fuzzy Associative Memories for Instrument Fault Detection," Annals of Nuclear Energy. Vol. 23 No. 9, pp. 739-756, June 1996.
- [49] M. Henry, "Automatic Sensor Validation," Control-and-Instrumentation, Vol. 27 No. 9, pp. 60-61, September 1995.
- [50] M. Henry, "Validating Data from Smart Sensors," Control Engineering. Vol. 41 No. 9, pp. 63-66, August 1994.
- [51] M. Henry, "Sensor Validation and Fieldbus," Computing and Control Engineering Journal. Vol. 6 No. 6, pp. 263-269, December 1995.
- [52] M. P. Henry, "Intelligent Sensors and the Need for New Metrics," IEE Colloquium (Digest). No. 236, IEE, Stevenage, England, pp. 3/1-3, IEE Computing and Control Division Colloquium on Fieldbus Devices - A Changing Future. London, UK, 1994.
- [53] M. P. Henry, "Hardware Compilation - a New Technique for Rapid Prototyping of Digital Systems - Applied to Sensor Validation," Control-Engineering-Practice, Vol. 3 No. 7, pp. 907-924, July 1995.

- [54] R. E. Henry et al, "Helping Operators Improve Plant Performance: HEATXPRT - An On-line Expert System," Proceedings of the 54th Annual Meeting of the American Power Conference. Chicago, IL, USA Vol. 54 part 1, pp. 671-680, 1992.
- [55] D. M. Himmelblau, "On-line Sensor Validation of Single Sensors Using Artificial Neural Networks," Proceedings of the American Control Conference, Seattle, Washington, June 1995.
- [56] K. E. Holbert et al, "Redundant Sensor Validation by Using Fuzzy Logic," Nuclear Science and Engineering, Vol. 118 No. 1, pp. 54-64, September 1994.
- [57] M. Hou and P. C. Muller, "Design of Robust Observers for Fault Isolation," IFAC/IMACS Safe process Conference, Baden-Baden, Germany, pp. 295-300, 1991.
- [58] M. A. Hunt, "Quantitative Pattern Recognition Using Nonlinear Model-based Analysis," Ph.D. Dissertation, The University of Tennessee, Knoxville – TN USA.
- [59] G. Ivakhnenko, Home Page at http://www.inf.kiev.ua/GMDH-home/GMDH_toc.htm August 26, 1998
- [60] R. Isermann, "Process Fault Detection Based on Modeling and Estimation Methods – A Survey," Automatica, Vol. 20, No. 4, pp. 387-404, 1984.
- [61] R. Isermann, "Fault Diagnosis of Machines via Parameter Estimation and Knowledge Processing – Tutorial Paper," Automatica, Vol. 29, No. 4, pp. 815-835, 1993.

- [62] Y. J. Kim, "Framework for an On-line Diagnostic Expert System for Intelligent Manufacturing," *Expert Systems with Applications*, Vol. 9 No. 1, pp. 55-61, 1995.
- [63] K. Kavaklioglu and B.R. Upadhyaya, "Monitoring Feedwater Flow Rate and Component Thermal Performance of Pressurized Water Reactors by Means of Artificial Neural Networks," *Nuclear Technology*, Vol.107, pp112-123, July 1994.
- [64] M. Kittamura, "Detection of Sensor Failures in Nuclear Plants Using Analytical Redundancy," *Transactions of the American Nuclear Society*, Vol. 34, pp. 581-583, 1980.
- [65] LabView User's Manual – National Instruments, 1998.
- [66] S. C. Lee, "Fault Detectability Analysis Based on Sensor Relationships for Space Power Systems," *Proceedings of the 28th Intersociety Energy Conversion Engineering Conference*. Atlanta, GA, USA, Vol. 1, pp. 19-24, 1993.
- [67] S. C. Lee, "Sensor Value Validation Based on Systematic Exploration of the Sensor Redundancy for Fault Diagnosis KBS," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24 No. 4, pp. 594-605, April 1994.
- [68] J. S. Lin, C. L. Resch, and J. V. Palmer, "Sensor Validation for On-line Diagnostics," *Instrumentation in the Aerospace Industry – Proceedings of the 40th International Instrumentation Symposium*, pp. 519-527, Baltimore, MD, USA, 1994.

- [69] D. G. Luenberger, "Observing the State of a Linear System," IEEE Transactions on Military Electronics, MIL-8, pp. 74-80, 1963.
- [70] Y. Maki and K. A. Loparo, "A Neural-Network Approach to Fault Detection and Diagnosis in Industrial Processes," IEEE Transactions on Control, Systems Technology, Vol. 5, No. 6, November 1997.
- [71] M. A. Massoumnia, G. C. Vergheses, and A. S. Willsky, "Failure Detection and Identification," IEEE Transaction on Automatic Control, AC-31, pp. 839-846, 1996.
- [72] L. J. Line and D. E. Labbe, "Mt. Storm on-line 'What-If' Performance Program," 4th International Power Generation Exhibition and Conference Part 9 and 10. Tampa, FL, USA, Vol. 9-10, pp. 1901-1912, 1994.
- [73] W. C. Merrill; J. L. Musgrave; T. H. Guo, "Integrated Health Monitoring and Controls for Rocket Engines," SAE Technical Paper Series 1992 SAE Aerospace Atlantic Conference. 31, pp. 1-5, Dayton, OH, USA, 1992.
- [74] R.L. Moore, "Distributed Embedded Intelligence Blazing Processing Trails," InTech, Vol. 44, No.3, pp. 50-53, March 1997.
- [75] M. A. Moradian et al, "TurbinAID: Turbine Artificial Intelligence Diagnostics," Instrumentation-in-the-Power-Industry, Proceedings. Vol. 34, pp. 535-550. ISA 34th and the First Annual ISA/EPRI Joint Controls and Automation Conference. St. Petersburg Beach, FL, USA, 1995.

- [76] V. M. Morgenstern, B. R. Upadhyaya, and O. Glocker, "Signal Anomaly Detection and Characterization," Internal Report prepared for The U.S. Department of Energy Contract No. DE-AC02-86NE37959.
- [77] A. Emami-Naeini and M. M. Akhter, "Effect of Model Uncertainty on Failure Detection: The Threshold Selector," IEEE Transactions on Automatic Control, Vol. 33, No. 12, December 1988.
- [78] A. E. Nicholson and J. M. Brady, "Dynamic Belief Networks for Discrete Monitoring," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 11, November 1994.
- [79] M. R. Napolitano et al, "Kalman Filters and Neural-Network Schemes for Sensor Validation in Flight Control Systems," IEEE Transactions on Control Systems Technology, Vol. 6, No. 5, September 1998.
- [80] P. M. Olin and G. Rizzoni, "Design of Robust Fault Detection Filters," Proceedings of the American Control Conference, Boston, MA., pp. 1522-1527, 1991.
- [81] H. V. Panossian and W. D. Ewing, "Real-time Failure Detection Algorithm for Space Shuttle Main Engine," IEEE Control Systems, pp. 16-23, August 1997.
- [82] A. E. Nicholson and J. M. Brady, "Dynamic belief networks for discrete monitoring," IEEE Transactions on Systems, Man and Cybernetics, pp. 1593-1610, November 1994.

- [83] R. J. Patton, J. Chen, and S. B. Nielsen, "Model-based Methods for Fault Diagnosis: Some Guidelines," IEEE Transactions of the Institute of Measurement and Control, Vol. 17, No. 2, 1995.
- [84] R. J. Patton and S. M. Kangethe, "Robust Fault Diagnosis Using Eigenstructure Assignment of Observers," Fault Diagnosis in Dynamic Systems, Prentice Hall, pp. 100-154, 1989.
- [85] A. Ray and R. Luck, "An Introduction to Sensor Signal Validation in Redundant Measurement Systems," IEEE Control Systems Magazine. Vol. 11 No. 2, pp. 44-49, February 1991.
- [86] J. I. Sackett, "Application of AI Technology to Nuclear Plant Operation," CoED-J. Vol. 9 No. 3, pp. 45-52, July-September 1989.
- [87] S. P. Sakti and R. Thorn, "On-line Validation of Modulating Sensors," Proceedings of the 3rd Australian Instrumentation and Measurement Conference, No. 94 part 5, pp. 321-325, Adelaide, Australia, 1994.
- [88] R. Sanz et al, "Computational Intelligence in Process Control," IEEE International Conference on Neural Networks Conference Proceedings 4. IEEE, Piscataway, NJ, USA, 94CH3429-8, pp. 2203-2208, 1994.
- [89] E. A. Scarl, J. R. Jamieson, and C. I. Delaune, "Diagnosis and Sensor Validation through Knowledge of Structure and Function," IEEE Trans Syst. Man Cybern. Vol. SMC-17 No 3, pp. 360-368, May/June 1987.

- [90] Simulink – Matlab User's Manual. Math Works Inc. 1997
- [91] D. Shieh and B. R. Upadhyaya, "Stochastic Estimation Approach for the Evaluation of Thermal-Hydraulic Parameters in Pressurized Water Reactors," Nuclear Technology, Vol. 73, pp. 19-29, April 1986.
- [92] D. M. Sopocy et al, "Development of an On-line Expert System. Cycle Chemistry and Demineralizer Operation Advisor," ASTM Special Technical Publication. No. 1102, pp. 52-65. Monitoring Water in the 1990's: Meeting New Challenges. Denver, CO, USA, 1996.
- [93] M. E. Tay, "Keeping Tabs on Plant Energy and Mass Flows," Chemical Engineering, pp. 82-88, September 1996.
- [94] E. Turkcan and J. G. E. Versteegh, "Neural Network Training by Decoupled Extended Kalman Filter for Detection of Signal and System Failures,"
- [95] D. M. Sopocy et al, "Development of an On-line Expert System. HEATXPRT," Advances in Instrumentation, Proceedings, Vol. 45 part 1, pp. 219-228. Proceedings of the ISA '90 International Conference and Exhibition Part 1 (of 4). New Orleans, LA, USA, 1991.
- [96] R. E. Uhrig, "Use of Neural Networks in the Analysis of Complex Systems," Proceedings of SPIE The International Society for Optical Engineering. Proceedings of the 3rd Workshop on Neural Networks: Academic / Industrial / NASA / Defense. Alabama, AL, USA, pp. 17-24, Vol. 1721, 1993.

- [97] R. E. Uhrig, "Opportunities for Automation and Control of the Next Generation of Nuclear Power Plants," Nuclear-Technology, Vol. 88 No. 2, pp. 157-165, November 1989.
- [98] B. R. Upadhyaya and E. Eryurek, "Application of Neural Networks for Sensor Validation and Plant Monitoring," Nuclear-Technology, Vol. 97 No. 2, pp. 170-176. 1990 International Conference on Fast Reactor Safety. Snowbird, UT, USA, February 1992.
- [99] B. R. Upadhyaya et al. "Monitoring and Prognosis of Plant Components," P/PM Technology, Vol.7, No.6, pp. 4349, December 1994.
- [100] B. R. Upadhyaya, O. Glockler, and J. Eklund, "Multivariate Statistical Signal Processing Technique for Fault Detection and Diagnostics," ISA Transactions, Vol.29, No.4, pp. 79-95, 1990.
- [101] B. R. Upadhyaya, "Sensor Failure Detection and Estimation," Nuclear Safety, Vol. 6 No. 1, pp. 32-42, January-February 1985.
- [102] B. R. Upadhyaya et al., "Development and Testing of an Integrated Signal Validation System for Nuclear Power Plants," Vol.1,2,3, Final Report prepared for the U.S. Department of Energy, DOE/NE137959-34,35,36, September 1989.
- [103] V. J. VanDoren, "Advanced Control Software Goes Beyond PID," Control Engineering, pp. 73-78, January 1998.

- [104] N. D. Walker Jr. and G. F. Wyatt-Mair, "Sensor Signal Validation using Analytical Redundancy for an Aluminum Cold Rolling Mill," *Control-Engineering-Practice*. Vol. 3 No. 6, p 753-760, June 1995.
- [105] T. Washio, M. Kitamura, and K. Sugiyama, "Qualitative Simulation of Power Plant Dynamics Based on Design Knowledge," *Control Theory Adv. Technol.* Vol. 2 No. 3, p 433-449, Sep 1986.
- [106] K. R. Wheeler, A. P. Dhawan, and C. M. Meyer, "Space Shuttle Main Engine Sensor Modeling using Radial-basis-Function Neural Networks," *Journal of Spacecraft and Rockets*. Vol. 31 No. 6, pp. 1054-1060, November-December 1994.
- [107] K. R. Wheeler and A. P. Dhawan, "SSME Parameter Estimation using Radial Basis Function Neural Networks," *IEEE International Conference on Neural Networks Conference Proceedings 5*. IEEE, Piscataway, NJ, USA, 94CH3429-8, pp. 3352-3357, 1994.
- [108] A. S. Willsky, "A Survey of Design Methods for Failure Detection in Dynamic Systems," *Automatica*, Vol. 12, pp. 601-611, 1976.
- [109] L. Xiaoli, Y. Yingxue, and Y. Zhejun, "On-line Tool Condition Monitoring System With Wavelet Fuzzy Neural Network," *Journal of Intelligent Manufacturing*, Vol. 8, pp. 271-276, 1997.

- [110] X. Xu, J. W. Hines, and R. E. Uhrig, "On-line Sensor Calibration Monitoring and Fault Detection for Chemical Processes," Proceedings of Maintenance and Reliability Conference, Vol.2 (of 2), pp. 55.01-55.10, 1998.
- [111] Z. Xinmin et al, "Artificial Neural Network for Sensor Failure Detection in an Automotive Engine," Conference Record IEEE Instrumentation and Measurement Technology Conference-1, pp. 167-170, part 1 (of 3), Hamamatsu, Japan, 1994.
- [112] C. Y. J. Yang and D. W. Clarke, "A Self-Validating Thermocouple," IEEE Transactions on Control Systems Technology, Vol. 5, No. 2, March 1997.
- [113] S. K. Yung and D. W. Clarke, "Local Sensor Validation," Measurements and Control. Vol. 22 No. 5, pp. 132-140, June 1989.
- [114] S. K. Yung, "Signal Processing in Local Sensor Validation," Ph.D. Dissertation, University of Oxford, Department of Engineering Science, Oxford, U.K. 1993.
- [115] Matlab Wavelets Toolbox, User's Manual. Math Works Inc. 1997.
- [116] Matlab Fuzzy-logic Toolbox, User's Manual. Math Works Inc. 1998.
- [117] K. C. Gross et al, "Application of a Model-based Fault Detection System to Nuclear Plant Signals," Proc. 9th Intl. Conf. On Intelligent Systems Applications to Power Systems, pp. 66-70, Seoul, Korea, July 6-10, 1997.
- [118] R. M. Singer et al, "Model-Based Nuclear Power Plant Monitoring and Fault Detection: Theoretical Foundations," Proc. 9th Intl. Conf. On Intelligent

Systems Applications to Power Systems, pp. 60-65, Seoul, Korea, July 6-10, 1997.

- [119] R. M. Singer et al, "Power Plant Surveillance and Fault Detection: Applications to a Commercial LWR," Proc. IAEA Tech. Mtg. On Advanced Technology Improving Availability and Reliability of LWR's, Argonne, Ill, Sept 8-12, 1997.
- [120] J. P. Herzog et al, "Theoretical Basis of the Multivariate State Estimation Technique (MSET)," ANL-NT-49, Argonne National Laboratory, Dec 1997.
- [121] J. P. Herzog et al, "MSET Modeling of Crystal River-3 Venturi Flow Meters," Proc. ASME/JSME/SFEN 6th Intl. Conf. on Nuclear Engineering, San Diego, CA, May 10-15, 1998.
- [122] A. Miron et al, "Effects of Parameter Variation on MSET Models of the Crystal River-3 Feedwater Flow System," Trans. Am. Nucl. Soc., Nashville, TN, June 1-6, 1998.
- [123] J. P. Herzog et al, "Multivariate State Estimation Technique (MSET) Surveillance System," Appendix H of On-line Monitoring of Instrument Channel Performance, TR-104965, EPRI, October 1998.
- [124] K. C. Gross, S. Wegerich, and R. M. Singer, "Advanced Monitoring System Detects Faults In Advance of Exceeding Operational Limits," Power Magazine, November 1998).

- [125] A. Miron et al, "Fault-Tolerance Improvement for a MSET Model of the Crystal River-3 Feedwater Flow System," Proc. IEEE Nuclear Science Symposium, Toronto November 1998.
- [126] K. C. Gross, S. W. Wegerich and N. Zavaljevski, 'Multivariate State Estimation Technique for Automated Surveillance of Industrial Machinery," Proc. 3rd Intl. Conf. On Industrial Automation, Montreal, Canada, June 7-9, 1999 [*Accepted for publication*].

APPENDICES

APPENDIX A

GROUP METHOD OF DATA HANDLING

User's Guide

APPENDIX A

GROUP METHOD OF DATA HANDLING

USER's GUIDE

This User's Guide describes the procedure for using the Group Method of Data Handling (GMDH) algorithm, illustrations of implementation, and a listing of Matlab™ computer codes.

A.1. Procedure for Using the Group Method of Data Handling (GMDH) Software

- 1) Collect the data you want to use and store it in a matrix format where each column is a system variable and each row is an observation. Example: for a system of 15 variables you made 300 observations, you should construct a matrix of order 300 x 15.
- 2) For better prediction (mapping), normalize the data.
- 3) From a phenomenological knowledge of your system, define the qualitative relationships of your system. For example, the water flow rate going out of a tank piping is a function of the water level in the tank and the outlet MOV position.
- 4) Create a Matlab script file similar to the one shown in the first attachment to this document. This file starts the GMDH computation for finding the best set of basic functions to be used to map the input to the output variables. It also defines the

number of layers needed in the GMDH structure. If the number of layers exceeds 30 the process is stopped.

- 5) At the end of this process, a variable called RESIDUAL contains the residuals of all possible combinations of basic functions. Using the Matlab function called SORT ([y, index]=sort(residual)) you have the residuals sorted from smallest to the highest values. The index variable contains the JobCounter values. JobCounter gives you the GMDH model structure. Normally you use the first JobCounter (index(1)).
- 6) You can use the best GMDH model by using index(1) instead of varying JobCounter variable from 1 to 2^{MaxPower} . By doing this you are making the best prediction of the output variable using the input variables.

A.1.1. Observations:

- a) The number of nodes in a new layer of a GMDH model is given by $m(m-1)/2$, where m is the number of independent variables for each layer (the inputs). If, for example, the number of independent variables is 4, then for the first layer $m=4$. For the second layer $m=6$, and for the third $m=15$, and so on. To avoid explosion of the number of nodes in a GMDH layer, the program always uses the best m nodes for the succeeding layer. That is, all the layers will always have m nodes.
- b) If the number of independent variables is equal to 2, the GMDH would stop in the first layer ($m(m-1)/2 = 1$). However, when this happens, the Matlab script file called TrainTest.m (used in the training phase), creates a third variable which is the ratio of

the first input to the second (x_1/x_2). This allows the GMDH to go beyond the first layer.

- c) If the number of independent variables is one, you should come up with some variants of this variable, such as $\sin(input)$, to increase the number of input variables for the GMDH training phase. Another option is the use of polynomial mapping instead of GMDH mapping for these situations.

A.2. Subroutines Called by the GMDH Algorithm

- 1) [TrainInputData, TrainOutputData, TestInputData, TestOutputData] =
TrainTest(InputData, OutputData, NPoints)

Description: Function used to create matrices for training and for testing GMDH models. The number of observations in the testing subset is user defined. The observations in each subset are randomly selected by the algorithm.

InputData: matrix of independent variables of size (measurement points x number of variables)

OutputData: the dependent variable (measurements x 1)

NPoints: Size of the Test data set. This is set to half the number of measurements. If this value is set to zero, the function will use the set of indices used in the previous case.

TrainInputData: Set of input data to be used for training purposes

TrainOutputData: Set of output data to be used for training purposes

TestInputData: Set of input data for testing the model.

TestOutputData: Set of output data for testing the GMDH model.

2) [Layer,LayerOrder,Coef] = GMDH_PBK(x_train,y_train,x_check,y_check)

Description: Basic function for obtaining GMDH models. This function will give the necessary parameters used in ANNt_PBK function for variable prediction.

x_train: same as TrainInputData

y_train: same as TrainOutputData

x_check: same as TestInputData

y_check: same as TestOutputData

Layer: number of layers created during training of the GMDH model.

LayerOrder: for each layer, the nodes are sorted in an increasing order. These orders are necessary to evaluate the model when using it for prediction.

Coef: Three-dimensional matrix of coefficients. These coefficients come from the best fitting models obtained during the GMDH training phase.

3) `Comb_Vector = fun3(x_train(:,i),x_train(:,j));`

Description: Returns the combined version of the pair of independent variables given as input of the function. The combination depends on the model currently being used.

`x_train(:,i)`: the i-th independent variable.

`x_train(:,j)`: the j-th independent variable.

`Comb_Vector`: One-dimensional matrix where each element is a combination of the two input variables. The combination depends on the model currently being used by the GMDH (defined by `JobCounter`).

4) `ModelSelection = MyBinaryGenerator(JobCounter,MaxPower)`

Description: Selects the basic functions to be used by the GMDH model. For each `JobCounter`, the model selected uses different basic functions.

`JobCounter`: variable from 1 to 2^{MaxPower} .

`MaxPower`: number of basic functions.

`ModelSelection`: binary vector that gives the basic functions selected for the current model used by the GMDH.

Figures A.1, A.2, and A.3 show the results of the best GMDH model for predicting the flow rate in the tank inlet piping as a function of Bypass and Inlet MOV

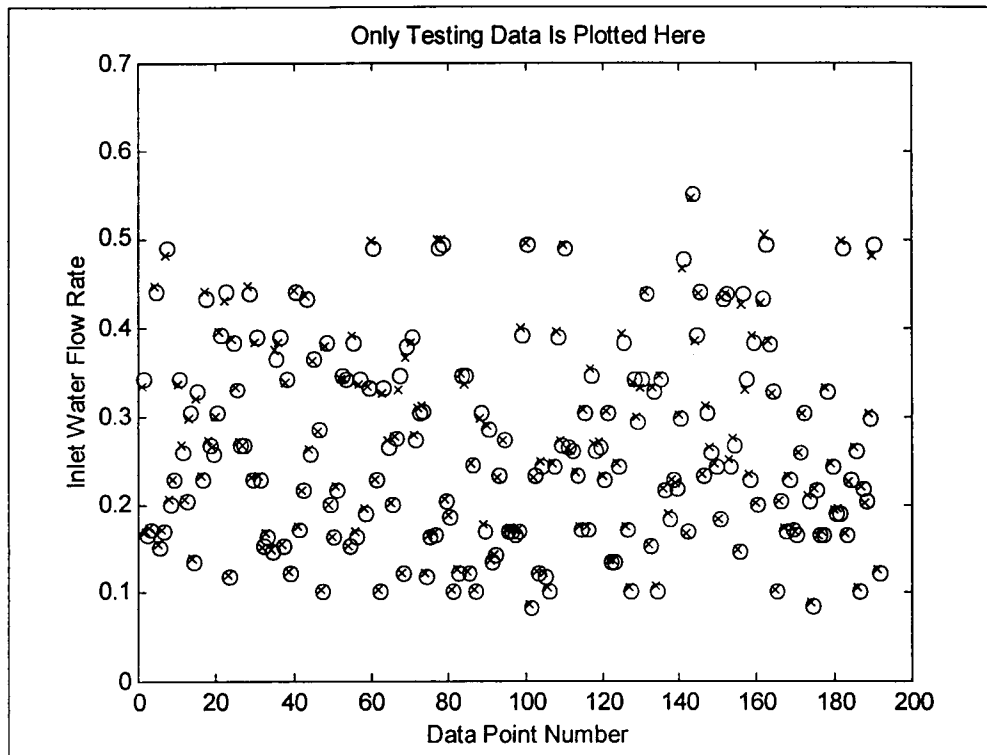


Figure A.1. The GMDH predicted values ("o") against the training output values ("x") for the Inlet Flow rate as a function of Bypass MOV and Inlet MOV positions.

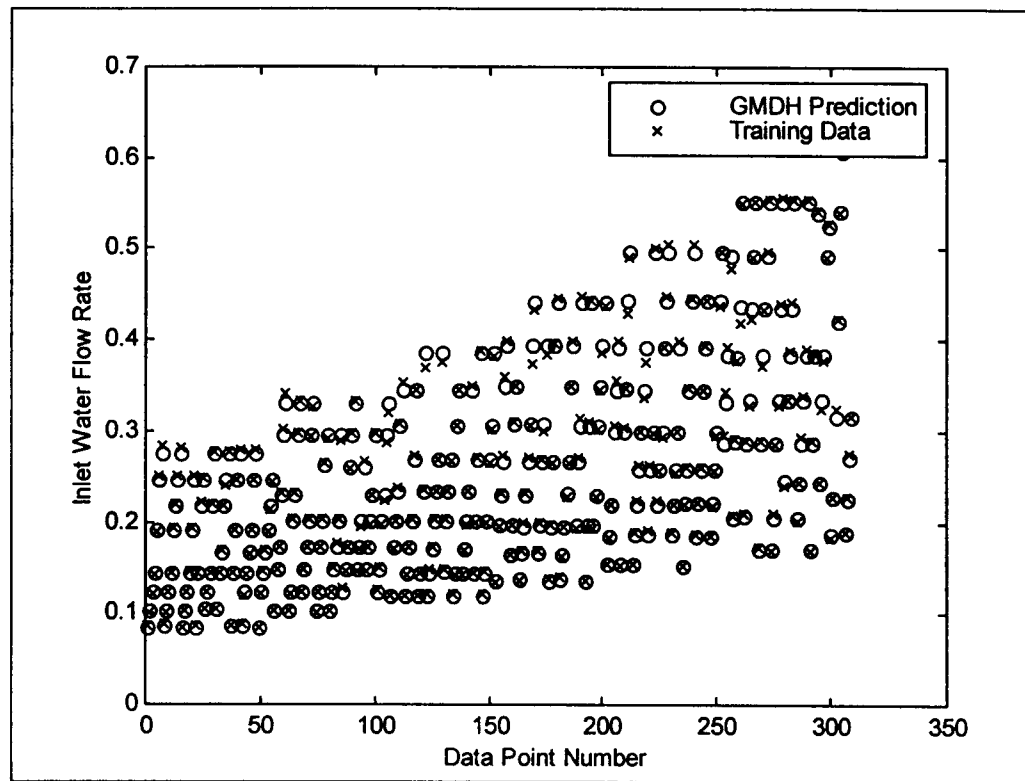


Figure A.2. The GMDH predicted values ("o") against the testing output values ("x") for the Inlet Flow rate as a function of Bypass MOV and Inlet MOV positions.

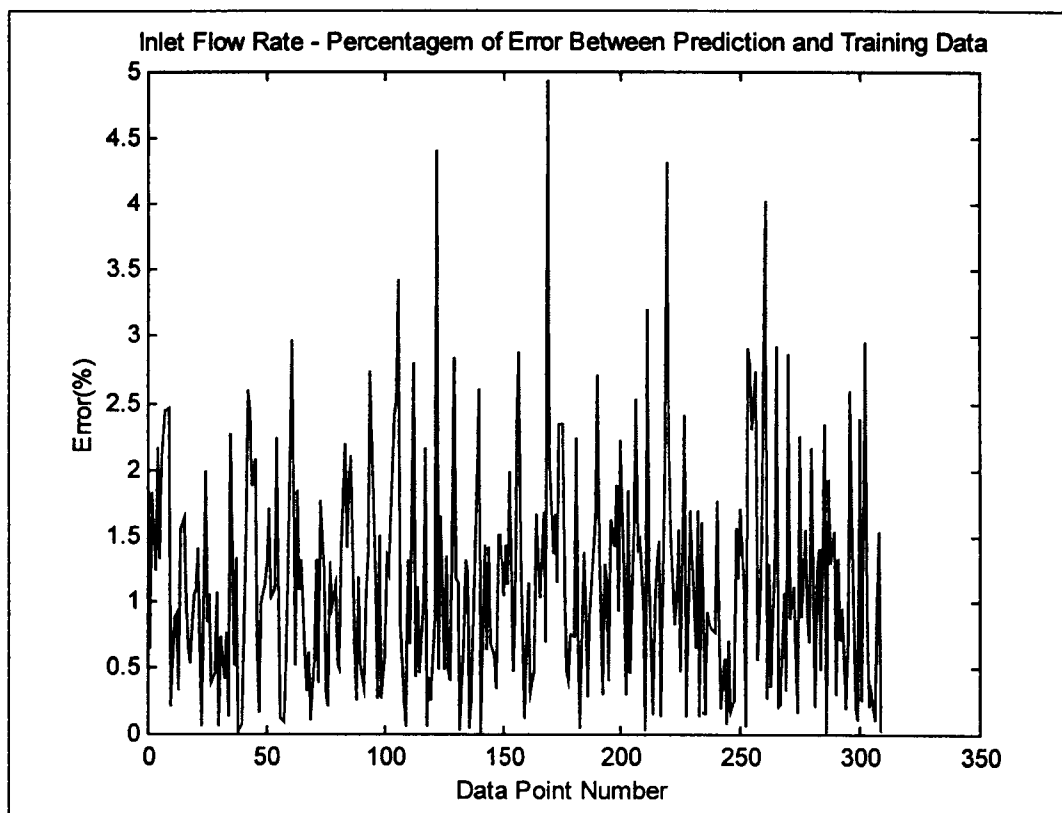


Figure A.3. The error (%) between the predicted values and the measurements for the Inlet Flow rate as a function of Bypass MOV and Inlet MOV positions.

positions. As we can see from these figures, the GMDH maps the input-output variables satisfactorily in both the training data and the testing data.

A.3. An Example of Obtaining a GMDH Model from a Data Set

% Outlet flow is a function of the following variables: WaterLevel and Outlet MOV Position

% Measured Water Level: data(:,8)

% Measured Outlet MOV Position: data(:,6)

% Measured Outlet Flow Rate: data(:,15)

% where data is experimental data from the LPWL system

% clear all previous variables stores in the Matlab environment

clear

% close any plot that was previously opened

close('all')

% define 2 global variables. JobCounter is used for defining the new set of GMDH basic functions to be used by the algorithm. MaxPower specifies the maximum number of terms used in the set of Basic functions.

global JobCounter MaxPower;

% get the current time. It is used just for information of how long did the whole process took to be completed

t0 = clock;

% load matrix mxn where m is the number of observations and n the number of variables. It is strongly recommended to normalize the data before.

load simulatedfaultfreedatabase;

data = simulatedfaultfreedatabase;

```

[TrainInputData, TrainOutputData, TestInputData, TestOutputData] =
TrainTest([data(:,8) data(:,6)],data(:,15),0);
clear data;

% creating the whole data base
x_alldata = [TrainInputData ; TestInputData];
y_alldata = [TrainOutputData ; TestOutputData];

MaxPower = 11;
residual = zeros(1,2^MaxPower-1);
OnesMatrix = ones(1,length(y_alldata));
den = (OnesMatrix*((y_alldata).^2));

for JobCounter=1:2^MaxPower-1
    [Layer,LayerOrder,Coef] =
GMDH_PBK(TrainInputData,TrainOutputData,TestInputData,TestOutputData);
    %save outletflow

    y_pred = ANNt_PBK(x_alldata,Layer,LayerOrder,Coef);
    res = sqrt(OnesMatrix*((y_alldata - y_pred).^2)/den);
    residual(JobCounter) = res;
    fprintf('JobCounter: %d Residual: %f\n',JobCounter,residual(JobCounter));

    if(0)
        figure
        plot(TestOutputData,'ko')
        hold
        y_pred_test = ANNt_PBK(TestInputData,Layer,LayerOrder,Coef);

```

```

        plot(y_pred_test,'kx')
    hold
    xlabel('Data Point Number')
    ylabel ('Outlet Water Flow Rate')
    title ('Only Testing Data Is Plotted Here')
        drawnow;

% extra plots
y_pred = ANNT_PBK(TrainInputData,Layer,LayerOrder,Coef);
figure;
plot(TrainOutputData,'ko')
    hold
    plot(y_pred,'kx')
    hold
    xlabel('Data Point Number')
    ylabel ('Outlet Water Flow Rate')
    legend('GMDH Prediction','Training Data')
    drawnow;

    figure
    plot(abs(TrainOutputData-y_pred)./TrainOutputData*100)
    xlabel('Data Point Number')
    ylabel ('Error(%)')
    title('Outlet Flow Rate - Percentagem of Error Between Prediction and Training
Data')
    end
end
etime(clock,t0)

```

```

function y = ANNt_PBK(x,Layer,LayerOrder,Coef)
[n m] = size(x);           % n = Number of Observations and m = Number of
Independent Variables
LayerCounter = 1;

while LayerCounter<Layer,

    % checking the performance of each GMDH node
    z = [];
    r = 1;
    for i=1:m-1;
        for j=(i+1):m
            Aux_Vector = fun3(x(:,i),x(:,j));
            z = [z Aux_Vector*Coef(:,r,LayerCounter)];
            r = r+1;
        end;
    end;

    % sorting columns of matrix Z
    x = z(:,(LayerOrder(:,LayerCounter)))';

    LayerCounter = LayerCounter + 1;
end;

% GMDH output
y=x(:,1);

```

```

function [CombinatoryVector] = fun3(x1,x2)

global MaxPower JobCounter;

[n m] = size(x1);

ModelSelection = MyBinaryGenerator(JobCounter,MaxPower);
CombinatoryVector = [ones(n,1)];

if (ModelSelection(11) == 1)
    CombinatoryVector = [CombinatoryVector x1 x2];
end
if (ModelSelection(10) == 1)
    CombinatoryVector = [CombinatoryVector x1.^2 x2.^2];
end
if (ModelSelection(9) == 1)
    CombinatoryVector = [CombinatoryVector x1.*x2];
end
if (ModelSelection(8) == 1)
    CombinatoryVector = [CombinatoryVector 1./x1 1./x2];
end
if (ModelSelection(7) == 1)
    CombinatoryVector = [CombinatoryVector 1./(x1.^2) 1./(x2.^2)];
end
if (ModelSelection(6) == 1)
    CombinatoryVector = [CombinatoryVector 1./(x1 + x2)];
end
if (ModelSelection(5) == 1)

```

```

    CombinatoryVector = [CombinatoryVector 1./(x1 .* x2)];
end
if (ModelSelection(4) == 1)
    CombinatoryVector = [CombinatoryVector x1./x2 x2./x1];
end
if (ModelSelection(3) == 1)
    CombinatoryVector = [CombinatoryVector x1./(x1 + x2) x2./(x1 + x2)];
end
if (ModelSelection(2) == 1)
    CombinatoryVector = [CombinatoryVector (x1 + x2)./x1 (x1 + x2)./x2];
end
if (ModelSelection(1) == 1)
    CombinatoryVector = [CombinatoryVector sin(x1*180/pi) sin(x2*180/pi)
cos(x1*180/pi) cos(x2*180/pi) ];
end

```

function[Layer,LayerOrder,Coef] = GMDH_PBK(x_train,y_train,x_check,y_check)

```

y_alldata = [y_train ; y_check];           % m is the number of independent variables)
RMIN = 1e99;                               % To be used for stopping the iteration
process
StopProcess = 0;                           % Flag: FALSE = 0, TRUE = 1
Layer = 1;                                 % Layer Counter - it starts at 1 (not
zero)
Alpha = 0.001;

while (~StopProcess)

```

```

    % creating the whole data base
    x_alldata = [x_train ; x_check];           % order nxm (where n is the total
number of observations and

    % initializing some variables
    [nt,m] = size(x_train);                    % nt = Number of Observations used for
Training, m = Number of independent variables
    n = size(x_alldata,1);                    % n = Number of Available Observations
    OnesMatrix = ones(1,(n-nt)); % matrix of unitary elements (order: 1x(n-nt) =
1xsize(y__check)

    % step #1: constructing new variables z1,z2,...,z(m 2)

    Coef_Matrix = [];
    z = [];
    y_aux = [];

    for i = 1:m-1
        for j = i+1:m
            % computing the best fit for the combination (xi,xj) (Aux_Vector below)
            Aux_Vector = fun3(x_train(:,i),x_train(:,j));
            %    Teta = Aux_Vector\y_train;
            Teta = inv(Aux_Vector'*Aux_Vector +
eye(size(Aux_Vector,2),size(Aux_Vector,2))*Alpha)*Aux_Vector'*y_train;
            Coef_Matrix = [Coef_Matrix Teta];

            % computing the z rows using all available data points

```

```

    Aux_Vector = fun3(x_alldata(:,i),x_alldata(:,j));
    z = [z Aux_Vector*Teta];

    % creating an auxiliar y_check to make step #2 computationally easier
    y_aux = [y_aux y_check];
end
end

% step #2: Screening out the least effective variables - Just a sort in this case

% computing r2
r2 = OnesMatrix*((y_aux - z(nt+1:n,:)).^2)/(OnesMatrix*(y_check.^2));

% sorting r2 and storaging the sorted indexes into LayerOrder matrix
% after command, r2 will have the values of r2 in increasing order ((m 2)x1)
% and Layer Order will have the initial columns of r2 values
% [r2,LayerOrder(:,Layer)] = sort(r2');
[Values,Indexes] = sort(r2');
r2 = Values(1:m,1);
    LayerOrder(:,Layer) = Indexes(1:m,1);

% replacing training and checking values by the columns of Z matrix
x_train = z(1:nt,LayerOrder(:,Layer));
x_check = z(nt+1:n,LayerOrder(:,Layer));

% salving Coefficient Matrix to be used by ??? to predict new values of dependent var
Coef(:, :, Layer) = Coef_Matrix(:, :);

```

```

% plotting y against the best fitted model GMDH got
if (0)
    subplot(2,1,1), plot(y_train, 'rx')
    hold
    subplot(2,1,1), plot(x_train(:,1), 'bo');
    hold

    subplot(2,1,2), plot(y_check, 'rx')
    hold
    subplot(2,1,2), plot(x_check(:,1), 'bo');
    hold

    drawnow
end

Layer = Layer+1;
% Step #3: Test for Optimality
if r2(1) <= RMIN & m > 2
    RMIN = r2(1);
    % Layer = Layer+1;
else
    StopProcess = 1;
end

    if Layer > 10
        fprintf('Max Number of Layers reached. Process stopped\n');
        StopProcess = 1;
    end
end
end

```

```
function [BinaryVector] = MyBinaryGenerator(Number, Power)
```

```
BinaryVector = zeros(1,Power);
```

```
% checking for positive and integer number
```

```
if ((Number <= 0) | (fix(Number)-Number)~= 0)
```

```
    fprintf('Number is negative,zero or not integer - Function MyBinaryGenerator can not  
be evaluated\n')
```

```
    return
```

```
end
```

```
% finding the highest power of two existent in Number
```

```
for i = Power:-1:0
```

```
    if Number == 2^i
```

```
        BinaryVector(Power-i) = 1;
```

```
        return
```

```
    elseif Number > 2^i
```

```
        BinaryVector(Power-i) = 1;
```

```
        break
```

```
    end
```

```
end
```

```
AuxNumber = 2^i;
```

```
% Finds the other bits below the highest power of two
```

```
for j=i-1:-1:0
```

```
    AuxNumber = AuxNumber + 2^j;
```

```
    if (AuxNumber == Number)
```

```

        BinaryVector(Power-j) = 1;
    return
elseif (AuxNumber < Number)
    BinaryVector(Power-j) = 1;
else
    AuxNumber = AuxNumber - 2^j;
end
end

% function TrainTest
% For given input and output matrices, this function randomly gets NPoints
% and put it in other matrices to be used as test matrices in a training
% procedure such as Neural Network

function [TrainInputData, TrainOutputData, TestInputData, TestOutputData] =
TrainTest(InputData,OutputData,NPoints)

if NPoints > 0
    Task = 1;
else
    Task = 2;
end

if Task == 1
    Matches = [];
    b = [];

    a = round(rand(NPoints,1)*size(InputData,1));

```

```

    for i = 1:length(a)
        for j = i+1:length(a)
            if a(i)==a(j)
                fprintf('they are the same: i:%d , j:%d\n',i,j)
                Matches = [Matches; [i j]];
            end
        end
    end

    for k = 1:length(a)
        if (k ~= Matches(:,2))
            b = [b; a(k)];
        end
    end

    TestInputData = InputData(b,:);
    TestOutputData = OutputData(b,:);

    aux = zeros(size(InputData));
    aux(b,1) = InputData(b,1);
    InputData = InputData-aux;
    b2 = find(InputData(:,1) ~= 0);

    TrainInputData = InputData(b2,:);
    TrainOutputData = OutputData(b2,:);

    save b b b2;
end

```

```

if Task == 2
    load b;
    TestInputData = InputData(b,:);
    TestOutputData = OutputData(b,:);
    TrainInputData = InputData(b2,:);
    TrainOutputData = OutputData(b2,:);
end

% if number of independent variables is equal to 2 then it creates
% an extra indep. var. such that the GMDH will be able to try deeper
% layers.
if (size(TestInputData,2) == 2)
    TestInputData = [TestInputData TestInputData(:,1)./TestInputData(:,2)];
    TrainInputData = [TrainInputData TrainInputData(:,1)./TrainInputData(:,2)];
end

```

A.4. SOME SIMULATED CASES STUDIED

The model should distinguish between two cases: changes in the system due to changes in one of its set points (Bypass and Outlet MOV positions, and water level) and changes in the system due to a faulty device. In the first case, the loop variable predictions should be very close to the actual values, while in the second case some discrepancies should be observed.

When a faulty component is present in the system, the algorithm must detect it and isolate it. The detection is made by comparing the predicted values and measured

values. If a discrepancy is larger than a predefined threshold, then a fault has been detected. As shown later, the fault can be isolated by using the results of the predicted values and the measured values in a rule-based expert system.

Some simulated cases studied using the GMDH models to detect the changes in the system set point, to detect faulty components and to isolate them using a rule-based expert system, are described.

A.4.1. Changes in system set points

A – Change in the water level set point

Predicted Inlet Flowrate: 0.009427

Model Inlet Flowrate: 0.009228

Percentage of Variation: -2.149558 (%)

Predicted Outlet Flowrate: 0.074271

Model Outlet Flowrate: 0.073825

Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.490487

Model PID Output: 0.497338

Percentage of Variation: 1.377414 (%)

Predicted Water Level: 0.042790

Model Water Level: 0.042857

Percentage of Variation: 0.156170 (%)

Water Level Setpoint: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

B – Changes in the Bypass MOV position

Predicted Inlet Flowrate: 0.009432
Model Inlet Flowrate: 0.009228
Percentage of Variation: -2.205635 (%)

Predicted Outlet Flowrate: 0.074271
Model Outlet Flowrate: 0.073825
Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.478500
Model PID Output: 0.485636
Percentage of Variation: 1.469425 (%)

Predicted Water Level: 0.042790
Model Water Level: 0.042857
Percentage of Variation: 0.156170 (%)

Water Level Setpoint: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.830000
Measured Position: 0.830000
Percentage of Variation: 0.000000 (%)

C – Changes in the Outlet MOV position

Predicted Inlet Flowrate: 0.008754
Model Inlet Flowrate: 0.008588
Percentage of Variation: -1.930489 (%)

Predicted Outlet Flowrate: 0.069062
Model Outlet Flowrate: 0.068706
Percentage of Variation: -0.518361 (%)

Predicted PID Output: 0.437957
Model PID Output: 0.444804
Percentage of Variation: 1.539406 (%)

Predicted Water Level: 0.042815
Model Water Level: 0.042857
Percentage of Variation: 0.099292 (%)

Water Level Setpoint: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: -0.000000 (%)

Outlet MOV Setpoint Position: 0.820000
Measured Position: 0.820000
Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

D – Changes in all the above three set points simultaneously

Predicted Inlet Flowrate: 0.009355
Model Inlet Flowrate: 0.009161
Percentage of Variation: -2.120764 (%)

Predicted Outlet Flowrate: 0.073629
Model Outlet Flowrate: 0.073286
Percentage of Variation: -0.467559 (%)

Predicted PID Output: 0.466817
Model PID Output: 0.474257
Percentage of Variation: 1.568847 (%)

Predicted Water Level: 0.045649
Model Water Level: 0.045714
Percentage of Variation: 0.142301 (%)

Water Level Setpoint: 0.045714
Measured Water Level: 0.045714
Percentage of Variation: 0.000000 (%)

Outlet MOV Setpoint Position: 0.820000
Measured Position: 0.820000
Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.820000
Measured Position: 0.820000
Percentage of Variation: 0.000000 (%)

A.4.2. Some Fault Cases

A- Inlet MOV Fault

Predicted Inlet Flowrate: 0.008791
Model Inlet Flowrate: 0.009228
Percentage of Variation: 4.742529 (%)

Predicted Outlet Flowrate: 0.074271
Model Outlet Flowrate: 0.073825
Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.490487
Model PID Output: 0.447438
Percentage of Variation: -9.621309 (%)

Predicted Water Level: 0.042790
Model Water Level: 0.042857
Percentage of Variation: 0.156170 (%)

Water Level Setpoint: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

B – Water Level Sensor Fault

Predicted Inlet Flowrate: 0.008456
Model Inlet Flowrate: 0.008306
Percentage of Variation: -1.805978 (%)

Predicted Outlet Flowrate: 0.074271
Model Outlet Flowrate: 0.066445
Percentage of Variation: -11.777340 (%)

Predicted PID Output: 0.416372
Model PID Output: 0.422887
Percentage of Variation: 1.540606 (%)

Predicted Water Level: 0.038575
Model Water Level: 0.042857
Percentage of Variation: 9.990798 (%)

Water Level Setpoint: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

C – Rosemount Mass Flowmeter Fault

Predicted Inlet Flowrate: 0.009427
Model Inlet Flowrate: 0.010144
Percentage of Variation: 7.068562 (%)

Predicted Outlet Flowrate: 0.074271
Model Outlet Flowrate: 0.073825
Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.490487
Model PID Output: 0.497338
Percentage of Variation: 1.377414 (%)

Predicted Water Level: 0.047030
Model Water Level: 0.042857
Percentage of Variation: -9.737045 (%)

Water Level Setpoint: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

D – Outlet MOV position

Predicted Inlet Flowrate: 0.007561
Model Inlet Flowrate: 0.007438
Percentage of Variation: -1.654478 (%)

Predicted Outlet Flowrate: 0.074271
Model Outlet Flowrate: 0.059505
Percentage of Variation: -24.814871 (%)

Predicted PID Output: 0.355361
Model PID Output: 0.360006
Percentage of Variation: 1.290273 (%)

Predicted Water Level: 0.034594
Model Water Level: 0.042857
Percentage of Variation: 19.280208 (%)

Water Level Setpoint: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Setpoint Position: 0.850000
Measured Position: 0.763120
Percentage of Variation: 10.221227 (%)

Bypass MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

E – Bypass MOV position

Predicted Inlet Flowrate: 0.009968
Model Inlet Flowrate: 0.009228
Percentage of Variation: -8.013686 (%)

Predicted Outlet Flowrate: 0.074271
Model Outlet Flowrate: 0.073825
Percentage of Variation: -0.603632 (%)

Predicted PID Output: 0.490487
Model PID Output: 0.548853
Percentage of Variation: 10.634087 (%)

Predicted Water Level: 0.042790
Model Water Level: 0.042857
Percentage of Variation: 0.156170 (%)

Water Level Setpoint: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.850000
Measured Position: 0.938044
Percentage of Variation: -10.358170 (%)

F – Outlet Flowmeter

Predicted Inlet Flowrate: 0.009427
Model Inlet Flowrate: 0.009228
Percentage of Variation: -2.149558 (%)

Predicted Outlet Flowrate: 0.074271
Model Outlet Flowrate: 0.066244
Percentage of Variation: -12.117199 (%)

Predicted PID Output: 0.414494
Model PID Output: 0.497338
Percentage of Variation: 16.657494 (%)

Predicted Water Level: 0.042790
Model Water Level: 0.042857
Percentage of Variation: 0.156170 (%)

Water Level Setpoint: 0.042857
Measured Water Level: 0.042857
Percentage of Variation: 0.000000 (%)

Outlet MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.850000
Measured Position: 0.850000
Percentage of Variation: 0.000000 (%)

G – Water Level Controller

Predicted Inlet Flowrate: 0.005949
Model Inlet Flowrate: 0.005957
Percentage of Variation: 0.135395 (%)

Predicted Outlet Flowrate: 0.048103
Model Outlet Flowrate: 0.047654
Percentage of Variation: -0.941647 (%)

Predicted PID Output: 0.264213
Model PID Output: 0.265879
Percentage of Variation: 0.626770 (%)

Predicted Water Level: 0.027751
Model Water Level: 0.027664
Percentage of Variation: -0.314502 (%)

Water Level Setpoint: 0.042857

Measured Water Level: 0.027664

Percentage of Variation: 35.450573 (%)

Outlet MOV Setpoint Position: 0.850000

Measured Position: 0.850000

Percentage of Variation: 0.000000 (%)

Bypass MOV Setpoint Position: 0.850000

Measured Position: 0.850000

Percentage of Variation: 0.000000 (%)

A.5. Rule-based Expert System

From the results obtained in the fault cases, a simple rule-based expert system was developed to detect and isolate faults in any loop component. Changes in the system set points are not considered as faults, even though the loop variable outputs may change from one to another steady-state value. The source code for fault detection and isolation is given below.

The fault detection and isolation algorithm is implemented as a Matlab function. The function input is the vector of error percentages of monitored loop variables. If the pattern of error vector is not recognized, the message "Fault Unknown" is returned. Otherwise the fault type is returned.

The source code of a sample program, which is used to generate a 10% fault in different loop components, is given in this section. This may be used to simulate a fault in a specific device.

Rule-Based Fault Isolation Code

```
function FaultType = FaultDetection(ErrorVector)
% number of monitored variables
NVars = 7;
% Number of monitored faults
NFaults = 8;
FaultVector = zeros(1,NVars);
Threshold = 3;
% list of possible faults
FaultTable = [ 1 0 1 0 0 0 0;
               0 1 0 1 0 0 0;
               1 0 0 1 0 0 0;
               0 1 0 1 0 1 0;
               1 0 1 0 0 0 1;
               0 1 1 0 0 0 0;
               0 0 0 0 1 0 0;
               0 0 0 0 0 0 0];

% Description of each fault
Faults = ['Inlet MOV Fault' 'Water Level Sensor Fault'
          'RMT Flow meter Fault '
          'Outlet MOV position '
          'Bypass MOV Position ']
```

```

        'Outlet Turbine Flowmeter'
        'PID controller Fault '
        'No fault detected ' ];

```

```

index = find(abs(ErrorVector)>Threshold);
counter = length(index);

```

```

for i = 1:counter
    FaultVector(index(i)) = 1;
end

```

```

for i=1:8
    if FaultTable(i,:) == FaultVector;
        FaultType = Faults(i,:);
        return
    end
end

```

```

FaultType = 'Fault Unkown';

```

```

% Cleaning all variables and closing any open plot
% Cleaning all variables and closing any open plot
clear
close ('all')

```

```

% time used to compute simulation time

```

```
t0 = clock;
```

```
% Used for putting ylabel in the automatic-generated plots
```

```
Variable = {'Bypass MOV position Setpoint','Inlet MOV position Setpoint','Outlet MOV  
position Setpoint','Measured Bypass MOV position','Measured Inlet MOV  
position','Measured Outlet MOV position','Water Level Setpoint','Measured Water  
Level','PID output','Dummy Variable','Dummy Variable','Dummy Variable','Measured  
Inlet Flowrate','Measured RMT Inlet Flowrate','Measured Turbine Inlet  
Flowrate','Measured Turbine Outlet Flowrate'};
```

```
% Creating the drift used for any requested device. It has maximum amplitude of 0.1  
(10%)
```

```
% The drift is saved in a lookup table (used by the model in the simulink)
```

```
x = (1000:7200)';
```

```
SensorDrift = (0.42*atan((x-4100)/400) + 0.05*rand(size(x))-0.025);
```

```
SensorDrift = (SensorDrift - SensorDrift(1))/12; % making changes in feet instead of  
inches
```

```
xprevious = 0;
```

```
xafter = 1e10;
```

```
yprevious = 0;
```

```
yafter = SensorDrift(length(SensorDrift));
```

```
x = ([xprevious x' xafter])';
```

```
SensorDrift = ([yprevious SensorDrift' yafter])';
```

LookupTable = [x SensorDrift];

% some initial values for some important model parameters

WaterLevelSetpoint = 1.7; % normally at 1.7 (in feet)

NoiseVar = 0.00001;

PumpSpeedInitialValue = 1;

MaxPumpOutput = 1; % lbm/s

OMOVToOPDiam = 0.005; %Conversion from MOV(%) position to Outlet Piping
Diameter (in)

TankDiameter = 5.73; %Tank inside diameter in inches

h0 = 12; %initial water level in the tank for linearized system (in inches)

g = 32.1522;

rho = 62.3000;

PIDp = 5.0; %1.0;

PIDi = 0.0; %0.5;

BypassMOVpos = 80;

OutletMOVpos = 80;

% Getting system variables steady-state values for defining the amount of drift in it
component

% Auxiliar data array. It is necessary for obtaining ss values and later it will be used
% for drift purposes

% Definition of what device is going to experiment drift. DO NOT CHANGE ANY OF
THE FOLLOWING INITIALIZATION

InletMOVPert = 0; InitialInletMOVValue = 0;

PumpSpeedPert = 0; InitialPumpSpeedValue = 0;

WaterLevelSensorPert=0; InitialWaterLevelValue = 0;

```

RMTInletFlowRatePert = 0; InitialRMTFlowValue = 0;
OutletMOVPosPert = 0;      InitialOutletMOVValue = 0;
BypassMOVPert = 0;        InitialBypassValue = 0;
OutletTFMPert = 0;      InitialOutletTFMValue = 0;

% Running a fault-free case for ss values
[t,x] = sim('loop04',[0 10000]);

% Defining a baseline data array for plot comparisons only.
InitialValue = simout(size(simout,1),:);

% Now, let's make a case where there is or there is not a faulty device
% Definition of what device is going to experiment drift
% Changes in a variable value from 0 to 1 makes the device to drift by 10%
% Currently the rule-based expert system identifies faults in the following loop
components:
% inlet MOV position, water level sensor, RMT inlet flowmeter, PID controller
% the PID controller fault studied here is the case where it can match the water level
setpoint;
% This is done by changing the PI integral constant to zero and the proportional to 6.0
(change
% previous initialization. Do not change PID variables below.
InletMOVPert = 0;      InitialInletMOVValue = InitialValue(2);
PumpSpeedPert = 0;      InitialPumpSpeedValue = PumpSpeedInitialValue;
WaterLevelSensorPert = 0; InitialWaterLevelValue = InitialValue(8);
RMTInletFlowRatePert = 0; InitialRMTFlowValue = InitialValue(13);
OutletMOVPosPert = 0;      InitialOutletMOVValue = InitialValue(3);
BypassMOVPert = 0;      InitialBypassValue = InitialValue(1);

```

```

OutletTFMPert = 0;                                InitialOutletTFMValue = InitialValue(15);

% Running this case
[t,x] = sim('loop04',[0 10000]);

VectorSize = size(simout,1)
standard = std( simout(VectorSize-100:VectorSize , :) );
MeanValue = mean(simout (VectorSize-100:VectorSize,:));

FaultVector = [];
% if you want to plot all loop variables before and after fault, set WantPlot to 1 (TRUE)
WantPlot = 0;
if (WantPlot)
    for i=1:15
        figure
        plot(t, simout(:,i), 'b');
        hold
        plot(t,ones(size(t))*InitialValue(i),'r')
        hold
        xlabel('Time (s)');
        ylabel (Variable(i));
        title ('System Behavior when there is a Faulty Device')
        if standard(i) > 0.00001
            % axis([0 t(length(t)) MeanValue(i)-20*standard(i)
MeanValue(i)+20*standard(i)]);
        end

%    if InitialValue(i) ~= 0

```

```

%           gtext(sprintf('Percentual Difference between Expected and Actual
Value = %.3f, (simout(size(simout,1),i) - InitialValue(i))/InitialValue(i)*100) );
%   else
%           gtext(sprintf('Absolute Difference between Expected and Actual
Value = %.3f, (simout(size(simout,1),i) - InitialValue(i)) ));
%   end
    drawnow;

    if InitialValue(i) ~= 0
        if ((simout(size(simout,1),i) - InitialValue(i))/InitialValue(i)) > 0.05
            aux = 1;
        elseif ((simout(size(simout,1),i) - InitialValue(i))/InitialValue(i)) < -0.05
            aux = -1;
        else
            aux = 0;
        end
    else
        if ((simout(size(simout,1),i) - InitialValue(i))) > 0.05
            aux = 1;
        elseif ((simout(size(simout,1),i) - InitialValue(i))) < -0.05
            aux = -1;
        else
            aux = 0;
        end
    end
    FaultVector = [FaultVector aux];
end
end

```

```

ProcessTime = etime(clock,t0);
fprintf ('Process Time: %f\n', ProcessTime);

%pFile = fopen('InletMOVPositionFuzzy.dat','w');
%for i = 60:size(simout,1)
%    LastPoint = i;
    LastPoint = size(simout,1);
    [LastData] = NormalizeData(simout(LastPoint,:));

    % remember, when there are only two independent variables it is necessary to
    create a
    % third one. This allows the GMDH goes deeper in layers to find the best model.
    In
    % our case I have been using the term var(1)/var(2) as the third variable.

    load inletflow
    InletFlow_Pred = ANNt_PBK([LastData(1) LastData(2)
LastData(1)/LastData(2)],Layer,LayerOrder,Coef);
    InletFlowDiff = (LastData(13)-InletFlow_Pred)/LastData(13)*100;
    fprintf('Predicted Inlet Flowrate: %f\nModel Inlet Flowrate: %f\nPercentage of
Variation: %f (%%)\n\n', InletFlow_Pred, LastData(13), InletFlowDiff);
    % pause

    load outletflow

```

```

OutletFlow_Pred = ANNt_PBK([LastData(8) LastData(3) LastData(8) /
LastData(3)], Layer, LayerOrder, Coef);
OutletFlowDiff = (LastData(15)-OutletFlow_Pred)/LastData(15)*100;
fprintf('Predicted Outlet Flowrate: %f\nModel Outlet Flowrate: %f\nPercentage of
Variation: %f (%%)\n\n', OutletFlow_Pred, LastData(15), OutletFlowDiff);
% pause

```

```

load PIDoutput
PID_Pred = ANNt_PBK([LastData(1) LastData(15) LastData(7)],Layer,
LayerOrder, Coef);
PIDDiff = (LastData(9)-PID_Pred)/LastData(9)*100;
fprintf('Predicted PID Output: %f\nModel PID Output: %f\nPercentage of
Variation: %f (%%)\n\n', PID_Pred, LastData(9), PIDDiff);
% pause

```

```

load WaterLevel
WaterLevel_Pred = ANNt_PBK([LastData(13) LastData(3)
LastData(13)/LastData(3)],Layer, LayerOrder, Coef);
WaterLevelDiff = (LastData(8)-WaterLevel_Pred)/LastData(8)*100;
fprintf('Predicted Water Level: %f\nModel Water Level: %f\nPercentage of
Variation: %f (%%)\n\n', WaterLevel_Pred, LastData(8), WaterLevelDiff);

```

```

WLSetpointError = (LastData(7) - LastData(8))/LastData(7)*100;
fprintf('Water Level Setpoint: %f\nMeasured Water Level: %f\nPercentage of
Variation: %f (%%)\n\n', LastData(7), LastData(8), WLSetpointError);

```

```

OutletMOVposError = (LastData(3) - LastData(6))/LastData(3)*100;

```

```

    fprintf('Outlet MOV Setpoint Position: %f\nMeasured Position: %f\nPercentage
of Variation: %f (%%)\n\n', LastData(3), LastData(6), OutletMOVposError);
    BypassMOVposError = (LastData(1) - LastData(4))/LastData(1)*100;
    fprintf('Bypass MOV Setpoint Position: %f\nMeasured Position: %f\nPercentage
of Variation: %f (%%)\n\n', LastData(1), LastData(4), BypassMOVposError);
    FaultType = FaultDetection([InletFlowDiff OutletFlowDiff PIDDiff
WaterLevelDiff WLSetpointError OutletMOVposError BypassMOVposError])
%    fprintf(pFile, '%f %f %f\n', InletFlowDiff, PIDDiff, (LastData(5)-
LastData(2))/LastData(2)*100);
%end

```

VITA

Paulo Brasko Ferreira was born in São Paulo, Brazil on July 19, 1962. He attended schools in the public system of São Paulo, Oswaldo Aranha, where he graduated from the High School in December 1980. He entered the University of São Paulo in February 1981 and received the Bachelor of Arts in Physics in 1984. In 1985 he started working for the Nuclear Energy Research Institute (IPEN) in São Paulo, where he worked for nine years. During those years, he also entered the Master's program in Nuclear Engineering at the University of São Paulo, officially receiving the Master's degree in 1988. Also, during this time, he worked at the Nuclear Power Plant, Angra-I, Rio de Janeiro, Brazil. In 1993 he started a program in the Nuclear Engineering Department at The University of Tennessee, Knoxville, Tennessee, USA, as a visiting scholar. In the fall semester of the following year, he entered the Ph.D. program in the Nuclear Engineering Department. He received the Ph.D. degree in August 1999.

He is presently working on the development and maintenance of existing nuclear analysis software as a senior engineer at ABB Combustion Engineering, Windsor, Connecticut.