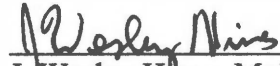
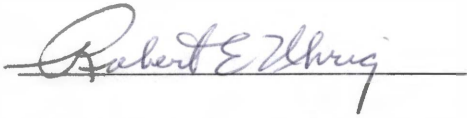
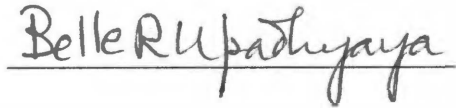


To the Graduate Council:

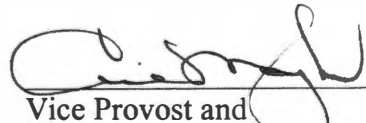
I am submitting herewith a thesis written by Brandon Peter Rasmussen entitled "Neural Network Partial Least Squares for Instrument Surveillance and Calibration Verification." I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.


J. Wesley Hines, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:


Vice Provost and
Dean of Graduate Studies

Neural Network Partial Least Squares for Instrument Surveillance and Calibration Verification

**A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville**

**Brandon Rasmussen
May 2002**

DEDICATION

Thesis
2002
.R37

This thesis is dedicated to my family and friends who have lent their support and guidance during my graduate studies. For those of you who have been there I dedicate this work to you, especially Shasten for her love and support.

ACKNOWLEDGEMENTS

I would like to thank the staff at the Electrical Power Research Institute's Instrumentation and Control Center in Kingston, TN, especially Cyrus Taft for his assistance in integrating the system at Tennessee Valley Authority's Kingston plant site.

I would also like to acknowledge the members of my thesis committee. Dr. J. Wesley Hines, who introduced me to artificial intelligence, and has guided me through this thesis, Dr. Robert E. Uhrig, who has taken the time to oversee, counsel, and review my work, and Dr. Upadhyaya who has lent his experience in artificial intelligence and signal validation to improve this work.

I would like to recognize Dennis Linkins, our project manager at Tennessee Valley Authority, and the Maintenance and Reliability Center at the University of Tennessee for their support.

I would also like to thank my fellow graduate students who have assisted in the creation of the graphical user interfaces presented in this thesis, Aleksey Urmanov and Rui Zhang.

Finally, I would like to thank the entire staff of the Nuclear Engineering department, especially Ellen Fisher and Kristin England for their assistance with travel

arrangements, preparation of project presentations, and anything else I may have requested.

ABSTRACT

Instrument surveillance and calibration verification (ISCV) systems are developed to monitor process sensors and produce correct values if the actual sensors are determined to be out of calibration, or have failed. The purpose of an ISCV system is to provide validated values to enable condition based calibrations of faulty sensors, to ensure the safety of a process by having validated process measurements, and to optimize process efficiencies by initiating control actions based on validated information. The use of ISCV systems is attractive to the nuclear power industry, where validated measurements will increase the confidence of operator's knowledge about the state of the process and reduce the probability of incorrect decisions based on unvalidated process measurements. Additional benefits would be realized by limiting unnecessary calibrations through the setting up of a condition based calibration schedule, based on the sensor fault reporting of the ISCV system, rather than the more common periodic calibration schedule.

An ISCV system has been developed and tested at Tennessee Valley Authority's coal-fired plant in Kingston, Tennessee. Process data, collected over the course of a year and a half at the Kingston plant site, were used to develop the system. The embedded empirical model of the ISCV system utilizes the Neural Network Partial Least Squares (NNPLS) algorithm, the development and implementation of which is the focus of this thesis. The algorithm and its associated ISCV system have been extensively evaluated with successful results.

NNPLS is an extension of the more common partial least squares (PLS). Nonlinear capabilities are available in the NNPLS algorithm through the use of small neural networks, whereas the standard PLS algorithm is strictly a linear technique. Sensor faults and drifts are detected using the sequential probability ratio test, and alarms are initiated by an alarm logic heuristic.

A set of graphical user interfaces have been created for the development of the NNPLS based ISCV models, and the subsequent monitoring of an appropriate set of signals either on-line or from historical data. These interfaces have been designed to transfer this technology to users without necessitating a theoretical understanding of the embedded modeling techniques.

Results are presented for monitoring 51 process sensors. For these signals, the mean absolute percent prediction errors for 80% of the signals tested were less than 2%. Some sensors were more difficult to model and exhibited higher prediction errors, most often this can be related to a lack of sufficient correlations with the problematic signal. Drift detection performance of the system ranged between 2-5%. This indicates that drifting or failed sensors will be identified after the signal measurement and the ISCV systems prediction deviate by more than 2-5% of the signal mean. In an online test, the system properly identified a failed sensor reporting measurements at a level greater than 5% of its mean signal value.

TABLE OF CONTENTS

1.0 INTRODUCTION	1
1.1 BACKGROUND	1
1.2 STATEMENT OF PROBLEM.....	3
1.3 CONTRIBUTIONS.....	6
1.4 ORGANIZATION OF DOCUMENT	7
2.0 LITERATURE REVIEW	8
3.0 THEORY.....	12
3.1 DATA COLLECTION	12
3.1.1 METHODS OF DATA COLLECTION	14
3.1.2 DESCRIPTION OF RECORDED DATA	16
3.1.3 EXAMPLES OF OPERATING CONDITION CHANGES	18
3.2 MEDIAN FILTERS.....	22
3.2.1 SMOOTHING OF IMPULSE NOISE	24
3.2.2 EDGE PRESERVATION.....	27
3.3 SEQUENTIAL TESTING	28
3.3.1 THE SEQUENTIAL PROBABILITY RATIO TEST	32
3.3.2 SPRT FOR NORMALLY DISTRIBUTED RANDOM VARIABLES.....	33
3.3.3 OPERATING CHARACTERISTIC FUNCTION FOR NORMALLY DISTRIBUTED RANDOM VARIABLES.....	34
3.3.4 AVERAGE SAMPLE NUMBER FUNCTION FOR NORMALLY DISTRIBUTED RANDOM VARIABLES.....	35
3.3.5 APPLICATION OF THE SPRT TO SIGNAL VALIDATION	37
3.3.6 ALARM LOGIC.....	39
3.4 MULTIPLE LINEAR REGRESSION.....	40
3.5 COLLINEARITY	42
3.6 NEURAL NETWORK PARTIAL LEAST SQUARES.....	46
3.6.1 FEATURES AND BENEFITS OF THE NNPLS ARCHITECTURE	46
3.6.2 NNPLS MODELING OVERVIEW	48
3.6.3 PLS ALGORITHM DESCRIPTION	50
3.6.4 NNPLS ALGORITHM DESCRIPTION	54
3.6.5 METHOD OF DETERMINATION OF THE MAXIMUM NUMBER OF LATENT VECTORS	58
3.6.6 SPRT PARAMETER AND ALARM LOGIC SETTING.....	58
4.0 NNPLS ISCV GRAPHICAL USER INTERFACES	61
4.1 NNPLS DEVELOPMENT INTERFACE.....	62
4.2 SQL CONNECTION INTERFACE.....	67
4.3 NNPLS MONITORING INTERFACE.....	69
5.0 RESULTS OF APPLICATIONS TO THE FOSSIL PLANT.....	71
5.1 PREDICTION PERFORMANCE	71
5.2 DRIFT DETECTION PERFORMANCE	73
6.0 CONCLUSIONS	77
6.1 CONCLUDING REMARKS.....	77

6.2 RECOMMENDATIONS FOR FUTURE WORK	79
LIST OF REFERENCES	81
APPENDICES	93
APPENDIX A	
NNPLS DEVELOPMENT INTERFACE SET-UP AND TUTORIAL	94
APPENDIX B	
SQL CONNECTION INTERFACE SET UP AND TUTORIAL	108
APPENDIX C	
NNPLS MONITORING INTERFACE SET UP AND TUTORIAL	118
APPENDIX D	
PREDICTION PLOTS.....	128
APPENDIX E	
TABLES OF RECORDED DATA	155
APPENDIX F	
MATLAB CODE.....	163
VITA.....	180

LIST OF TABLES

Table 1: Description and Dates of Recognized Incidents	17
Table 2: Representative quantities of the expected error values induced by impulse noise, and the reduction of these errors based on certain probabilities of occurrence of impulse noise, and various filter window sizes.....	26
Table 3: NNPLS Prediction Performance Results.....	72
Table 4: Data Storage Method Based on Front Panel Control Settings	103
Table 5: List of signals recorded via the MATLAB script program (4/2000 –1/2001)	155
Table 6: List of signals recorded via the SQL connection interface (6/2001 –11/2001)	159

LIST OF FIGURES

Figure 1: Conceptual diagram of the ISCV system interactions.....	5
Figure 2: Fossil power generation system [ASME PTC PM-193].	14
Figure 3: Condition changes for combustion air flow (%).	20
Figure 4: Condition changes for Reheat Attemp A before spray temp (DegF).....	20
Figure 5: Condition changes for station service load with respect to gross generation.....	21
Figure 6: Operating characteristic function for $x \sim N(\theta, \sigma)$, where $H_0: \theta = \theta_0 = 0$, $H_1: \theta = \theta_1 = 1$, $\alpha = 0.0001$, and $\beta = 0.1$	35
Figure 7: The average sample number (ASN) as a function of θ , for $x \sim N(\theta, \sigma)$, where $H_0: \theta = \theta_0 = 0$, $H_1: \theta = \theta_1 = 1$, $\alpha = 0.0001$, $\beta = 0.1$, and $\sigma^2 = 4$	36
Figure 8: Sample distributions for the null hypothesis, $H_0: \theta = \theta_0$, $x \sim N(\theta_0, \sigma^2)$, $\theta_0 = 0$, $\sigma^2 = 4$, and the alternative hypothesis $H_1: \theta = \theta_1$, $x \sim N(\theta_1, \sigma^2)$, $\theta_1 = 10$, $\sigma^2 = 4$..	38
Figure 9: Relationship between the determinant and condition number of a two redundant signal predictor variable matrix with increasing addition of noise.	45
Figure 10: NNPLS inferential unit schematic.	48
Figure 11: Geometric representation of the relationship between the input and output residual matrices in the PLS-1 algorithm.	52
Figure 12: Schematic of the SISO-ANN architecture embedded in the inner relationships of the NNPLS architecture.....	55
Figure 13: Simplified flow diagram of the development interface.	63
Figure 14: Development interface front panel.....	66
Figure 15: SQL interface front panel.....	68
Figure 16: Monitoring interface front panel.	70
Figure 17: Feedwater flow 1 simulated drift test, alarm initiated after 2.8% drift.	74
Figure 18: Feedwater flow 2 response to drift test in feedwater flow 1 signal.	75
Figure 19: Front panel controls to identify and describe the training data file.	95
Figure 20: Front panel indicators and controls for viewing the data and statistics.	96
Figure 21: Front panel controls for filtering the data.	98
Figure 22: Front panel control for model updating or building.....	100
Figure 23: Front panel controls and indicators for model development.	101
Figure 24: Front panel plot controls and display of inputs, predictions, and training errors.....	105
Figure 25: Flow diagram for the SQL connection interface.....	109
Figure 26: SQL interface tag name and data storage controls.....	110
Figure 27: Sample tag list files and formats.	111
Figure 28: SQL interface database parameter controls.	113
Figure 29: SQL interface connection status indicators and run control.	114

Figure 30: SQL interface signal tag name, measurement, and prediction indicators..... 115

Figure 31: SQL interface indicators of the applied SELECT and INSERT statements.... 116

Figure 32: SQL interface display of connection or execution errors. 117

Figure 33: Monitoring interface data file control. 120

Figure 34: Monitoring interface signal controls and indicators. 122

Figure 35: Monitoring interface graphical display..... 124

Figure 36: Monitoring interface tabulated display. 125

1.0 INTRODUCTION

1.1 Background

As condition-based maintenance strategies are becoming more prevalent, new technologies are being developed to ascertain the condition of plant equipment. The main purpose of an Instrument Surveillance and Calibration Verification (ISCV) system is to monitor the condition of process sensors and their associated instrument chains. Current practices often rely on periodic manual calibration schedules, which may result in failed or out of calibration sensors operating for periods of time up to the calibration frequency. In addition, unnecessary calibrations may also be performed when measurement devices maintain proper operating characteristics for periods exceeding the calibration frequency. The potential for faulted sensors to go unnoticed can cause poor product quality, poor economic performance, and unsafe conditions. Performing unnecessary calibrations leads to an increased risk of instrument damage.

An ISCV system consists of an empirical model and a set of associated modules to deal with data handling, data processing, statistic analyses, alarm logic, and other tasks. The main focus of this work is the development of an empirical model based on the widely used Partial Least Squares (PLS) technique. While many techniques can be applied for this task, the three most widely studied for plant-wide monitoring are Autoassociative

Artificial Neural Networks (AANN), Kernel Regression (KR), and Neural Network Partial Least Squares (NNPLS), which is the focus of this thesis.

The NNPLS technique differs from AANNs and KR in that it is a transformation based technique. The transformation eliminates instabilities caused by collinear data which are an issue that must be resolved in both of the other stated methods. The standard partial least squares (PLS) algorithm is a bilinear, empirical modeling technique introduced in the 1960's [Wold 1966]. In the present work, the technique is augmented to entertain non-linear relationships and is applied to a fossil-powered electrical generation plant. The use of this ISCV system is also presented for application to nuclear power plants as well as other large-scale processes.

The main features of the system draw from the fields of statistics for data processing tasks and decision logic, and from Chemometrics for the empirical model itself. Neural networks are also employed for these data-driven modeling tasks. Understanding the neural network partial least squares (NNPLS) algorithm is facilitated through a knowledge of basic statistical techniques such as data processing, data filtering, and linear regression, as well as more advanced topics such as principal component analysis (PCA) and the sequential probability ratio test (SPRT). These topics will be discussed in this thesis along with the integration of the NNPLS algorithm into an ISCV system.

1.2 Statement of Problem

The objective of an ISCV system is to monitor process sensors and produce correct values if the sensors are determined to be out of calibration, or have failed. Instrument surveillance increases the reliability of decisions and calculations based on process measurements through the continuous monitoring of the residual outputs from the ISCV system. The residual outputs are the differences between the measured values and the estimated values provided by the embedded empirical model. Calibration verification ensures the proper operation of process sensors and indicates when sensor re-calibration is necessary through continuous statistical evaluation of the residual outputs.

An ISCV system has been developed to monitor process sensors. The completed system has been tested at Tennessee Valley Authority's coal-fired plant in Kingston, TN. To ensure that adequate data are available for model development, a large reserve of data was collected, over the course of approximately 1 year, at the Electrical Power Research Institute's Instrument and Control center (EPRI I&C center) at the Kingston plant site.

The embedded empirical model of the ISCV system utilizes the NNPLS algorithm, which has been extensively evaluated with successful results and compared to results obtained with a competing methodology, AANNs. The NNPLS algorithm constructs inferential models for the process sensors using orthogonal projections of the data. Each sensor in the model has a separate inferential model, and the combined set of all

of the inferential models results in an autoassociative architecture, in which the outputs of the model are estimates of the inputs. The linear design of the standard partial least squares algorithm is enhanced through the replacement of the linear regression with small neural networks.

A set of graphical user interfaces have been created for the development of the NNPLS based ISCV models, and the subsequent monitoring of an appropriate set of signals either on-line or from historical data. The on-line feature of the monitoring interface uses Open Data Base Connectivity (ODBC) compliant Structured Query Language (SQL) constructs to complete the tasks of reading and writing data values to and from a database.

The NNPLS-ISCV system is intended to be a front-end processor such that plant operational and safety systems have the validated data values available to make performance calculations and suitably adjust the plant control systems. During this work, the ISCV system was integrated with the Kingston plant computer associated with the 9th unit of the site. Figure 1 shows the conceptual diagram of the system.

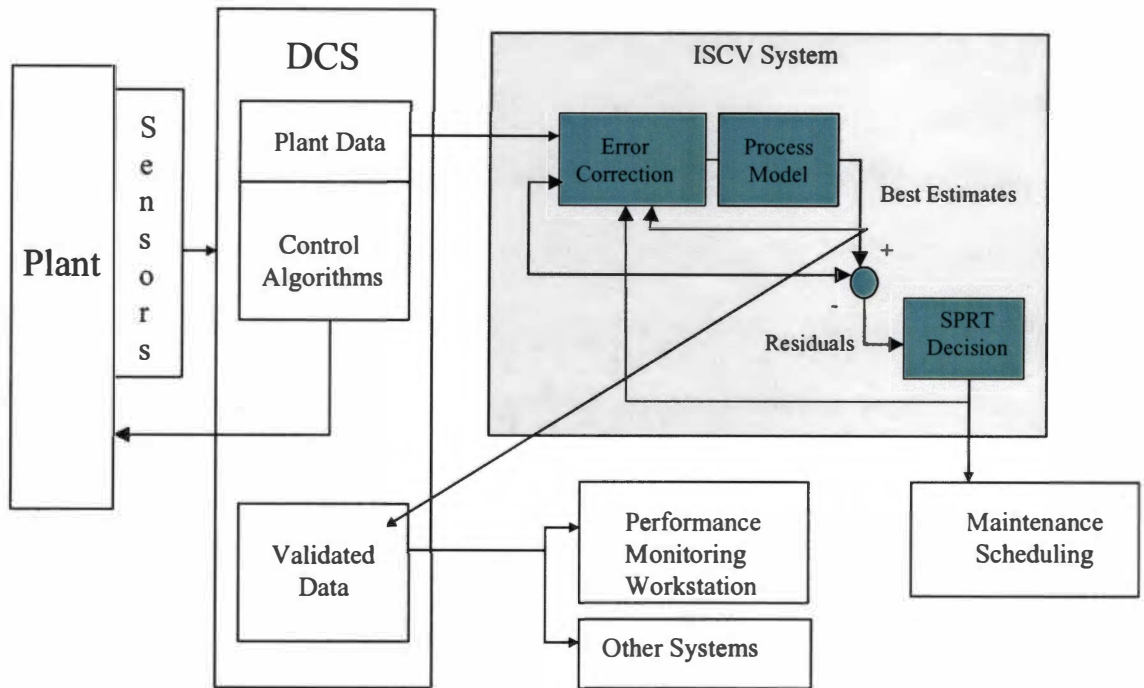


Figure 1: Conceptual diagram of the ISCV system interactions.

Data are retrieved from the plant sensors and the appropriate data are passed to the ISCV system. The sensor validation system has an error correction module that substitutes the best estimates for sensor values determined to be faulty. This keeps the other inferential process models using validated data and operating optimally. The process model produces estimates of the included sensors, and the SPRT decision module outputs the sensor status. If a sensor is determined to be faulty, maintenance can be arranged via the maintenance scheduling module. The validated ISCV system estimates are returned to the plant database where they are made available to operators and the plant control algorithms. Optimum efficiency will be realized when control

actions are based on correctly validated values, rather than the actual output of the plant sensors.

The performance of the NNPLS-ISCV system using the Kingston plant data was sufficiently accurate for detecting drifts of 2-5%, and under normal operations, estimation errors were generally less than 2%. While the prediction accuracy of the ISCV system is an important feature, it was not the primary concern of this work. The ISCV system presented was designed to handle large groups of sensors (>50) with adequate precision, minimal required oversight in development, and inherent stability through regularization.

1.3 Contributions

The main contribution of this work is the development and implementation of an alternate methodology for designing ISCV systems for large scale processes. The benefits of the NNPLS design: inherent regularization and minimal oversight or knowledge, have led to interest from other parties in the field of sensor validation. The Halden Reactor Project, in Halden Norway, has initiated a month long collaboration with the University of Tennessee to evaluate the use of the NNPLS design in their existing Plant Evaluation and Analysis by Neural Operators (PEANO) software.

1.4 Organization of Document

A literature review of ISCV techniques and the NNPLS methodology is provided in Chapter II. Chapter III begins with a description of the data, and the continuous collection of new data to allow for additional models to be added if desired. The theoretical basis for all of the applied methodologies is then provided, including: median filtering, the SPRT, and the NNPLS algorithm. Additional discussions are given regarding collinearity issues, multiple linear regression, and the benefits of the NNPLS algorithm in lieu of these considerations. Chapter IV describes the set of graphical user interfaces written for the creation and implementation of the NNPLS ISCV system, and full details of each interface are provided in Appendices A, B, and C. The prediction results and drift detection results are summarized in Chapter V. Conclusions are drawn in Chapter VI, and recommendations for future work are presented in Chapter VII. Prediction results are shown for each of the 51 signals in appendix D. Appendix E contains tables of the signals available in the data archive created during this study. Appendix F contains the necessary MATLAB code to develop and implement a NNPLS ISCV system.

2.0 LITERATURE REVIEW

On-line calibration verification systems have been under study at the University of Tennessee beginning in the 1980's [Upadhyaya 1985, 1989], with investigations into the application of artificial intelligence techniques to nuclear power plants. Researchers at Argonne National Laboratory continued with similar research throughout the late 1980's and 1990's [Mott 1987] in which they developed the Multivariate State Estimation System (MSET) which has gained wide interest in the US Nuclear Industry [Gross 1998]. The major European participant in this area is the Halden Research Project where Dr. Paolo Fantoni and his multi-national research team have developed a system termed Plant Evaluation and Analysis by Neural Operators (PEANO) [Fantoni 1998] and applied it to the monitoring of nuclear power plant sensors.

Numerous data-based technologies have been used by major researchers in the field including autoassociative neural networks [Fantoni 1998, Hines 1998, Upadhyaya 1992], fuzzy logic [Hines 1997], principal component analysis [Qin 1999], non-linear partial least squares [Qin 1992, Rasmussen 2000a], and kernel based techniques such as MSET [Singer 1996] and the Advanced Calibration Monitor (ACM) [Hansen 1994]. The three major technologies used in the Nuclear Power Industry that use different data-based prediction methods are a kernel based method (MSET), a neural network based method (PEANO and the University of Tennessee AANN), and a transformation method (NNPLS).

The use of the NNPLS algorithm for sensor validation in highly-dimensional spaces is the subject of this thesis. The competing data-driven modeling paradigm of AANNs was also evaluated during the preparation of this thesis; however, the similar performances obtained with both methods, and the benefits of the NNPLS architecture for highly dimensional applications preclude the extensive evaluation and performance reporting of AANNs in this study. The benefits of the NNPLS ISCV design stem from the inherent regularization of the algorithm, and the ease with which highly dimensional data sets are modeled. The sensitivity of the overall model is reduced due to the use of a set of inferential models, rather than a single model with a multidimensional set of response variables. The basis of the NNPLS architecture [Holcomb 1992, Malthouse 1997, Qin 1996] is a linear projection based modeling paradigm, Partial Least Squares (PLS) [Geladi 1986, Hoskuldsson 1988, Kresta 1994], which performs a linear regression between predictor and response variables. The extension of PLS to capture nonlinearities comes from the replacement of the linear regression with an appropriate set of one hidden layer neural networks. The stability of these small neural networks is substantially greater than a complementary fully connected AANN due to their simple error surfaces and small number of weight and bias values.

A significant amount of work has been reported using Artificial Neural Networks (ANNs) to model process systems. For the purposes of ISCV, a commonly used ANN structure is the Autoassociative architecture (AANN) [Dong 1994, Hines 1998, Nabeshima 1995, Upadhyaya 1992]. The capability of ANNs to model nonlinear

relationships within the data has become an indispensable feature of process models, and has improved their prediction accuracy over their equivalent linear models. There are several significant issues that arise in developing process models with ANNs, which include variable selection, architecture selection, sensitivity, and the effects of collinearity on neural network predictions [Qin 1997].

The combination of Partial Least Squares (PLS), which is also referred to as Projection to Latent Structures, and ANNs leads to an alternate method of empirical modeling for ISCV systems. The standard PLS technique, initially developed by Wold [1966], is a technique in which both the input data (X) and output data (Y) are transformed to non-identical coordinate systems based on the covariance between X and Y. The linear transformation of the PLS method provides a supervised reduction of the process system's data while preserving the information most relevant to the specific sensor(s) to be modeled. In addition, the reduced data set is derived such that the transformed variables are orthogonal, eliminating the well known problems associated with large collinear data sets in linear modeling [Wold 1984] as well as ANNs [Qin 1997]. Following this transformation, the PLS method prescribes a linear regression to relate the reduced data to the sensor(s) being modeled. By replacing the linear regression with a single hidden layer ANN, nonlinear relationships can also be incorporated into the model.

In recent years, the use of feed forward ANNs in conjunction with the standard PLS technique has been gaining importance, and is often referred to as Nonlinear Partial

10

Least Squares (NLPLS) [Holcomb 1992, Malthouse 1997, Qin 1992, Rasmussen 2000a, Xu 1997]. An alternate approach using Radial Basis Function Networks, PLS/RBF has also been proposed [Wilson 1997a, 1997b]. To avoid confusion between the NLPLS techniques incorporating quadratics and splines [Baffi 1999a, Wold 1989, Wold 1992], the term Neural Network Partial Least Squares (NNPLS) will be adopted when referring to the combination of PLS and ANNs as has been reported elsewhere [Baffi 1999b, Qin 1996].

3.0 THEORY

This chapter is divided in several sections: first the need and methodology of data collection is presented, second the basis of median filtering is presented, followed by the theory of sequential testing and the SPRT. Multiple linear regression (MLR) is then discussed, and collinearity is briefly explained and related to the tasks of MLR. Finally the NNPLS algorithm is developed starting with a description of the standard PLS algorithm, and extending via incorporation of neural networks to the final algorithm. Additional features of the NNPLS-ISCV system are then explained, relating to the parameter settings used to develop the model for which the results are reported. These additional features are orthogonal component selection, SPRT parameter setting, and alarm logic parameter setting.

3.1 Data Collection

In order to develop accurate empirical models, it is imperative that a substantial reserve of historical data be available. One of the assumptions made in any empirical modeling technique is that the data used to train or create models are accurate and representative of the nominal plant operating conditions. Over the 2 years that we have been working with the Kingston plant's data we have noted various changes in the relationships between the signals. In all modeling techniques, the choice of training or development data is a crucial step in producing an optimal model for the given process. To have a

vast reserve of data is an invaluable resource for the creation of sufficiently accurate signal validation models. The task of sorting through the data becomes time consuming when large sources of data are available; however, techniques can be applied to retain portions of the data that are most representative of the process. These data are chosen based on their being most likely from the true operating mode. The data is recorded in instantaneous samples to preserve the actual values provided by the plant sensors, and these data can be filtered in a variety of ways to suit the particular needs of the user.

Tennessee Valley Authority's (TVA) Kingston, TN, coal-fired fossil power generating facility was the site for all data recording. The task of data recording was performed at the Electrical Power Research Institute's (EPRI) Instrumentation and Control (I&C) Center associated with the 9th unit of the plant, and located at the plant site. The mission of the EPRI I&C Center, constructed in 1994, is to make utilities more productive and profitable by increasing the migration to better controls, instrumentation, and information management in power plants. The focus of the I&C Center is to develop improved algorithms for process control and the accompanying advanced instrumentation. Figure 2 demonstrates the interrelationships among three sub-systems of this power generating cycle: boiler system (steam generator), turbine/generator and condenser/heat rejection [ASME PTC PM-193]. Because only a few signals from the condensing system were provided, the research concentrates on two major sub-systems: the boiler system and the turbine system.

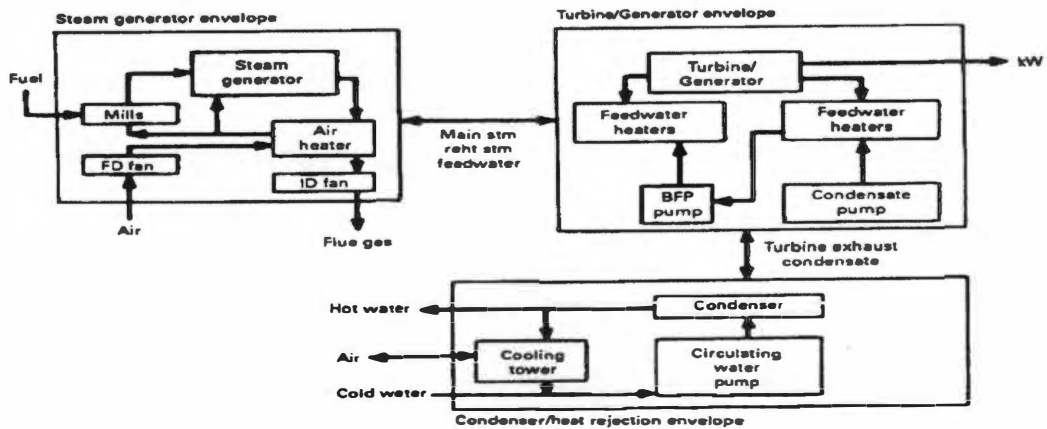


Figure 2: Fossil power generation system [ASME PTC PM-193].

The types of signals recorded during this work are pressures, temperatures, flow rates, and levels. These signals are associated with the various components of the system: steam generator, boiler feed pumps, feedwater heaters, air preheaters, pulverizers, and condensers. The Kingston plant is a load following system, with the unit gross generation of the 9th unit ranging from 80-190MW.

3.1.1 Methods of Data Collection

All of the data were recorded at the EPRI I&C center at the Kingston plant site. The data is from the 9th unit of the plant, and was recorded between 4/2000 and 11/2001. A large amount of data has been recorded using a MATLAB [1999] script utilizing Dynamic Data Exchange (DDE) commands. DDE sends data between applications using Windows messages according to a documented protocol. DDE uses a hierarchy of three names, the SERVICE, the TOPIC and the ITEM. A DDE conversation is established using the service and topic names as a pair. It is convenient to name the pair

a CHANNEL. The item part of the name is used to identify the particular data or command being requested by the client once a conversation is established. The three MATLAB functions that were used in the DDE script were `ddeinit.m`, `ddereq.m`, and `ddeterm.m`. All of these functions are available in the base software package. The `ddeinit` function returns a channel handle related to the initiated conversation, based on the service and topic specified. The data is retrieved via the `ddereq` function based on the conversation handle, and the specified item. After all of the required data is retrieved, the `ddeterm` function is executed to terminate the current conversation. The script used to retrieve data from the server at the Kingston plant is provided in appendix F. DDE was the initial method of data collection used from 4/24/2000 to 1/29/2001. The MATLAB script program created a single file for each day of data recorded with a filename structure 'tvadataddmmyyyy.mat' (dd=day, mm=month, yyyy=year).

To facilitate the writing of validated data to the plant computer, a more flexible means of communication was adopted and implemented in a graphical user interface. The interface utilized structured query language (SQL) commands to read and write information along an established connection with the plant computer. This interface was created with National Instrument's LabView software [2000], and is discussed later in the SQL connection interface section of chapter IV. The data received by the SQL connection interface was stored in binary files with a similar filename format as used before, with the exception of the filename extension, which was changed to `.dbl`.

Data requests for both methods of collection were made in one minute increments based on the internal clock of the computer. Due to varying spans of time to complete each data request, the exact sampling rate is not 1 minute. The time information of the recorded data is available in each of the files.

3.1.2 Description of Recorded Data

Using the MATLAB script, data was recorded as instantaneous 1 minute samples over a period beginning on 4/24/2000 and ending on 1/29/2001. Signal descriptions for the data recorded with MATLAB are provided in appendix E. Note that from 4/24/2000 to 6/14/2000, data was only recorded for the first 89 signals in the list. Beginning on 6/15/2000, data was recorded for all of the 140 signals in the list. The reason for the increased number of signals was to include all of the signals that the Performance Monitoring Workstation (PMW) requests as input so that data would be available for these signals if new models were required. The PMW for the Kingston plant monitors the outputs of the instrument devices which are critical to the process and executes calculations based on these values to provide efficiencies and other important parameters to plant operators. The ISCV system at the Kingston plant site was to be integrated into the PMW so that the calculated parameters would be based on the validated data. The prototype model required only the first 89 signals, and thus the original data collection program was tailored to the specific needs of the original model. Data collection with the MATLAB script was continuous excluding the exceptions as noted below in Table 1.

Table 1: Description and Dates of Recognized Incidents

INCIDENT	BEGIN DATE AND TIME	END DATE AND TIME
Scheduled Outage	5/6/2000 – 20:49	5/19/2000 – 19:58
Server Down	5/31/2000 – 17:00	6/15/2000 – 12:36
Data Collection Error	6/16/2000 – 00:00	6/18/2000 – 13:29
Unknown Loss of Power	8/24/2000 – 22:43	8/26/2000 – 10:57
Unknown Loss of Power	9/4/2000 – 9:44	9/11/2000 – 11:46
Data Collection Error	10/11/2000 – 10:56	10/18/2000 – 12:35
Unknown Loss of Power	10/18/2000 – 12:35	10/20/2000 – 4:30
Collection Stopped	12/7/2000 – 11:40	1/18/2001 – 11:31
Erroneous Measurements	1/20/01 – 00:00	1/25/01 – 00:00

The periods for which incidents occurred are further explained below:

Scheduled Outage – data were recorded during this time, but is not suitable for model development.

Server Down – data were recorded during this time, but is not suitable for model development. The files contain constant values for all signals. The server went down, and the last value in memory was being continuously recorded during this time.

Data Collection Errors – data were unavailable during these periods, due to programming errors and an isolated power trip, which caused the computer to re-boot.

Unknown loss of Power and Erroneous Measurements – data were recorded during these periods, but is not suitable for model development. During the loss of power

periods the unit gross generation is at or near zero. During the period of erroneous measurements, the recorded data was near zero for 2 days, and then was near 1000MW for 2 days.

Collection Stopped – data were unavailable during this period. During this time testing began with the new SQL connection interface, and to isolate errors more easily the MATLAB data collection program was stopped.

Data collection was intermittently paused while the new SQL interface was being tested and brought online. Data has been recorded with the SQL interface almost continuously beginning in June 2001. Some periods were not recorded due to errors in updating the interface, and some periods were not recorded due to plant outages or server errors. Data collection was stopped in November of 2001. Signal descriptions of the data recorded by the SQL interface are provided in appendix E. This list is slightly different than the previous list of signals incorporated into the MATLAB script to eliminate a few unnecessary signals.

3.1.3 Examples of Operating Condition Changes

Over time, it is expected that the process being monitored will exhibit operating condition changes which will affect the relationships between the signals associated with the process. While these changes are not unexpected, they raise difficulties when attempting to compile a set of data which is representative of the process under study.

If a particular signal operates at different levels, and these level changes are not similarly represented in related signals, the true value of the parameter being measured is not clearly known. One would either have to use all of the data available for the given sensor, or use engineering judgment to remove portions where the operational level appears to be improper. If neither of these approaches results in a suitable model for the particular sensor, it may need to be omitted from the set of monitored signals for two reasons. First, poor validation results would lead to a lack of confidence in the validation system. Second, if unreliable or implausible sensor data is used to develop an empirical model, it can adversely affect the performance of the remaining related signals. Operating condition changes may also occur which result in changes to all of the process sensors simultaneously. Often these are slight changes that are not instantaneous, but happen over periods of 6 months or longer. These types of changes are less problematic and, if these changes do not move the process sensor measurements outside of the operating regions represented in the data used to develop the model, should not cause the ISCV system to decrease in accuracy. An example of long-term changes is shown in figure 3, which shows a plot of the combustion air flow signal. It shows small operating condition changes between January, June and August. Figure 4 indicates a change in the RH Attemp A before spray signal that is so large that we have classified it as abnormal after discussions with the staff at the EPRI I&C Center.

Long-term effects or changes may necessitate the retraining of the empirical model if its performance is compromised due to operating condition changes. Performance may

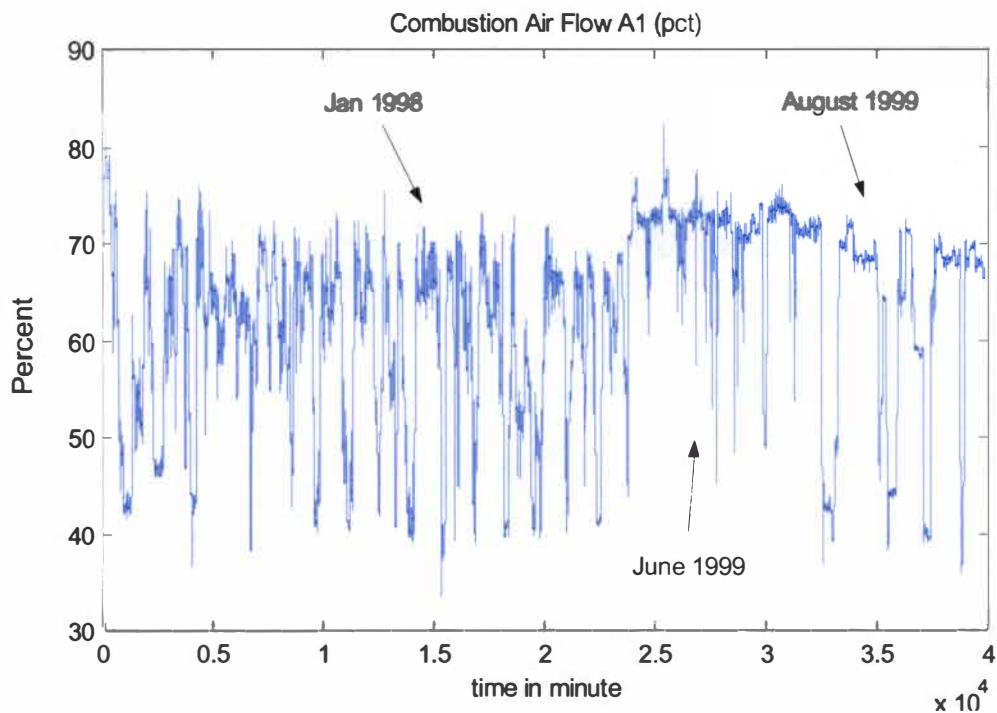


Figure 3: Condition changes for combustion air flow (%).

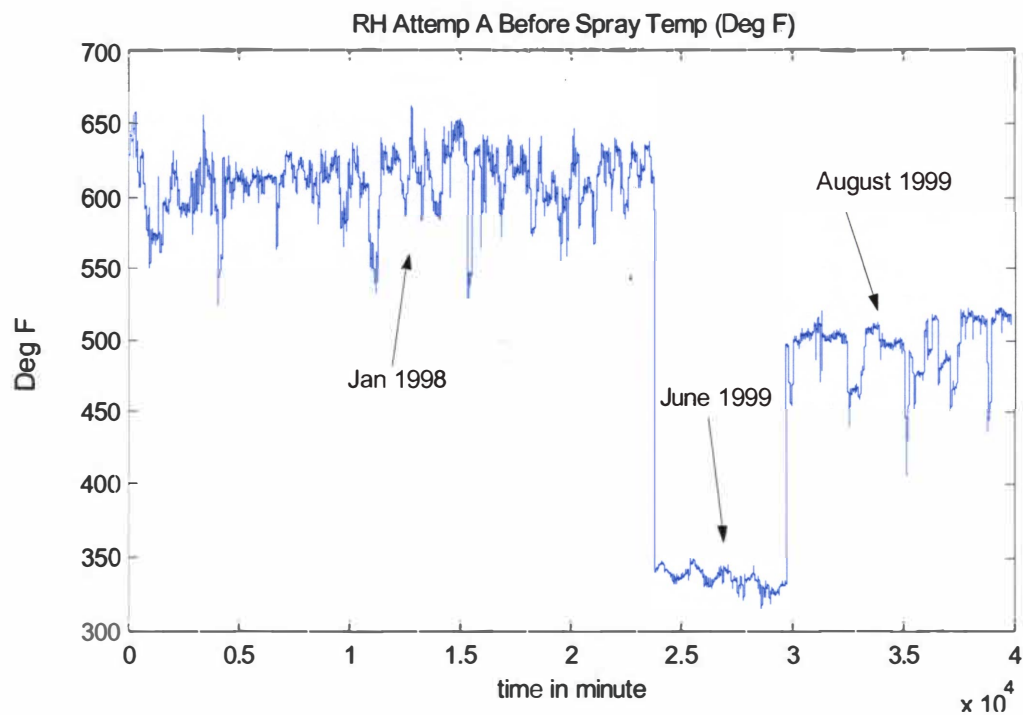


Figure 4: Condition changes for Reheat Attemp A before spray temp (DegF).

be compromised if the relationships between the signals change, or if the signal values move outside of the range represented in the development data. Neural networks perform poorly when presented with inputs that are outside of the range represented in the training data, due to their nonlinear activation functions. The ability of an empirical model to predict values outside of the training data range is referred to as generalization.

Condition changes have been noted more recently in the station service load signal (figure 5). Prior to the change in the station service load signal, the correlation coefficient between the station service load signal and the unit gross generation signal was 0.9950. Such a high correlation value ensures that the NNPLS model for

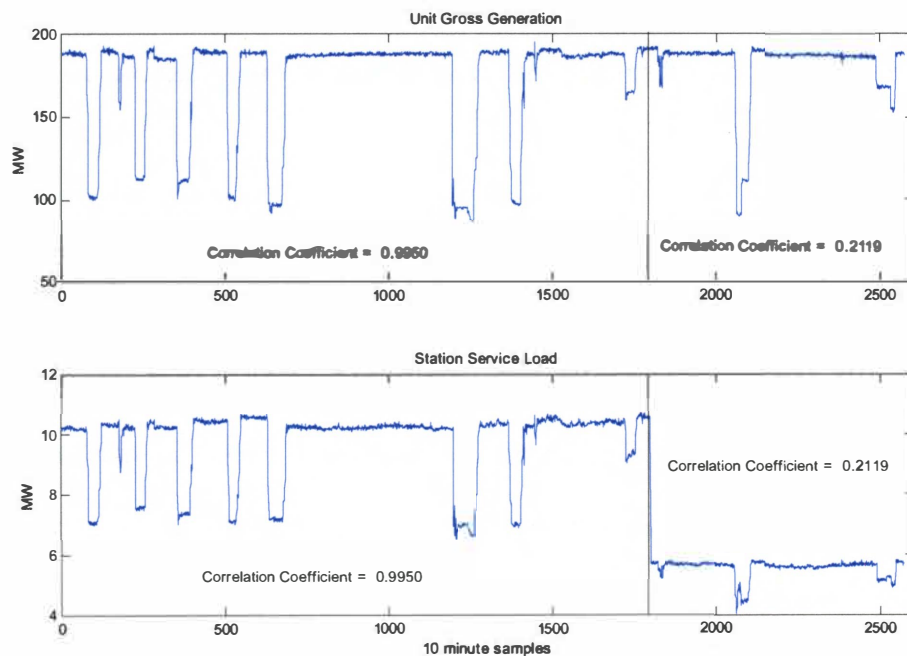


Figure 5: Condition changes for station service load with respect to gross generation.

either signal will rely heavily on the other. Though other highly correlated signals will also play a role in the predictions, this drastic change will inevitably affect the ability of the system to make predictions for these signals. The correlation coefficient after the change is 0.2119. This extreme change was classified as a sensor fault.

The presentation of operating condition changes was intended to illustrate one of the primary issues of empirical modeling, training data selection. The training data represents the cumulative knowledge of the process from which the empirical model will be derived. Processes inevitably will change over time, thus it is important to obtain an adequate set of training data that contains all of the expected conditions in which the process will operate. Not having a representative training set for a system to be monitored is a problem. When the system moves to a new operating regime, that would not be expected based on the historical data, we cannot expect the system to perform with high accuracy.

3.2 Median Filters

Median filtering is a nonlinear signal processing technique useful for noise and outlier suppression. The median is a member of the class of order statistics, and median filtering uses a moving window, of specified length, which replaces the central element in the window with the median value of all points contained in the window. The median value is determined with respect to the original values in the signal, and does

not depend on the median value inserted at the previous point x_{i-1} to determine the median value at the current location x_i . The two properties of median filters that propose its use over moving average filters are their edge preservation characteristics, their abilities in smoothing of impulse noise, and the fact that they do not induce phase shifts [Tyan 1981].

Assume a set of ordered points:

$$x_1, x_2, \dots, x_n$$

where:

$$x_1 \leq x_2 \leq \dots \leq x_n$$

The median is, for n odd, the middle number, and for n even, is the mean of the two middle numbers. Other definitions have also been given for even n , but vary only slightly [Justusson 1981].

A median filter of size n , on a sequence $\{x_i, i \in \mathbb{Z}\}$, can be defined as:

$$y_i = \underset{n}{Median}(x_{i-v}, \dots, x_i, \dots, x_{i+v}), i \in \mathbb{Z}$$

where:

$\mathbb{Z}$ = the set of all natural numbers

$$v = \frac{n-1}{2} = \text{filter rank}$$

n = filter window size

y_i = the i^{th} output of the median filter

Using similar notation, a moving average filter can be defined as:

$$z_i = (x_{i-v} + \dots + x_i + \dots + x_{i+v}) / n, i \in \mathbb{Z}$$

where:

z_i = the i^{th} output of the moving average filter

Observation of the above equations indicates that median filters will preserve edge characteristics of a signal, whereas a moving average filter will change an edge into a ramp. Similar definitions can be described for two-dimensional filters [Justusson 1981], but are not applicable to this work. In addition, if not explicitly stated, all discussions will be limited to filter window sizes where n is odd.

3.2.1 Smoothing of Impulse Noise

Impulse noise can be described as signal disturbances of very large positive or negative values of short duration. Causes of impulse noise in process sensor data are electrostatic discharges, multiplexer problems, proximity of welding cables to measuring instrument electronics, as well as others. Median filters are well suited to suppress such noise provided that the window size is chosen to be at least twice the width of the impulses, or at least twice the number of impulses within a given window. In other words, the number of spikes that can be eliminated by median filtering, within a fixed window, is equal to the rank of the filter window. For beginning and ending elements of a vector (signal), an appropriate number of zeros are included in the set of elements used to determine the median values. For example, the resultant value of a median filter on the first element of a signal will be the median value of the first element, the subsequent elements up to the filter rank, and a number of zeros equal to the filter rank. This fact adds a minor distortion at the beginning and ending points of a

24

median filtered signal and has a negligible effect on model development tasks. If small amounts of data are being used and this distortion effect is a concern, the first few elements of a signal, up to the filter rank, should not be filtered.

Impulse noise (spikes) can be defined by:

$$x_i = \begin{cases} z_i & \text{with probability } p \\ s_i & \text{with probability } 1 - p \end{cases}$$

where:

x_i = the i^{th} value of a signal

z_i = an independent random variable with uniform distribution on the interval $[0, d]$

s_i = the i^{th} "true" value of a signal

Spikes in the signal will occur with probability p , and the numerical value of a spike at the i^{th} point is given by z_i . Similarly, the absence of a spike at the i^{th} point is given by $1 - p$, and the numerical value of the signal at this point is given by s_i . Define:

$$Q(n, p) = P [\text{correct reconstruction of the signal at the } i^{\text{th}} \text{ point}] = P(y_i = s_i)$$

The number of spikes in a filter window has a binomial distribution, leading to the following result [Justusson 1981]:

$$Q(n, p) = \sum_{k=0}^{(n-1)/2} \binom{n}{k} p^k (1-p)^{n-k}$$

In addition, the expected size of an error at any point will be reduced from:

$$E(z_i) = 0.5 * d \quad \text{to:} \quad E(z_i)^* = \frac{E[\text{Median}(x_i)]}{1 - Q(n, p)} * d$$

where:

$$E[\text{Median}(x_i)] = \sum_{k=(n+1)/2}^n \frac{k - (n-1)/2}{k+1} \binom{n}{k} p^k (1-p)^{n-k}$$

Table 2 shows representative values for some of these quantities.

Table 2: Representative quantities of the expected error values induced by impulse noise, and the reduction of these errors based on certain probabilities of occurrence of impulse noise, and various filter window sizes.

Rank	Window Size (n)	p	$1 - Q(n, p)$	$Q(n, p)$	% Size of Original Expected Error
1	3	0.1	0.0280	0.9720	67.86
1	3	0.2	0.1040	0.8960	69.73
2	5	0.1	0.0086	0.9914	51.64
2	5	0.2	0.0579	0.9421	53.59
3	7	0.1	0.0027	0.9973	41.77
3	7	0.2	0.0333	0.9667	43.95
4	9	0.1	0.0009	0.9991	35.11
4	9	0.2	0.0196	0.9804	37.36

In table 2, p is the probability of occurrence of impulse noise at any point. The magnitude of the impulse noise is in the interval $[0, d]$, where $d = 1$. $Q(n, p)$ is the probability of correct reconstruction of the signal at any point. The last column displays the percent size of the original expected error which could occur at any point following the application of the specified median filter, that is $[E(z_i)/E(z_i)^*] * 100\%$.

For obvious reasons, moving average filters are not well suited for filtering of impulse noise. For example, a large spike present in a series of 3 points will result in a median filtered signal with no remnant of the erroneous point; whereas, a moving average filtered signal will still exhibit some remnant of the original spike. Moving average filters are more beneficial in some cases; e.g., for normal white noise. Normal white noise is less of a concern in this work due to the distributions already being somewhat normal around a mean value, and the overall precision of the method in general.

The general purpose of filtering in the present work is to eliminate spikes within the data. An erroneous spike may cause an instantaneous trip of an ISCV system alarm. Alarm logic is introduced to reduce the chances of this phenomenon; however, if several spikes occur in succession, the alarm logic cannot guarantee prevention of a false alarm.

3.2.2 Edge Preservation

The second advantageous quality of median filters is edge preservation. When dealing with edges, moving average filters result in a substantial blurring effect. This effect is much less reduced if median filtering is applied. An edge is defined as being at a point $i = 1$, where for $i \leq 0$, x_i is $N(0, \sigma^2)$ and for $i \geq 1$, x_i is $N(h, \sigma^2)$. It is important to note that for relatively small edge heights, there is not much of a difference between

median and moving average filter results; however, for large edge heights ($> 3\sigma$) the advantage of median filtering becomes clear.

For the purposes of this work, the edge preservation qualities of median filters are of less interest than the impulse noise reduction. In a load following system edges are present in the signals; however, these edges are not very frequent. More detailed discussions of the statistical and deterministic properties of median filters, as well as modifications such as weighted median filters and iterated median filters, are available in the literature [Justusson 1981, Tyan 1981].

3.3 Sequential Testing

The number of observations required by a sequential test depends on the outcome of the observations and is therefore, a random variable. The sequential method for testing a hypothesis, H , requires a rule for making one of the following three decisions:

To accept H .

To reject H .

To continue to the next iteration by taking an additional observation.

This sequential process continues until one of the first two decisions is reached. A sequential test is completely defined by the sets R_m^0 , R_m^1 , and R_m for all positive integers m , where the sets are mutually exclusive and add up to the entire sample space M_m . If the first observation, x_1 , lies in R_1^0 , H is accepted, if it lies in R_1^1 , H is

rejected, or a second observation is made if x_1 lies in R_1 . If a second observation is made, the sample set becomes (x_1, x_2) and similar to above, its location is to be determined to be in one of R_2^0 , R_2^1 , or R_2 . This process continues until at the m^{th} observation, the sample set is determined to lie in either R_m^0 , or R_m^1 .

It is assumed that the distribution of x is known and can be given by a function $f(x, \theta_1, \dots, \theta_k)$; however, the true value of the parameters remains unknown. Thus, the probability of accepting a null hypothesis, H_0 , is a function of these parameters which will be simply referred to as θ . This function is referred to as the operating characteristic function, and will be written as $OC(\theta)$, in lieu of Wald's [1945] original presentation of the function as $L(\theta)$ because the uppercase L will be reserved for the likelihood function. Thus, the probability of accepting H_0 is $OC(\theta)$ and the probability of rejecting H_0 is $1-OC(\theta)$. For any parameter point, θ , the probability of making a correct decision can be obtained directly from the operating characteristic function.

Consider $f(x, \theta)$ where there is only one unknown parameter. The null hypothesis, H_0 , is the statement that the true parameter value is included in the set ω . Where ω is the set of all parameter points, θ , which are consistent with H_0 . If H_0 is the hypothesis that θ is less than or equal to θ_0 , then ω is the set of all values of θ for which $\theta \leq \theta_0$, a boundary point exists at θ_0 , and a set $\bar{\omega}$ can be defined as the set of all values of θ

for which $\theta > \theta_0$. An alternative hypothesis could also be used where ω is the set of all values for which $\theta_0 \leq \theta \leq \theta_1$, there are two boundary points, and $\bar{\omega}$ is the set of all values for which $\theta < \theta_0 \cup \theta > \theta_1$. If the true value of θ lies near a boundary point, the preference for acceptance or rejection of H_0 will be only slight. If θ lies on a boundary point, there will be no preference for one action over the other. In general, the parameter space is divided into three mutually exclusive zones:

- Zone 1: A zone where acceptance of H_0 is strongly preferred.
- Zone 2: A zone where rejection of H_0 is strongly preferred.
- Zone 3: A zone where a strong preference, for acceptance or rejection of H_0 , does not exist.

As an example of these divisions, assume x is normally distributed with known standard deviation and unknown mean, then there is only one unknown parameter $\theta = \bar{x}$. Assume that the most desirable parameter value is $\theta = \theta_0$; such that, H_0 may be defined as $|\theta - \theta_0| < c$, where c is a constant at the boundary of ω and $\bar{\omega}$. The zone definitions, in this case, are then:

- Zone 1 (acceptance): $|\theta - \theta_0| < c - \Delta$
- Zone 2 (rejection): $|\theta - \theta_0| > c + \Delta$
- Zone 3 (indifference): $c - \Delta \leq |\theta - \theta_0| \leq c + \Delta$

Where Δ is a small positive value, and is introduced so that decisions of acceptance or rejection of H_0 are not made at or near the boundary.

To come up with an operating characteristic function, consider that for any θ in zone 1, the probability of rejecting H_0 (incorrect decision) should be less than or equal to a preassigned value α , that is $1 - OC(\theta) \leq \alpha$. Similarly, for any θ in zone 2, the probability of accepting H_0 (incorrect decision) should be less than or equal to a preassigned value β , that is $OC(\theta) \leq \beta$. Both α and β have values between 0 and 1. In short, a sequential test is admissible if it satisfies: $1 - OC(\theta) \leq \alpha$ for any θ in zone 1, and $OC(\theta) \leq \beta$ for any θ in zone 2. Consider a simple hypothesis tested against a single alternative. The null hypothesis, H_0 , is the statement that $\theta = \theta_0$, and the alternative hypothesis, H_1 , is the hypothesis that $\theta = \theta_1$. The two associated numbers, α and β , are also selected. If H_0 is true, α is the probability that H_0 will be rejected. This is referred to as an error of the first kind. If H_1 is true, β is the probability that H_0 will be accepted. This is referred to as an error of the second kind.

The number of observations required by a sequential test is a random variable, denoted as n . The expected value of n can be determined based solely on the distribution of x , which is dependent only on θ . From this, the average sample number function can be defined as $E_\theta(n)$. The operating characteristic function describes how well the test makes correct decisions, and the average sample number function quantifies the cost, in terms of the number of observations required.

3.3.1 The Sequential Probability Ratio Test

Let $f(x, \theta)$ denote the distribution of x . Let H_0 be the hypothesis that $\theta = \theta_0$, or that the distribution of x is $f(x, \theta_0)$. Let H_1 be the hypothesis that $\theta = \theta_1$, or that the distribution of x is $f(x, \theta_1)$. For any integer m , the probability that a sample, $x_1, \dots, x_m$, is obtained is given by the likelihood functions:

$$L_m(\theta_0) = \prod_{i=1}^m f(x_i, \theta_0) \quad \text{When } H_0 \text{ is true.}$$

$$L_m(\theta_1) = \prod_{i=1}^m f(x_i, \theta_1) \quad \text{When } H_1 \text{ is true.}$$

Rather than compute the likelihood functions directly, the logarithm of these functions can be obtained:

$$\ell_m(\theta_0) = \log[L_m(\theta_0)] = \sum_{i=1}^m \log f(x_i, \theta_0) \quad \text{When } H_0 \text{ is true.}$$

$$\ell_m(\theta_1) = \log[L_m(\theta_1)] = \sum_{i=1}^m \log f(x_i, \theta_1) \quad \text{When } H_1 \text{ is true.}$$

After each observation, the probability ratio of the log likelihood functions is computed:

$$\Lambda_m = \frac{\ell_m(\theta_1)}{\ell_m(\theta_0)}$$

The decision rule is then implemented as follows:

If $\Lambda_m \geq \log A$, H_0 is rejected and the process is terminated.

Else if $\Lambda_m \leq \log B$, H_0 is accepted and the process is terminated.

Else an additional observation is taken, i.e. $\log B < \Lambda_m < \log A$.

The relationships among the quantities α, β, A , and B have been derived and discussed extensively in the literature [Wald 1945,1947]. These four quantities are related by inequalities; however, the equalities below give adequate results with minimal increases in the number of samples required by the test.

$$A = \frac{1-\beta}{\alpha}, \text{ and } B = \frac{\beta}{1-\alpha}.$$

3.3.2 SPRT for Normally Distributed Random Variables

In the processing of residuals for signal validation, the residuals are considered to be normally distributed random variables with a known variance and an unknown mean.

The SPRT for a given set of residual observations can be completely defined by:

1. x represents the residual observations, $(x_1, x_2, \dots, x_n)$.
2. θ is the unknown parameter under the hypotheses, H_0 and H_1 .
3. $f(x, \theta) = N(\theta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left[\frac{1}{2\sigma^2}\right](x-\theta)^2}$, where σ^2 is known.
4. $H_0 : \theta = \theta_0$ is the null hypothesis.

$H_1 : \theta = \theta_1$ is the alternative hypothesis.

$$5. \quad \Lambda_m = \frac{\ell_m(\theta_1)}{\ell_m(\theta_0)} = \frac{\sum_{i=1}^m \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left[\frac{1}{2\sigma^2}\right](x_i - \theta_1)^2} \right)}{\sum_{i=1}^m \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left[\frac{1}{2\sigma^2}\right](x_i - \theta_0)^2} \right)} = \sum_{i=1}^m \left[x_i(\theta_1 - \theta_0) - \frac{m}{2}(\theta_1^2 - \theta_0^2) \right]$$

6. The decision rule is:

If $\Lambda_m \geq \log A$, H_0 is rejected and the process is terminated.

Else if $\Lambda_m \leq \log B$, H_0 is accepted and the process is terminated.

Else an additional observation is taken, i.e. $\log B < \Lambda_m < \log A$.

7. The constants A and B are determined so that the test will have the prescribed strength (α, β) . Determination of these constants in practice is given by:

$$A = \frac{1 - \beta}{\alpha}, \text{ and } B = \frac{\beta}{1 - \alpha}.$$

3.3.3 Operating Characteristic Function for Normally Distributed Random Variables

The operating characteristic function, $OC(\theta)$, is the probability that the null hypothesis is accepted when θ is the true value of the parameter. An approximate form of the operating characteristic function, as presented by Wald [1947], is shown below:

$$OC(\theta) \sim \frac{A^{h(\theta)} - 1}{A^{h(\theta)} - B^{h(\theta)}}$$

For the case of $x \sim N(\theta, \sigma^2)$, where σ^2 is known:

$$h(\theta) = \frac{\theta_1 + \theta_0 - 2\theta}{\theta_1 - \theta_0}$$

An example of the operating characteristic function is shown in figure 6.

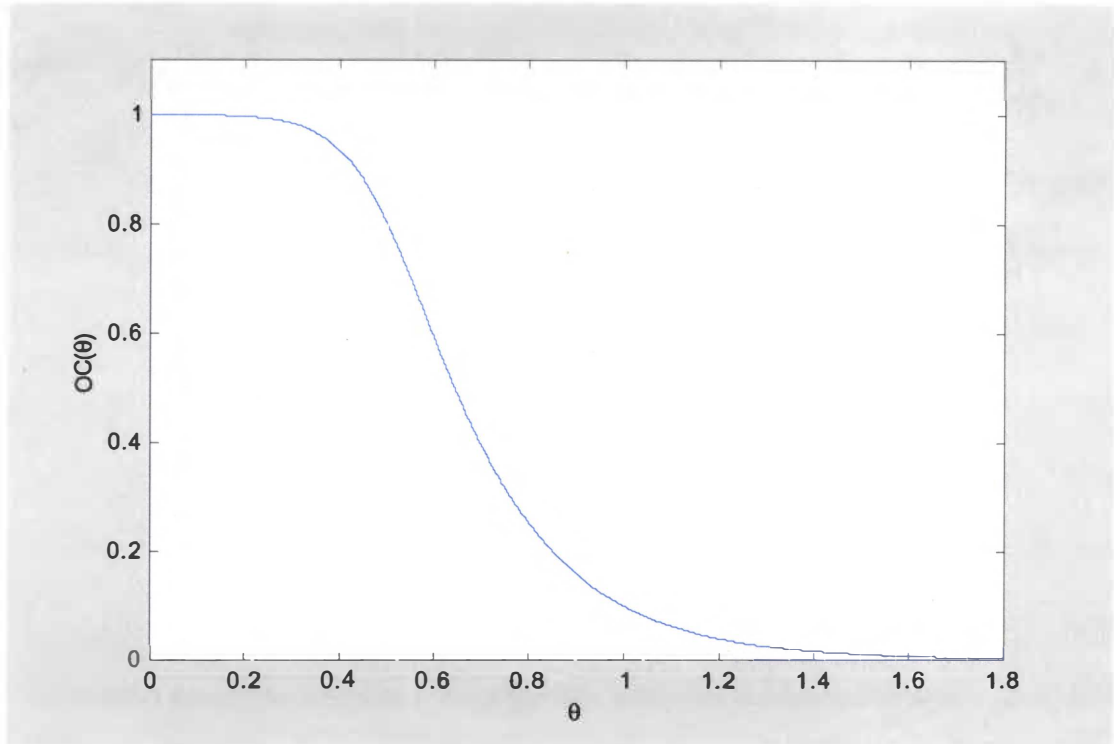


Figure 6: Operating characteristic function for $x \sim N(\theta, \sigma)$, where $H_0: \theta = \theta_0 = 0$, $H_1: \theta = \theta_1 = 1$, $\alpha = 0.0001$, and $\beta = 0.1$.

Figure 6 indicates that the probability of accepting the null hypothesis increases to a value of 1, as the true value of the parameter approaches the null hypothesis.

3.3.4 Average Sample Number Function for Normally Distributed Random Variables

The average sample number function is the expected value of the number of observations required to make a decision. An approximation for the average sample number function, $E_0(n)$, is given below [Wald 1947], for the case of $x \sim N(\theta, \sigma^2)$ where σ^2 is known, and figure 7 shows a representative result.

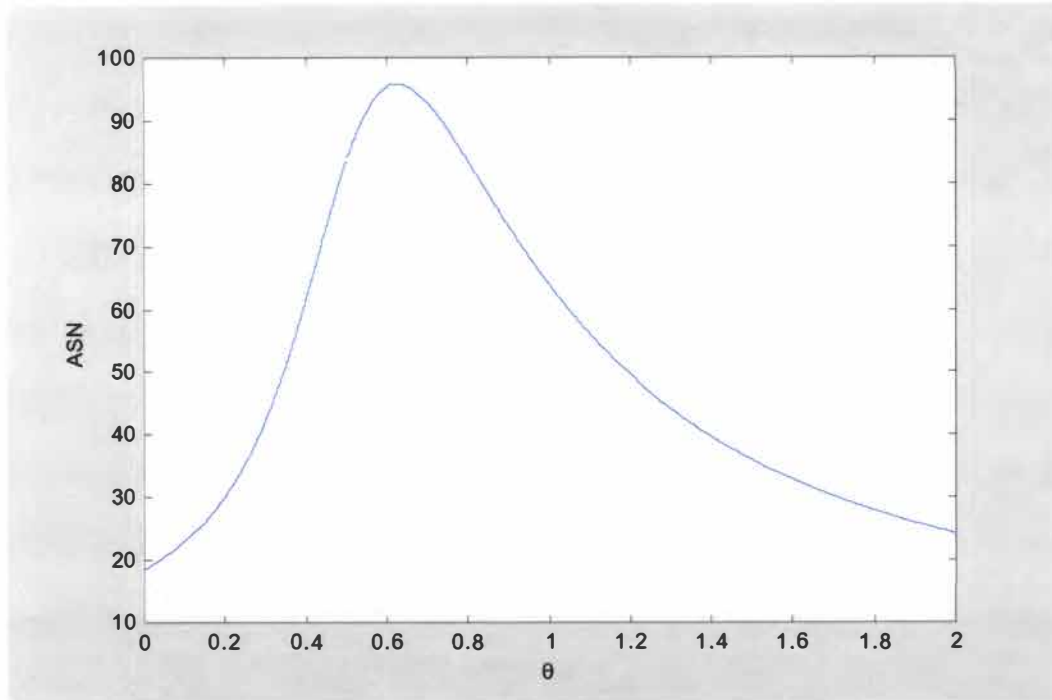


Figure 7: The average sample number (ASN) as a function of θ , for $x \sim N(\theta, \sigma)$, where $H_0 : \theta = \theta_0 = 0$, $H_1 : \theta = \theta_1 = 1$, $\alpha = 0.0001$, $\beta = 0.1$, and $\sigma^2 = 4$.

$$E_{\theta}(n) \sim \frac{OC(\theta) \log B + (1 - OC(\theta)) \log A}{\frac{1}{2\sigma^2} [2(\theta_1 - \theta_0)\theta + \theta_0^2 - \theta_1^2]}$$

The average sample number is lower at θ_0 than it is at θ_1 due to the selection of α and β . If these parameters are set equal to one another, the average sample number would be the same at θ_0 and θ_1 . For the case shown above, the lower limit of the test is $B = -2.3$, whereas the upper limit is $A = 9.1$. Thus, due to the cumulative nature of the test, if the true value of θ is near $H_0 : \theta = \theta_0 = 0$, the number of observations required for a decision is much less than that which would occur if the true value of θ is near $H_1 : \theta = \theta_1 = 1$. The average sample number function provides a guide as to how many

36

observations one could expect, for a defined test, before a fault hypothesis can be decided upon. This is beneficial in preparing drift simulation studies of developed signal validation models.

3.3.5 Application of the SPRT to Signal Validation

The precedent for the use of the SPRT for sensor drift and fault detection in ISCV systems has been set through its reported use for these applications in the literature [Erbay 1997, Eryurek 1994, Gross 1991, Humenik 1990, Li 1999, Olvera 1993, Ray 1989, Upadhyaya 1987, Wrest 1996, Xu 2000]. In applying the SPRT to signal validation, it is assumed that the null hypothesis is given by $H_0 : \theta = \theta_0 = 0$. This allows for a simplification of the likelihood ratio to the form:

$$\Lambda_m = \Lambda_{m-1} + \frac{\theta_1}{\sigma^2} \left(x_m - \frac{\theta_1}{2} \right).$$

Acceptance of the null hypothesis indicates that the residuals are more likely to belong to an unfaulted distribution, characteristic of a normal operating condition for the sensor under test, rather than from a faulted distribution. The variance of the distributions is based on the residuals obtained during model development. Figure 8 shows the normal distributions under the null hypothesis and a specific alternative hypothesis, given the specified variance. Rejection of the null hypothesis, acceptance of the alternative hypothesis, indicates that the residuals are more likely to belong to a faulted distribution. The mean of the faulted distribution can be set using a variety of techniques, most often related to the error observed during model development. After

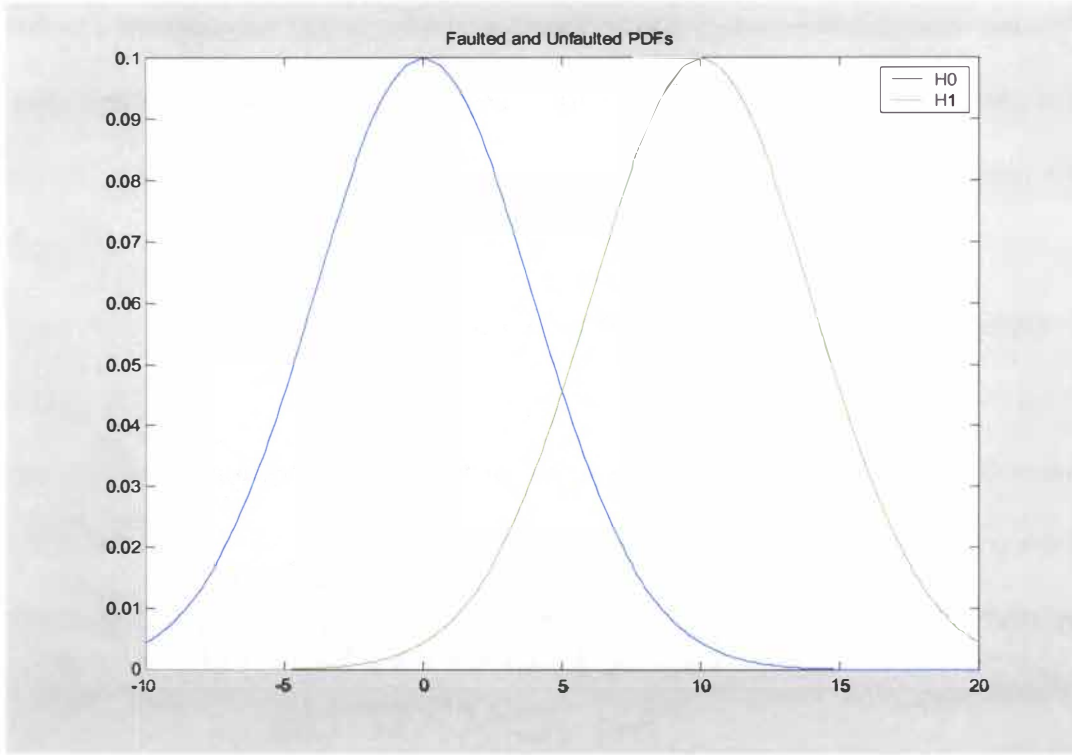


Figure 8: Sample distributions for the null hypothesis, $H_0: \theta = \theta_0$, $x \sim N(\theta_0, \sigma^2)$, $\theta_0 = 0$, $\sigma^2 = 4$, and the alternative hypothesis $H_1: \theta = \theta_1$, $x \sim N(\theta_1, \sigma^2)$, $\theta_1 = 10$, $\sigma^2 = 4$.

setting the variance and the faulted mean, θ_1 , the desired strength of the test must be set. Recall that α is defined as the probability of rejecting H_0 , when H_0 is indeed true. Thus, α is often referred to as the false alarm probability. To increase the confidence of faulted decisions, α is set at 0.01%. Similarly, β is defined as the probability of accepting H_0 , when H_1 is true. Thus, β is often referred to as the missed alarm probability. Due to the presence of noise in sensor calibration monitoring, β is set at 10% to reduce the possibility of erroneous residuals resulting in the acceptance of the fault hypothesis, H_1 .

Signal validation requires continuous processing of residuals. The SPRT decision rule states that acceptance or rejection of the null hypothesis shall result in termination of the test. To allow for decisions to be made about the distribution of residuals under operating condition changes through time, following acceptance of either hypothesis the SPRT is reinitialized to zero and a subsequent test is begun.

3.3.6 Alarm Logic

The presence of signal noise adds further difficulties to the decision making process. In general, the acceptance of the faulted hypothesis at any time is not a reliable indication of the calibration status of a sensor. For this reason, engineering judgment must be applied to determine how many times, within a specified period of time, the faulted hypothesis should be accepted before the sensor in question can be confirmed to be out of calibration. This is a user selectable heuristic that will affect the sensitivity of the signal validation system. In this work, the number of required fault hypotheses is 3, within a time frame of 60 minutes. This heuristic has been determined to be suitable for the procurement of the desired drift detection capabilities warranted by the current signal validation model.

3.4 Multiple Linear Regression

Multiple linear regression is one of the simplest and most common empirical modeling techniques. In addition, the derivation of the MLR solution is straight forward, and easily highlights the resultant complications in solving for the model parameters under conditions of collinear data. A collinear data matrix exhibits near or exact linear dependence between its columns. The arguments regarding the complications related to empirical modeling with collinear data have been extended to Kramer's [1992] autoassociative artificial neural network (AANN) architecture, which has been used extensively at the University of Tennessee [Upadhyaya 1992, Xu 1999], and by other researchers [Fantoni 1998], for ISCV applications.

In matrix notation the MLR equation can be presented as:

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \boldsymbol{\varepsilon}$$

$$\begin{array}{c} \mathbf{y} \\ \left[\begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_n \end{array} \right] \\ (nx1) \end{array} = \begin{array}{c} \mathbf{X} \\ \left[\begin{array}{cccccc} 1 & x_{11} & x_{12} & \cdot & \cdot & \cdot & x_{1m} \\ 1 & x_{21} & x_{22} & \cdot & \cdot & \cdot & x_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & x_{n1} & x_{n2} & \cdot & \cdot & \cdot & x_{nm} \end{array} \right] \\ (n \times p) \end{array} \bullet \begin{array}{c} \mathbf{b} \\ \left[\begin{array}{c} b_0 \\ b_1 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{array} \right] \\ (px1) \end{array} + \begin{array}{c} \boldsymbol{\varepsilon} \\ \left[\begin{array}{c} \varepsilon_1 \\ \varepsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon_n \end{array} \right] \\ (nx1) \end{array}$$

The least squares solution is found by minimizing the sum of squares of the residuals, with respect to the regression coefficient vector $\mathbf{b}$:

$$\min|\boldsymbol{\varepsilon}| = \min \sum_{i=1}^n \varepsilon_i^2$$

The ϵ_i 's are assumed independent, normally distributed random variables with a mean of zero, and a constant variance.

$$|\epsilon| = (y - Xb)^T (y - Xb)$$

$$\frac{\partial |\epsilon|}{\partial b} = -X^T y + X^T X b$$

Setting equal to zero and solving for b results in the normal equations.

$$X^T X b = X^T Y$$

The solution requiring the solution of p normal equations with p unknowns is thus:

$$b = (X^T X)^{-1} X^T Y$$

Confirmation that the solution is a minimum is obtained by computing the second derivative, and recognizing that the second derivative is a positive semi-definite matrix.

$$\frac{\partial^2 |\epsilon|}{\partial b^2} = 2X^T X$$

Prediction for future independent observations is then obtained through:

$$\hat{y}_i = x_i (X^T X)^{-1} X^T Y$$

The stability of the MLR solution can be related to the variances of the regression coefficients, b , which are given by:

$$\text{var}(b) = \sigma^2 (X^T X)^{-1}$$

Where $\sigma^2 = \text{var}(\epsilon)$.

In viewing the required computations of the MLR solution, and the variance of the regression coefficients, it is obvious that collinear data will result in unstable, and highly variable, estimates for the regression coefficients. Large magnitude regression

coefficients may lead to models which exhibit a greater variance at the output than that which is present at the input. This is an obvious result, since excessively large magnitude regression coefficients will force large changes at the output when only slight changes are presented at the input.

3.5 Collinearity

A matrix X is said to be collinear if the columns in X are approximately or exactly linearly dependent. This can also be stated as the existence of a very small angle between two column vectors of X . Collinearity means that the matrix X will have some dominating types of variability that carry most of the available information [Martens 1989]. A square matrix is said to be singular if there is at least one linear dependency among the rows (or columns) of the matrix. For the case of a singular matrix the determinant is zero: $|X|=0$. The rank of a matrix is defined as the maximum number of linearly independent columns (or rows). Based on this, matrices containing linearly dependent columns are said to be rank-deficient. Another term often applied to collinear, rank-deficient matrices is ill-conditioned.

In relation to MLR, the matrix of concern is $X^T X$. Any linear dependence that exists in a matrix X will be preserved in the new matrix $X^T X$. Situations where the determinant of $X^T X$ is near zero lead to unstable estimates of the regression coefficients which may be unreasonably large or have the wrong sign [Massart 1997].

When the determinant of a matrix is zero its inverse does not exist, and when it is near zero, its inverse contains values of extremely large magnitude. This can be further illustrated by the analogy that inverting a collinear matrix is the multivariate equivalent of dividing a number by zero; thus, leading to the extremely large values in the inverted matrix when near linear dependence is present. Two common measures of collinearity are condition number (CN), and variance inflation factor (VIF).

$$CN = \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} \quad \text{where } \lambda_{\max} \text{ and } \lambda_{\min} \text{ are the largest and smallest eigenvalues of } \mathbf{X}^T \mathbf{X}.$$

The condition number measures the ratio between the variation accounted for by the principal component with the largest variance and the variation of the principal component with the smallest variance.

$$VIF_k = \frac{1}{1 - R_k^2} \quad \text{where } R_k^2 \text{ is the squared correlation coefficient between } \mathbf{x}_k$$

(considered here as the dependent variable) and $\hat{\mathbf{x}}_k$ (the estimation of $\mathbf{x}_k$ produced via regression of the remaining independent variables). Since R_k^2 represents the variation in a single variable $\mathbf{x}_k$ that can be explained by the remaining variables $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{k-1}, \dots, \mathbf{x}_{k+1}, \dots, \mathbf{x}_m\}$, a large value means that the given variable $\mathbf{x}_k$ is almost a linear combination of the other variables [Massart 1997]. VIF measures the increase in variance compared to an orthogonal design and values greater than 5 or 10 are generally considered large, indicating poor estimation properties based on the regression coefficients.

Collinearity in a data set leads to an ill-posed problem that causes inconsistent results when data based models such as MLR or neural networks are used [Hines 1999a]. Often in large scale process systems redundant measurements of critical parameters are obtained. These redundant measurements will exhibit a high linear correlation reduced from a theoretical value of 1 for true redundancy, by instrument error, instrument locations, and the addition of noise through signal processing. Two redundant signals can be simulated

$$x_1 = \sin(x) + f * \text{noise}_1$$

$$x_2 = \sin(x) + f * \text{noise}_2$$

where: x is an $n \times 1$ vector of increasing values in the interval $0 \leq x \leq 2\pi$

noise_1 and noise_2 are $n \times 1$ vectors of random numbers chosen from a normal distribution with mean zero and variance one. The random numbers were then linearly scaled to be between -1 and 1 . f is the fractional noise contribution to the sinusoidal signals.

Figure 9 shows the relationship between the condition number and the determinant of $X^T X$, where $X = [x_1 \ x_2]$, as the fractional noise contribution increases.

When the noise contribution is zero, the condition number is infinite and the determinant is zero. This is the case of exact linear dependence. As fractional noise is added to the vectors in the X matrix, the linear dependence decreases. Along with this decrease, the condition number of the $X^T X$ matrix decreases and the determinant increases. Considering redundant measurements, where one could assume approximate

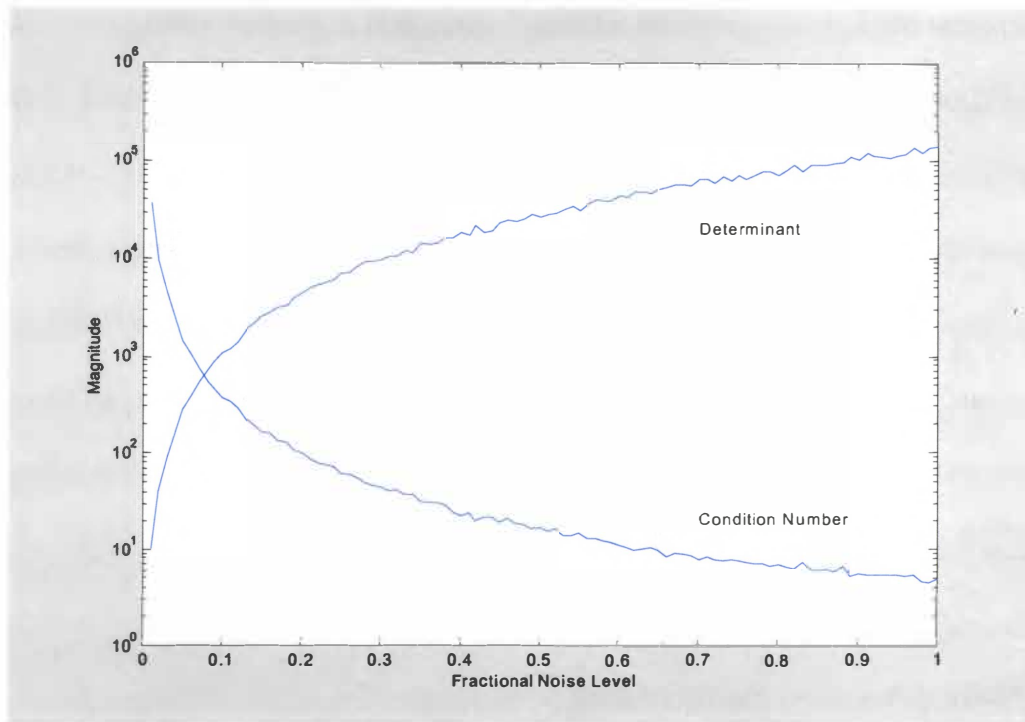


Figure 9: Relationship between the determinant and condition number of a two redundant signal predictor variable matrix with increasing addition of noise.

conditions being analogous to the above example with $f \leq 0.1$ it is apparent that the $\mathbf{X}^T \mathbf{X}$ matrix is ill-conditioned. This will produce multivariate regression models whose regression coefficients will exhibit large variances under conditions of arbitrary data selection for model development.

Data sets from large scale processes such as an electrical generating facility are often plagued with collinearity. Thus, MLR is not a suitable method of empirical modeling for these situations. The use of AANNs is also inhibited by collinearity for similar reasons [Qin 1997]. Some type of regularization method should be utilized when dealing with collinear data. Regularization can be integrated into neural network

development through the use of cross validation training and robust training techniques [Hines 2000, Xu 2000]. Other methods of regularization include truncated singular value decomposition, and ridge regression [Hines 1999b, Hoerl 1970]. Alternate methods of dealing with collinear data are transformation techniques, where the data is transformed to an equivalent set of orthogonal vectors, without any loss of information. Orthogonalization of the data eliminates the problems associated with collinearity. Transformation techniques include Principal Component Analysis (PCA) [Jolliffe 1986], and PLS [Hoskuldsson 1995].

3.6 Neural Network Partial Least Squares

This section describes the features and benefits of the NNPLS architecture. An overview is provided, followed by a description of the standard PLS algorithm. The NNPLS algorithm is then developed and additional discussions about the latent vector selection and alarm parameter settings are presented.

3.6.1 Features and Benefits of the NNPLS Architecture

Many new approaches in signal validation have incorporated regularized modeling strategies to alleviate problems associated with collinearity, as well as to allow for the additional benefits of data compression. Regularized modeling techniques attempt to solve collinearity issues through the introduction of a bias, while orthogonal techniques such as PCA and PLS remove the issues of collinearity by transforming the data to a

new set of orthogonal vectors without the loss of any information. PCA and PLS allow for data compression by using only the most relevant information in the data for the specified model. PCA bases the transformation to relevant information on the $\mathbf{X}^T\mathbf{X}$ matrix, while PLS bases the transformation on the $\mathbf{X}^T\mathbf{Y}$ matrix. For the purposes of inferential modeling, the latter is clearly superior due to its reliance on the covariance between the input data and the desired output, rather than the variance of the input data alone.

The ordering of the orthogonal components in the PLS framework allows for the highest variability, relevant to the desired output, to be obtained through the mapping of the first few orthogonal components of any given model. The higher ordered components will contain the noise that was present in the data, and thus this noise will not contribute to the final model. This also leads to a noise reduction in the NNPLS predictions, when compared to the noise present in the measured data.

The inferential design reduces the sensitivity of predictions to perturbations of the same signal at the input, due to the feature of omitting a given signal's measurement from directly influencing its own prediction. The move from fully connected AANNs to single input single output neural networks increases the stability of the neural network weight solutions due to the small number of parameters being optimized at any one time. Additional regularization features are incorporated into the NNPLS architecture through the use of cross validation training of the neural networks in the inner relationships of the model.

3.6.2 NNPLS Modeling Overview

The overall NNPLS based ISCV system design consists of a set of inferential models. Each parameter to be monitored requires a separate inferential model. An inferential model infers a prediction of a specific sensor's measurement based on correlated sensors.

The primary needs of this architecture are that the data are strongly related, and as in all modeling techniques, the data used to develop the inferential models must be representative of the full range of the signals under normal operating conditions.

[Hines 2000, 2001a, 2001b, Rasmussen 2000a, 2000b]

The sensor values that are used as inputs to each inferential model include all sensor values except for those of the specific sensor being modeled. A schematic of a NNPLS inferential model is provided in figure 10.

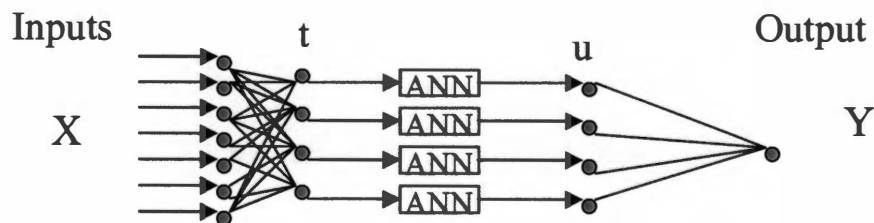


Figure 10: NNPLS inferential unit schematic.

The theoretical basis for PLS is explained in detail in many publications and texts [Geladi 1986, Hoskuldsson 1995, Martens 1989]. Data is scaled to have a mean of zero, and a standard deviation of one prior to its introduction to the PLS algorithm. PLS is an iterative algorithm that sequentially decomposes the input data set into orthogonal score vectors. The output data set is also decomposed into a new set of vectors, though orthogonality is not a constraint. The transformations of the input data set and output data set to their respective score vectors, t and u , are derived such that the first set of score vectors explains the maximum covariance between the input data set (X) and output data set (Y). The standard PLS algorithm then prescribes a linear regression of the input score vector onto the output score vector. The variability in the input data set explained by the first input score is then subtracted from the input data set, resulting in an input residual. Similarly, the variability in the output data set explained by the first output score is subtracted from the output data set, resulting in an output residual. The input and output residuals represent the remaining variability, from which further information can be extracted, about the covariance between the input and output data sets. The second set of score vectors is then determined such that the maximum variance in the input residual related to the output residual is captured, with the additional constraint of orthogonality with the previous input score vector. After the second set of score vectors is obtained, the input and output residuals are again reduced by the variability explained by the second set of score vectors. This iterative process continues until the information left in the residual matrices is negligible, and can be attributed to noise, or unrelated variation with respect to the desired output. Based on

the desired accuracy of the model, and the level of noise present in the data, the number of input score vectors to be included in the model can be determined.

3.6.3 PLS Algorithm Description

The PLS algorithm is capable of mapping a set of predictor variables to a set of response variables. When there is only one response variable, the standard PLS algorithm can be reduced to an algorithm referred to as PLS-1 [Gil & Romera 1998]. Data is scaled to have a mean of zero, and a standard deviation of one prior to its introduction to the PLS-1 algorithm. The linear PLS-1 algorithm is presented below, followed by a diagram and a discussion of the equations involved at each step:

$a = 1$	Initialize iteration index
$E_0 = X$	Input residual matrix
$f_0 = y$	Output residual vector
$w_a^* = E_{a-1}^T f_{a-1}$	Calculation of transformation weights
$w_a = w_a^* / \ w_a^*\ $	Normalization
$t_a = E_{a-1} w_a$	Calculation of input scores
$p_a^T = t_a^T E_{a-1} / t_a^T t_a$	Calculation of input loadings
$b_a = t_a^T f_{a-1} / t_a^T t_a$	Inner relationship regression
$f_a = f_{a-1} - t_a b_a$	Calculation of output residual vector

$$E_a = E_{a-1} - t_a p_a^T$$

Calculation of input residual matrix

if $a < A$

Update iteration index

$$a = a + 1$$

goto 4

else

end

Nomenclature:

$a =$ Iteration index

$m =$ The number of data samples

$n =$ The number of predictor variables

$A =$ The optimal number of score vectors, or rank of the model

$X = (mxn)$ Matrix of predictor variables

$y = (mx1)$ Response variable

$w_a = (nx1)$ The weight vector used to calculate the a^{th} score vector

$t_a = (mx1)$ The a^{th} score vector

$p_a = (nx1)$ The a^{th} input loading vector

$b_a = (1x1)$ The a^{th} regression coefficient of the inner relationship

$E_a = (mxn)$ The a^{th} input residual matrix

$f_a = (mx1)$ The a^{th} output residual vector

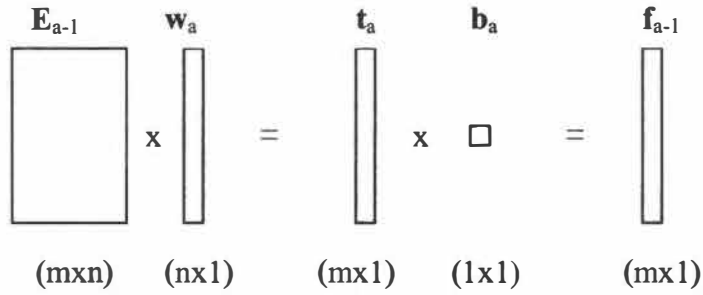


Figure 11: Geometric representation of the relationship between the input and output residual matrices in the PLS-1 algorithm.

The relationships between the input residual matrix and output residual vector are diagrammed in figure 11, and the steps of the algorithm are further explained below.

Steps 1-3: Initialize the dimension (a) to 1, the input residual matrix to X and the output residual vector to y .

Steps 4-6: The input score vectors (t_a) contain the maximum covariance between the variables in the input residual matrix and the output residual vector. A method to determine the weight vectors w_a , such that the maximum covariance is contained in the score vectors is by solution of the eigenvalue problem:

$$E_{a-1}^T f_{a-1} f_{a-1}^T E_{a-1} w_a = \lambda_{a-1} w_a$$

For $a = 1$ this reduces to:

$$X^T y y^T X w_1 = \lambda w_1$$

The first score vector is then:

$$t_1 = X w_1$$

The loading vector is then determined by a linear regression of the input residual matrix onto the score vector, which for $a = 1$ is equivalent to:

$$p_1^T = t_1^T X / t_1^T t_1,$$

For $a > 1$ the loading vectors are calculated by step 6 of the PLS-1 algorithm.

Step 7: This is referred to as the inner relation. In this step the output residual vector is regressed onto the input score vector. For $a = 1$ this is equivalent to:

$$b_1 = t_1^T y / t_1^T t_1$$

Steps 8-9: After the first inner and outer relations are constructed, the variance of the input data accounted for in the first score vector is given by $t_1 p_1^T$, and the related variation in the response variable is given by $t_1 b_1$. This information is then removed from the input and output data resulting in the residual matrices as shown in steps 8 and 9 of the PLS-1 algorithm (for $a = 1$, $f_{a-1} = y$, $E_{a-1} = X$).

Step 10: It is difficult to determine the optimal dimension of the model without some measure of the model's performance at each dimension. As the model is iteratively constructed, a measure of the model's performance is determined for each dimension. A common performance measure is the sum squared error (SSE), and is computed with respect to an independent set of data that was not used during model development. The optimal dimension is then defined as the dimension at which the SSE was at its minimum. For large models, a bound is usually set for the maximum dimension, and all dimensions are computed up to the maximum.

3.6.4 NNPLS Algorithm Description

The NNPLS algorithm is the same as the PLS-1 algorithm, except that rather than use a simple linear regression in the inner relation, a set of SISO-ANNs are used. In the linear PLS-1 algorithm, the inner relationship is: $f_{a-1} = t_a b_a$. The PLS-1 algorithm can be extended to a non-linear algorithm, referred to as NNPLS-1, by replacing steps 7 and 8 with:

NNPLS-1 Step 7: $TRAIN\{N_a(t_a, f_{a-1})\}$ Train an ANN to map the inner relationship

Step 8: $f_a = f_{a-1} - N(t_a)$ Calculation of output residual vector

Where N_a represents a neural network mapping of the a^{th} inner relationship. The neural networks used in the inner relation contain a single hidden layer with two neurons containing hyperbolic tangent activation functions, and a single output neuron with a linear activation function. While there are a variety of training algorithms that can be applied, the one implemented is the scaled conjugate gradient training algorithm [Moller 1993]. A diagram of the NNPLS-1 inner relation for a single set of score vectors is shown in figure 12.

Each score vector (t_a) requires a SISO-ANN as shown above. The neural network weights and biases for each set of score vectors are determined during the current iteration (step 7 of NNPLS-1), and the explained variation is subtracted from the input residual via network simulation (step 8 of NNPLS-1). The decision to use two hidden

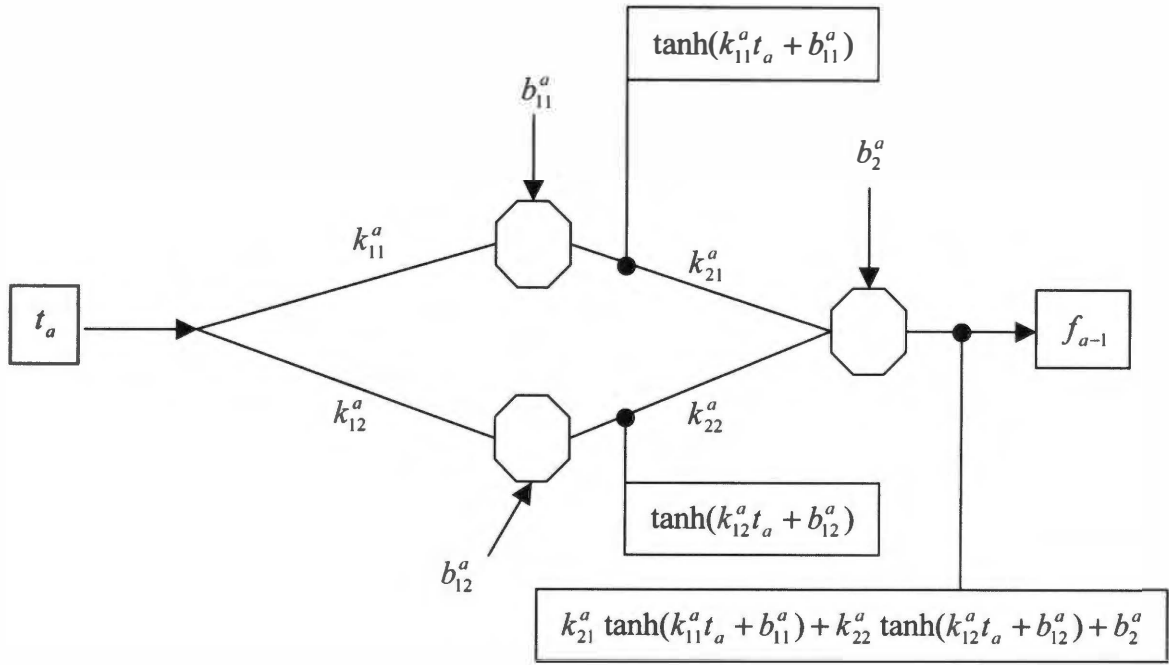


Figure 12: Schematic of the SISO-ANN architecture embedded in the inner relationships of the NNPLS architecture.

neurons was to provide the neural networks with a significant number of degrees of freedom without increasing the complexity of the model beyond that which is required. Through experimentation, it was noted that two hidden neurons often provided a better solution than a single hidden neuron, and greater than two hidden neurons did not result in improvements great enough to warrant their inclusion.

The NNPLS-1 algorithm is designed to accept any number of input variables to construct a model for a single response variable. For the purposes of ISCV, the required architecture is autoassociative, such that for a given set of input variables, the model provides a prediction at the output for each variable in the input set. For each response variable a separate inferential model is developed. The response variable for a given model is omitted from the input set to avoid the effects of spurious values of a

given variable on its own response. Thus, the NNPLS-1 model infers a response variable's value based on the values of the related predictor variables. The complete model for a single response variable will be referred to as an inferential unit. An inferential unit consists of a specific set of predictor variables, transformation weights, and neural network weights. The other variables associated with the NNPLS-1 algorithm are required only during model development, and are unnecessary for prediction. The required parameters of an inferential unit for prediction of the j^{th} response variable are given as:

W_j = matrix of input transformation weights

N_j = the neural network weights for the inner relationship

$$= [k_{11}^a \ k_{12}^a \ k_{21}^a \ k_{22}^a \ b_{11}^a \ b_{12}^a \ b_2^a]$$

The prediction from the j^{th} inferential unit is given by:

$$\hat{y}_j = \sum_{a=1}^A [k_{21}^a \tanh(k_{11}^a t_a^j + b_{11}^a) + k_{22}^a \tanh(k_{12}^a t_a^j + b_{12}^a) + b_2^a]$$

The inclusion of all available variables in the input set, except for the particular response variable, requires no oversight regarding variable selection. While this method works well for the majority of the response variables used in this study, there are instances where selective omission of input variables improves the model's prediction error. Recent publications have indicated that the performance of the PLS algorithm may be degraded when inputs are included in the model which are not relevant to the

desired output [Alsberg 1998, Lindgren 1994, Osborne 1997, Woodward 1996]. The derivation of the PLS outer transformation ensures that the score vectors contain the maximum covariance; however, the possibility of random fluctuations in the data resulting in a biased covariance must be considered. The PLS algorithm may become unreliable if a significant number of peculiarities exist in the data such that the estimated covariance values are strongly biased towards these peculiarities. Peculiarities in the data may be due to instrument noise, instrument error, or chance correlations in generally uncorrelated variables [Woodward 1996]. Similarly, Seasholtz & Kowalski [1993] have indicated that the ability of the algorithm's outer transformation to determine the input variables, that are most highly correlated with the output variables, is limited.

Drift detection relies on the difference between the ISCV process model's inputs and outputs. In the NNPLS design, the inferential model for a given parameter does not include that parameter as an input to the model. A drift in a specific parameter, that does not produce a similar drift in its correlated parameters, will have no effect on the NNPLS design's prediction of that parameter; therefore, an isolated parameter drift can be easily detected.

3.6.5 Method of Determination of the maximum number of latent vectors

The maximum number of latent vectors is determined based on a heuristic dependent only on the number of inputs available in a given inferential model. The heuristic is defined below:

$p =$ total number of signals, note that the number of inputs to any inferential model will be $p - 1$.

$mf =$ the maximum number of latent vectors to retain in an inferential model.

$$\left\{ \begin{array}{l} p \leq 20, mf = INT(0.7 * p) \\ 20 < p \leq 40, mf = INT(0.6 * p) \\ 40 < p \leq 70, mf = INT(0.5 * p) \\ 70 < p \leq 100, mf = INT(0.4 * p) \\ p > 100, mf = INT(0.3 * p) \end{array} \right\}$$

INT rounds the product to the nearest integer. The initial settings for the determination of mf are set relatively high intentionally. The NNPLS algorithm in most cases will use only a fraction of this number as the optimal number of latent vectors. The optimal number is found based on cross validation of each possible number of latent vectors up to mf . The initial high estimates were meant to accommodate a wide variety of data types with nonlinear relationships and varying noise levels.

3.6.6 SPRT Parameter and Alarm Logic Setting

There are 4 parameters which must be set to define a SPRT for a given residual: The false alarm probability, the missed alarm probability, the degradation mean, and the

variance of the faulted distribution. Two additional settings are also available to implement sensor alarm logic.

The false alarm probability is set at $\alpha = 0.01\%$, and the missed alarm probability is set at $\beta = 10\%$. All of the signals in a given model will be subject to the same SPRT probabilities.

The alarm logic parameters are used to control the rate at which signal alarms are produced. An alarm is produced if the "number of alarms in increment to produce true alarm" is exceeded within the specified "increment". The default settings are 3 alarms in a 60 sample increment. This is set based on the current data which is sampled in one minute increments; thus, if the degradation hypothesis is accepted three times within an hour, an alarm will be initiated.

The variance and degradation mean parameters are set within the development interface. The variance is computed as $9\sigma_r^2$, where σ_r^2 is the variance of the post-scaled residual. Note that a variance multiple of 9 has been inserted to slow the rate of increase of the SPRT decision. The degradation mean is set at 5% of the mean value of a given signal. The variance controls the rate of increase or decrease of the SPRT decision, and the degradation mean positions the normal distribution of the degradation hypothesis. Adjustment of these parameters may be required if the alarm rate is not

performing well. These changes can be made in conjunction with adjustments to the alarm logic and false and missed alarm probability settings.

4.0 NNPLS ISCV GRAPHICAL USER INTERFACES

The development interface, monitoring interface, and the structured query language (SQL) connection interface have been created using National Instrument's LabView software with embedded C++ programs stored in dynamic link libraries. Each of the interfaces can be packaged as an executable file that can be run from any modern PC without the need for software licenses.

The NNPLS algorithm presents optimal performance when the data provided exhibit a high degree of linear correlation. Redundant information, or nearly redundant information, is useful and enables the algorithm to utilize its signal averaging capabilities.

It is recommended that the minimum number of sensors be 10. While no maximum number is definable, the time and development issues become more significant as the number of sensors exceeds 50. The number of data points and the integrity of these data are directly related to the performance of the developed models. These data should adequately represent the normal operating conditions of the sensors, and should span their entire operating ranges. Recommended quantities of data for normal process sensors are 500 to 5000 data points. Note that all sensors in the data file must have the same number of data samples.

The data to be evaluated with the monitoring interface should be consistent with that used to develop the embedded NNPLS model. The embedded model is written to a specific directory, by the development interface, from which the monitoring interface will load the model parameters. A full run of the development interface is required before the monitoring interface can be used. The following sections briefly introduce each of three main interfaces of the ISCV system. Full detailed descriptions are provided in the appendices.

4.1 NNPLS Development Interface

The development interface creates a NNPLS ISCV model that can later be used in the monitoring module. The inputs to the development interface include files that describe the data names, data units of measure, and the data used to develop the model. The development module outputs the model and files containing data necessary for proper monitoring and alarming such as noise levels and drift detection thresholds. The methodology embedded within the development interface follows the theoretical basis explained previously to construct ISCV models. All of the necessary parameters for a complete NNPLS ISCV model are computed and stored. These parameters include the NNPLS transformation weights, the neural network weights for the inner relationship models, and the SPRT mean and variance values. Additionally, appropriate statistics of the data, and the selected median filter properties are recorded for data scaling and filtering. These parameters are stored in a specified location that is accessed by the NNPLS monitoring interface during instrument surveillance activities.

The required parameters for model development consist of a set of training data, and a set of validation data. The file format is specified as a tab delimited ASCII text file. Additional controls are provided to adjust the median filter rank applied to both data sets, and an option is available to save the resultant model parameters to a file. In addition to building complete models for a data set, individual models can be updated, assuming that a full model containing the specific sensor model to be updated already exists. All of these controls are selectable from the front panel of the development interface. The development interface has a complementing monitoring interface that can be used to assess the performance of the developed model on a set of test data during process operation. A simple schematic of the development interface is shown below in figure 13.

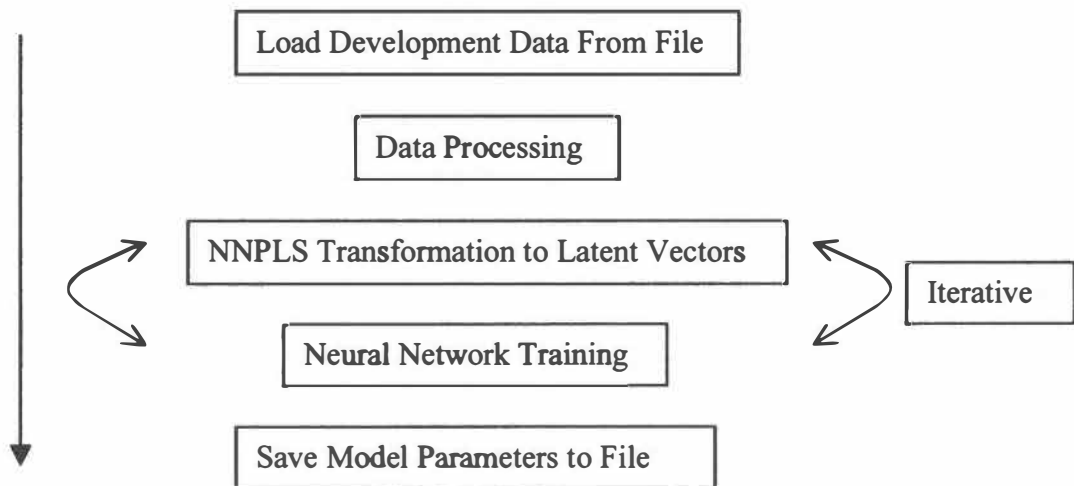


Figure 13: Simplified flow diagram of the development interface.

The major tasks performed in the development phase are:

1) Load Development Data From File - The training and validation data for the development interface must be stored in a file, and be in ASCII tab delimited spreadsheet format. The names and paths of the files will be entered and the number of sensors represented in the data files along with the number of data points in the files will be echoed to the screen when loaded. This provides a method to verify the correct data files were properly loaded.

2) Data Processing – The data pre-processing methods consist of data scaling to produce training data with a zero mean and a variance of one. Additionally, the data may be median filtered based on the user selectable median filter size. This is useful in removing outliers and excessive noise.

3) Iterative PLS Transformation to Latent Vectors and Neural Network Training – These calculations are performed using C++ programs which implement the NNPLS algorithm.

4) Save Model Parameters to File – This allows for the transferal of the new model parameters to the monitoring interface.

In addition to the controls, the development interface also displays the numerical data read from the specified data files along with the mean and standard deviation of the training data. A graphical display is also provided with controls to select

1. The data to be plotted:
 - training data
 - training prediction
 - training error
 - training data and prediction
 - validation data
 - validation prediction
 - validation error
 - validation data and prediction
2. The sensor to be plotted
3. The data interval to be plotted.

These plots are useful for presenting a graphical image of the model performance. The front panel of the development interface is shown on the following page.

The front panel of the development interface (figure 14) displays the applicable user controls and indicators. The use of the development interface is explained in detail via a tutorial included in appendix A.

ON **Nonlinear Partial Least Squares Model Building**

DATA PREPROCESSING

TRAINING DATA
 TRAINING DATA FILE

VALIDATION DATA
 VALIDATION DATA FILE

SIGNALS

SIGNAL POINT

POINTS

	96.68	97.72	98.54	99.17	99.53	MEAN	STD
	269.46	271.09	272.41	273.63	274.68	69.71	24.85
	528.45	535.24	541.32	547.00	552.07	223.10	44.69
	51.74	51.83	51.92	51.97	51.98	358.65	141.20
	74.55	74.58	74.51	74.44	74.39	48.84	2.62
						63.06	11.80

SIGNAL POINT

POINTS 2

	56.85	58.04	59.23	60.09	60.95
	207.63	209.31	210.99	212.23	213.47
	286.64	291.15	295.65	299.14	302.62
	48.00	48.11	48.21	48.21	48.21
	58.41	58.75	59.08	59.27	59.47

NNPLS MODEL BUILDING

UPDATING ☐ BUILDING ☒

FILTER

 FILTER WINDOW SIZE

CREATE MODELS
 SIGNAL TO MODEL (0 - ALL)

SAVE MODEL

☒ SAVE

PROGRESS INDICATOR

# OF SENSORS	15
TRAINING POINTS	0
VALIDATION POINTS	0
MODELING SIGNAL	1

PLOT CONTROL

 START POINT SENSOR #
 POINTS TO PLOT ALL ☐
 TRAIN MAE VAL MAE # COMPONENTS

Figure 14: Development interface front panel.

4.2 SQL Connection Interface

The SQL connection interface has been designed to work with the development and monitoring modules of the ISCV system. The creation of the SQL connection interface has been facilitated by National Instrument's SQL toolkit, which provides a bridge between the G programming language of LabView and an ODBC compliant database.

The use of the SQL connection interface is not required. If monitoring tasks will be implemented on historical data, all functions can be carried out directly from the monitoring interface.

The SQL connection interface enables on-line monitoring from a server database. The required inputs to the SQL connection interface are two data files containing signal tag names, a set of database parameters, a directory name for data storage, and a sampling rate. All of these settings are accessible from the front panel of the interface. The interface maintains a continuous connection to the server during operation. A single SQL SELECT statement, and a single SQL INSERT statement are executed upon each iteration at a sampling rate that is user definable. The SELECT statement retrieves the necessary data, and after processing and validating the data, the validated data is written back to the server via the INSERT statement. The SQL connection interface handles data archiving tasks as well, storing data files in a designated directory with filenames according to the contents of the file and the appropriate date. The front panel of the SQL connection interface (figure 15) displays the applicable user controls and

SQL Connection Interface

Connection Status

RUN

Sampling Rate (milliseconds)

60000

Iteration

11

Successful Connect

Successful Disconnect

Database Parameters

Data Source Name

foxboro snapshot

Table Name

foxsnapshot

Tag Column Name

tag

Value Column Name

value

Tag Name and Data Storage

Full Tag List for Archive

%C:\Pls_d\alltags2.txt

Monitored Tag List

%C:\Pls_d\tags6.txt

Data Directory

c:\pls_dll\data_storage

Signal Tag Names and Data

	Data Tags	Verified Tags	Measurements	Predictions
0	9BM:B0.PNT	9BMB0PNT	1016.17	1025.11
0	9BM:B0A.PNT	9BMB0APNT	1016.51	1022.65
	9DSAI2:B16.PNT	9DSAI2B16PNT	1098.22	1107.22
	9DSAI2:B33.PNT	9DSAI2B33PNT	514.97	499.27
	9DSAI2:B34.PNT	9DSAI2B34PNT	180.27	182.25
	9DSAI2:B35.PNT	9DSAI2B35PNT	74.14	72.23
	9DSAI2:B36.PNT	9DSAI2B36PNT	48.48	50.24
	9DSAI3:B2.PNT	9DSAI3B2PNT	108.96	111.95
	9DSAI3:B3.PNT	9DSAI3B3PNT	167.04	162.18
	9DSAI3:B36.PNT	9DSAI3B36PNT	263.84	255.94
	9DSAI3:B38.PNT	9DSAI3B38PNT	244.20	233.10
	9DSAI3:B39.PNT	9DSAI3B39PNT	467.26	458.67
	9DSAI3:B4.PNT	9DSAI3B4PNT	191.55	201.24
	9DSAI3:B42.PNT	9DSAI3B42PNT	338.19	337.54
	9DSAI3:B43.PNT	9DSAI3B43PNT	193.10	187.14
	9DSAI3:B44.PNT	9DSAI3B44PNT	169.07	172.33
	9DSAI3:B45.PNT	9DSAI3B45PNT	178.85	185.60

SQL Statement Indicators

SQL Read Statement

SELECT value FROM foxsnapshot WHERE tag IN ('9AFAI1:B10.PNT','9AFAI1:B11.PNT','9AFAI1:B12.PNT','9AFAI1:B16.PNT','9AFAI1:B19.PNT','9AFAI1:B6.PNT','9AFAI1:B8.PNT','9BM:B0.PNT','9BM:B0A.PNT','9CBAI1:B23.PNT','9CBAI1:B24.PNT','9CBAI1:B25.PNT','9CBAI1:B26.PNT','9CBAI1:B27.PNT','9CBAI1:B28.PNT','9CBAI1:B29.PNT','9CBAI1:B30.PNT','9CBAI1:B31.PNT','9CBAI1:B32.PNT','9CBAI1:B33.PNT','9CBAI1:B34.PNT','9CBAI1:B35.PNT','9CBAI1:B36.PNT','9CBAI1:B37.PNT','9CBAI1:B38.PNT','9CBAI1:B39.PNT','9CBAI1:B4.PNT','9CBAI1:B40.PNT','9CBAI1:B41.PNT','9CBAI1:B42.PNT','9CBAI1:B43.PNT','9CBAI1:B44.PNT','9CBAI1:B45.PNT','9CBAI1:B46.PNT','9CBAI1:B47.PNT','9CBAI1:B48.PNT','9CBAI1:B49.PNT','9CBAI1:B50.PNT','9CBAI1:B51.PNT','9CBAI1:B52.PNT','9CBAI1:B53.PNT','9CBAI1:B54.PNT','9CBAI1:B55.PNT','9CBAI1:B56.PNT','9CBAI1:B57.PNT','9CBAI1:B58.PNT','9CBAI1:B59.PNT','9CBAI1:B60.PNT','9CBAI1:B61.PNT','9CBAI1:B62.PNT','9CBAI1:B63.PNT','9CBAI1:B64.PNT','9CBAI1:B65.PNT','9CBAI1:B66.PNT','9CBAI1:B67.PNT','9CBAI1:B68.PNT','9CBAI1:B69.PNT','9CBAI1:B70.PNT','9CBAI1:B71.PNT','9CBAI1:B72.PNT','9CBAI1:B73.PNT','9CBAI1:B74.PNT','9CBAI1:B75.PNT','9CBAI1:B76.PNT','9CBAI1:B77.PNT','9CBAI1:B78.PNT','9CBAI1:B79.PNT','9CBAI1:B80.PNT','9CBAI1:B81.PNT','9CBAI1:B82.PNT','9CBAI1:B83.PNT','9CBAI1:B84.PNT','9CBAI1:B85.PNT','9CBAI1:B86.PNT','9CBAI1:B87.PNT','9CBAI1:B88.PNT','9CBAI1:B89.PNT','9CBAI1:B90.PNT','9CBAI1:B91.PNT','9CBAI1:B92.PNT','9CBAI1:B93.PNT','9CBAI1:B94.PNT','9CBAI1:B95.PNT','9CBAI1:B96.PNT','9CBAI1:B97.PNT','9CBAI1:B98.PNT','9CBAI1:B99.PNT','9CBAI1:B100.PNT')

SQL Insert Statement

INSERT INTO foxsnapshot (tag,value) VALUES ('9BMB0PNT',1025.11),('9BMB0APNT',1022.65),('9DSAI2B16PNT',1107.22),('9DSAI2B33PNT',499.27),('9DSAI2B34PNT',182.25),('9DSAI2B35PNT',72.23),('9DSAI2B36PNT',50.24),('9DSAI3B3PNT',111.95),('9DSAI3B36PNT',255.94),('9DSAI3B38PNT',233.10),('9DSAI3B39PNT',458.67),('9DSAI3B4PNT',201.24),('9DSAI3B42PNT',337.54),('9DSAI3B43PNT',187.14),('9DSAI3B44PNT',172.33),('9DSAI3B45PNT',185.60)

Error Management

Error Out After CONNECT

status

no error

code

0

text

Error Out After SELECT

status

no error

code

0

text

Error Out After INSERT

status

no error

code

0

text

Error Out After DISCONNECT

status

no error

code

0

text

Figure 15: SQL interface front panel.

68

indicators. The use of the SQL connection interface is explained in detail via a tutorial included in Appendix B.

4.3 NNPLS Monitoring Interface

The NNPLS monitoring interface can be utilized after an ISCV model has been created and stored through the use of the development interface. The monitoring interface can retrieve data from a file; or can be linked to a server using the SQL connection interface. Median filtering and data scaling (mean centered and unit variance) are then employed for data processing. The processed data is then passed to the prediction model, which outputs estimations of this data. The sequential probability ratio test is used to determine the operational status of the sensors, and alarms are reported by boolean indicators on the front panel. Sensor replacement is initiated and maintained for any sensor reporting an alarm. The results are presented in graphical and tabulated formats on the front panel of the interface (figure 16). All of the necessary model parameters are read from the files created by the development interface. The use of the NNPLS monitoring interface is explained in detail via a tutorial included in Appendix C.

Nonlinear Partial Least Squares Monitoring Module



Figure 16: Monitoring interface front panel.

5.0 RESULTS OF APPLICATIONS TO THE FOSSIL PLANT

The final NNPLS-ISCV model for which results are reported contains 51 signals, as listed in table 3. These signals were selected based on their relevance to the calculations made by the PMW of the 9th unit of the Kingston plant. Additional considerations based on the relationships between signals, and the consistent output of the various measurement devices also assisted in the determination of this set of signals. The prediction performance results are presented first, followed by an example of the drift detection performance of the system. Drift detection was consistently within 2-5% of the mean value of the signals. This indicates that drifting or failed sensors will be identified after the signal measurement and the ISCV systems prediction deviate by more than 2-5% of the signal mean. In an online test, the system properly identified a failed sensor reporting measurements at a level greater than 5% of its mean signal value.

5.1 Prediction Performance

The signal descriptions, units, mean absolute error (MAE), mean values, and error values are displayed below. The error values are calculated as $(MAE/Mean)*100\%$. Absolute error is the absolute difference between a measurement and its corresponding prediction.

Table 3: NNPLS Prediction Performance Results

	Description	Unit	MAE	Mean	(MAE/Mean)*100
1	First Stage Pressure A	PSIG	17.0317	1154.40	1.48
2	First Stage Pressure B	PSIG	15.1	1154.50	1.31
3	Measured Steam Flow - Raw	KLB/HR	11.6143	1206.20	0.96
4	FW Heater 1 Extraction Pressure	PSIG	5.4476	576.46	0.95
5	FW Heater 3 Extraction Pressure	PSIG	3.4901	204.13	1.71
6	FW Heater 4 Extraction Pressure	PSIG	0.9058	83.03	1.09
7	FW Heater 5 Ext Press - Abs	PSIA	0.5137	53.08	0.97
8	FW Heater 8 Water In Temp	DegF	1.0748	118.34	0.91
9	FW Heater 8 Water Out Temp	DegF	0.6068	171.35	0.35
10	FW Heater 5 Drain Temperature	DegF	4.0645	254.01	1.60
11	FW Heater 6 Drain Temperature	DegF	0.7877	248.82	0.32
12	FW Heater 6 Extraction Temp A	DegF	4.2992	450.93	0.95
13	FW Heater 7 Water Out Temp	DegF	0.6713	195.57	0.34
14	FW Heater 7 Extraction Temp A	DegF	3.9738	320.75	1.24
15	FW Heater 7 Drain Temperature	DegF	1.4883	196.35	0.76
16	FW Heater 8 Drain Temperature	DegF	0.5563	173.34	0.32
17	FW Heater 8 Extraction Temp A	DegF	4.6754	190.94	2.45
18	FW Heater 6 Water Out Temp	DegF	0.7858	243.73	0.32
19	FW Heater 5 Water Out Temp	DegF	0.7035	261.67	0.27
20	Feedwater Flow #1 - Raw	KLB/HR	18.1922	1248.40	1.46
21	Feedwater Flow #2 - Raw	KLB/HR	18.0916	1258.90	1.44
22	Selected Drum Pressure	PSIG	9.0942	1918.70	0.47
23	FW Heater 3 Water In Temp	DegF	23.4568	248.37	9.44
24	FW Heater 2 Water In Temp	DegF	4.8889	367.66	1.33
25	FW Heater 1 Water In Temp	DegF	4.6021	405.19	1.14
26	FW Heater 1 Water Out Temp	DegF	5.3092	447.70	1.19
27	FW Heater 1 Drain Temperature	DegF	9.0778	416.76	2.18
28	FW Heater 2 Drain Temperature	DegF	11.9921	396.30	3.03
29	FW Heater 3 Drain Temperature	DegF	3.1733	368.29	0.86
30	FW Heater 1 Extraction Temp	DegF	5.5081	733.43	0.75
31	FW Heater 3 Extraction Temp	DegF	5.1578	873.71	0.59
32	FW Heater 4 Extraction Temp	DegF	6.002	705.40	0.85
33	Cold Reheat Pressure at Turbine	PSIG	6.626	350.14	1.89
34	Deaerator Pressure	PSIG	5.7358	72.61	7.90
35	FW Heater 2 Extraction Pressure	PSIG	10.3346	320.76	3.22
36	Cold Reheat Temp at Turbine	DegF	2.5448	626.52	0.41
37	Condensate Flow - Raw	KLB/HR	39.7181	943.90	4.21
38	Station Service Load	MW	0.2397	9.37	2.56
39	Hot Reheat Pressure at Turbine	PSIG	5.2743	318.80	1.65
40	Reheat Outlet Temperature #1	DegF	4.5456	978.15	0.46
41	Reheat Outlet Temperature #2	DegF	4.92	978.56	0.50
42	Superheat Outlet Temperature #2	DegF	3.6999	998.71	0.37
43	Superheat Outlet Temperature #1	DegF	4.4785	999.06	0.45
44	RH Outlet Header Temperature A	DegF	13.7757	981.14	1.40
45	Hot Reheat Temp at Int Vlv A	DegF	5.477	977.77	0.56

Table 3: continued

46	Hot Reheat Temp at Int Vlv B	DegF	4.9904	978.01	0.51
47	Excess O2 A in Reheat Furnace	%Vol	0.2128	3.16	6.74
48	Excess O2 B in Reheat Furnace	%Vol	0.2159	2.88	7.50
49	Unit Gross Generation	MW	2.6343	169.93	1.55
50	Throttle Pressure at Stop Vlv A	PSIG	8.9533	1818.50	0.49
51	Throttle Pressure at Stop Vlv B	PSIG	8.9774	1819.70	0.49

The data used to evaluate the NNPLS system consisted of approximately 2 months of data from October and November of 2000. One minute instantaneous sampling was used to acquire this data, as was done for the data used to develop the model. This data was is an independent set that was not used during the development of the system.

The prediction results indicate that approximately 80% of the signals in the selected set have prediction errors below 2%. Errors in the range of 2-4% were obtained for 5 signals, and errors greater than 4% were obtained for 5 signals. The NNPLS-ISCV system was developed as a method to handle large signal sets with minimal oversight. The accuracy of the model was sufficient for the purposes of drift detection at the Kingston plant site. For cases in which a higher degree of accuracy is required, small inferential models could be designed and tailored for specific signals.

5.2 Drift Detection Performance

The drift detection capabilities of the system are related to the training errors observed during model development. The drift levels for the signals in the models will be greater than the prediction error incurred during model development, combined with the noise

in the signals. Each signal will exhibit different drift detection precision depending on the NNPLS model's approximation of the signal during development, and the amount of noise in the signal.

Figures 17 and 18 show the response of the NNPLS ISCV system for the feedwater flow 1 and feedwater flow 2 signals during a simulated drift test. The feedwater flow 1 signal was drifted 10% per day, while the feedwater flow 2 signal was not modified. In figure 17, the drifting measurement can clearly be seen, and the SPRT module initiates an alarm after the drift reaches 2.8% of the signal mean. The coincident response of the

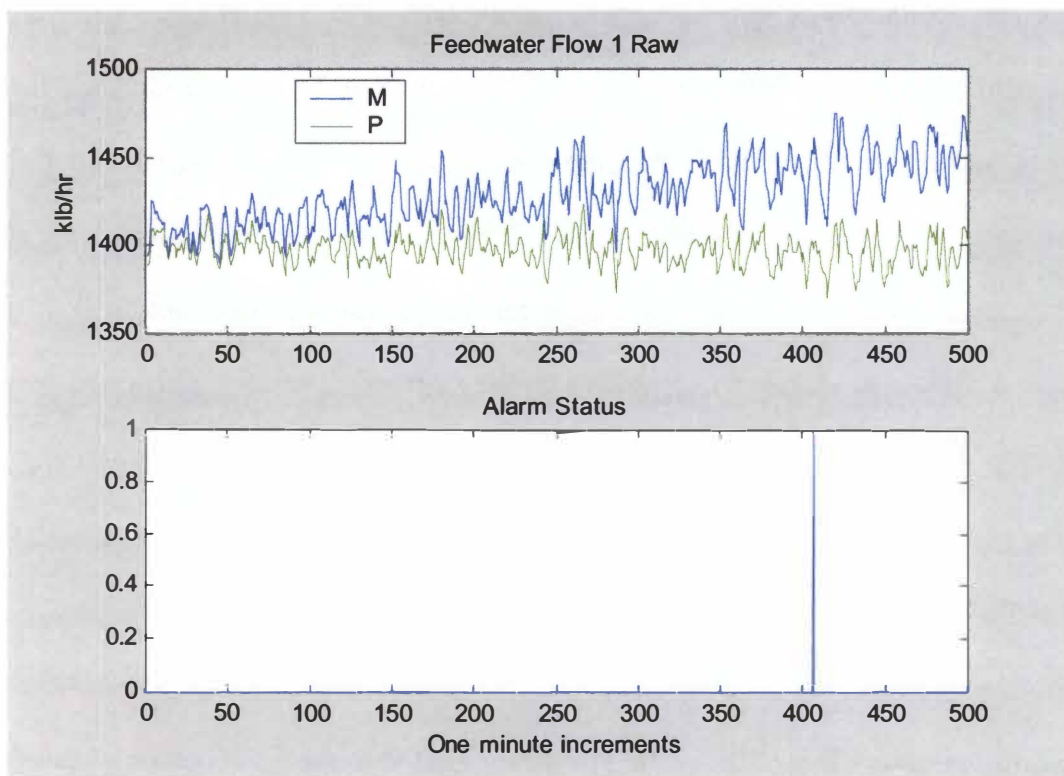


Figure 17: Feedwater flow 1 simulated drift test, alarm initiated after 2.8% drift.

feedwater flow 2 signal is shown in figure 18. The close similarity between the measurement and prediction of the feedwater 2 flow data indicates that the drift in the feedwater flow 1 signal does not significantly affect the prediction of the non-drifting signal. The correlation between these 2 signals is very high ($>.95$), and one might expect the prediction of one to be influenced by a drifting measurement in the other. This illustration shows that the NNPLS system is robust to signal drifts, and maintains sufficient prediction performance for other correlated signals when drifts do occur.

The range of the drift detection levels for the 51 monitored signals was between 2% and 5% of the signal mean value. The NNPLS-ISCV system was developed as a method to

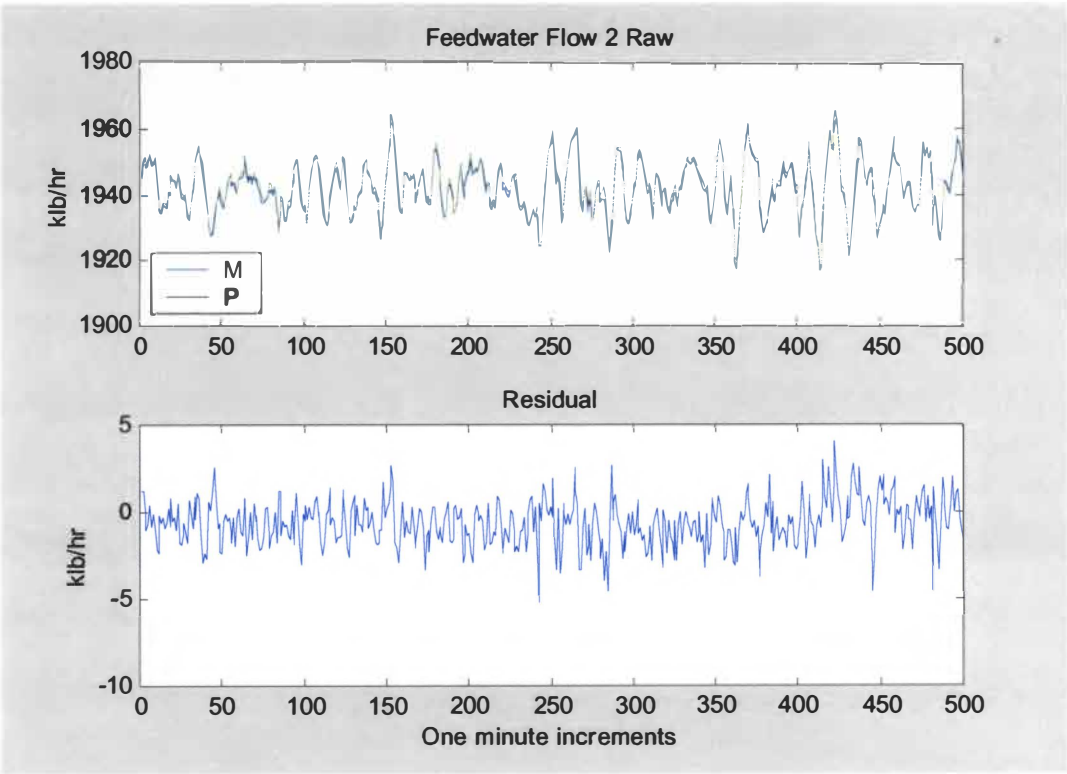


Figure 18: Feedwater flow 2 response to drift test in feedwater flow 1 signal.

handle large signal sets with minimal oversight. The drift detection performance is representative of what would be expected for a large scale ISCV system developed to cover the entire operating range of a process. For cases in which a higher degree of accuracy is required, small inferential models could be designed and tailored for specific signals.

6.0 CONCLUSIONS

6.1 *Concluding Remarks*

The primary objective, to develop and implement an ISCV system that continuously monitors process sensors during plant operation, has been completed. Currently 51 sensors are included in the validation system model. Estimation errors between the ISCV system and the measured sensor values were presented in table 3. Figures of the measured and estimated values are shown in appendix D for each of the 51 signals. Most absolute prediction errors are less than 2% of the mean values of the sensors. A few exceptions can be seen, and the model's decreased performance in these cases can be traced back to inadequate availability of correct historical data and in some cases a lack of high enough correlations to produce sufficient models.

The implemented NNPLS-ISCV system successfully identifies failed or degraded sensors using the SPRT, and replaces the faulted sensor signals with surrogate signals comprised of the ISCV model's estimations for these sensors. The decision to replace a sensor is made by a sensor alarm logic module which assesses recent trends before reporting an alarm. Alarm logic decreases false alarm reports and adds to the stability of the overall system by eliminating its ability to make instantaneous alarm decisions that may be caused by erroneous measurements, missing data, etc. Drift detection performance of the ISCV system ranges from 2-5% of signal mean. Alarms can be

cleared manually when a faulted sensor is repaired or recalibrated. If the process has moved to a new operating regime, the model will have to be redeveloped. This can be done on a sensor by sensor basis using the update option of the Development Module of the ISCV system, or the entire model may need to be redeveloped if the change affected a large number of signals.

The final set of NNPLS-ISCV system interfaces contains 2 main modules, and an optional module for connection to a database via structured query language (SQL) constructs. The required inputs and parameter settings of the interfaces are outlined in the user manuals included in Appendices A, B, and C. The set of interfaces has been designed with enough flexibility to allow for application to any process that contains sufficient correlations and historical data representative of the entire operating range to use for model development. The modules can be packaged as executables, or can be run with National Instrument's LabView software. The latter option will provide the greatest flexibility for designing models, and monitoring their performance; however, the executable versions avoid the need for user licenses.

The contributions of this work are the presentation of an alternate methodology for designing ISCV systems for large scale processes. The NNPLS ISCV system poses several advantages to the more common AANN methodology. Advantages include a linear complexity increase with model size, no reasonable limitations on the model size, inherent regularization eliminating the need for extended training algorithms to prevent model overfitting. Additionally, collinearity is dealt with in an effective manner and

solutions obtained are stable and reproducible. The benefits of the NNPLS design will be most evident when used to monitor a large process with a large number of correlated sensors.

6.2 Recommendations for Future Work

Additional enhancements to the NNPLS methodology may come from implementing a variable or component selection algorithm. In cases where the calculated sample covariance values are biased by the inclusion of non-relevant variables, a variable selection approach may lead to more consistent solutions. Component selection could also be implemented, where rather than select individual variables, one could select individual components. To become even more specific, one could eliminate a specific variable's contribution to a specific component. Work on algorithms such as this have been reported, thus there is a basis for this method already existent in the literature.

To provide a measure of the reliability of the ISCV system's performance, a reliability assessment module could be incorporated. This module would determine a reliability measure based on how similar the current data sample is to the data which was used to develop the model. This would assist in outlier identification, such that data samples which lie outside of the development region would be given a low reliability value. This type of reliability assessment could be combined with uncertainty bounds on the system predictions. An uncertainty analysis of the NNPLS model would provide the additional abilities to provide predictions with an associated uncertainty. These

uncertainty bounds could also be utilized in the reliability assessment module to provide a system which not only provides validated values, but also indicates the reliability of the system at the current operating point, and bounds the uncertainty of the system at that point.

Finally, to best validate the performance of the NNPLS model for the purposes of ISCV, its performance should be directly compared to the performance of alternate methods. These comparisons should be done on several reference data sets. All of the models should be developed or trained using the same data, and similarly validated and tested. Each data set should have different characteristics, for example, high correlations, weak correlations, strong nonlinear relationships, etc. In this way, the benefits of each empirical modeling paradigm can be related to the specific characteristics of the data. The modeling paradigms which should be compared include multiple linear regression, ridge regression, kernel regression, principal component regression, partial least squares, neural network partial least squares, and autoassociative artificial neural networks.

LIST OF REFERENCES

ASME, American Society of Mechanical Engineers, PTC PM-193.

Alsberg, B.K., and D.B. Kell, and R. Goodacre (1998), "Variable Selection in Discriminant Partial Least Squares Analysis," *Analytical Chemistry*, **70**, 4126-4133.

Baffi, G., Martin, E.B., Morris, A.J.(1999a), "Non-linear projection to latent structures revisited: the quadratic PLS algorithm," *Computers in Chemical Engineering* vol. 23, 395-411.

Baffi, G., Martin, E.B., Morris, A.J. (1999b), " Non-linear projection to latent structures revisited: the neural network PLS algorithm," *Computers in Chemical Engineering* vol. 23, 1293-1307.

Dong, D., and T. McAvoy, (1994), "Sensor data analysis using autoassociative neural networks," *Proceedings of the World Congress on Neural Networks*, **1**, 161-166, San Diego, CA.

Erbay, A. S. and B.R. Upadhyaya, (1997), "A Personal Computer-based On-line Sensor Validation System for Nuclear Power Plants," *Nuclear Technology*, **119**, 63-75.

Eryurek, E. (1991), M.S. thesis, "Development and Application of Multi-layer Neural Networks for Estimation of Power Plant Variables," The University of Tennessee Department of Nuclear Engineering, Knoxville, TN.

Fantoni, P., S. Figedy, A. Racz, (1998), "A Neuro-Fuzzy Model Applied to Full Range Signal Validation of PWR Nuclear Power Plant Data," FLINS-98, Antwerpen, Belgium.

Geladi, P., and B. R. Kowalski (1986), "Partial Least Squares Regression: A Tutorial," *Analytica Chimica Acta*, **185**, 1-17.

Gil, J.A., and R. Romera (1998), "On Robust Partial Least Square (PLS) Methods," *Journal of Chemometrics*, **12**, 365-378.

Gross, K.C., S. W. Wegerich, R. M. Singer, and J. E. Mott (1998), "Industrial Process Surveillance System," US Patent #5,764,509.

Gross, K.C., and Keith E. Humenik (1991), "Sequential Probability Ratio Test for Nuclear Plant Component Surveillance," *Nuclear Technology*, **93**, 131-137.

Hansen, E.J., and M.B. Caudill (1994), "Similarity Based Regression: Applied Advanced Pattern Recognition for Power Plant Analysis", 1994 EPRI-ASME Heat Rate Improvement Conference; Baltimore, Maryland.

Hines, J.W., and D.J. Wrest (1997), "Signal Validation Using an Adaptive Neural Fuzzy Inference System," *Nuclear Technology*, August, 181-193.

Hines, J.W., and R.E. Uhrig, (1998), "Use of Autoassociative Neural Networks for Signal Validation," *Journal of Intelligent and Robotic Systems*, Kluwer Academic Press, February, 143-154.

Hines, J.W., A.V. Gribok, I. Attieh, and R.E. Uhrig (1999a), "Regularization Methods for Inferential Sensing in Nuclear Power Plants," *Fuzzy Systems and Soft Computing in Nuclear Engineering*, Ed. Da Ruan, Springer.

Hines, J.W., A.V. Gribok, I. Attieh, and R.E. Uhrig (1999b), "The Use of Regularization in Inferential Measurements," Presented at the Enlarged Halden Programme Group (EHPG) Meeting, Loen, Norway, May 24-29.

Hines, J.W., A.V. Gribok, I. Attieh, and R.E. Uhrig (2000), "Neural Network Regularization Techniques for a Sensor Validation System," American Nuclear Society Annual Meeting, San Diego, California, June 4-8.

Hines, J.W., and B. Rasmussen (2000b), "On-Line Sensor Calibration Verification: A Survey," Presented at the 14th International Congress and Exhibition on Condition Monitoring and Diagnostic Engineering Management, Manchester, England, September.

Hines, J.W., and B. Rasmussen (2001a), "Continuous Calibration Verification at a Coal Fired Power Plant," Presented at ICOMS-2001, The International Conference of Maintenance Societies, Melbourne, Australia, June.

Holcomb, T.R., and M. Morari (1992), "PLS/Neural Networks," *Computers chem. Engng.*, **16**, 393-411.

Hoskuldsson, A.(1988), "PLS Regression Methods," *Journal of Chemometrics*, **2**, 211-228.

Hoskuldsson , A. (1995), "A Combined Theory for PCA and PLS," *Journal of Chemometrics*, **9**, 91-123.

Humenik, K., and K.C. Gross (1990), "Sequential Probability Ratio Tests for Reactor Signal Validation and Sensor Surveillance Applications," *Nuclear Science and Engineering*, **105**, 383-390.

Justusson, B.I. (1981), "Median Filtering: Statistical Properties in Two-Dimensional Digital Signal Processing II: Transforms and Median Filters," Ed. T.S. Huang, Springer-Verlag , Berlin.

Kramer, M.A. (1992), "Autoassociative neural networks," *Computers in Chemical Engineering*, vol. 16, 313-328.

Kresta, J.V., T.E. Marlin, and J.F. MacGregor (1994), "Development of Inferential Process Models Using PLS," *Computers chem. Engng.*, **18**, 597-611.

Li, R.L., and L. Wang (1999), "Fault Detection Using Sequential Probability Ratio Test," *Proceedings of the 1999 Winter Meeting of IEEE Power Engineering Society*, Piscataway, NJ, **2**, 938-943.

Lindgren, F., P. Geladi, S. R  nner and S. Wold (1994), "Interactive Variable Selection (IVS) for PLS. Part 1: Theory and Algorithms," *Journal of Chemometrics*, vol. 8, 349-363.

Malthouse, E.C., A.C. Tamhane, and R.S.H. Mah (1997), "Nonlinear Partial Least Squares," *Computers chem. Engng.*, **21**, 875-890.

Martens, H., and T. Naes (1989), *Multivariate Calibration*, John Wiley and Sons, Chichester.

Massart, D.L., B.G.M. Vandeginste, S.N. Deming, Y. Michotte, and L. Kaufman (1988), *Chemometrics: A textbook*, Elsevier Science Publishers, Amsterdam.

MATLAB (1999), *High performance numeric computation and visualization software*, version 5.3, The Mathworks Inc., Natick, MA.

Moller, M. F. (1993), "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, 525-533.

Mott, Young, and R.W. King (1987), "Pattern Recognition Software for Plant Surveillance," US DOE Report.

Nabeshima, K., Susuki, K., and Turkan, T. (1995), "Real-time nuclear power plant monitoring with hybrid artificial intelligence systems," *9th Power Plant Dynamics, Control & Testing Symposium*, Vol. 2, pp. 55.01, University of Tennessee, May 24-26.

National Instruments (2000), LabView Software, version 6.0, National Instruments Corporation, Austin, TX.

Olvera, J.R. (1993), M.S. thesis, "Instrument Calibration Verification," The University of Tennessee, Department of Nuclear Engineering, Knoxville, TN.

Osborne, S.D., R.B. Jordan, and R. Künnrmeyer (1997), "Method of Wavelength Selection for Partial Least Squares," *Analyst*, **122**, 1531-1537.

Qin, S.J., and T.J. McAvoy (1992), "Nonlinear PLS Modeling Using Neural Networks," *Computers in Chemical Engineering*, vol. 16, n. 4, 379-391.

Qin, S.J., and T.J. McAvoy (1996), "Nonlinear FIR Modeling via Neural Net PLS Approach," *Computers chem. Engng.*, **20**, 147-159.

Qin, S. (1997), "Neural networks for intelligent sensors and control --- Practical issues and some solutions," In *Neural Systems for Control*, Chapter 8, Edited by O. Omidvar and D. L. Elliott, Academic Press.

Qin, S.J and W. Li (1999), "Detection, identification and reconstruction of faulty sensors with maximized sensitivity," *AIChE J.*, 45(9).

Rasmussen, B., J.W. Hines, and R.E. Uhrig (2000a), "Nonlinear Partial Least Squares Modeling for Instrument Surveillance and Calibration Verification," published in the proceedings of the Maintenance and Reliability Conference (MARCON 2000), Knoxville, TN, May 7-10.

Rasmussen, B., J.W. Hines, and R.E. Uhrig (2000b), "A Novel Approach to Process Modeling for Instrument Surveillance and Calibration Verification," Proceedings of The Third American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation and Control and Human-Machine Interface Technologies, Washington DC, November 13-17.

Ray, A. (1989), "Sequential Testing for Fault Detection in Multiply-Redundant Systems," *ASME Journal of Dynamic Systems, Measurement and Control*, **111**, 329-332.

Seasholtz, M.B. and B. Kowalski (1993), "The Parsimony Principle Applied to Multivariate Calibration," *Anal. Chim. Acta*, vol. 277, 165-177.

Singer, R.M., K.C. Gross, J.P. Herzog, R.W. King, and S.W. Wegerich (1996), "Model-Based Nuclear Power Plant Monitoring and Fault Detection: Theoretical Foundations," Proc. 9th Intl. Conf. on Intelligent Systems Applications to Power Systems, Seoul, Korea.

Tyan, S.G. (1981), "Median Filtering: Deterministic Properties in Two-Dimensional Digital Signal Processing II: Transforms and Median Filters," Ed. T.S. Huang, Springer-Verlag, Berlin.

Upadhyaya, B.R., (1985), "Sensor Failure Detection and Estimation," *Nuclear Safety*.

Upadhyaya, B.R., Wolvaardt, F.P., and Glockler, O. (1987), "An Integrated Approach for Sensor Failure Detection in Dynamic Systems," Research Report prepared for the Measurement & Control Engineering Center, Report No. NE-MCEC-BRU-87-01.

Upadhyaya , B.R., and K. Holbert (1989), "Development and Testing of an Integrated Signal Validation System for Nuclear Power Plants," DOE Contract DE-AC02-86NE37959.

Upadhyaya, B.R., and E. Eryurek (1992), "Application of Neural Networks for Sensor Validation and Plant Monitoring," *Nuclear Technology*, vol. 97, 170-176.

Wald, A. (1945), "Sequential tests of statistical hypothesis," *Ann. Math. Statist.* vol. 16, 117-186.

Wald, A. (1947), *Sequential Analysis*, John Wiley & Sons, New York.

Wilson, D.J.H., G.W. Irwin, and G. Lightbody (1997a), "Neural Networks and Multivariate SPC," *Proceedings of the IEE Colloquium on Fault Diagnosis in Process Systems*, London, UK, n. 174, 5/1-5/5.

Wilson, D.J.H., G.W. Irwin, and G. Lightbody (1997b), "Nonlinear PLS Modeling Using Radial Basis Functions," *Proceedings of the American Control Conference*, Albuquerque, NM, 3275-3276.

Wold H. (1966), "Non-linear Estimation by Iterative Least Squares Procedures," In: David, F. (Ed.), *Research Papers in Statistics*, Wiley, NY.

Wold, S., Ruhe, A., Wold, H., Dunn, W.J. III (1984), "The collinearity problem in linear regression: The partial least squares approach to generalized inverses," *SIAM J. Sci. Stat. Comput.*, vol. 5, 735-743.

Wold, S., N.K. Wold, and B. Skagerberg (1989), "Nonlinear PLS Modeling," *Chemometrics and Intelligent Laboratory Systems*, 7, 53-65.

Wold, S. (1992), "Nonlinear Partial Least Squares Modeling II. Spline Inner Relation," *Chemometrics and Intelligent Laboratory Systems*, 14, 71-84.

Woodward, A.M., A. Jones, X. Zhang, J. Rowland, and D.B. Kell (1996), "Rapid and Non-invasive Quantification of Metabolic Substrates in Biological Cell Suspensions Using Non-linear Dielectric Spectroscopy with Multivariate Calibration and Artificial Neural Networks," *Bioelectrochemistry and Bioenergetics*, 40, 99-132.

Wrest, D. J. (1996), M.S. thesis, "Instrument Surveillance and Calibration Verification through Plant Wide Monitoring using Autoassociative Neural Networks," The University of Tennessee, Department of Nuclear Engineering, Knoxville, TN.

Xu, D.L., Martin, E.B., Morris, A.J. (1997), "Nonlinear projection to latent structures," *Proceedings of the 13th World Congress, International Federation of Automatic Control*, 259-264, Pergamon, Oxford, UK.

Xu, X., J.W. Hines, and R.E. Uhrig (1999), "Sensor Validation and Fault Detection Using Neural Networks," published in the proceedings of the *Maintenance and Reliability Conference (MARCON 99)*, Gatlinburg, TN, May 10-12.

Xu, X. (2000), PhD dissertation, "Automated Neural Network-Based Instrument Validation System," The University of Tennessee, Nuclear Engineering Department, Knoxville, TN.

APPENDICES

APPENDIX A

NNPLS Development Interface Set-Up and Tutorial

The use of the NNPLS development interface is described beginning with preliminary set-up procedures followed by illustrations and discussions.

Preliminary Set-Up

Prior to starting development, a directory needs to be created as `c:\pls_dll\new_model`. This directory will be used to hold the developed model. After creating the directory, open the LabView virtual instrument with the filename `development.vi`. Sample data files have been provided (training data – `train.txt`, validation data – `val.txt`, test data - `test.txt`). To begin, depress the run arrow on the Labview toolbar and make sure the switch at the top of the front panel is set to ON.

Data File Formatting and Construction

The interface supports tab delimited ASCII text. If using MATLAB to save your training and test data sets, please use the following MATLAB code.

```
>>dlmwrite('test.txt',test,'t');
```

Where test.txt is the file name and test is the data matrix. If using a spreadsheet such as excel, simply save the data as a tab delimited test file.

Setting the file paths

1. Insert the correct file names and corresponding paths (A and C in figure 19).

The browse tabs are provided to ease in locating the data files.

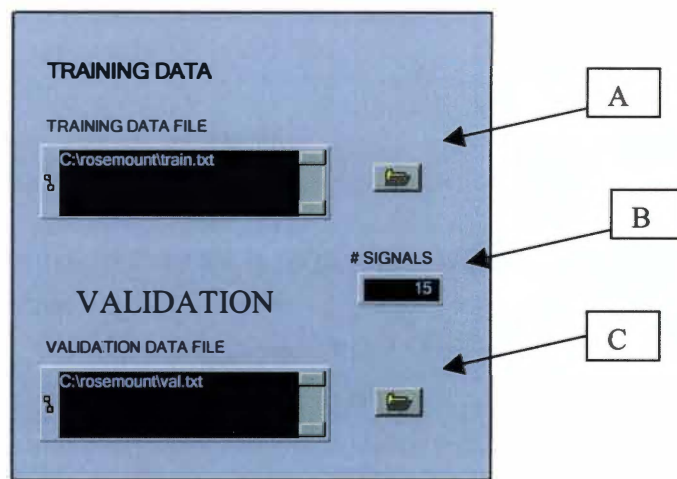


Figure 19: Front panel controls to identify and describe the training data file.

A: Training Data File - set to identify the name and location of the training data file. The data file must be a tab delimited text file, and the filename extension **MUST** BE .txt.

B: # Signals – indicates the number of signals that are represented in the data files.

C: Validation Data File- set to identify the name and location of the validation data file. The data file must be a tab delimited text file, and the filename extension **MUST**

BE.txt. Note that the number of signals in the validation data file must be the same as that in the training data file.

Viewing the Sensor Data and Statistics

2. Depress the LabView run arrow on the LabView toolbar.

The sensor information should now be displayed as shown in figure 20.

3. The signal and point controls can be used to scroll through the training and validation data tables. Note that location [0,0] is the first sample for the first sensor in the data file. A sensor's data is displayed along its corresponding row.

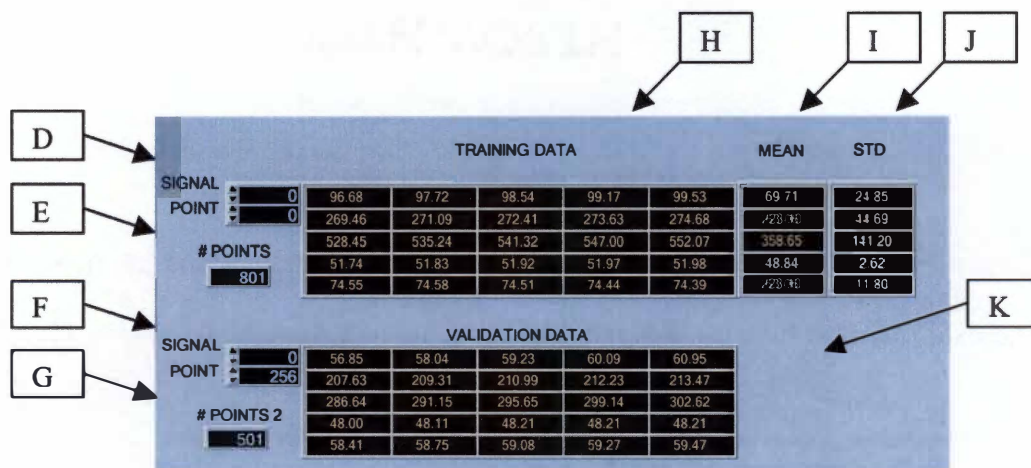


Figure 20: Front panel indicators and controls for viewing the data and statistics.

D: Signal - controls the signal which is displayed along the top row of the training data table. Note that 0 refers to the first signal in the data.

E: Point - controls the data point which is displayed down the leftmost column of the training data table. Note that 0 refers to the first point in the data.

F: Signal - controls the signal which is displayed along the top row of the validation data table. Note that 0 refers to the first signal in the data.

G: Point - controls the data point which is displayed down the leftmost column of the validation data table. Note that 0 refers to the first point in the data.

H: Training Data – table of information read from the specified training data file.

I: Mean – display of the mean values of the training data.

J: STD – display of the standard deviations of the training data.

K: Validation Data – table of information read from the specified validation data file.

Data Filtering and Saving the Developed Model

4. It is recommended that some filtering be applied to the data. The median filter window size (M) should be set according to the level of noise in the data. A general range for this window size is 3-10. In some cases if the data was recorded as 30 minute

averages, median filtering may not be necessary. To see the effect that the filtering has on the data, referring to figure 24, set control X to "Training Data" and set control DD to a specific signal for evaluation. The result of varying filter window sizes on the displayed signal can be viewed as control M is adjusted.

5. After setting the median filter controls to the desired values, ensure that the save model control box (O in figure 21) is checked, unless the purposes are for evaluation only. Note that the save model control will result in the developed model to be written to the directory c:\pls_dll\new_model, and will write over any previous model (unless the update option is selected, see step 6 below).

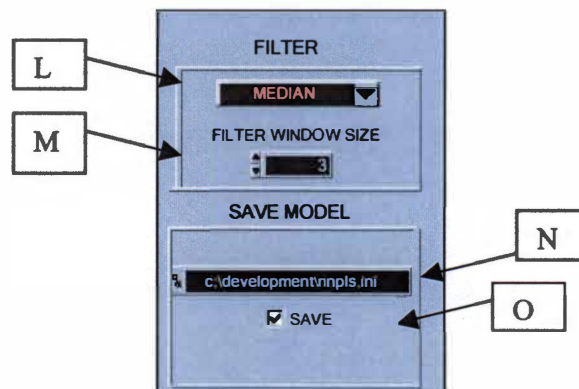


Figure 21: Front panel controls for filtering the data.

L: Filter Option – the two choices are none, and median.

M: Window Size – controls the length of the median filter window. The larger the window, the greater the effect the filter will have. Note that an alternate measure for window size is rank. The relationship between the two measures is:

$$\text{window size} = 2 \times \text{rank} + 1.$$

N: Configuration Settings – supplies the path and names of the stored model parameters. This is a generated file, and the paths are preset such that all model parameters are stored in a directory `c:\pls_dll\new_model`. These settings should not be altered because the monitoring interface reads these files from this directory.

O: Save Control – when checked, new models developed will be written to the locations and file names specified in the configuration settings file. This box must be checked prior to depressing the Go tab in the create models section in order to save the model parameters to file. Checking the box after the algorithm is run will not save the model parameters. See control Q in figure 23 below.

Model Building or Updating

6. Set the update/build control to build if the development data is new, and this is the first run through the development interface. If a full model corresponding to the development data has already been created, and the purpose is to improve the performance of an individual sensor prediction in the model, set the build/update control to update. The recommended procedure is that on the first run through, a full

model is created and stored. After evaluating the performance, either through the development interface graphical display (figure 24) or with the monitoring interface, if updates to specific signals are required return to the development interface and set this control to update. Successful updates require that a more representative data set be selected so that the performance can be enhanced. When the update option is selected and the save model control box is checked, the new model for the selected sensor (control R in figure 23) will be inserted into the previously developed full model.



Figure 22: Front panel control for model updating or building.

P: Updating/Build Control – This control allows for the updating of an individual sensor model contained within a previously generated full model for the given set of signals represented in the data. Model updates can be applied using different data than that which was used to develop the original model; however, the new data sets must contain the same signals, in the same order, as the original data. Setting this control to building will result in the creation of a full set of model parameters, assuming that control R in figure 23 is set to 0, and the save model control box is checked (control O in figure 21).

Controls for Creating Models and Viewing the Progress Indicator

7. Set the signal to model control to the sensor for which a model is desired. If this value is set to zero, models will be developed for all sensors in the training data set.
8. Depress the Go tab (Q). This will begin the development algorithm for models corresponding to the selection indicated by control (R).
9. The indicator will turn green while the development algorithm is running, and the progress indicators will display the current activity. The indicator light will turn off when the development is complete.

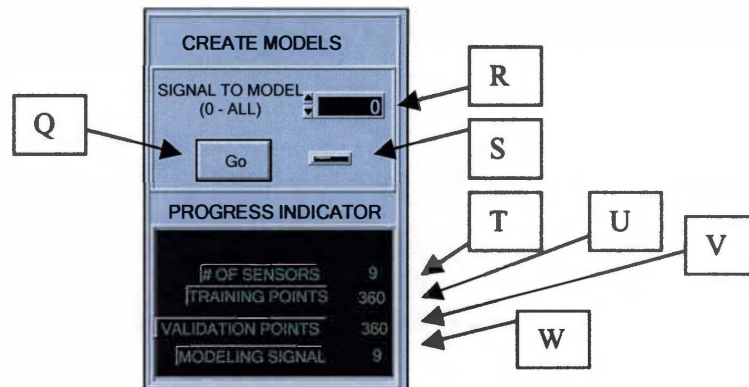


Figure 23: Front panel controls and indicators for model development.

Q: Go – depressing this control initiates the development algorithm. If the save model control (O in figure 21) is checked, each time this control is depressed, the new

model parameters will be saved, writing over the previous parameters in the files, corresponding to the selection of which signals to model (R).

R: Signal to Model - controls which signals the development algorithm will construct models for. If this value is set to zero models will be developed for all sensors in the data set. If set to a specific sensor only one model will be developed. This control should be set accordingly with the save model control box, (O in figure 21) and the build/update control (figure 22). See the discussion below for further explanation.

S: Indicator Light – indicates (turns green) when the development algorithm is running. Note that all other controls will be very slow to respond while the algorithm is running. It is not recommended that controls be adjusted while the algorithm is running.

T: # of Sensors – indicates the number of sensors in the data file.

U: Training Points – Indicates the number of training points being used to develop the current model.

V: Validation Points – Indicates the number of validation data points being used to develop the current model.

W: Modeling Signal – indicates the signal for which a model is being developed. Note that if control U is set to 0, this number will increase in +1 increments up to the number of sensors displayed in T.

Further Discussion on the controls for creating models

There are three controls that will influence the development and storage of the models. Table 4 below shows these 3 controls, and based on the different possible combinations of these controls, the characteristics of the stored models is described.

Table 4: Data Storage Method Based on Front Panel Control Settings

Control	Build/Update	Signal to Model	Save Model Control
Type I settings	Build	0 (all)	checked
Type II settings	Build	specific sensor	checked
Type III settings	Update	0 (all)	checked
Type IV settings	Update	specific sensor	checked

Type I settings – The developed model will be a full model for all signals represented in the data. The stored model will overwrite any previous models, and will be a full model.

Type II settings – The developed model will contain a model for the specified sensor; however, the model parameters will be of full dimension with zeros inserted in the parameter spaces for all other signals. The stored model will be of full dimension, but will produce proper predictions for only one signal.

Type III settings – The developed model will be a full model, and the stored model will also be a full model. This setting will produce the same result as the Type I setting.

Type IV settings – The developed model will contain a single model for the specified sensor. In this case the model parameters will not be fully dimensional. The previously stored model will be read in, and the new parameters for the specified sensor will be inserted into the previous model. The stored model will contain the previous models, and the newly updated model.

Note that the save model control must be checked for the above settings to produce the described effect.

Controls and Display for Evaluation of Input Data, Prediction, and Error

10. The controls and indicators shown below in figure 24 allow for the evaluation of the developed models. Review the control descriptions below the figure for additional information.

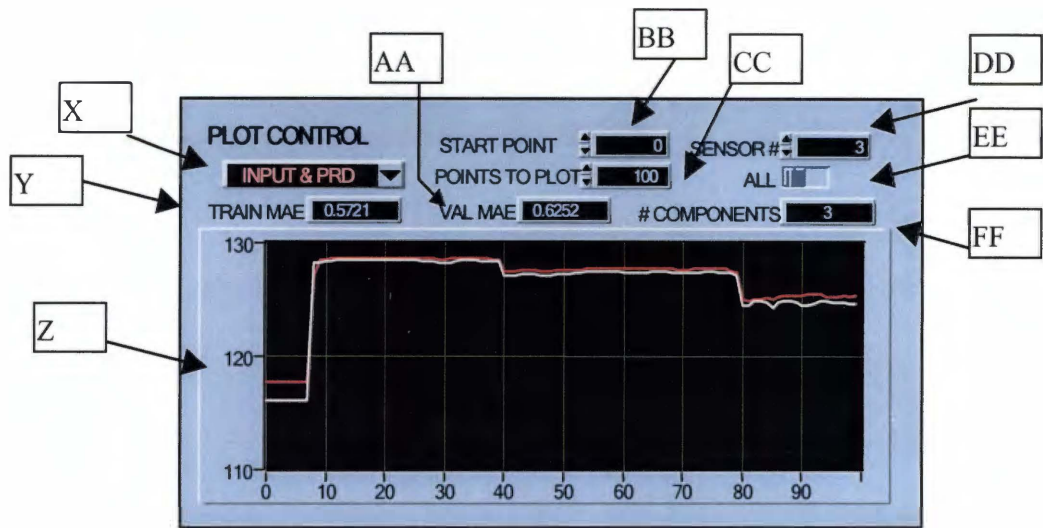


Figure 24: Front panel plot controls and display of inputs, predictions, and training errors.

X: Display Control – controls the current display. The options are none, training data, training prediction, training error, training data & prediction, validation data, validation prediction, validation error, validation data & prediction. Predictions and errors can only be displayed after the development algorithm has been executed.

Y: Train MAE – indicates the mean absolute error on the scaled training data corresponding to the sensor specified by control DD.

Z: Display – graphical presentation of selection based on current control settings.

AA: Val MAE – indicates the mean absolute error on the scaled validation data corresponding to the sensor specified by control DD.

BB: Start Point – controls the leftmost point of the display. Note that 0 refers to the first point in the data file.

CC: Points to Plot – controls the number of points in the display.

DD: Sensor # - controls the sensor being displayed.

EE: Plot All Signals Option – will override control DD and plot all of the signals in the training data set, and/or their predictions or errors, based on the setting of control X.

FF: # Components – indicates the number of latent vectors or components which were used in the model displayed according to control DD.

Files Generated

A number of files are generated by the development interface and stored in specific directories. With the exception of the *nnpls.ini* file, all stored files are written to the *c:\pls_dll\new_model* directory. All of the generated files are briefly described below.

- *c:\development\nnpls.ini*: This is a generated file which indicates the locations and filenames of the stored model parameters.
- *c:\pls_dll\new_model\nn.txt*: This file contains the number of nodes used in the neural networks (set at a default value of 2).
- *c:\pls_dll\new_model\ni.txt*: This file contains total number of signals represented in the data files.

- *c:\pls_dll\new_model\mf.txt*: This file contains the maximum number of latent vectors allowable by the developed model.
- *c:\pls_dll\new_model\rank2.txt*: This file contains the rank of the applied median filter used during development so that a filter of the same rank may be applied during monitoring.
- *c:\pls_dll\new_model\theta1.txt*: This file contains the degradation means for use in the sequential probability ratio test.
- *c:\pls_dll\new_model\variance.txt*: This file contains the residual variances for use in the sequential probability ratio test.
- *c:\pls_dll\new_model\nnpls_xm.dbl*: This file contains the mean values for pre-process scaling of the input data sets for each inferential model.
- *c:\pls_dll\new_model\nnpls_xs.dbl*: This file contains the standard deviation values for pre-process scaling of the input data sets for each inferential model.
- *c:\pls_dll\new_model\nnpls_ym.dbl*: This file contains the mean values for post-process scaling of the output predictions for each inferential model
- *c:\pls_dll\new_model\nnpls_ys.dbl*: This file contains the standard deviation values for post-process scaling of the output predictions for each inferential model.
- *c:\pls_dll\new_model\nnpls_w.dbl*: This file contains the NNPLS transformation weights to be applied to the input data sets for each inferential model.
- *c:\pls_dll\new_model\nnpls_p.dbl*: This file contains the NNPLS input loadings required for matrix deflation of the input data during model development and later required during monitoring.
- *c:\pls_dll\new_model\nnpls_q.dbl*: This file contains the NNPLS output loadings which are used for matrix deflation of the output data during model development, and matrix inflation of the output predictions during development.
- *c:\pls_dll\new_model\nnpls_wn.dbl*: This file contains the neural network weights and biases for all of the inferential models developed.
- *c:\pls_dll\new_model\nnpls_f.dbl*: This file contains the number of latent vectors retained (determined by cross validation) in each inferential model.

APPENDIX B

SQL Connection Interface Set Up and Tutorial

The use of the SQL connection interface is described beginning with preliminary set-up procedures followed by illustrations and discussions.

Preliminary Set-Up

An important note is that if the SQL connection interface is going to be utilized with the monitoring module, the data must be organized alphabetically by tag name in the development data matrices. This is due to the inclusion of an ORDER BY *tag column name* at the end of the SQL SELECT statement. This is intended to alleviate problems associated with miss ordering of the data causing erroneous prediction results. As an example of the proper order consider 3 signals with tag names A, B, and C. During model development the data should be organized in a $n \times 3$ matrix, where n is the number of samples. The first column of the matrix should contain the data for signal A, the second column should contain the data for signal B, and so on.

A simple flow diagram of the SQL connection interface is shown in figure 25. After loading the tag name files, data is retrieved from the server and stored. This represents the data archive from which data will be extracted and sent to the validation model.

Following validation, the measured data and validated data are also stored. The storage of this data provides a history of the performance of the model.

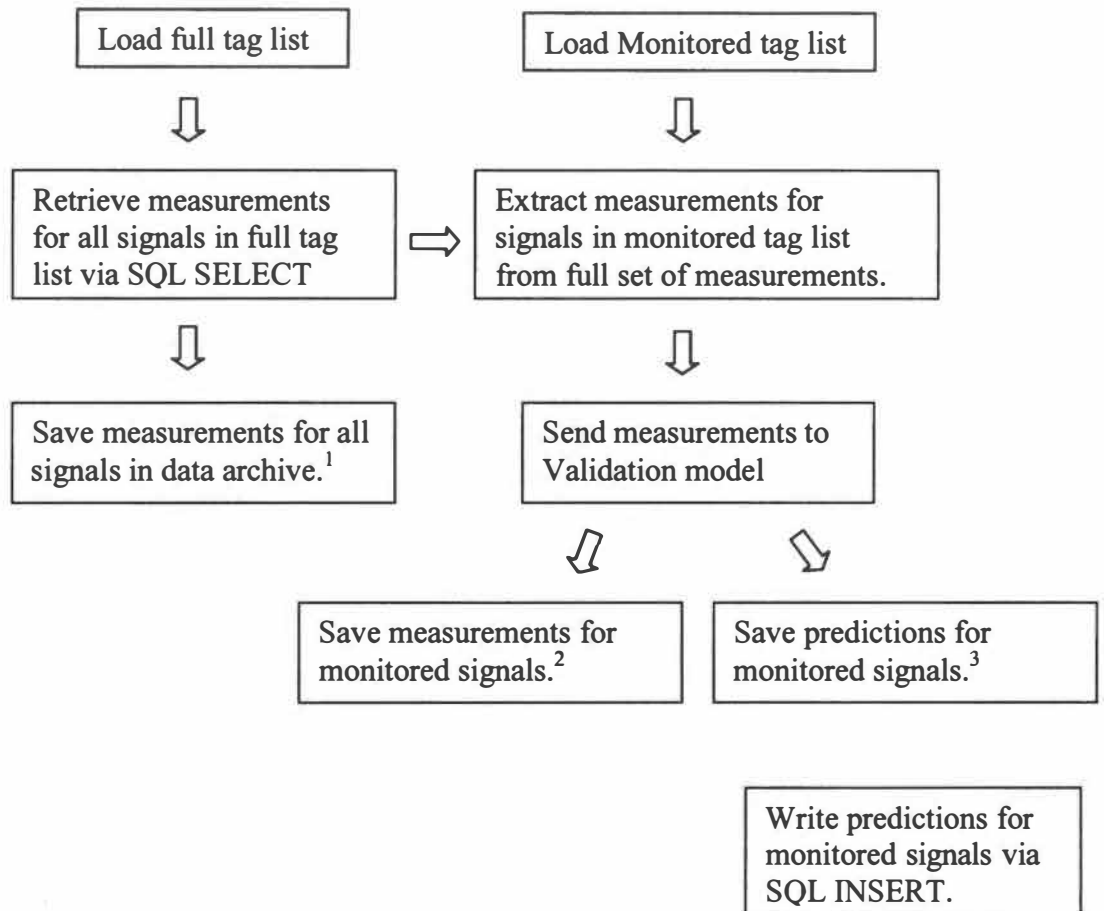
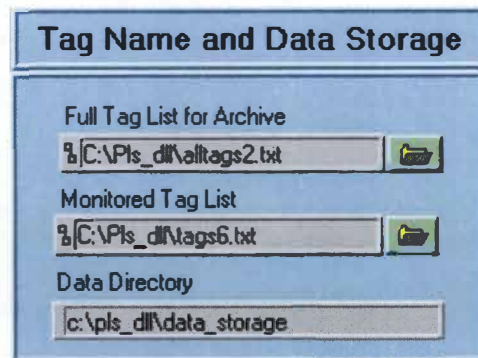


Figure 25: Flow diagram for the SQL connection interface.

- ¹ Data archive filename, according to figure 26 settings, will be c:\pls_dll\data_storage\data_archiveMMDDYYYYY.dbl.
- ² Measured data filename, according to figure 26 settings, will be c:\pls_dll\data_storage\measurementsMMDDYYYYY.dbl. The data in these files will vary from the corresponding data in the data archive files because this data is saved after median filtering has been applied.
- ³ Predicted data filename, according to figure 26 settings, will be c:\pls_dll\data_storage\predictionsMMDDYYYYY.dbl.
Where: MM is a two digit number representing the current month
DD is a two digit number representing the current day
YYYY is a four digit number representing the current year

Creating tag list names and setting the data storage directory

This section describes the required formats for the signal tag lists, and the methods of data storage. The 3 required inputs are shown below in figure 26.



The screenshot shows a dialog box titled "Tag Name and Data Storage". It contains three input fields, each with a folder icon button to its right:

- Full Tag List for Archive: %C:\Pis_dlltags2.txt
- Monitored Tag List: %C:\Pis_dlltags6.txt
- Data Directory: c:\pis_dll\data_storage

Figure 26: SQL interface tag name and data storage controls.

The "full tag list for archive" file format should be a tab delimited text file with each tag name being entered on a separate line. These tag names are used to create the SQL SELECT statement. The "monitored tag list" contains 3 columns:

1. Tag name
2. Verified tag name
3. Index of tag in the "full tag list for archive"

Sample files are presented below in figure 27. The dual file structure of the interface was designed so that a full set of signals could be recorded while the interface is running. All of the signals in the full tag list may not be required by the current validation model; however, if future models require additional signals a full set of data will be readily available.

Full Tag List		
9DRAFT:B1.PNT		
9DRAFT:B4.PNT		
9DSAI2:B16.PNT		
9DSAI2:B17.PNT		
9DSAI2:B18.PNT		
9DSAI2:B19.PNT		
9DSAI2:B20.PNT		
9DSAI2:B27.PNT		
9DSAI2:B33.PNT		
9DSAI2:B34.PNT		
9DSAI2:B35.PNT		
9DSAI2:B36.PNT		
9DSAI3:B2.PNT		
9DSAI3:B3.PNT		
9DSAI3:B35.PNT		
9DSAI3:B36.PNT		

Monitored Tag List		
9DSAI2:B16.PNT	9DSAI2B16PNT	3
9DSAI2:B20.PNT	9DSAI2B20PNT	7
9DSAI2:B34.PNT	9DSAI2B34PNT	9
9DSAI2:B35.PNT	9DSAI2B35PNT	10
9DSAI2:B36.PNT	9DSAI2B36PNT	11
9DSAI3:B2.PNT	9DSAI3B2PNT	14
9DSAI3:B3.PNT	9DSAI3B3PNT	16

Figure 27: Sample tag list files and formats.

It is recommended that the tag lists be alphabetized according to the measured tag names, because the SQL SELECT statement contains an ORDER BY *tag column name* (referring to figure 28 settings this would read ORDER BY tag). The validated tag names in the example are the same as the measured tag names with the exception of the punctuation characters. The user can define any method for assigning validated tag names.

The data storage directory (see figure 28) is the location to which three files will be written:

1. data_archiveMMDDYYYYY.dbf

2. measurementsMMDDYYYYY.dbl

3. predictionsMMDDYYYYY.dbl

(M, D, and Y are digits for month, day, and year)

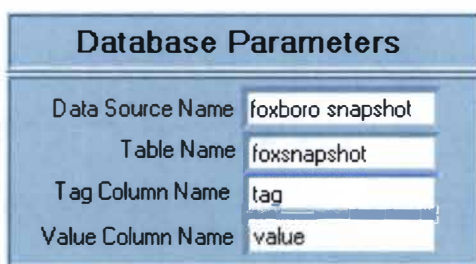
The data archive file will be the largest of the three files, assuming that not all of the tags in the full tag list are in the monitored tag list. The measurement and prediction files will be the same size. The files are written in standard binary format, and can easily be read into Matlab with the following function:

```
function [X]=read_double(filename,number_signals);  
% filename is a string argument,  
% e.g. filename='data_archive10102001.dbl';  
% number_signals is the number of signals contained in the requested  
% file.  
fid_x=fopen(filename,'r');  
X=fread(fid_x,[number_signals inf],'double');  
fclose(fid_x);X=X';
```

Three files will be written for each day the interface is run. This keeps the file sizes small for simple file transfer and management. The data archive files will contain the exact values returned from the data source. These files are an important source of data for model updating. The measured data files will contain median filtered data, assuming that median filtering was implemented during the development of the validation model. The prediction data files will contain the validation model's outputs corresponding to the data in the measured data files. These archives enable easy access to the measured data and the corresponding predictions.

Setting the Database Parameters

The database parameters are entered in the text controls as shown in figure 28. These parameters are database specific and should be entered according to the definitions of the appropriate database.



Database Parameters	
Data Source Name	foxboro snapshot
Table Name	foxsnapshot
Tag Column Name	tag
Value Column Name	value

Figure 28: SQL interface database parameter controls.

Viewing the Connection Status and Running the Interface

After creating the signal tag lists, the interface can be initiated by first clicking the "Run" tab as shown in figure 29. Set the desired sampling rate, and then press the run arrow on the LabView toolbar. The "Successful Connect" indicator will turn green and the iteration counter indicates the number of successful iterations.



Figure 29: SQL interface connection status indicators and run control.

The recommended method to stop the operation of the SQL connection interface is to depress the "Run" button as shown in figure 29. The successful disconnect indicator will illuminate after the iteration counter increases by one increment. After the disconnection is confirmed the interface can be stopped completely by pressing the run tab on the LabView toolbar. In this way, connections to a server will not be maintained after the interface is stopped. In some cases, if a disconnect command is not executed the process will not be deleted and multiple connections may be established decreasing server availability.

Viewing the Tag Names, Data, and Predictions

The signal tag names and data display (figure 30) enables the user to view the relevant information. The data tag column displays the tag names of the signals that are being sent to the monitoring interface. These tag names are those that are used in the SQL SELECT statement. The verified tags column shows the corresponding tag to which the validated data will be written. The measurements column displays the most recent data

Signal Tag Names and Data				
	Data Tags	Verified Tags	Measurements	Predictions
0	9BM:B0.PNT	9BMB0PNT	1016.17	1025.11
0	9BM:B0A.PNT	9BMB0APNT	1016.51	1022.65
	9DSAI2:B16.PNT	9DSAI2B16PNT	1098.22	1107.22
	9DSAI2:B33.PNT	9DSAI2B33PNT	514.97	499.27
	9DSAI2:B34.PNT	9DSAI2B34PNT	180.27	182.25
	9DSAI2:B35.PNT	9DSAI2B35PNT	74.14	72.23
	9DSAI2:B36.PNT	9DSAI2B36PNT	48.48	50.24
	9DSAI3:B2.PNT	9DSAI3B2PNT	108.96	111.95
	9DSAI3:B3.PNT	9DSAI3B3PNT	167.04	162.18
	9DSAI3:B36.PNT	9DSAI3B36PNT	263.84	255.94
	9DSAI3:B38.PNT	9DSAI3B38PNT	244.20	233.10
	9DSAI3:B39.PNT	9DSAI3B39PNT	467.26	458.67
	9DSAI3:B4.PNT	9DSAI3B4PNT	191.55	201.24
	9DSAI3:B42.PNT	9DSAI3B42PNT	338.19	337.54
	9DSAI3:B43.PNT	9DSAI3B43PNT	193.10	187.14
	9DSAI3:B44.PNT	9DSAI3B44PNT	169.07	172.33
	9DSAI3:B45.PNT	9DSAI3B45PNT	178.85	185.60

Figure 30: SQL interface signal tag name, measurement, and prediction indicators.

retrieved from the server, and the predictions column displays the corresponding validated values.

Viewing the Constructed SQL Statements

At a rate specified by the sampling rate defined in figure 29, a single SQL SELECT statement will be executed. The tag names in the SELECT statement are those

contained in the "full tag list for archive" (see figure 26). Following extraction of the required data for the current model, the data will be sent to the validation model. The resultant predictions will then be written to the validated tag names contained in the second column of the "monitored tag list" (see figure 26). This task is achieved through the use of a single SQL INSERT statement. Representative SQL statements are shown below in figure 31.

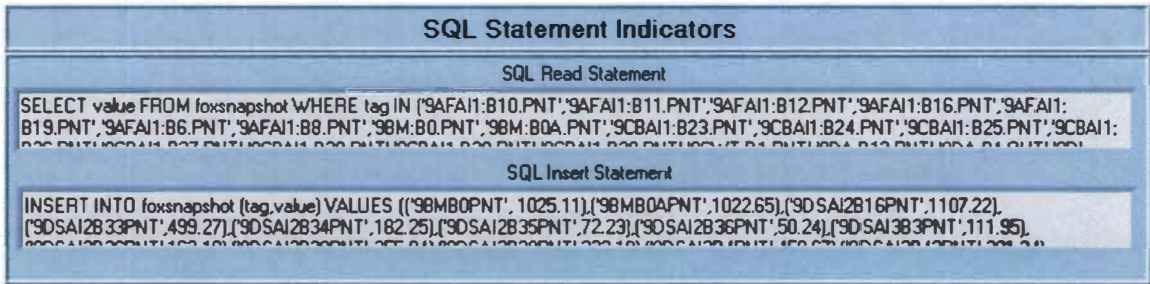


Figure 31: SQL interface indicators of the applied SELECT and INSERT statements.

Viewing the Error Management Display

The error management display shown below in figure 32 indicates to the user any SQL connection errors and execution errors. The 4 possible SQL interface errors may occur during connection/disconnection attempts, or during execution of the SELECT or INSERT statements. Each of these cases has an appropriately labeled error dialog indicator. These indicators are intended to assist the user in debugging, and isolating communication problems between the SQL interface and the respective server.

Error Management																											
Error Out After CONNECT	Error Out After SELECT	Error Out After INSERT	Error Out After DISCONNECT																								
<table><tr><td>status</td><td>code</td></tr><tr><td>no error</td><td>0</td></tr><tr><td>text</td><td></td></tr></table>	status	code	no error	0	text		<table><tr><td>status</td><td>code</td></tr><tr><td>no error</td><td>0</td></tr><tr><td>text</td><td></td></tr></table>	status	code	no error	0	text		<table><tr><td>status</td><td>code</td></tr><tr><td>no error</td><td>0</td></tr><tr><td>text</td><td></td></tr></table>	status	code	no error	0	text		<table><tr><td>status</td><td>code</td></tr><tr><td>no error</td><td>0</td></tr><tr><td>text</td><td></td></tr></table>	status	code	no error	0	text	
status	code																										
no error	0																										
text																											
status	code																										
no error	0																										
text																											
status	code																										
no error	0																										
text																											
status	code																										
no error	0																										
text																											

Figure 32: SQL interface display of connection or execution errors.

APPENDIX C

NNPLS Monitoring Interface Set Up and Tutorial

The use of the monitoring interface is described beginning with preliminary set-up procedures followed by illustrations and discussions.

Preliminary Set-Up

Prior to using this interface, the development interface must be run, and a set of model parameters must be available in a directory `c:\pls_dll\new_model`. This is the directory that the development interface is set to write the model parameter files to. Please consult the development interface tutorial for further assistance.

Before getting started, 3 text files need to be created and stored in the same directory as above (`c:\pls_dll\new_model`):

1. `description.txt` – contains the description for the signals in the model in the proper order. An example description would be Unit Gross Generation. Put one description per line in the file.
2. `tag.txt` – contains a specific tag or identifier for each of the signals. Again in the proper order, and having one entry per line. An example would be 9JGROSS.

3. unit.txt – contains the signal units for each of the signals in the proper order, with one entry per line. Example, MW.

A data file must be available as well, in a tab delimited text format, where the number of columns is equal to the number of signals in the current model. No restrictions are present on the number of data samples (rows). Ensure that the columns are ordered in the same manner as they were during the construction of the models using the development interface. A sample data file has been provided (test.txt) in the required format (tab-delimited text), which corresponds to the train.txt and val.txt files also supplied for use in development. The front panel of the monitoring interface can be accessed by opening the LabView virtual instrument Monitor.vi.

Brief Instructions on Using the Monitoring Interface

1. After completing the preliminary set-up as above, set a proper path and filename for the data to test. Also set paths and filenames for the stored measurements and predictions. These are set to default as c:\temp\lvpredictions.txt and c:\temp\lvmeasurements.txt. LabView is set to overwrite these files if they already exist.
2. Set the sampling rate, and run the VI by depressing the run arrow on the LabView toolbar.

3. Aside from stopping the VI and beginning with a new data file, there are really no other controls except for the plot control for the specification of which sensor's data should be plotted. Descriptions of all of the features of the monitoring interface are provided in the following sections.

Setting the File Names and Paths in the Data File Control

Figure 33 shows the required data file control settings. These file paths and file names should be specified prior to running the monitoring interface. The browse tabs are provided to ease file selection and location tasks. In all cases the file name extensions should be used (.txt). This is recommended to ease in the transfer of data from one application to another. Each of the controls shown in figure 33 is described in more detail below.

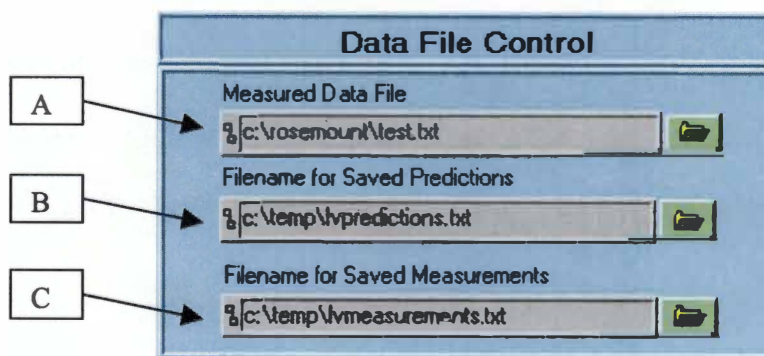


Figure 33: Monitoring interface data file control.

A: Measured Data File – input data file corresponding to the most recent model. The format must be tab-delimited text, where the number of columns is equal to the number of signals in the model.

B: Filename for Saved Predictions – filename where the predictions will be stored for reference. If the monitoring interface is run more than once without changing this filename, the file will be overwritten. This is generally a desired result during testing stages to avoid excessive file creation.

C: Filename for Saved Measurements – filename where the measurements will be stored. Note that these values will be median filtered with a filter rank corresponding to the setting used during development. This is to directly compare the inputs and outputs of the NNPLS model for reference, since the predictions are based on the filtered data. If no filter was used during development, than no filter will be applied here either.

Operating the Signal Controls

Figure 34 shows the two signal controls as well as the display of signal information corresponding to the selected signal. Descriptive and numerical information is displayed in the signal control box, and further explanations about the displayed information are provided below.

Signal Control	
Sampling Rate (sec)	Signal to Plot
0	1
Tag Name	
9FFW1	
Signal Description	
Feedwater Flow #1 - Raw	
Measurement	
1387.46	klb/hr
Prediction	
1392.77	klb/hr
Residual	
-5.30	klb/hr

Figure 34: Monitoring interface signal controls and indicators.

D: Sampling Rate – control to set the sampling rate in seconds. Since the program is set to read from a file, the only usefulness of a sampling rate is to slow the program down so the numerical values can be observed visually. If desired, this control can be set to zero and the resultant measurements and predictions can be retrieved from the data files as set in B and C of figure 33.

E: Signal to Plot – controls the signal plotted in the graphical display of figure 35.

F: Tag Name – indicates the signal tag name corresponding to the signal selected by control E above.

G: Description, Measurement, Prediction, and Residual – these indicators provide a description of the signal, and the signal's current measured, predicted and residual

values. The residual is the prediction subtracted from the measurement. These indicators also correspond to the signal selected by control E above.

Graphical Display

The signal represented in the graphical display is controlled by control E of figure 34. The upper plot shows the measured signal (red), and its corresponding prediction (white). The lower plot shows the residual difference. These plots are set to not exceed a horizontal range of 100 data points, though they will continuously be updated through the removal of the oldest values to make room for the most current values. If desired this can be adjusted by changing the “Number of Points to Plot” within the Monitor.vi diagram. It’s labeled accordingly and is near the upper left of the diagram (not the front panel). Also if x-axis values are not desired, right-click on the plots with the arrow-tool, and you can adjust the display properties (look under the visible items option from the pop-up menu after the right-click). This can be done from the front panel if the VI is paused or stopped. Figure 35 shows an example of the graphical display, and each plot is further described below.

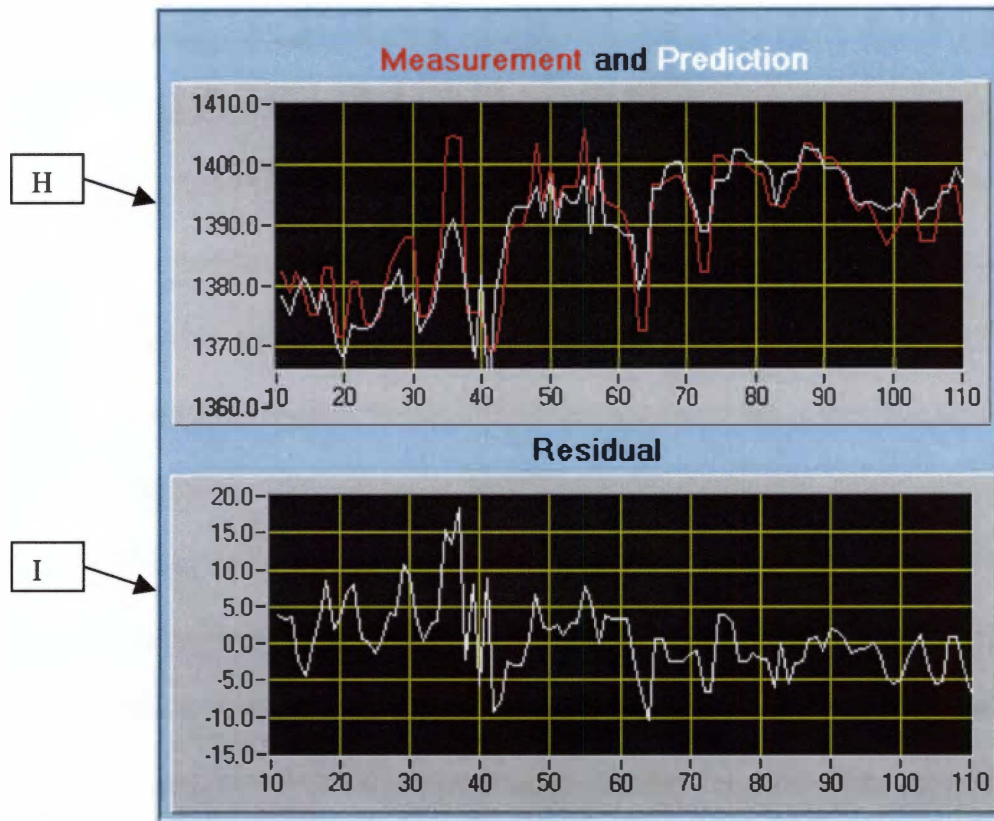


Figure 35: Monitoring interface graphical display.

H: Measurement and Prediction Plot – Display of the most recent 100 measurements and predictions. The plotted measurements have been filtered according to the median filter rank specified in the `c:\pls_dll\new_model\rank2.txt` file. This file is written by the development interface and should not be altered manually. The predictions are made based on the median filtered measurements.

I: Residual Plot – Display of the most recent 100 residuals. The residuals are calculated based on the difference between the filtered measurements, and the predictions made based on the filtered measurements. The residuals are based on the

post-processed signals, and correspond to the difference between the signals plotted in the measurement and prediction plot.

Tabulated Display

The tabulated display is also available to view the predictions for all of the signals simultaneously. In addition, the status of the sensors is indicated along with controls to reset validation for sensors that had entered alarm conditions. The display is set to adjust automatically to accommodate all of the signals in the input data file, though the border surrounding the tabulated display may need to be adjusted manually if it is not large enough to accommodate all of the signals in the specified data file. Figure 36 shows an example of the tabulated display, and each of the indicators is further described below.

Tag Names	Signal Descriptions	Signal Units	Measurements	Predictions	Residuals	Status	Reset Status	Alarm Reset
9FFW1	Feedwater Flow #1 - Raw	klb/hr	1356.359	1349.394	6.965			Reset
9FFW2	Feedwater Flow #2 - Raw	klb/hr	1367.861	1360.016	7.845			Reset
9UGROSS	Unit Gross Generation	MW	183.749	186.219	-2.470			Reset
91STSTGA	First Stage Pressure A	psig	1262.915	1256.820	6.095			Reset
9PTHROTTLEA	Throttle Pressure at Stop Vlv A	psig	1818.588	1819.499	-0.911			Reset
9PTHROTTLEB	Throttle Pressure at Stop Vlv B	psig	1819.816	1820.726	-0.910			Reset

Figure 36: Monitoring interface tabulated display.

J: Tag Names – Displays the tag names of the specified data file. These tag names are read from the c:\pls_dll\new_model\tag.txt file.

K: Signal Descriptions – Displays the signal descriptions as read from the c:\pls_dll\new_model\description.txt file.

L: Signal Units – Displays the signal units as read from the c:\pls_dll\new_model\unit.txt file.

M: Measurements – Displays the most recently read measured value from the specified data file. Note that these values are reported after median filtering has been applied.

N: Predictions – Displays the most recent predictions based on the filtered measurements.

O: Residuals – Displays the current residual calculated as filtered measurement minus prediction.

P: Status – Displays the sensor alarm status. The example in figure 36 shows all sensors are operating correctly. An alarm would be indicated by a change in color, of the corresponding boolean indicator, to red. When a sensor's status indicator is red, its

measured values are not being utilized in the validation model. Its values are replaced with the validation model's prediction of that sensor. To override sensor replacement, the alarm reset controls can be used (see control R below).

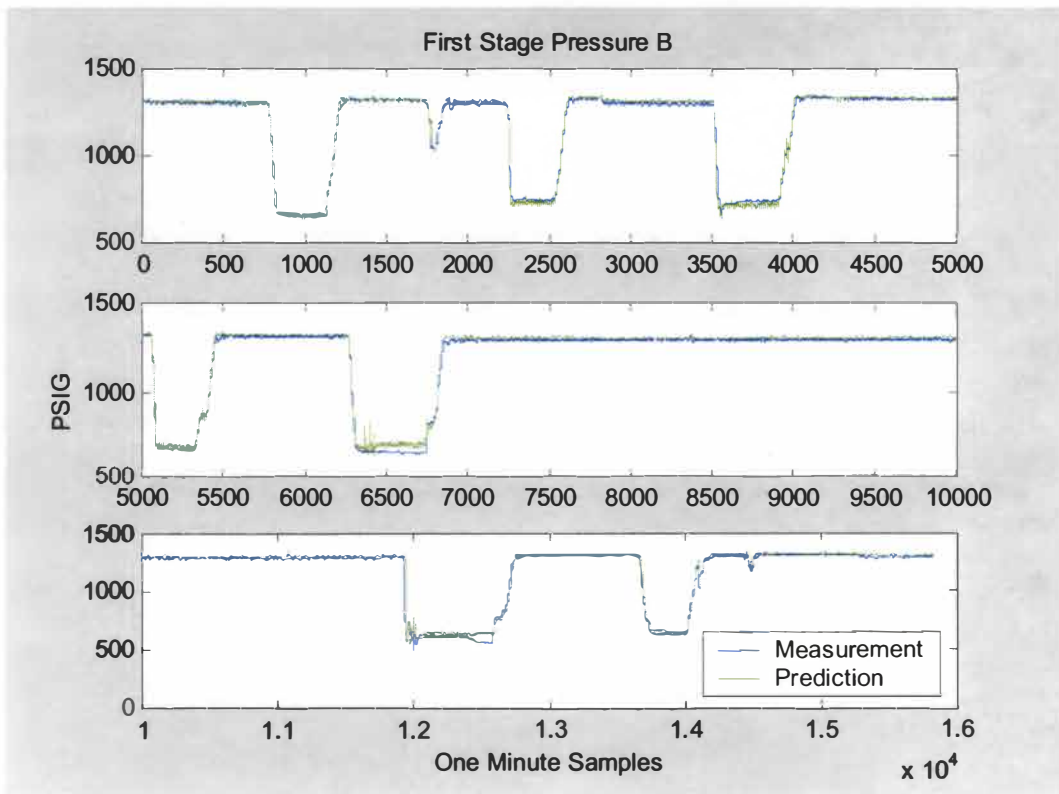
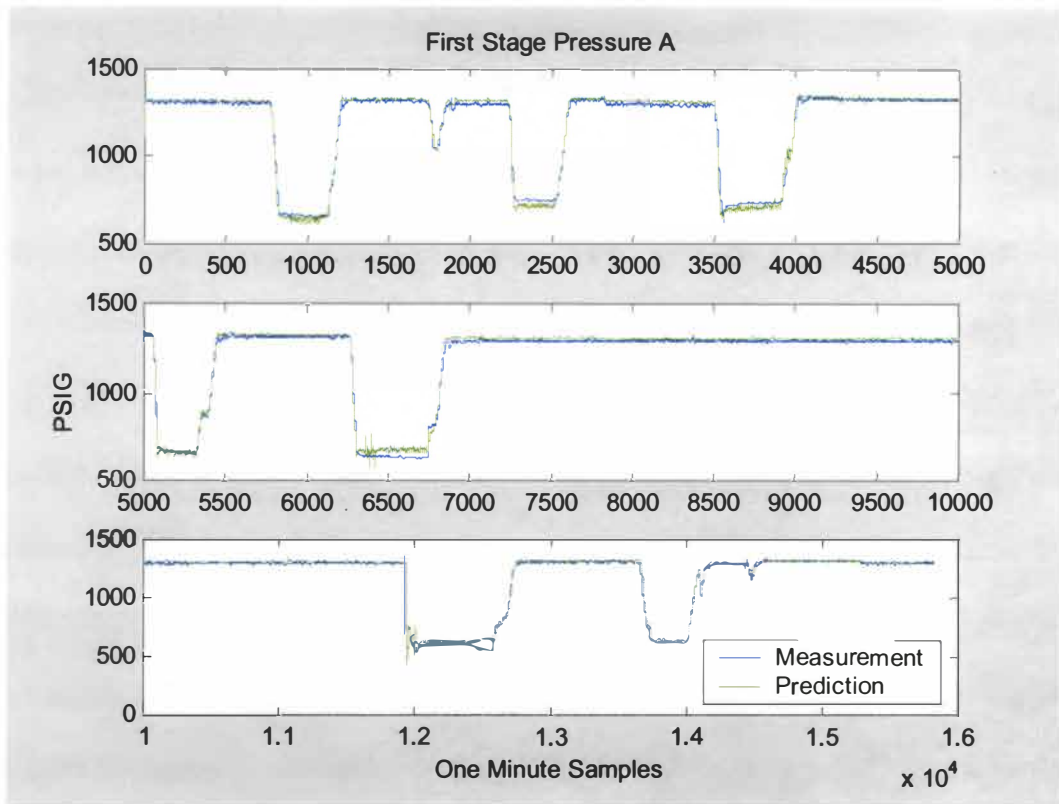
Q: Reset Status – These indicators show when an alarmed sensor can be successfully re-inserted into the validation model without producing an alarm. For example, a drifted sensor's "Status" will turn red, indicating an alarm. The reset status for that sensor will also change to red at this time. When the validation system determines that the alarmed sensor can be successfully re-inserted, its reset status indicator will change to green. At this time, the alarm reset control can be utilized to re-insert the sensor's measurements into the validation model.

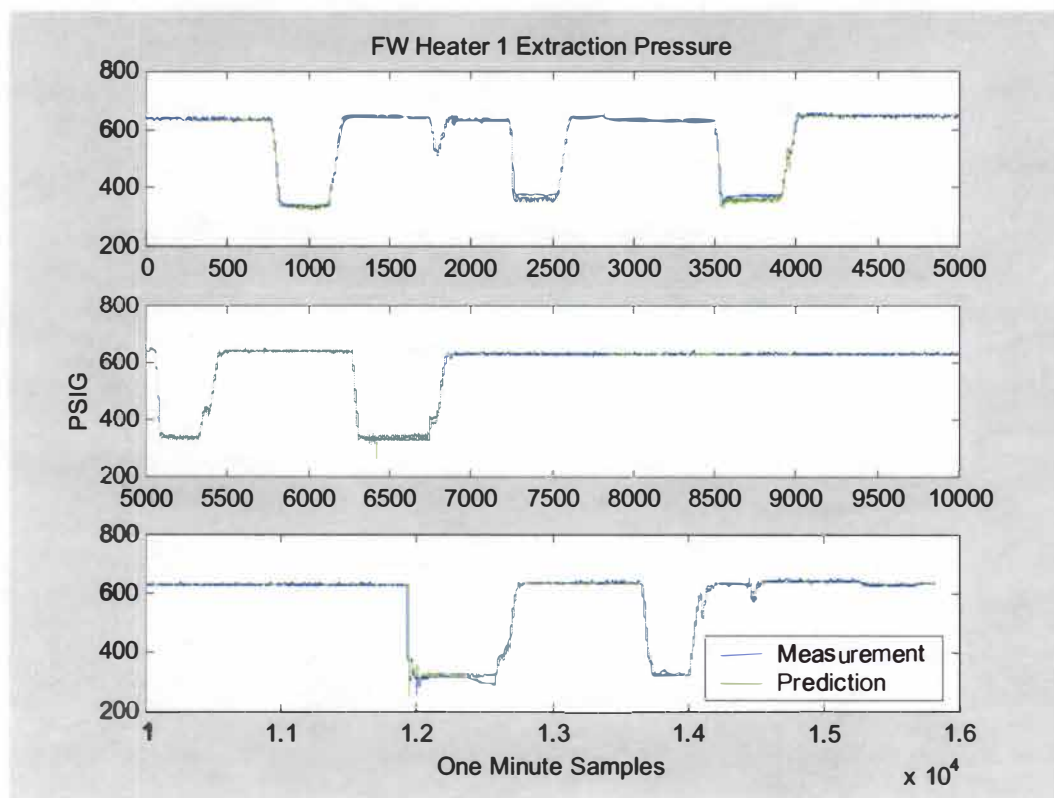
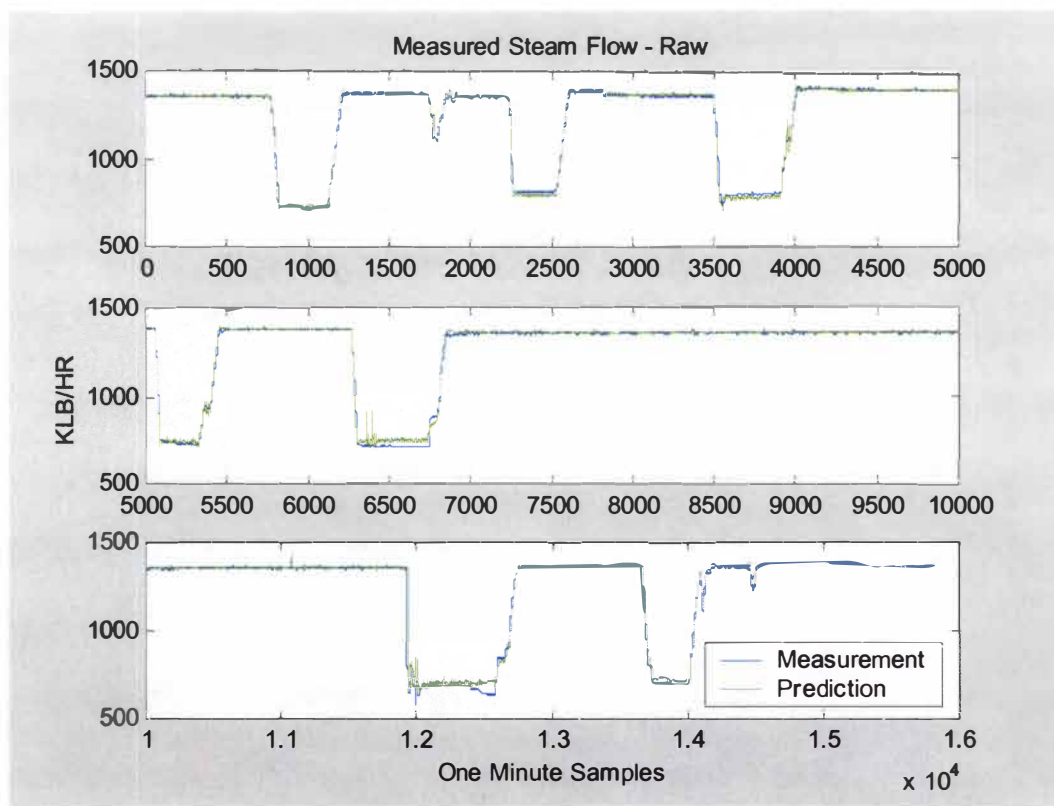
R: Alarm Reset – These controls will interrupt sensor replacement when utilized. Their execution will result in the corresponding sensor's re-introduction into the validation model. If used when no sensors are under alarm conditions, no effect will be observed. These controls should be used only after an alarmed sensor's "Reset Status" indicator is green.

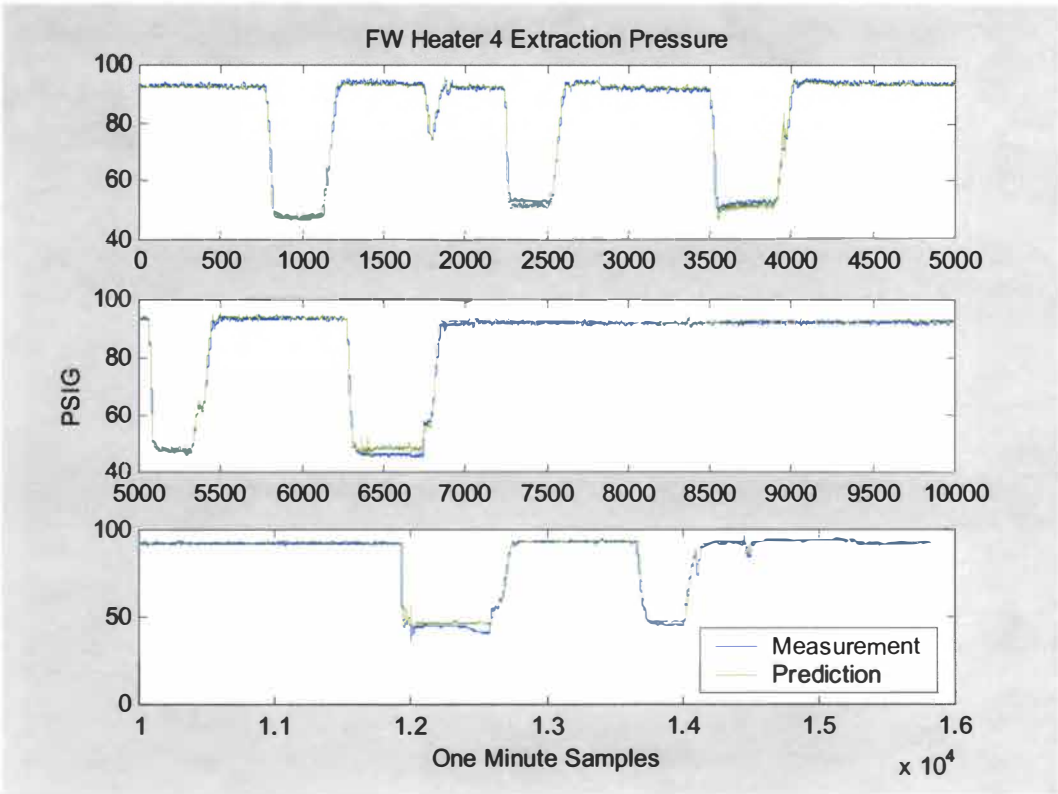
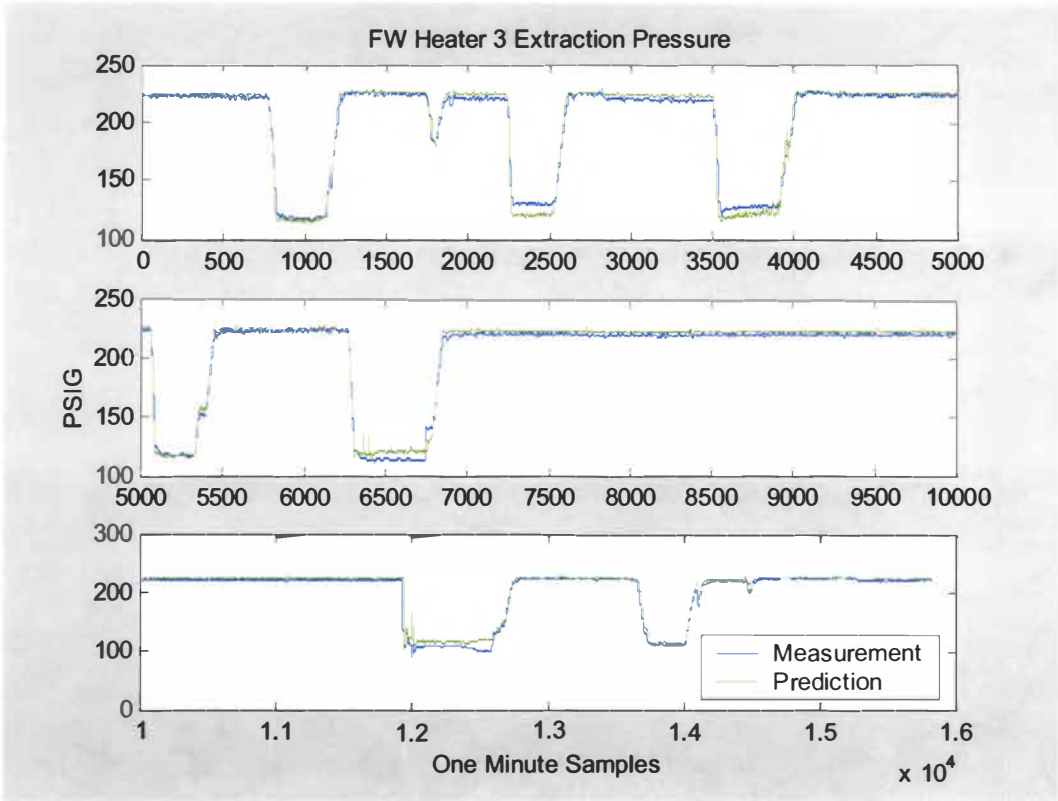
APPENDIX D

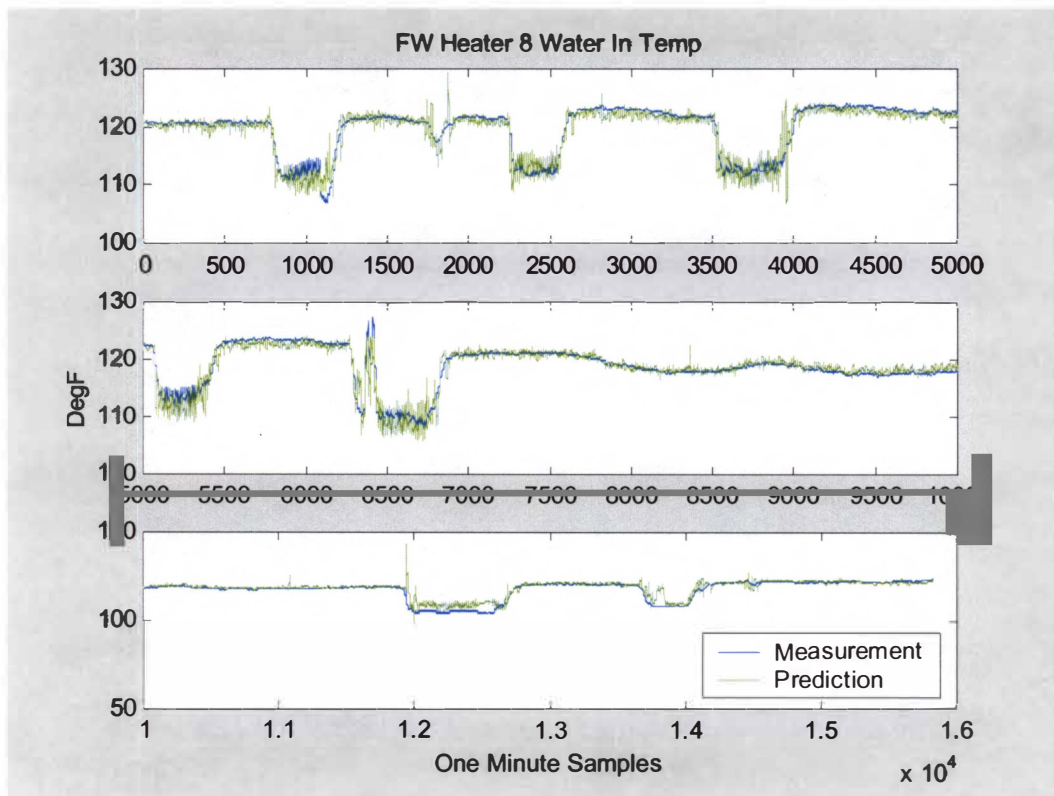
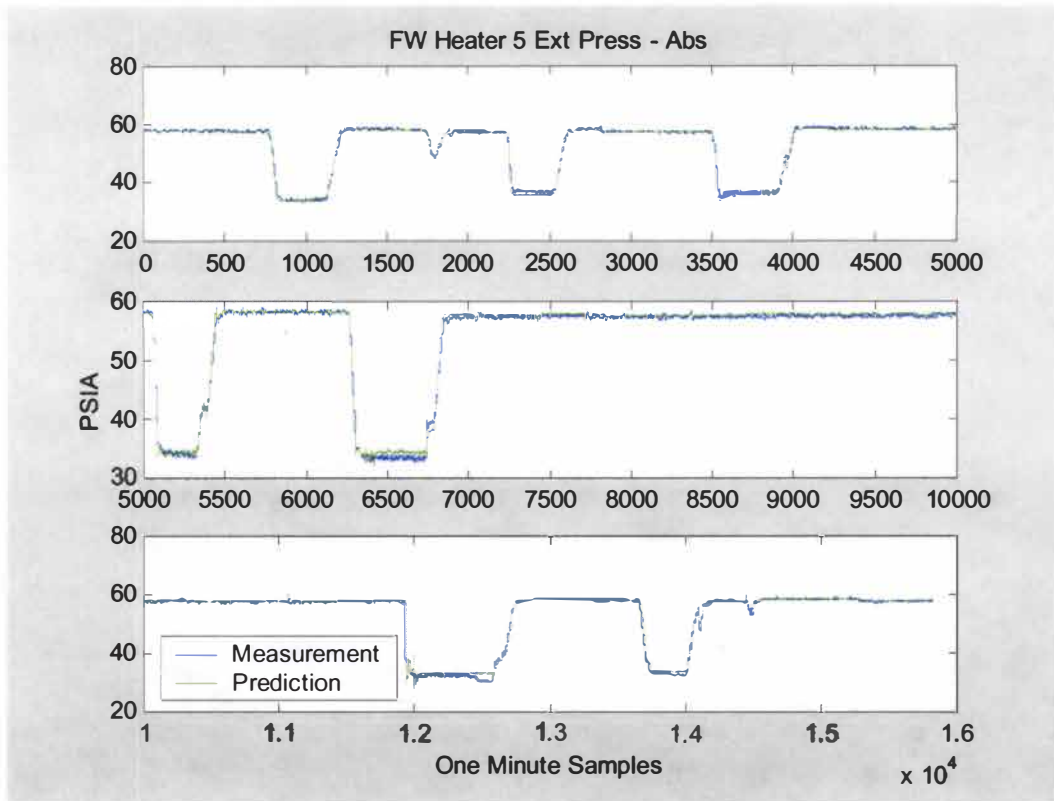
Prediction Plots

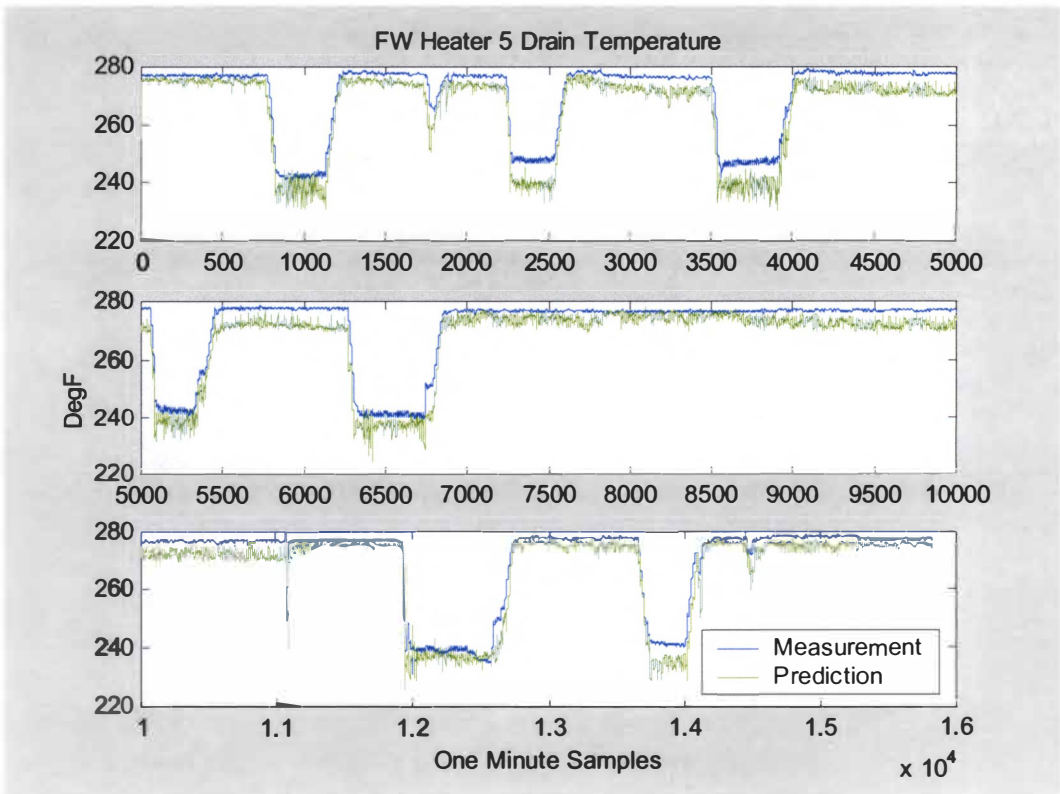
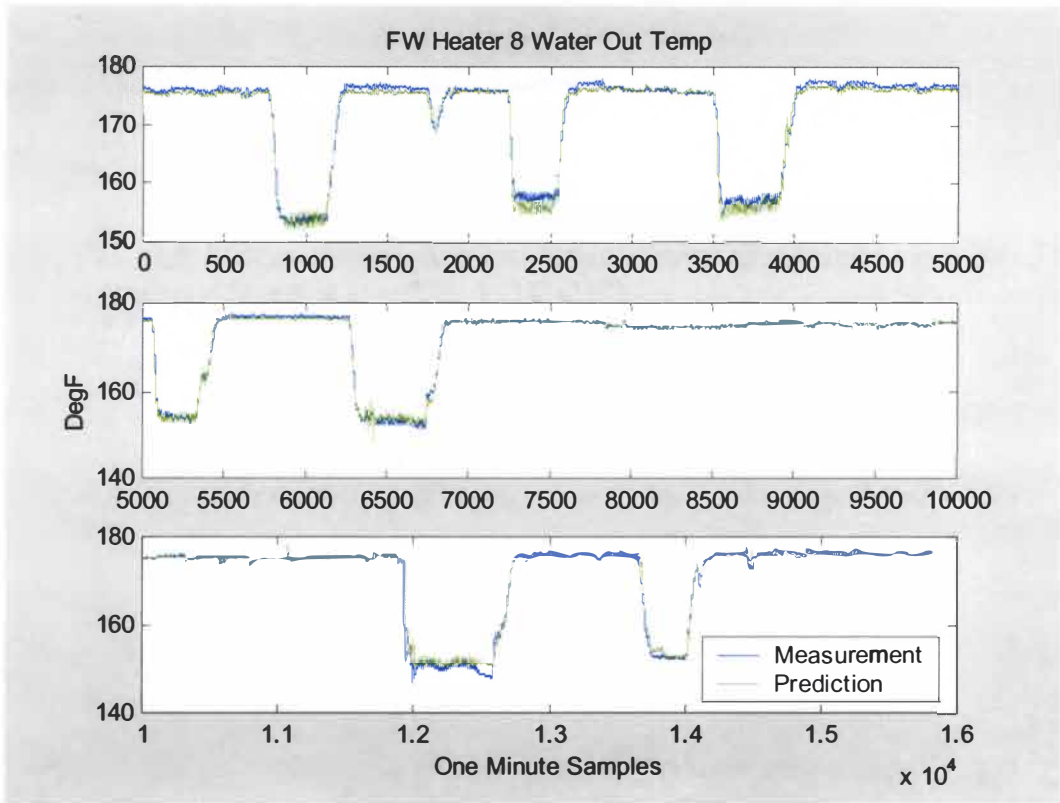
The plots in this appendix show representative prediction performances for each of the 51 signals in the NNPLS ISCV model. Each plot is divided into 3 subplots, where the samples are continued from one plot to the next. For example, the first subplot contains the first 5000 consecutive samples, the second subplot contains the next 5000, etc. The total number of samples shown for each signal is approximately 15000.

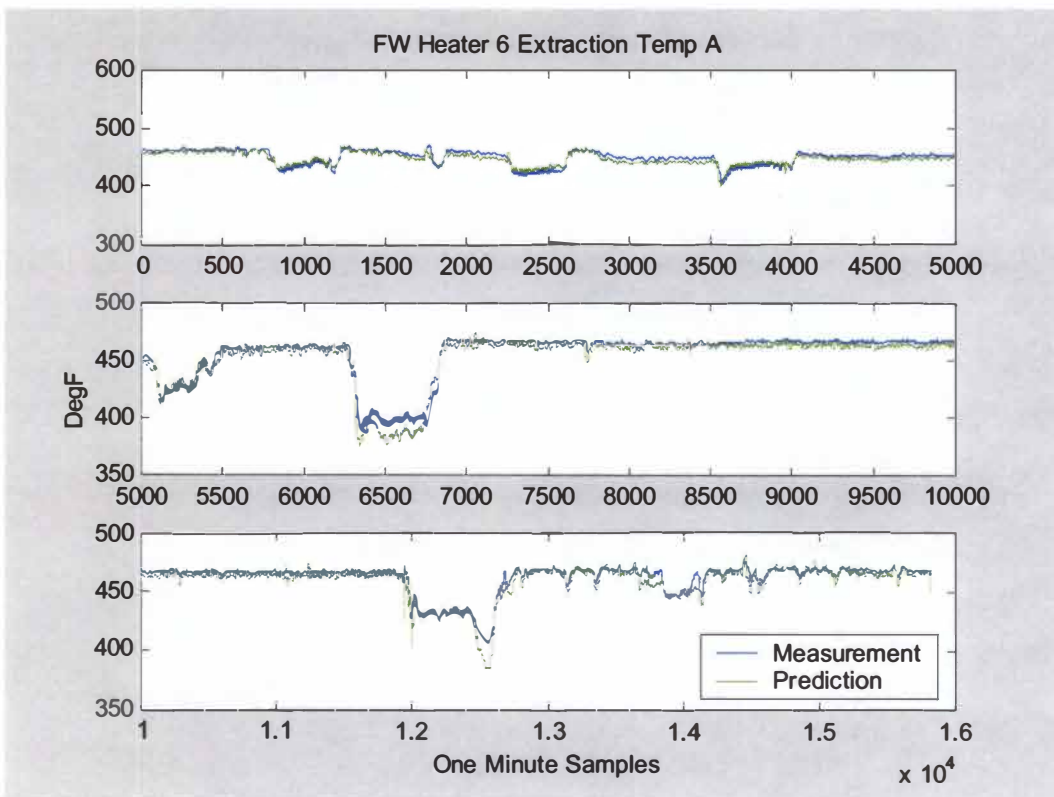
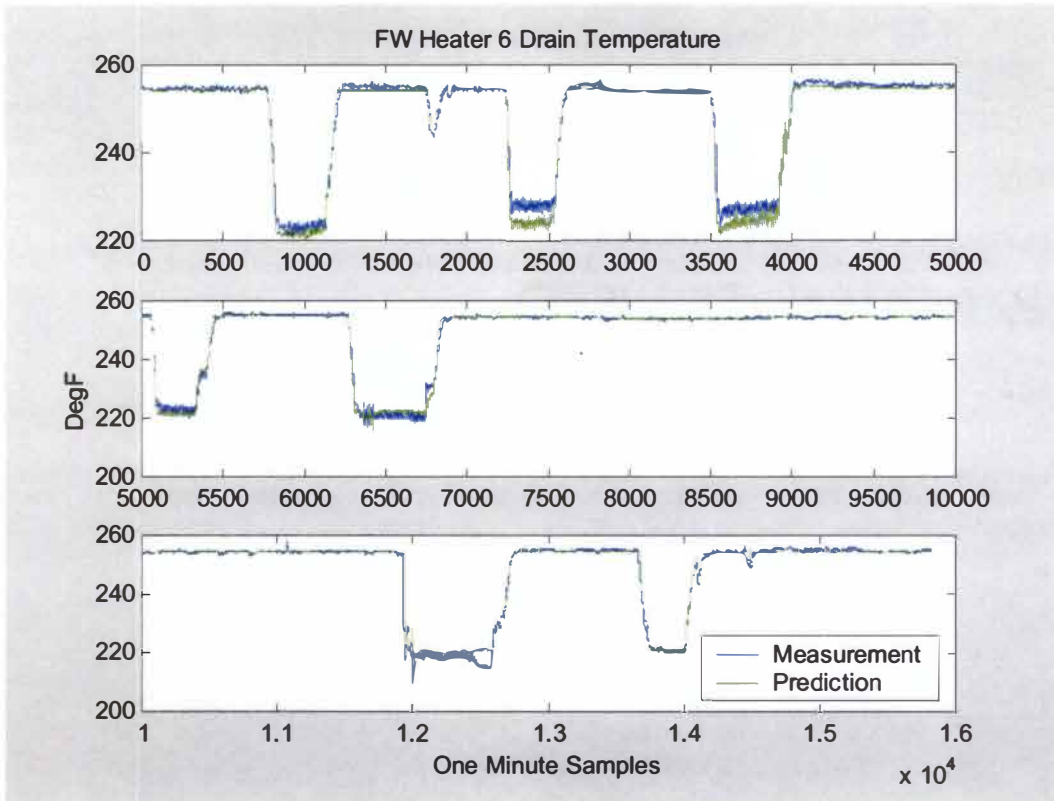


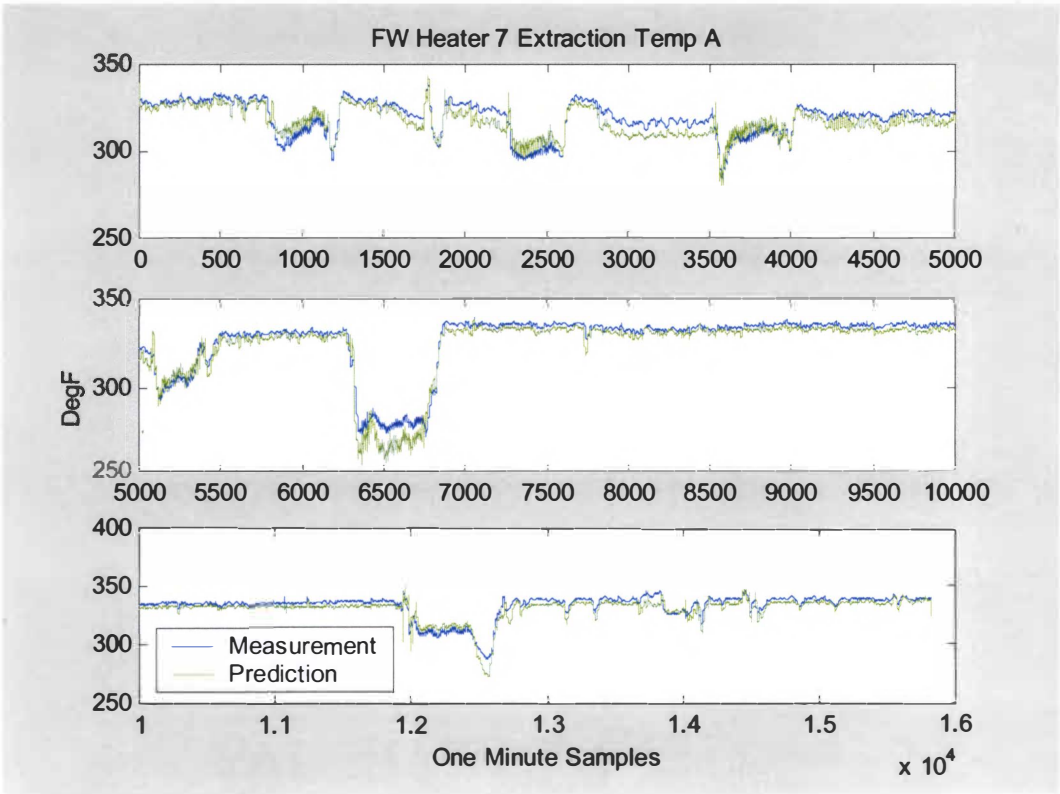
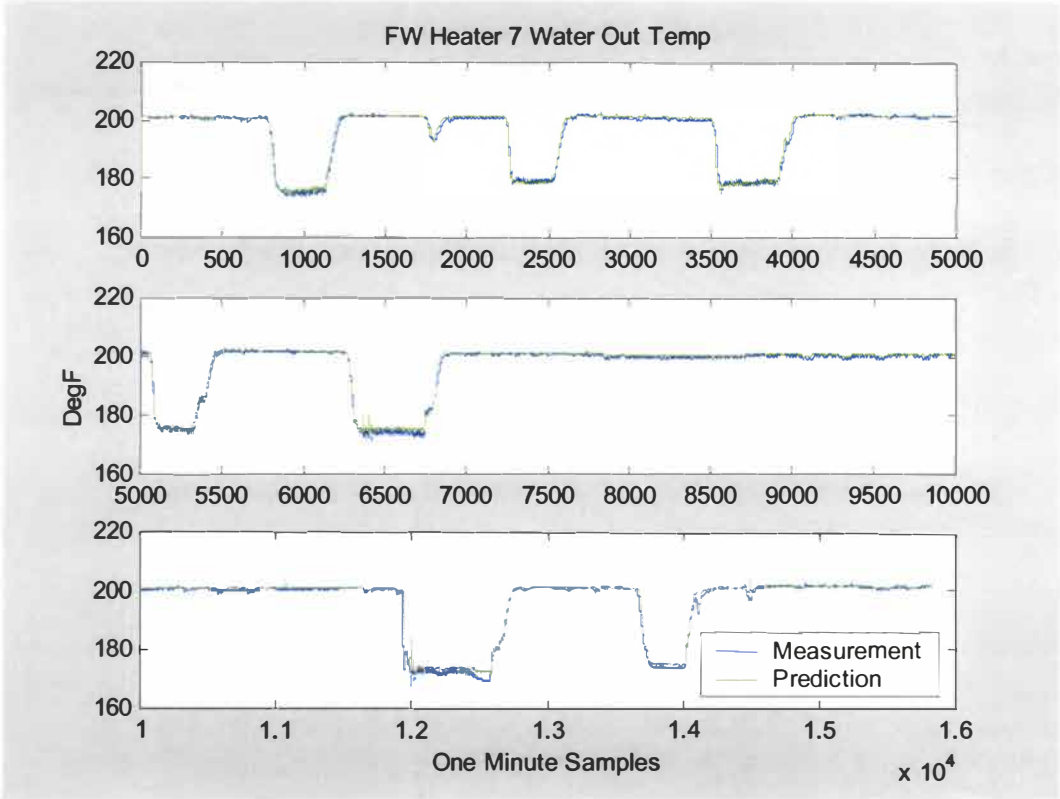


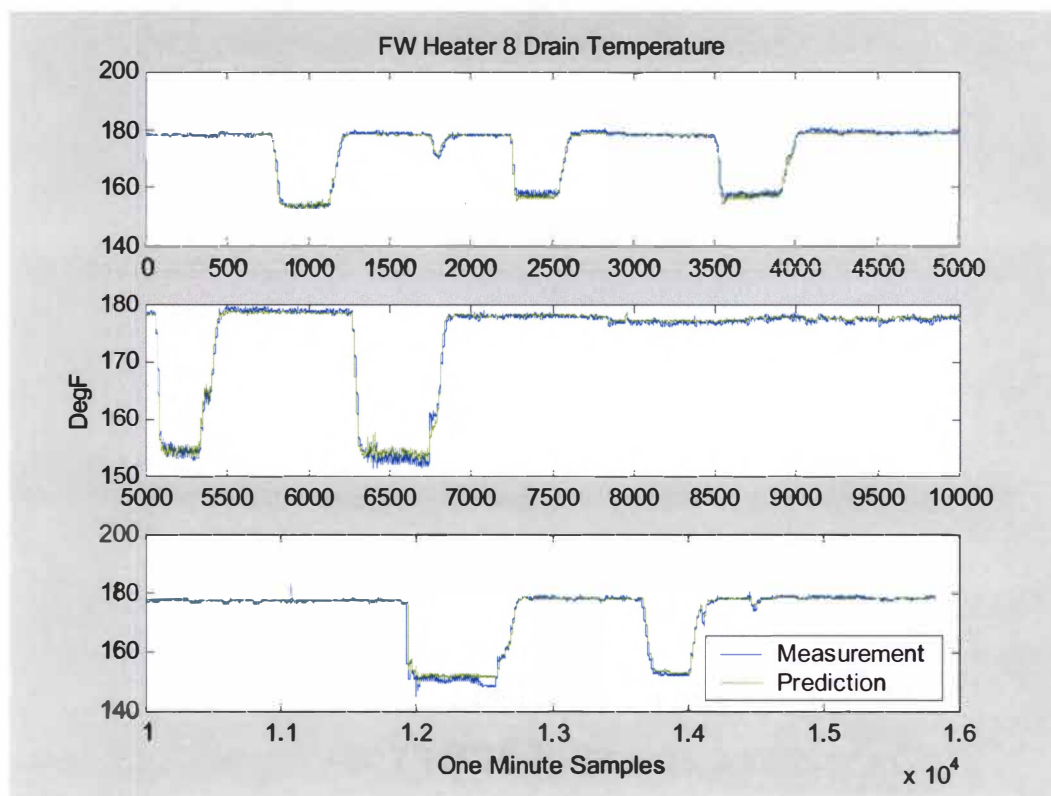
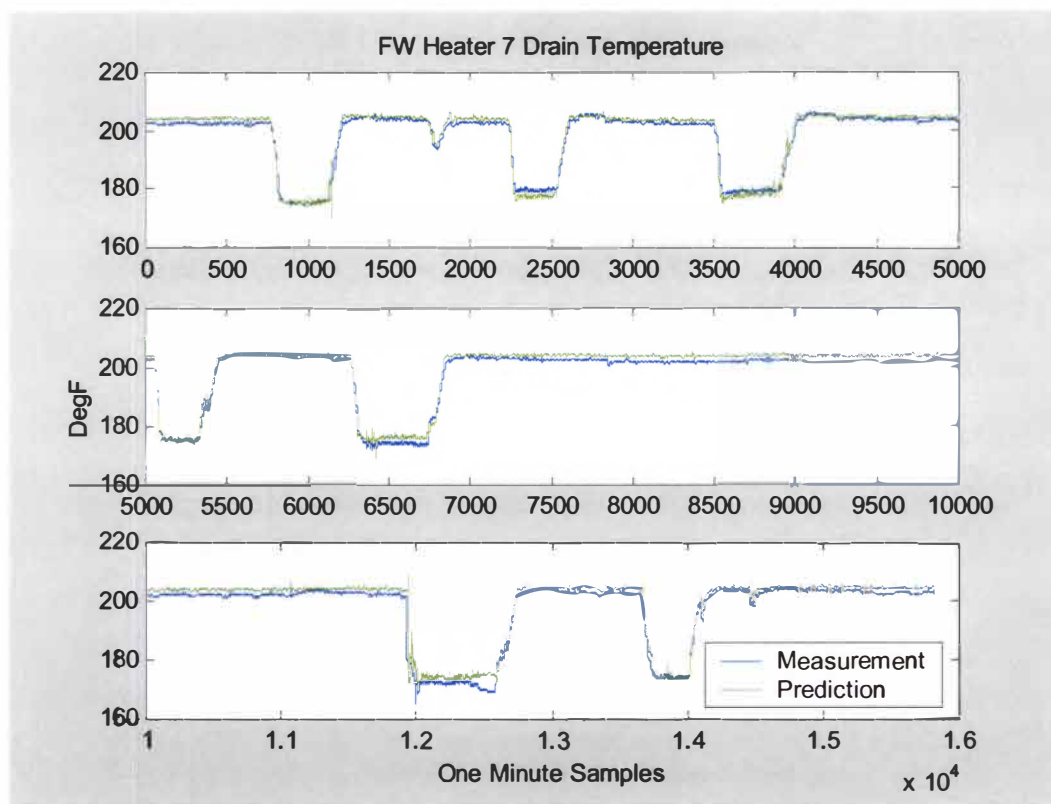


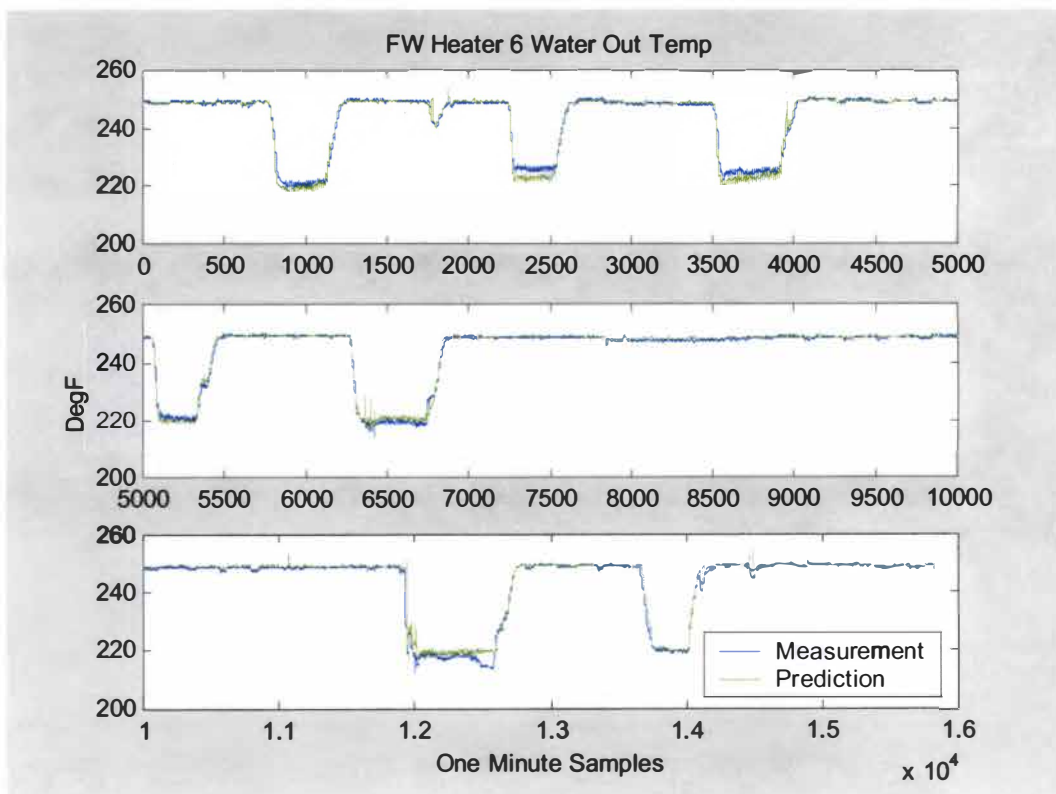
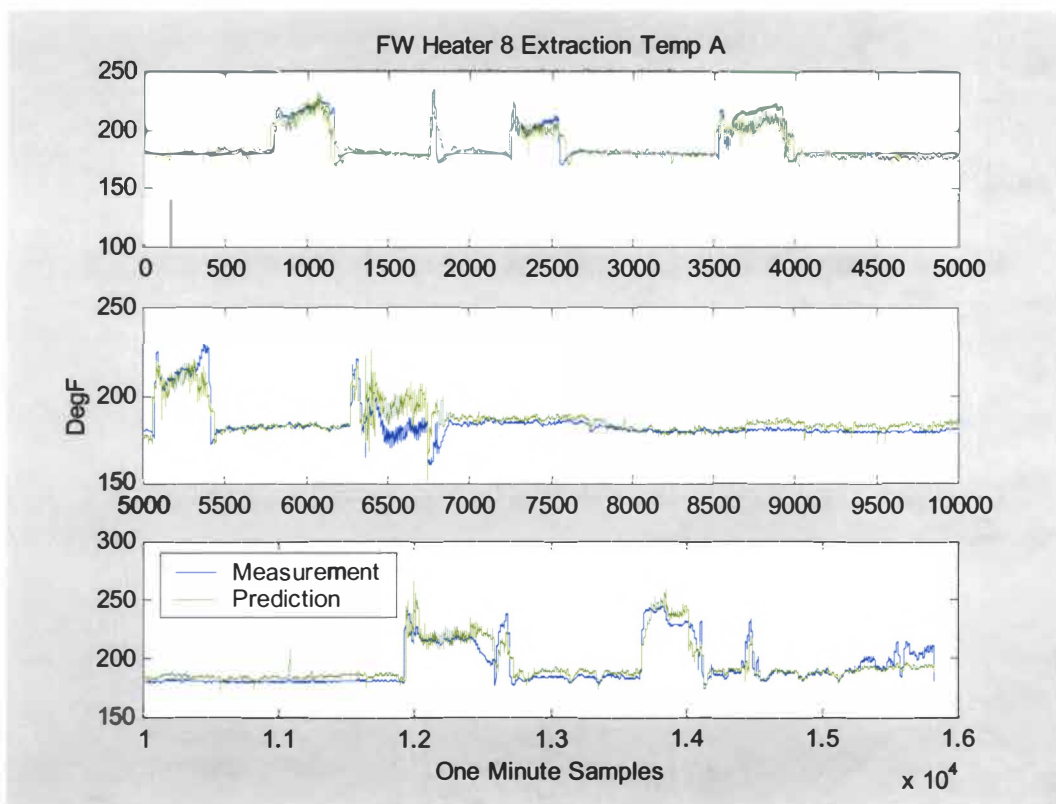


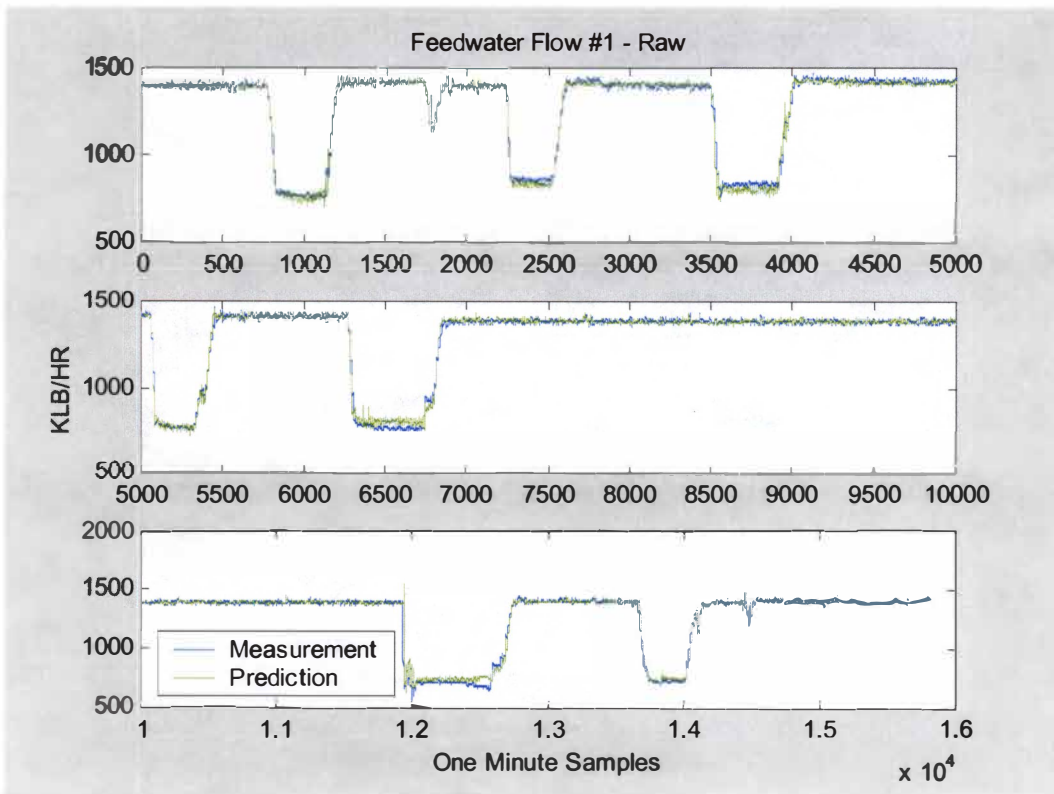
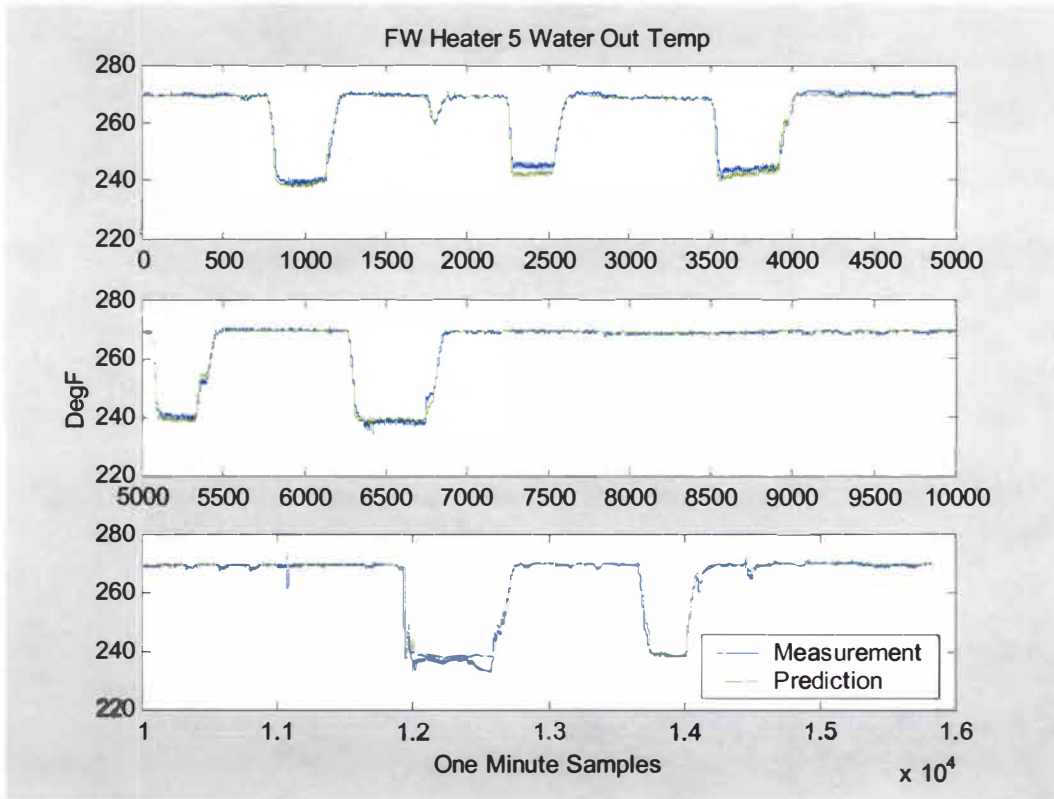


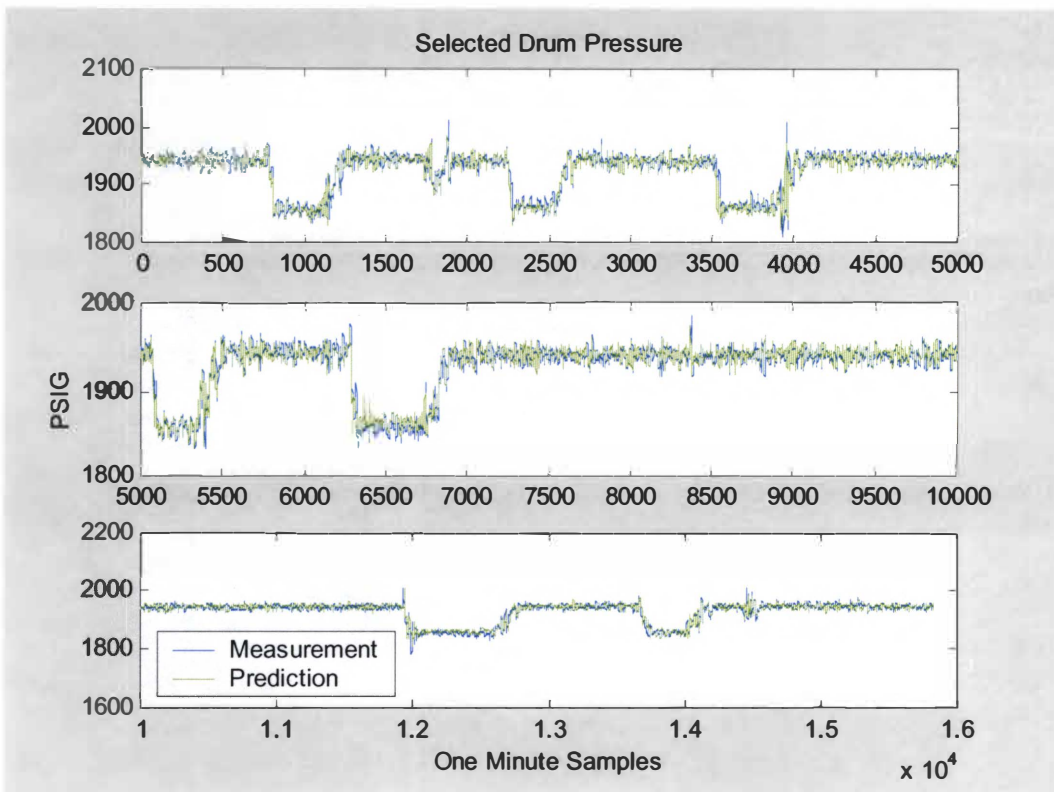
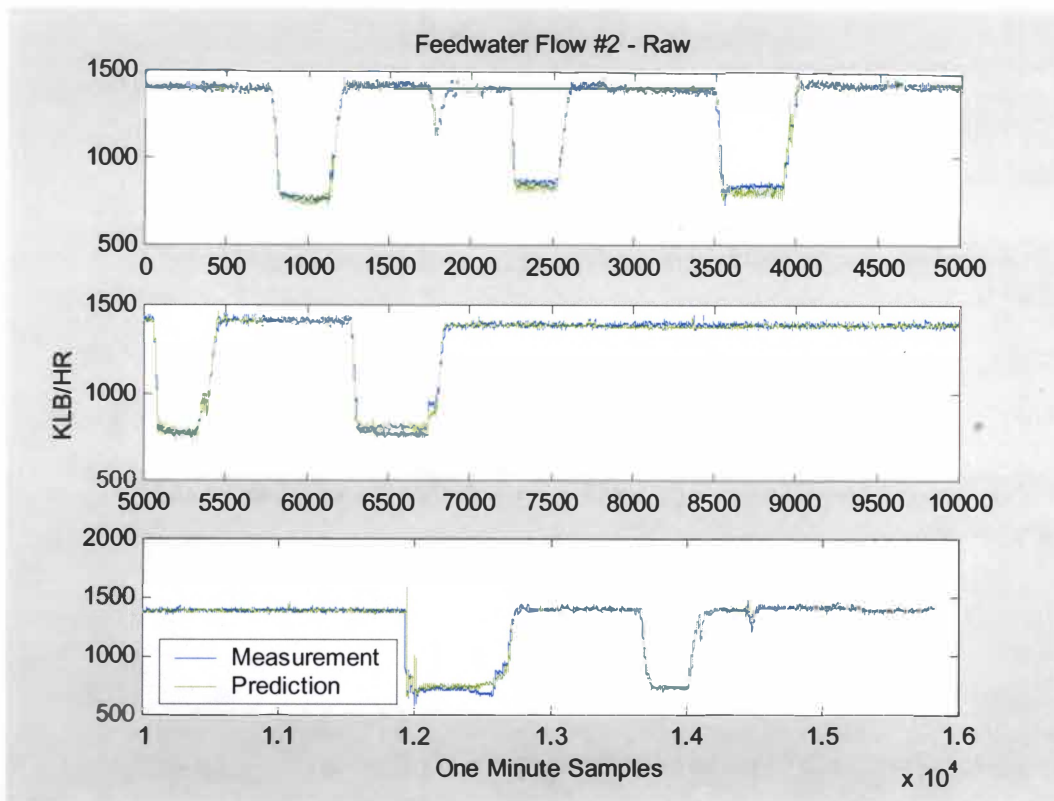


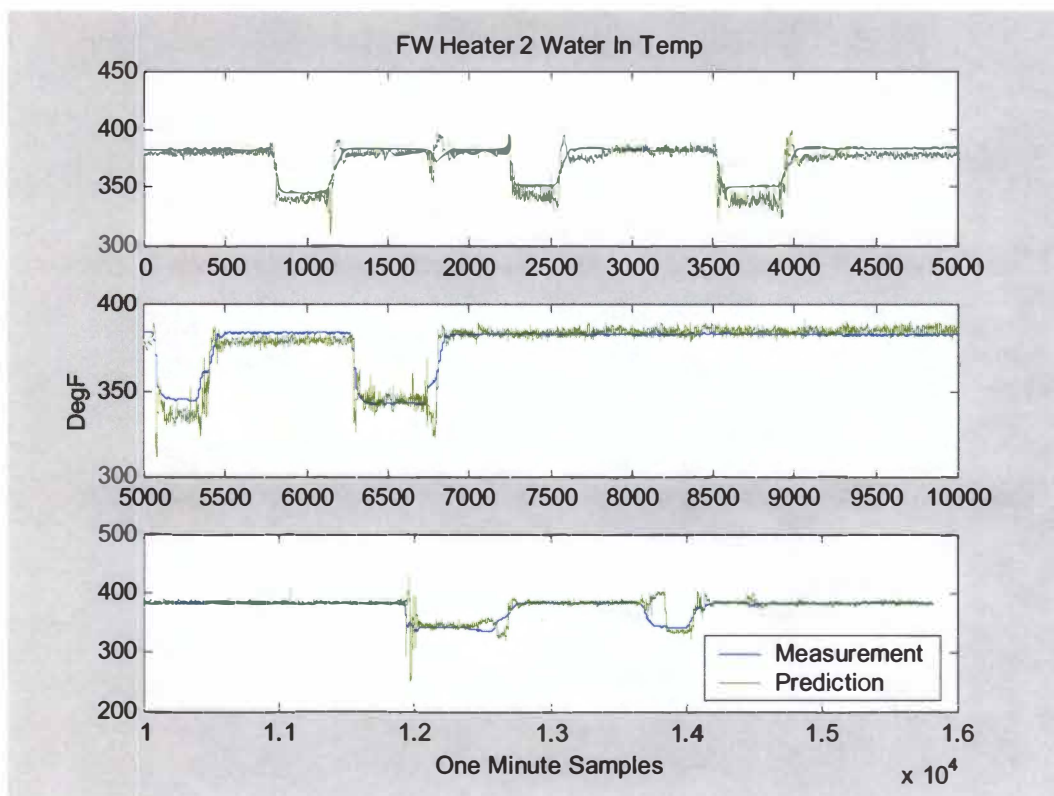
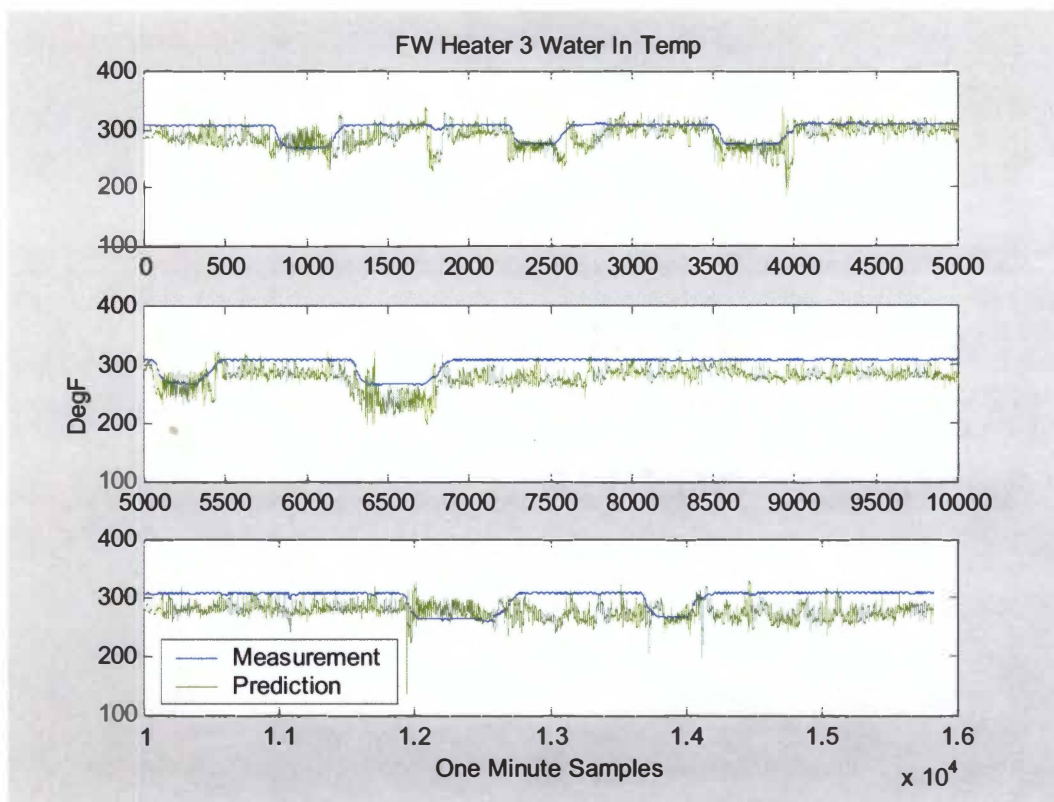


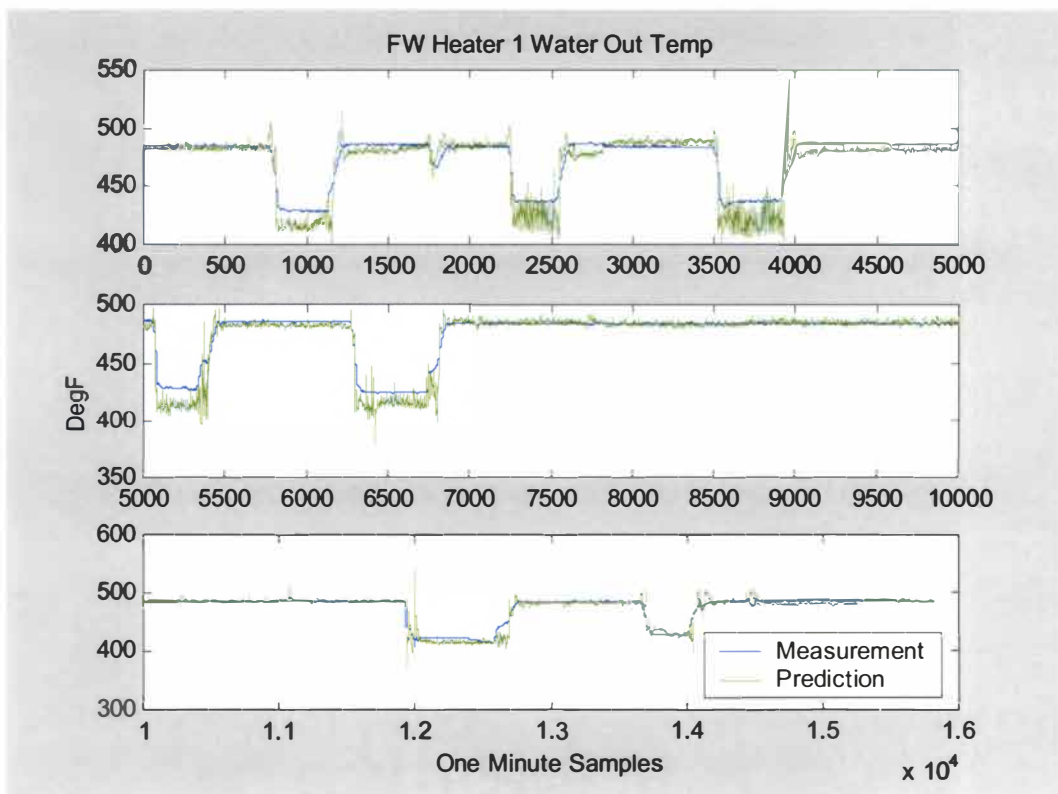
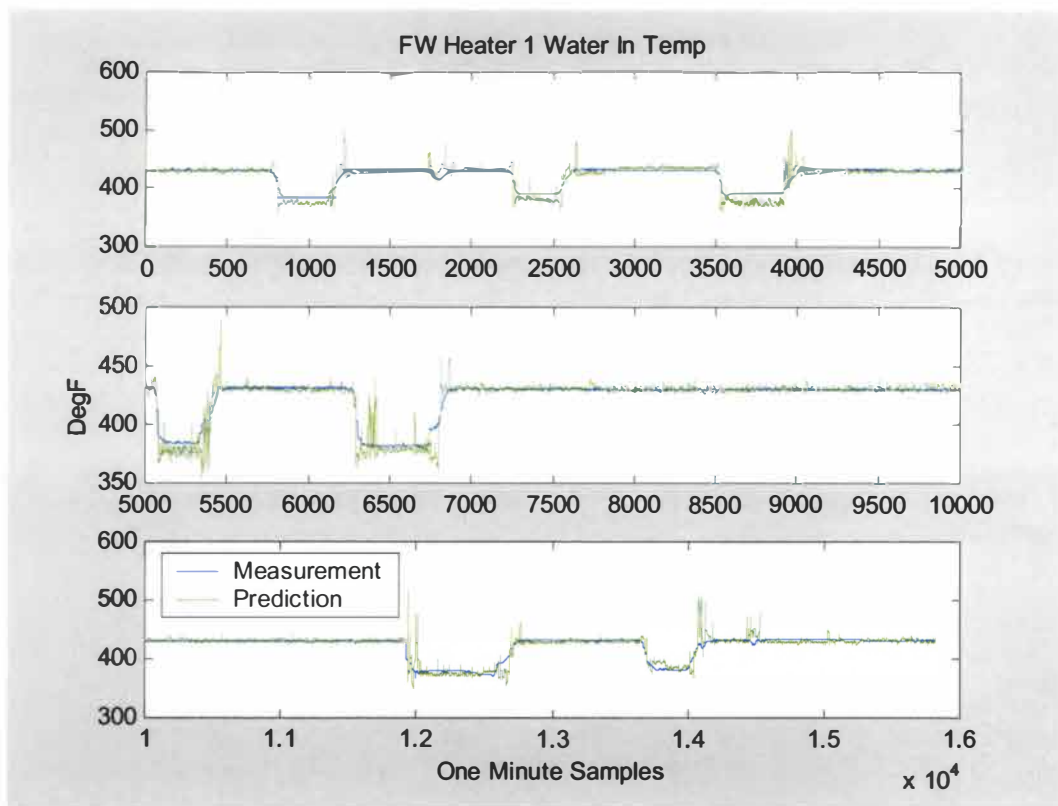


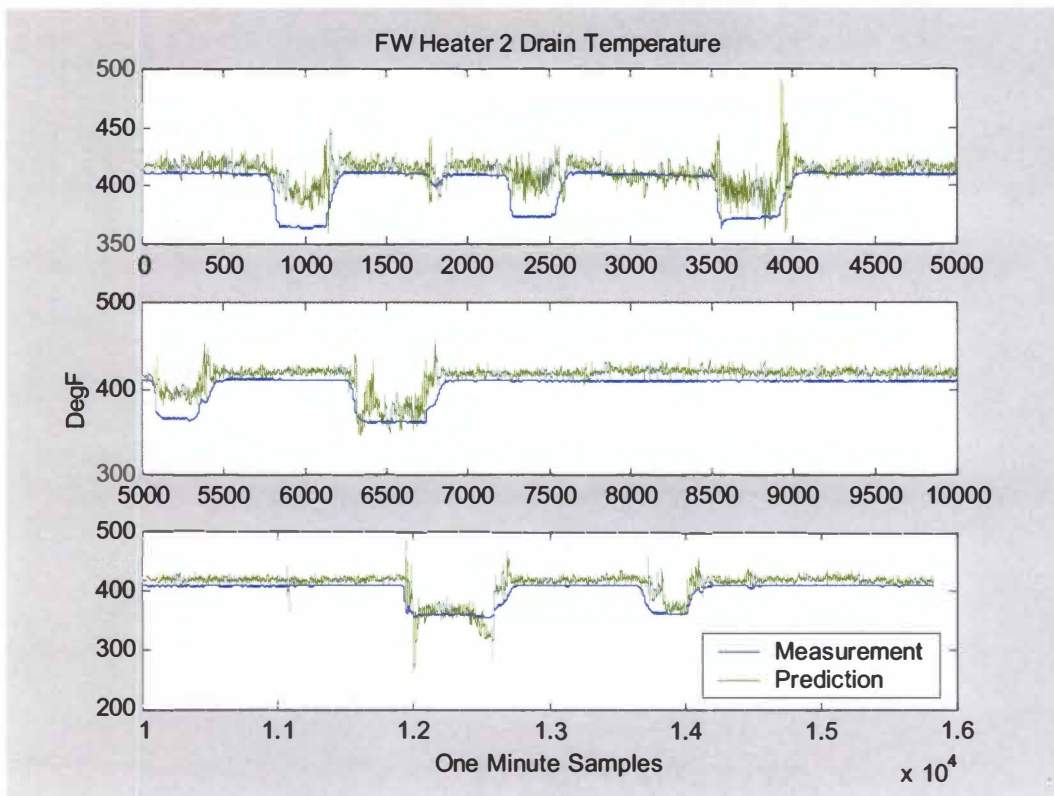
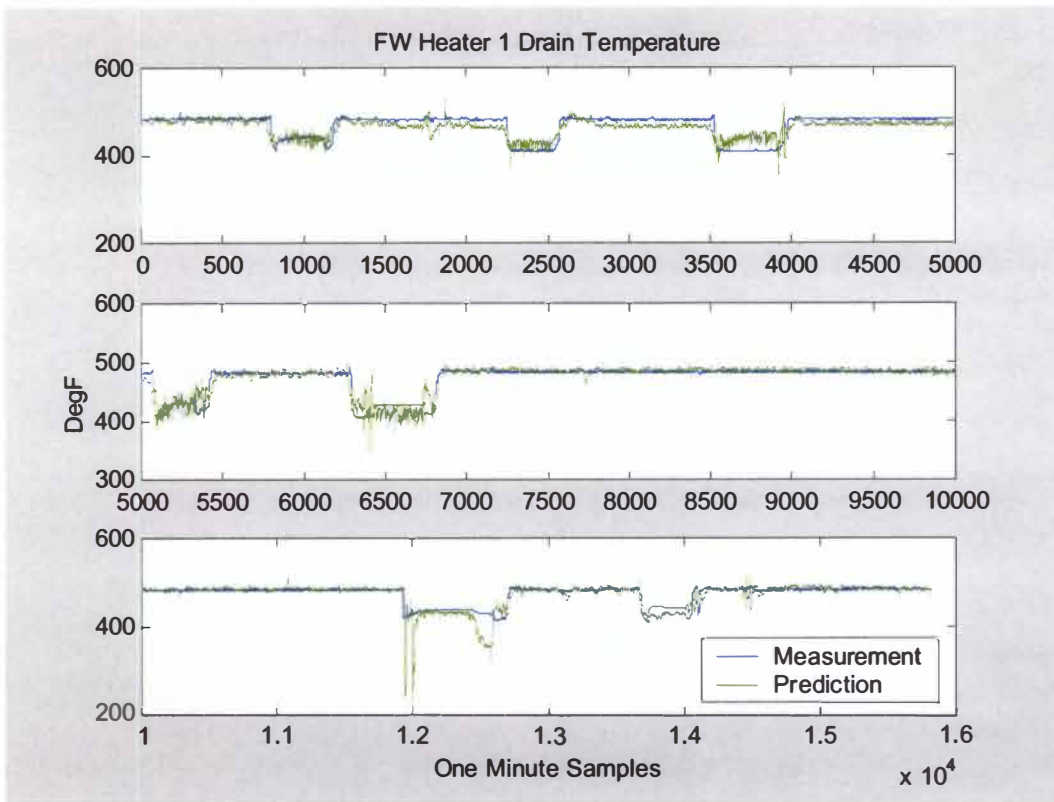


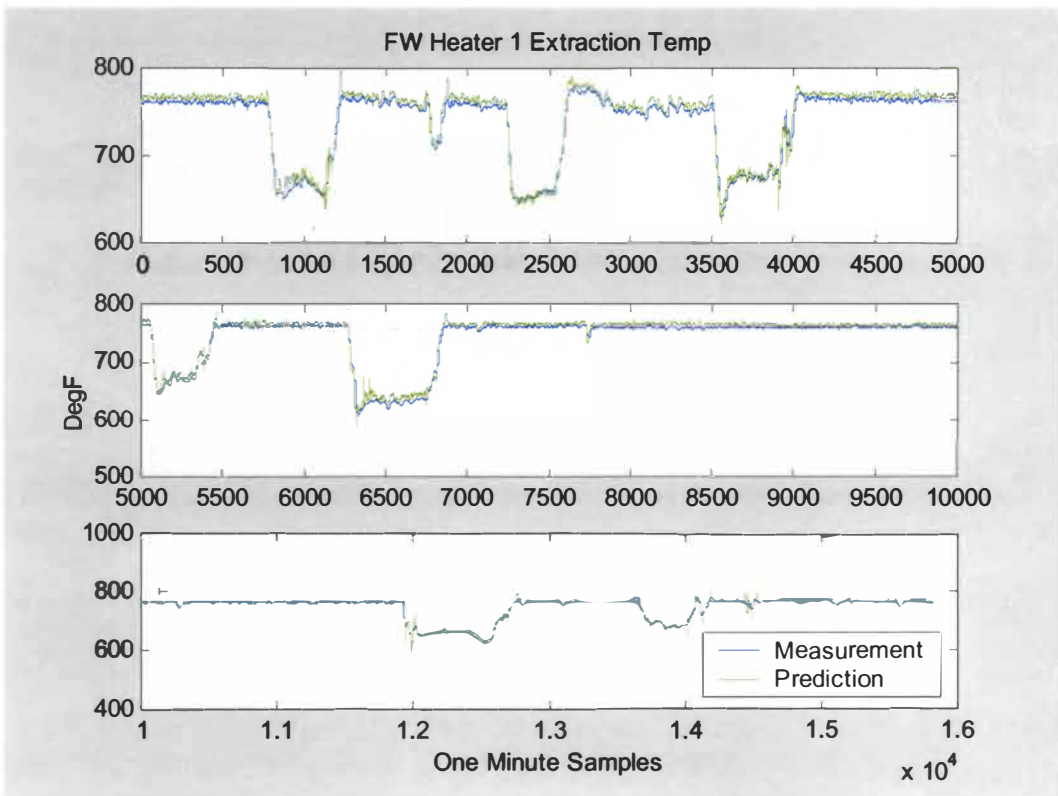
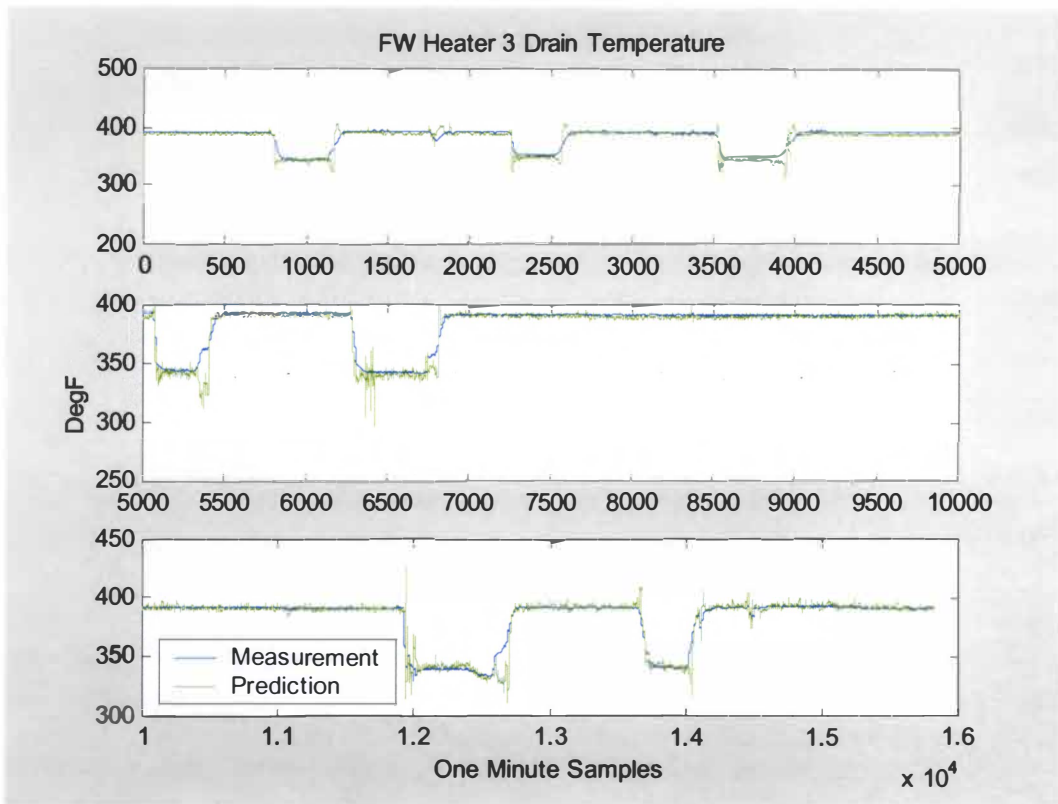


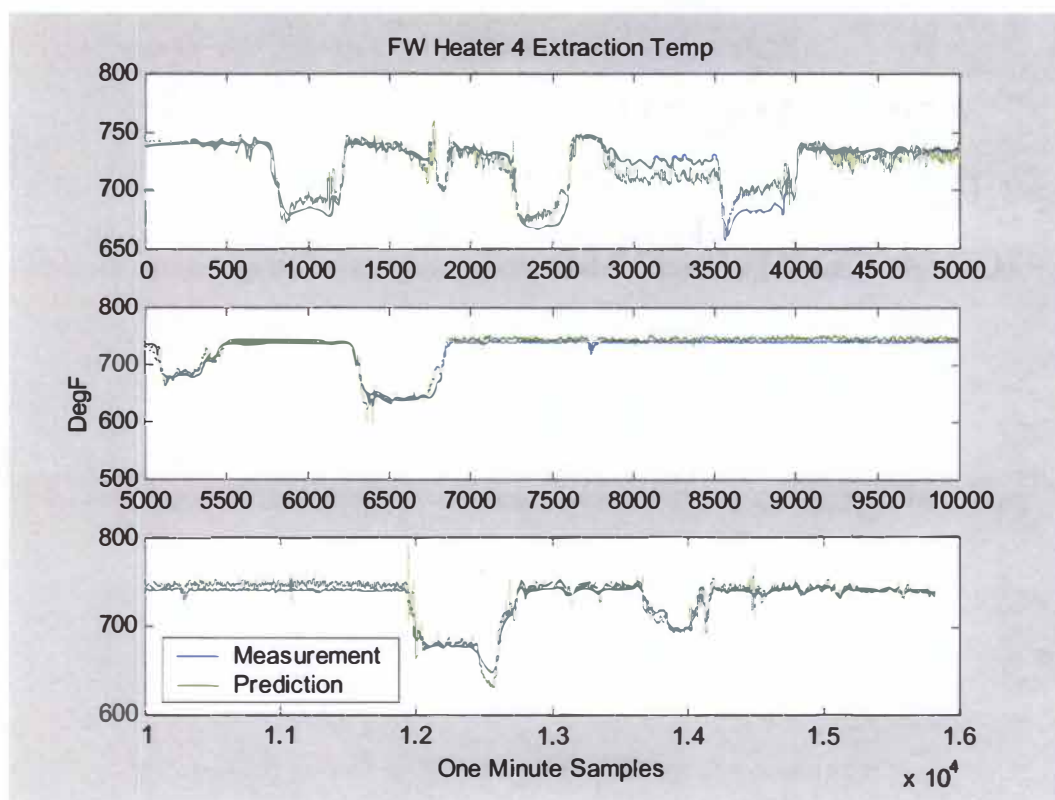
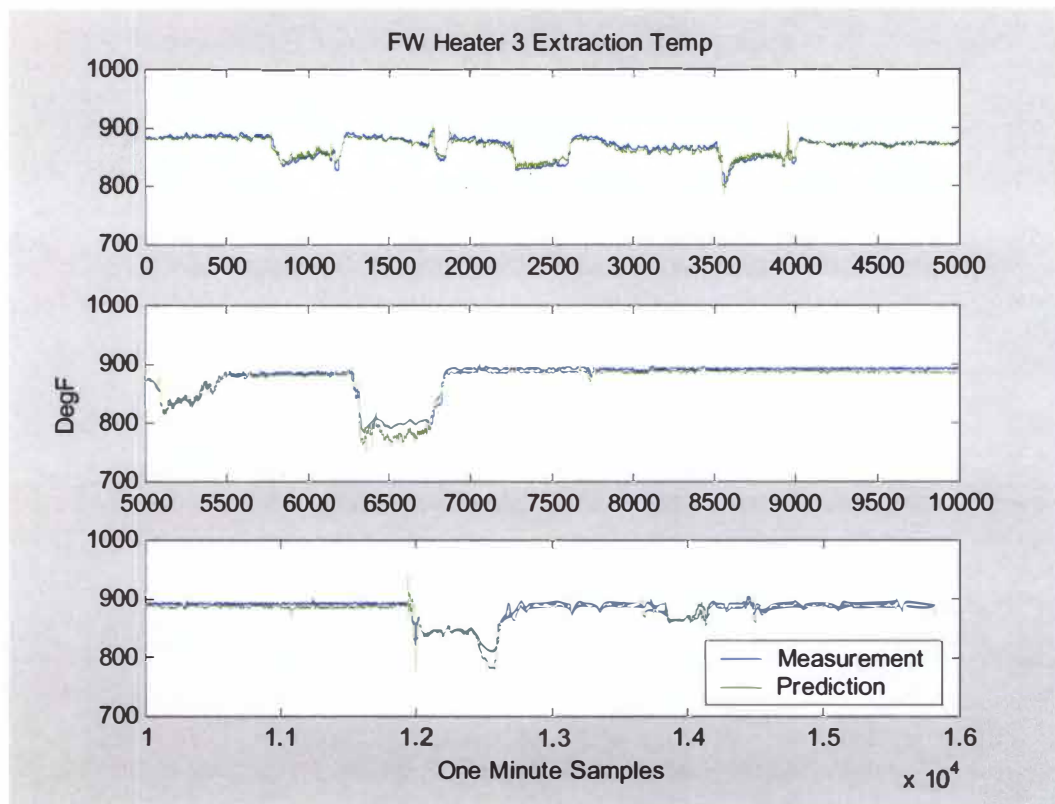


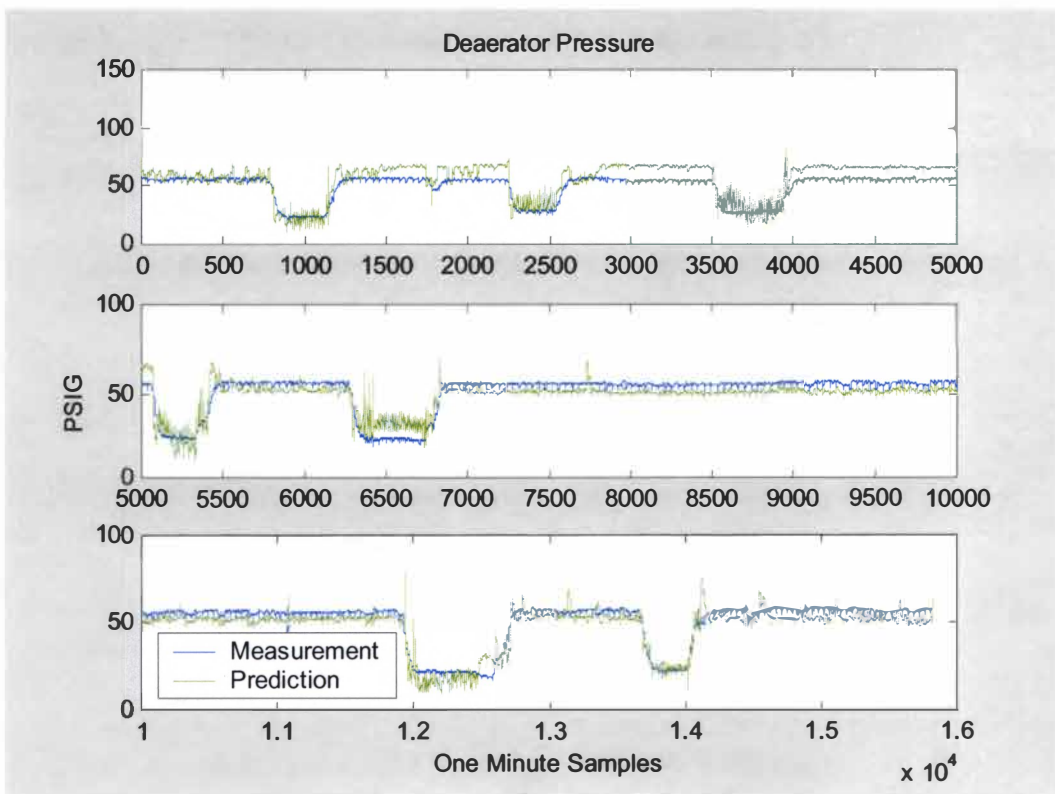
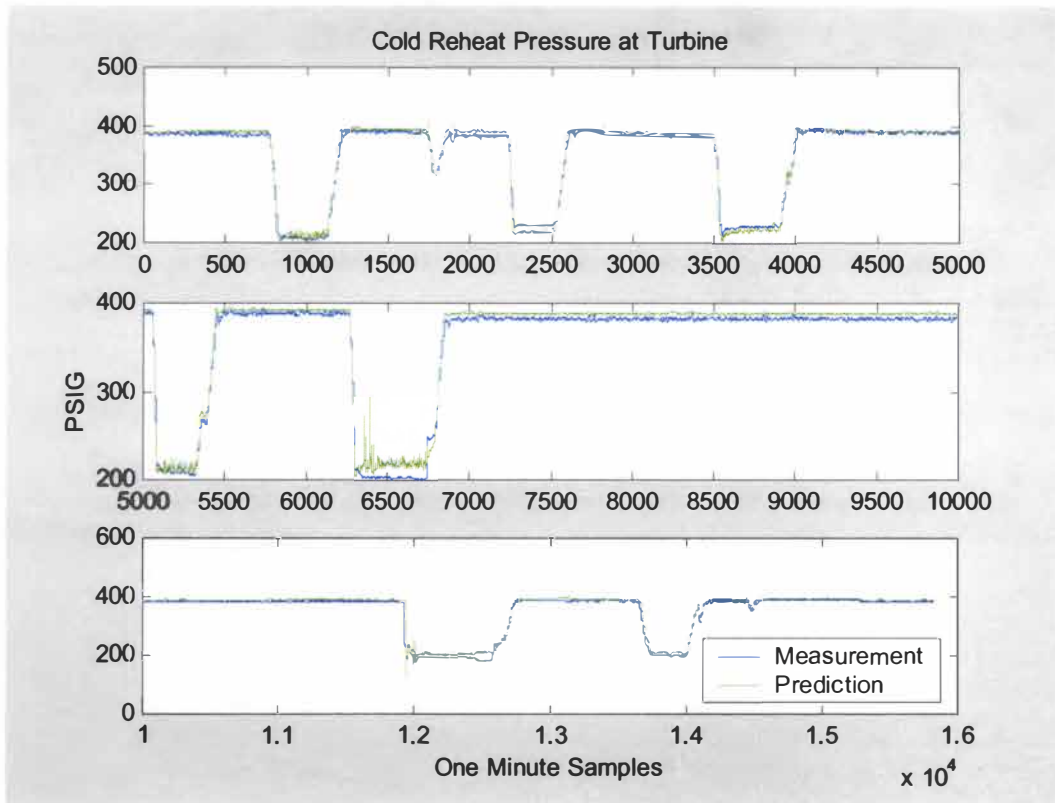


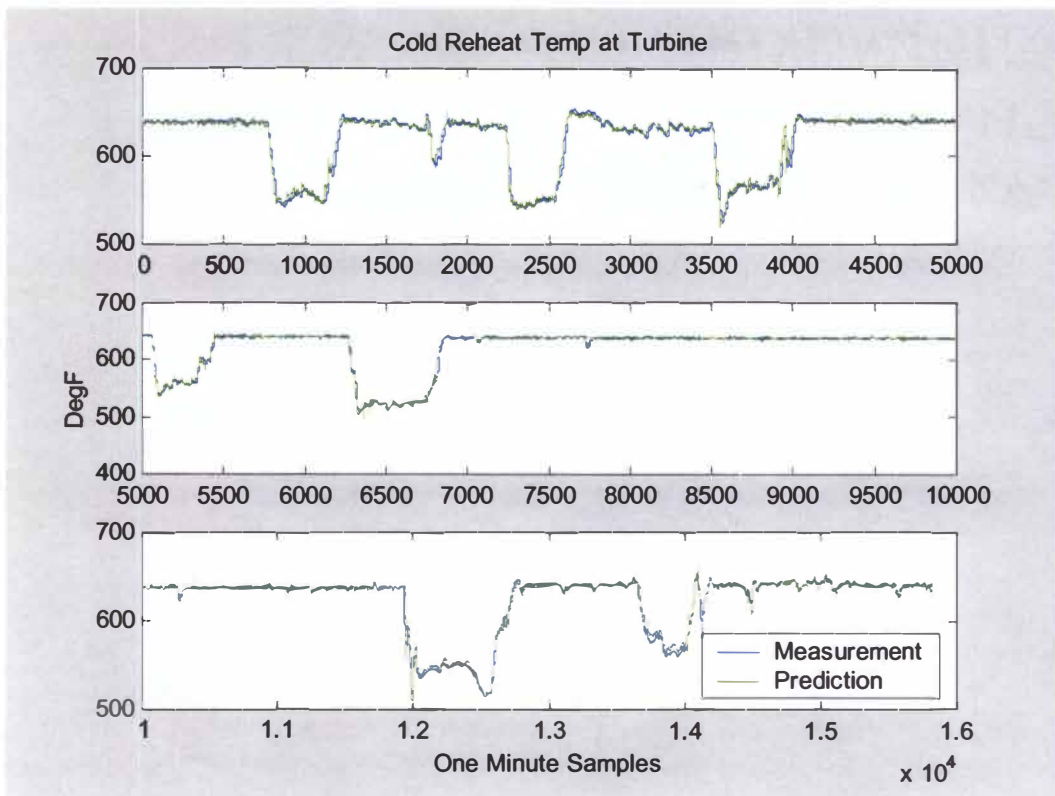
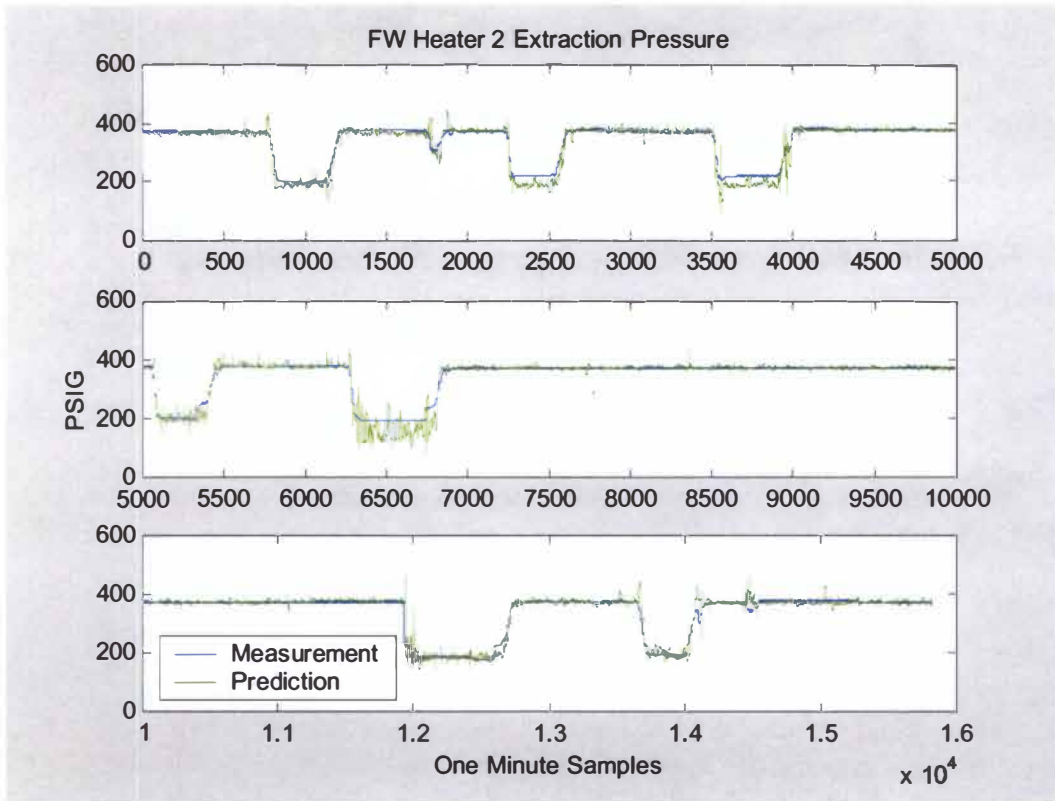


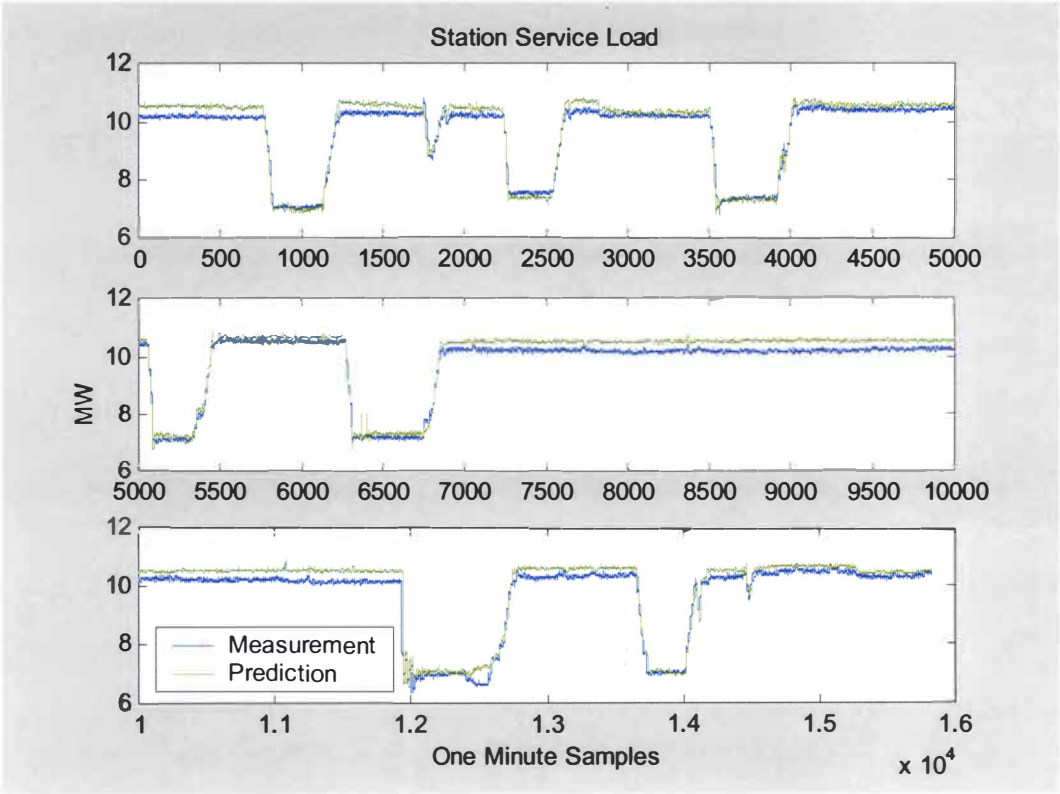
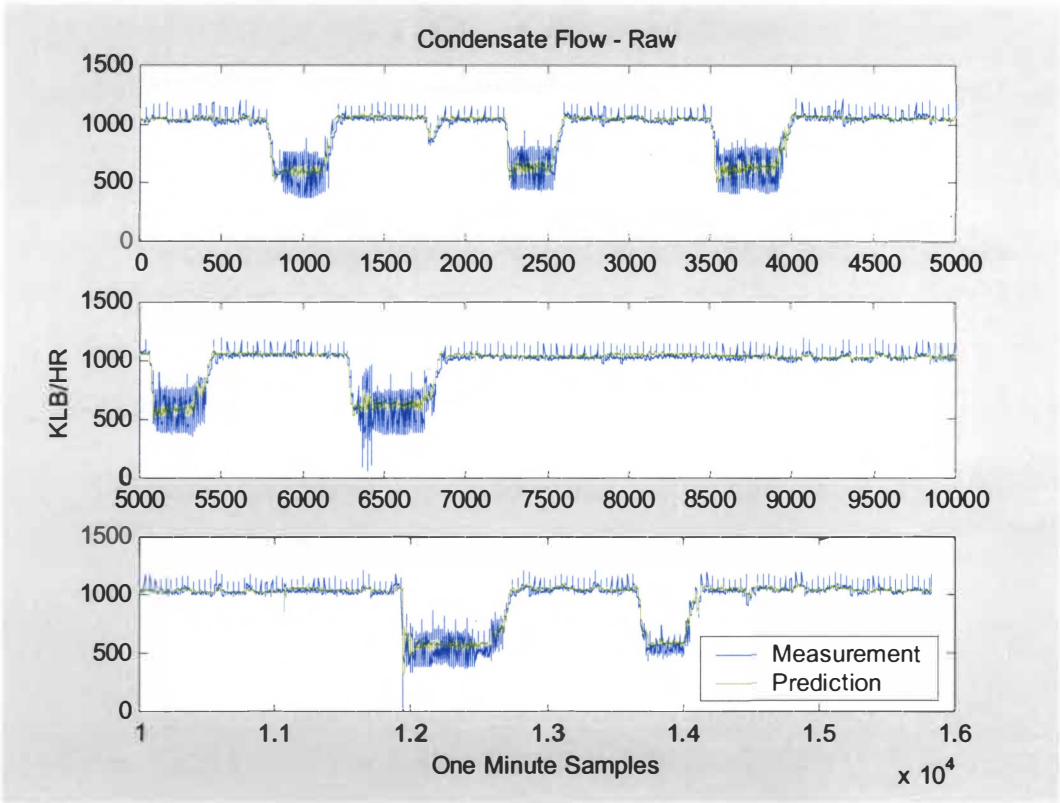


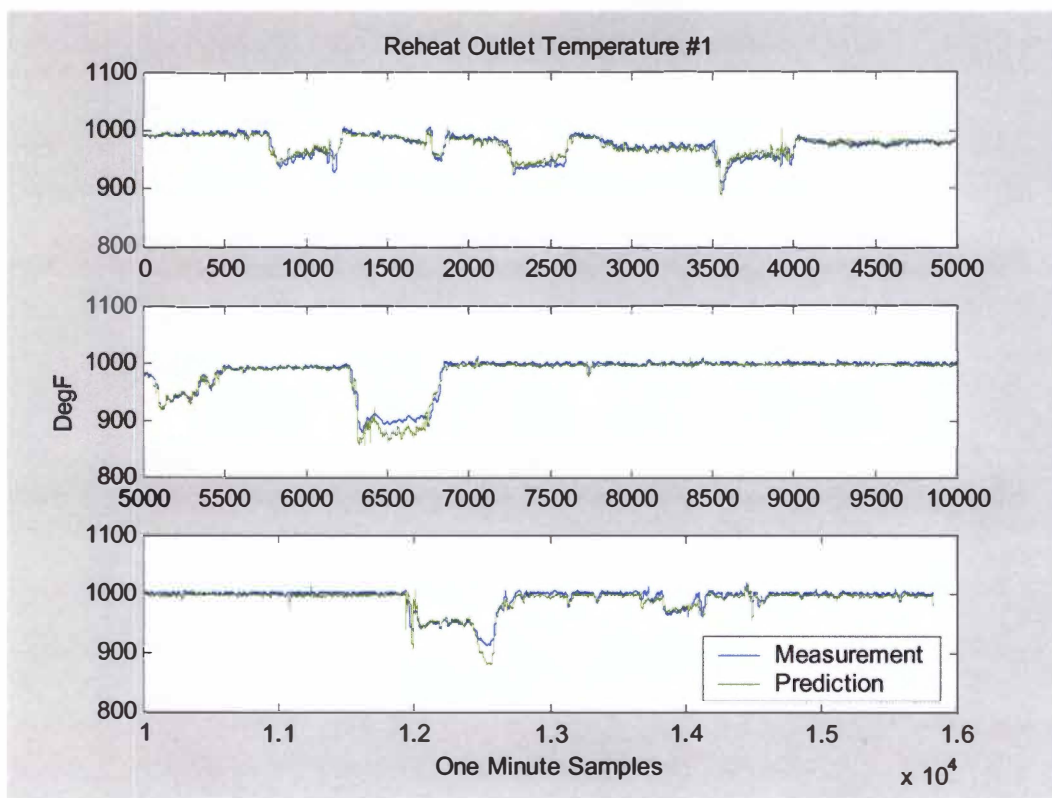
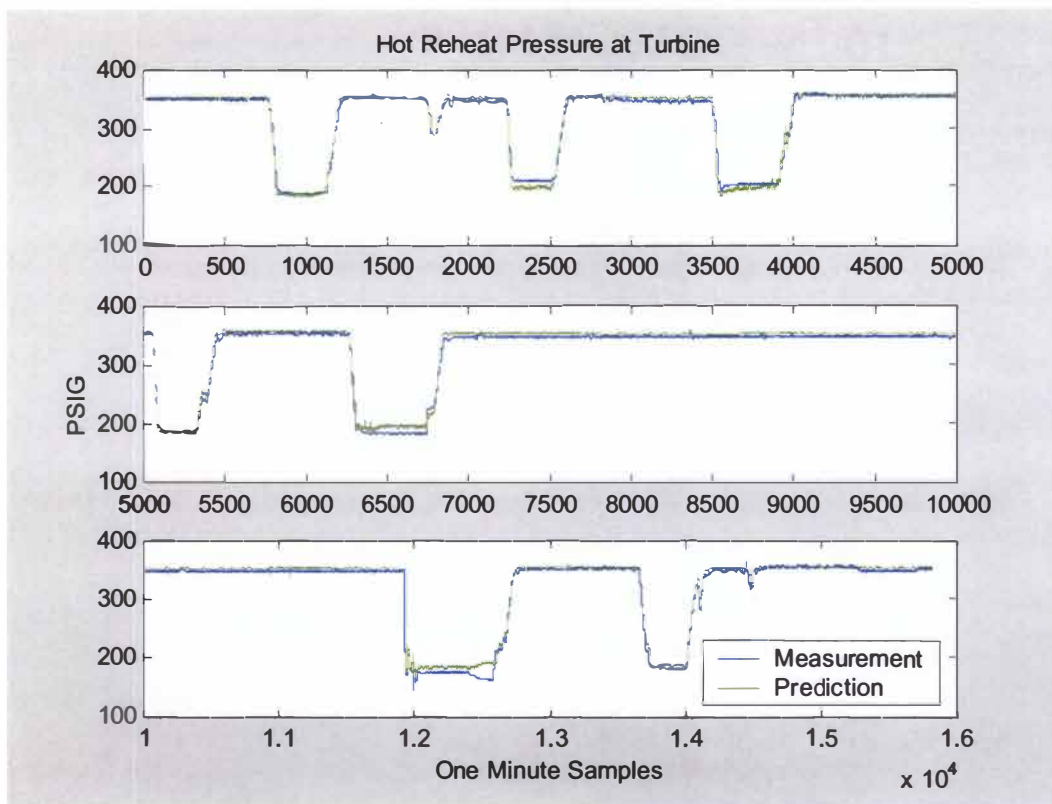


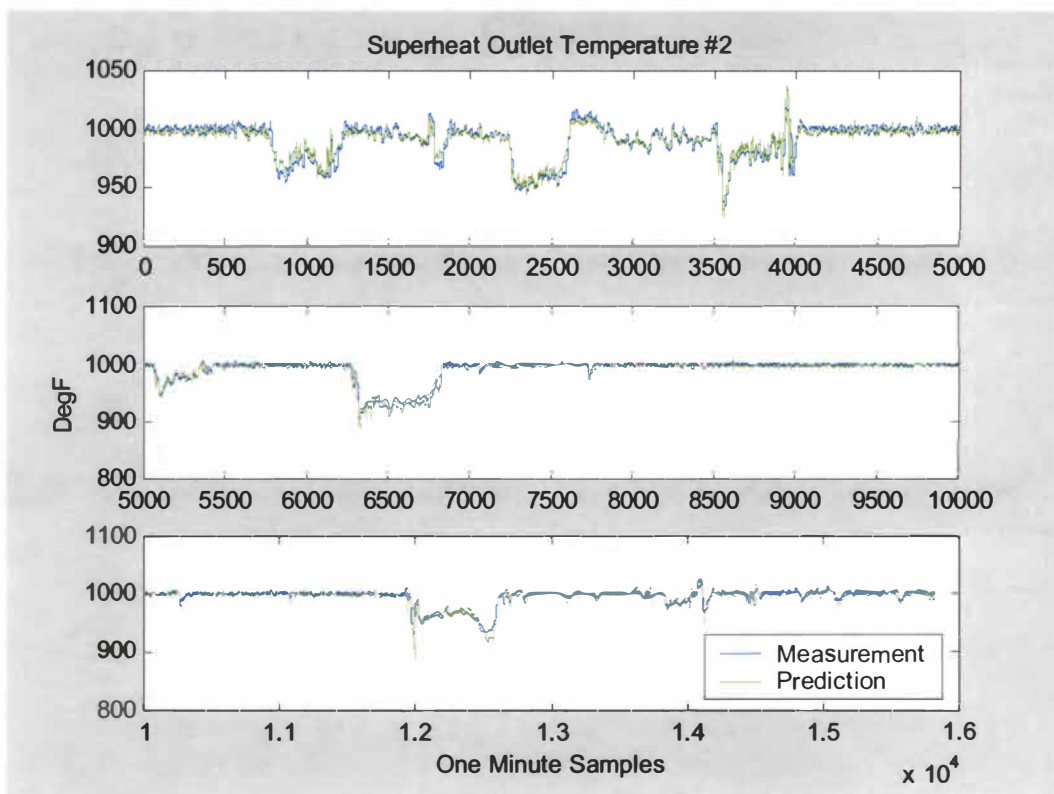
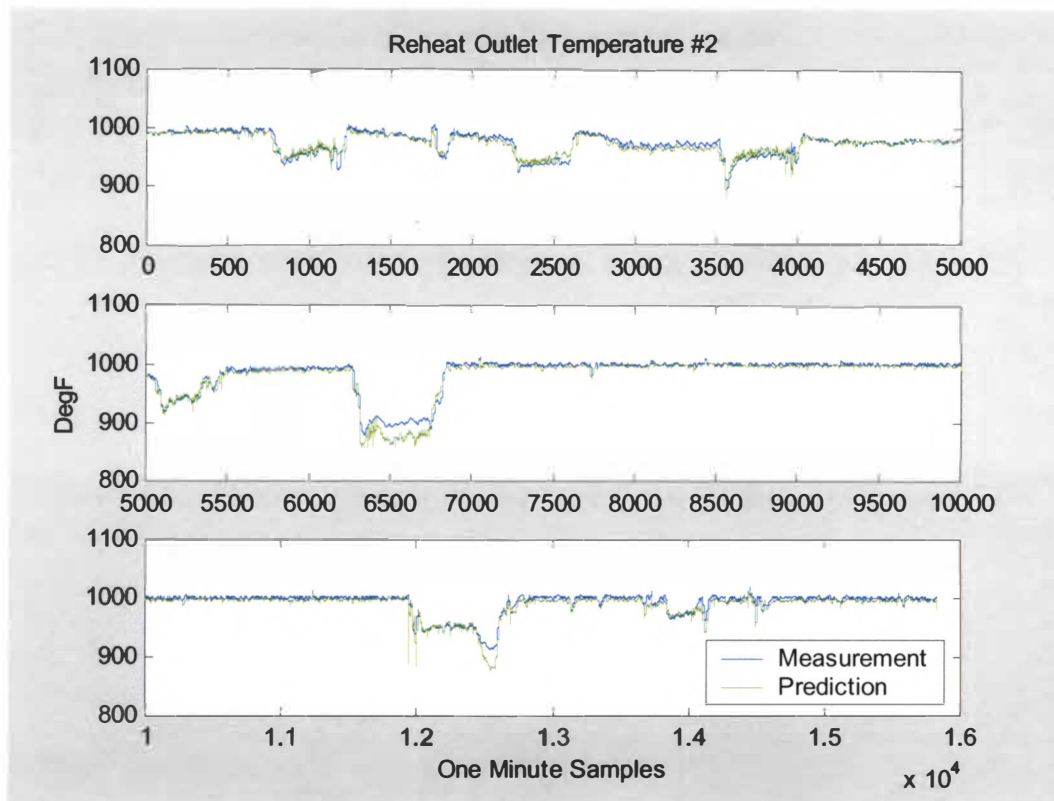


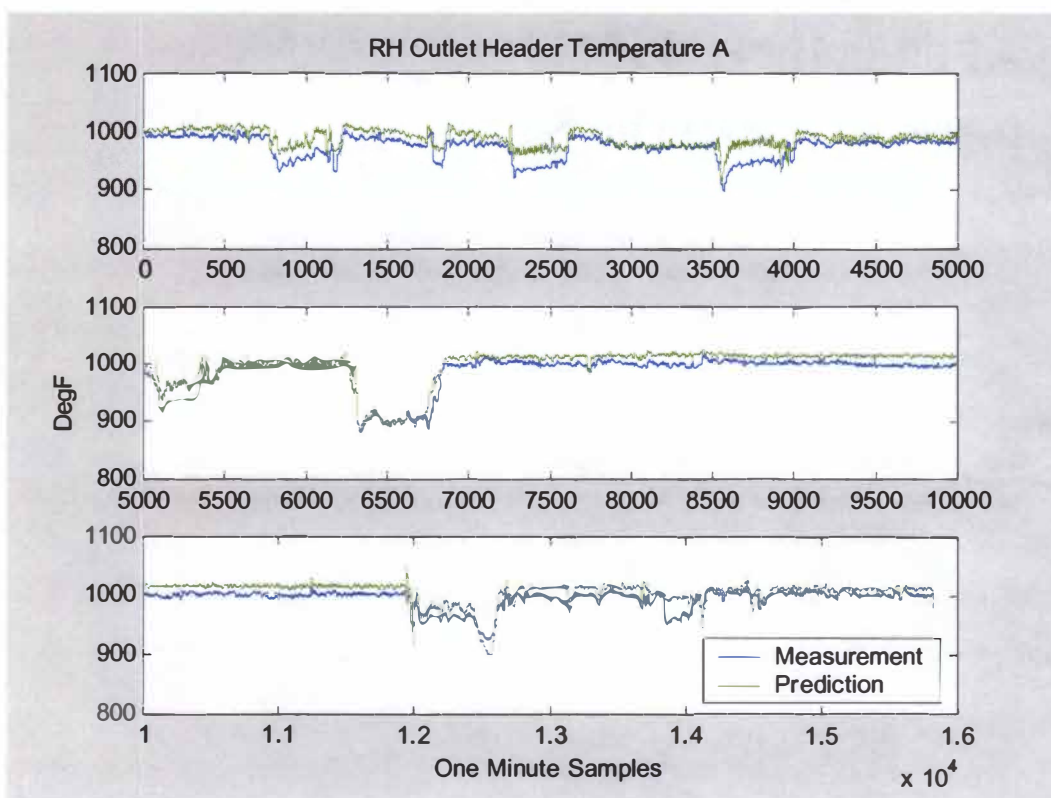
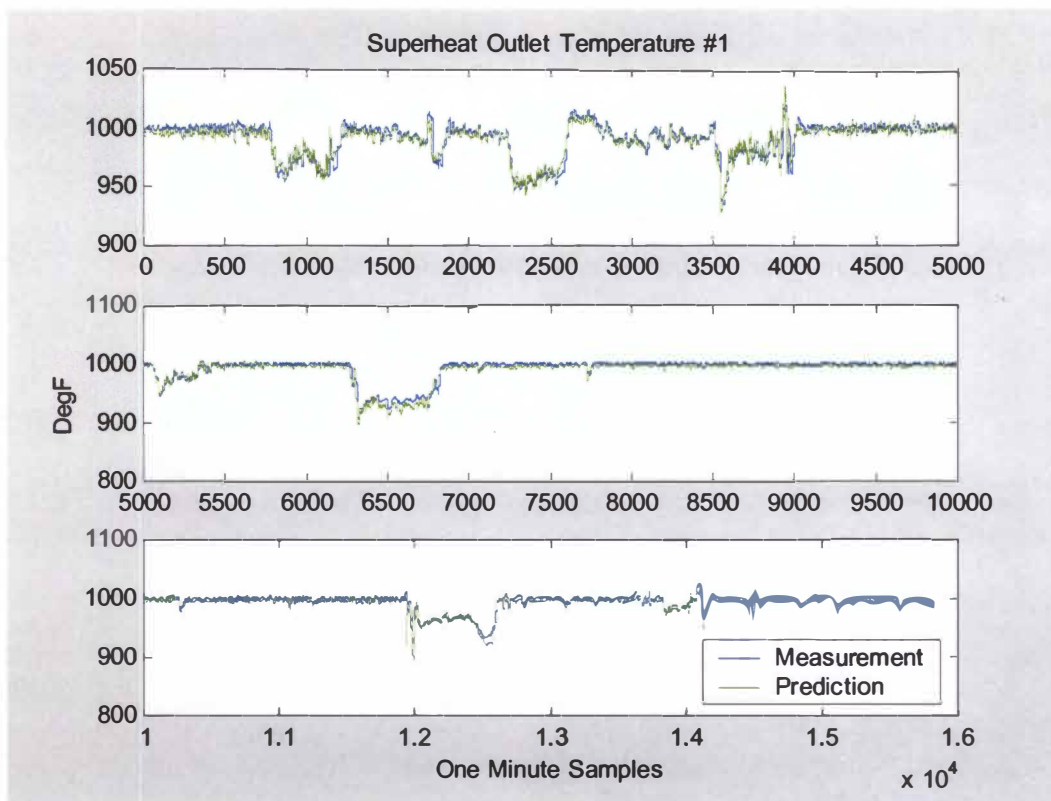


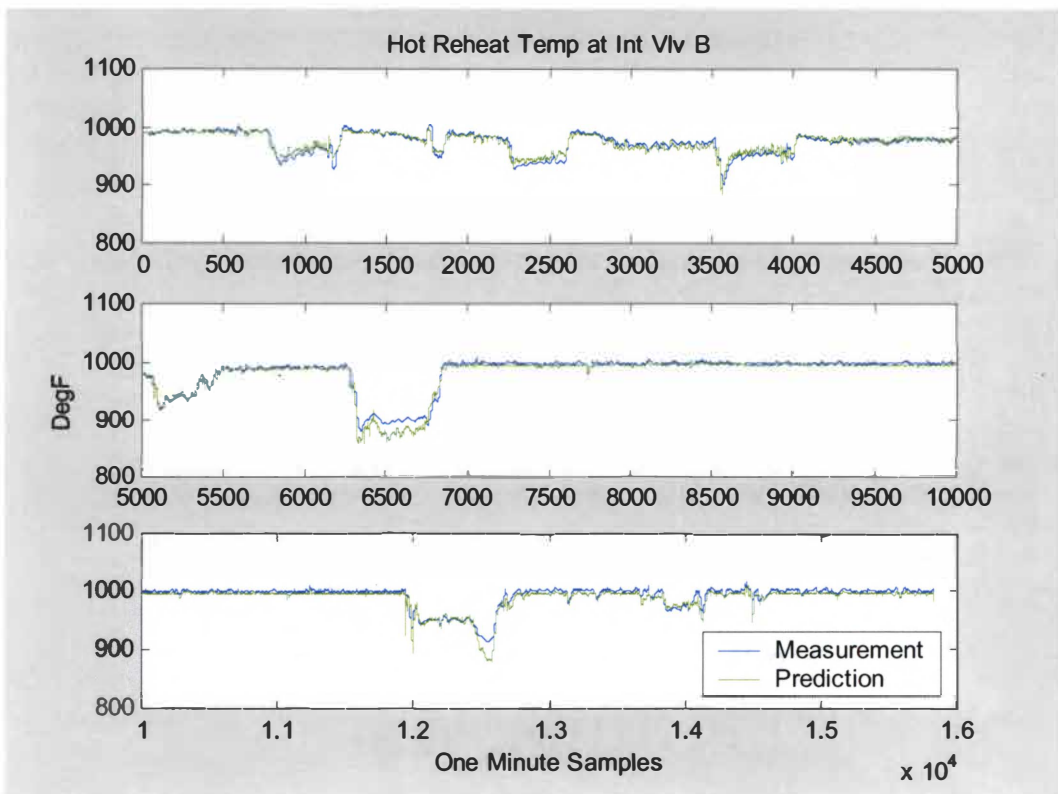
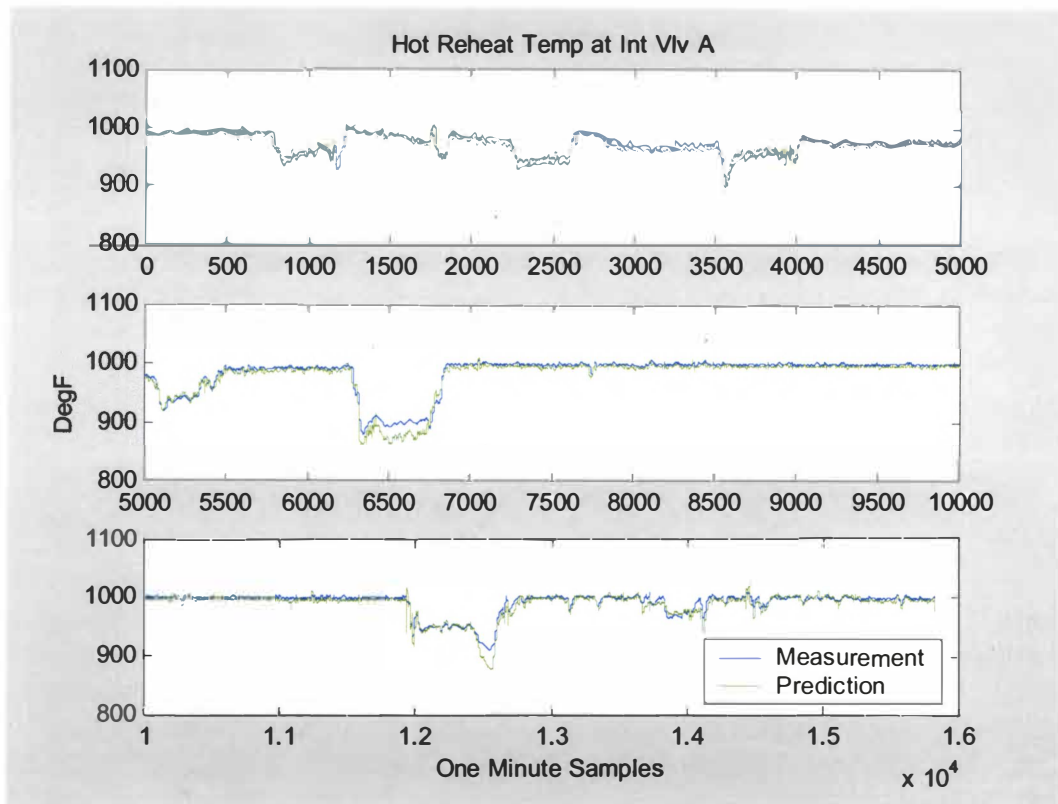


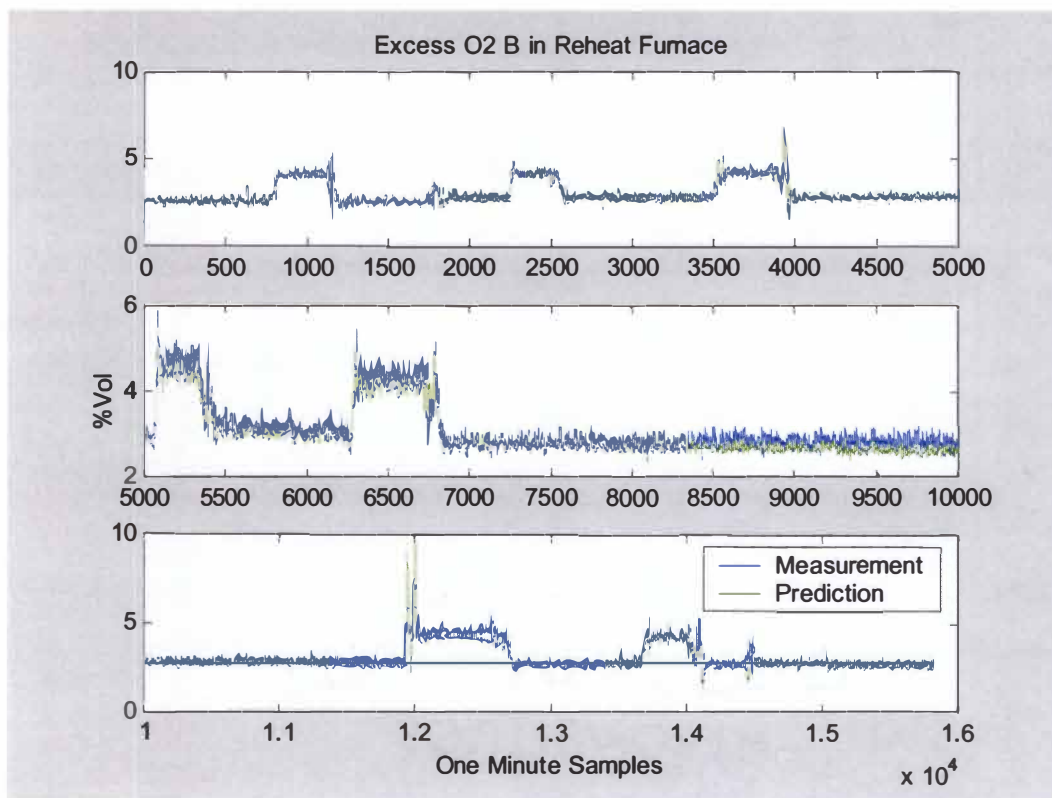
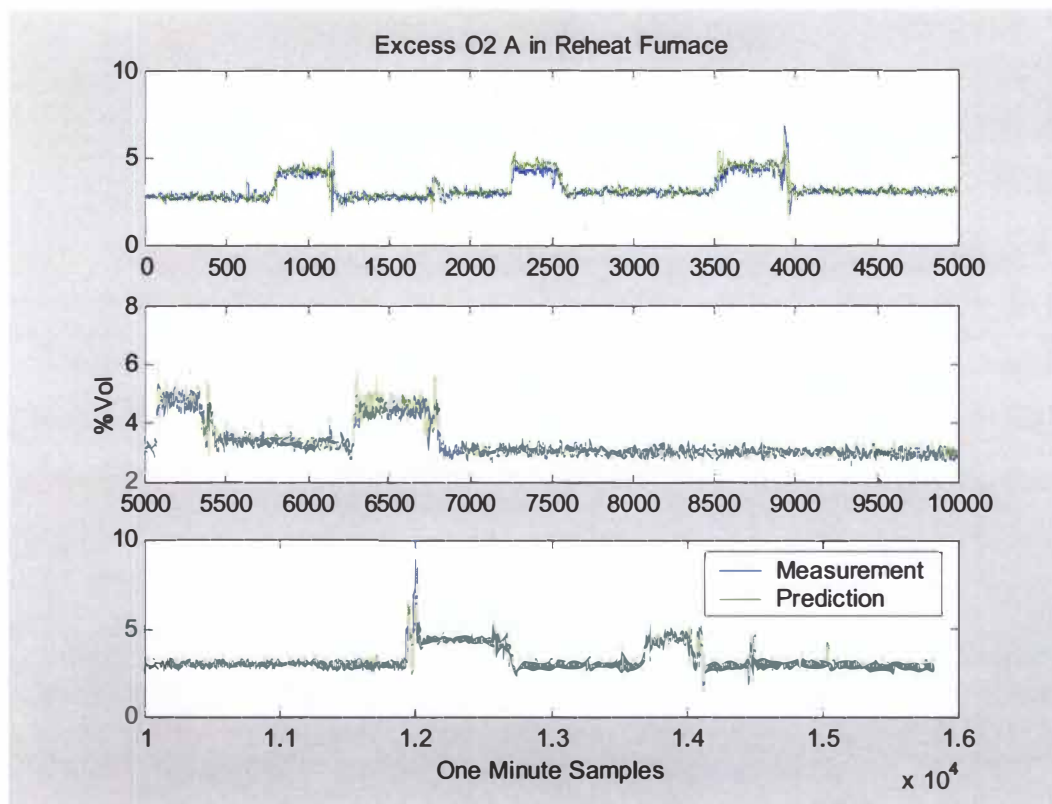


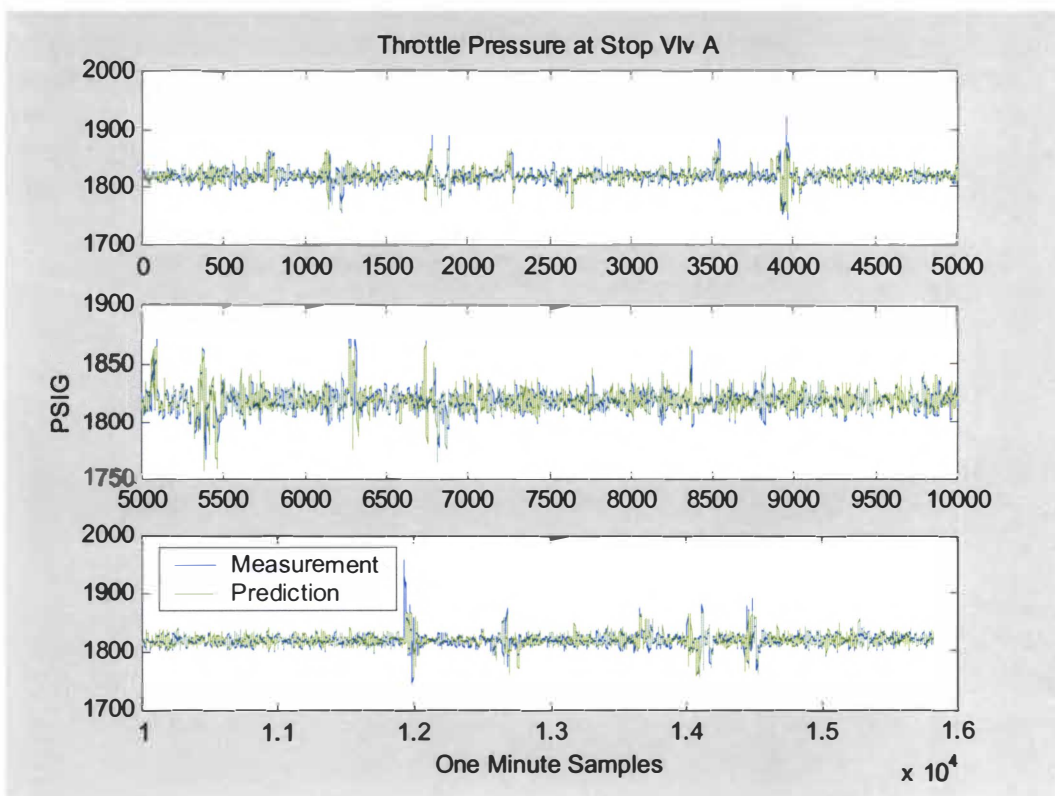
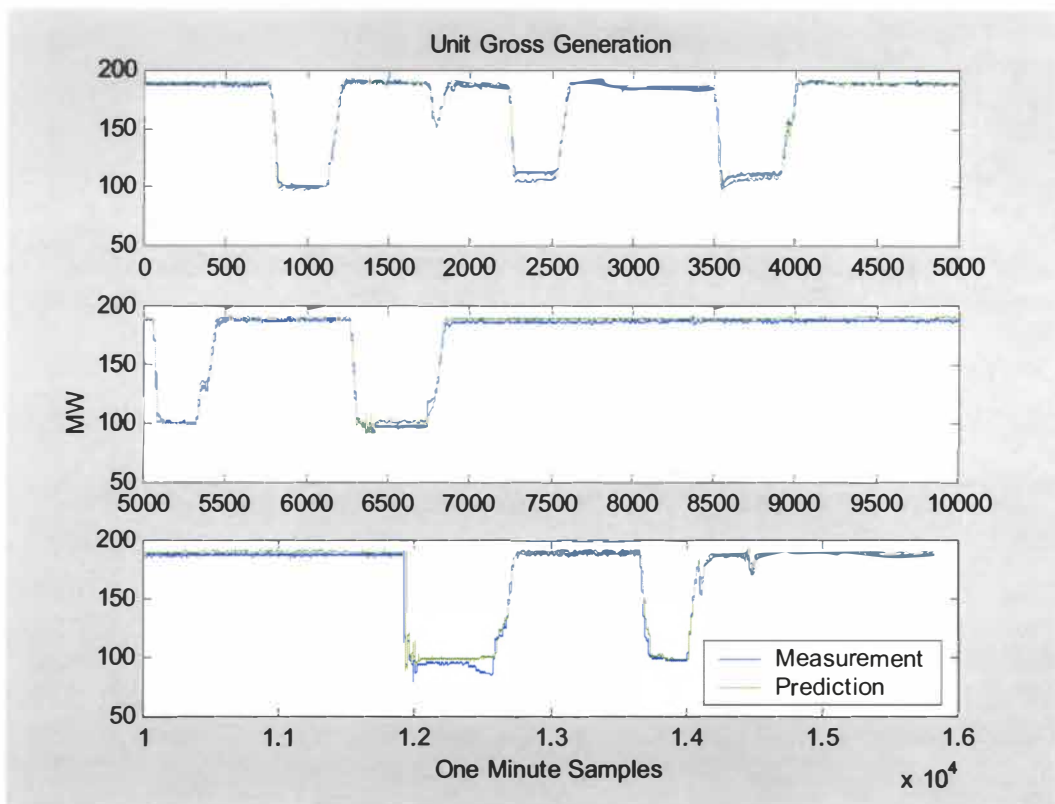


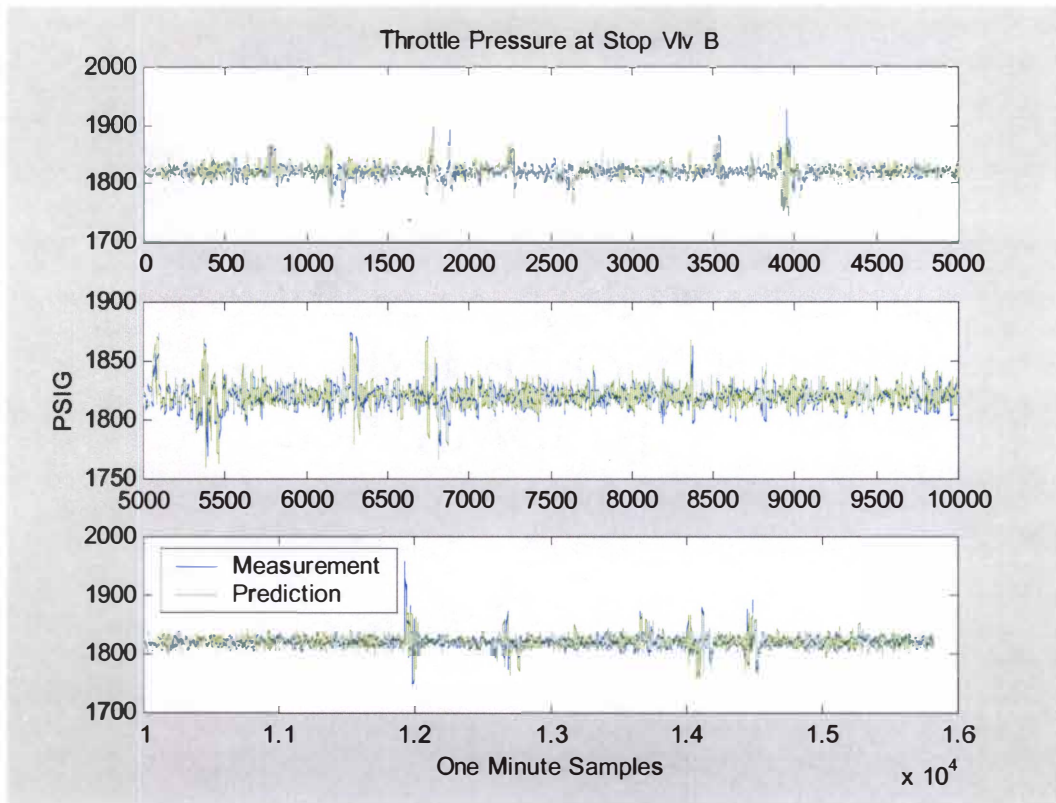












APPENDIX E

Tables of Recorded Data

Table 5: List of signals recorded via the MATLAB script program (4/2000 –1/2001)

#	PMW Name	Foxboro ID	Description	Unit
1			Month	
2			Day	
3			Hour	
4			Minute	
5			Second	
6	9FAIRA1	u9aw/9TA:B1.PNT	Combustion Air Flow A1	Pct
7	9FAIRB1	u9aw/9TA:B3.PNT	Combustion Air Flow B1	Pct
8	9FBWCLEAK	u9aw/9DSAI2:B20.PNT	BCW Pump Leakoff Flow - Raw	KLB/HR
9	9FCOND	u9aw/9HW:B3.PNT	Condensate Flow - Raw	KLB/HR
10	9FFW1	u9aw/9FW:B7.PNT	Feedwater Flow #1 - Raw	KLB/HR
11	9FFW2	u9aw/9FW:B8.PNT	Feedwater Flow #2 - Raw	KLB/HR
12	9FHW PUMPS	u9aw/9DA:B13.PNT	Hotwell Pumps Discharge Flow	KLB/HR
13	9FSTEAM	u9aw/9DSAI2:B16.PNT	Measured Steam Flow - Raw	KLB/HR
14	9JGROSS	u9aw/9TC:B1.PNT	Unit Gross Generation	MW
15	9JSS	u9aw/9LCAI1:B1.PNT	Station Service Load	MW
16	9LFWH1	u9aw/9HP_HTRS:B2.PNT	FW Htr 1 Level	Inches
17	9LFWH3	u9aw/9HP_HTRS:B3.PNT	FW Htr 3 Level	Inches
18	9O2RHA	u9aw/9TA:B14.PNT	Excess O2 A in Reheat Furnace	%Vol
19	9O2RHB	u9aw/9TA:B15.PNT	Excess O2 B in Reheat Furnace	%Vol
20	9P1STSTGA	u9aw/9BM:B0.PNT	First Stage Pressure A	PSIG
21	9P1STSTGB	u9aw/9BM:B0A.PNT	First Stage Pressure B	PSIG
22	9PCRHT	u9aw/9FWAI1:B3.PNT	Cold Reheat Pressure at Turbine	PSIG
23	9PDEA	u9aw/9FWAI1:B4.PNT	Deaerator Pressure	PSIG
24	9PDRUM	u9aw/9FW:B9.OUT	Selected Drum Pressure	PSIG
25	9PFW	u9aw/9FWAI1:B6.PNT	FW Entering Economizer Pressure	PSIG
26	9PFWH1EXT	u9aw/9DSAI2:B33.PNT	FW Heater 1 Extraction Pressure	PSIG
27	9PFWH2EXT	u9aw/9FWAI1:B7.PNT	FW Heater 2 Extraction Pressure	PSIG
28	9PFWH3EXT	u9aw/9DSAI2:B34.PNT	FW Heater 3 Extraction Pressure	PSIG
29	9PFWH4EXT	u9aw/9DSAI2:B35.PNT	FW Heater 4 Extraction Pressure	PSIG
30	9PFWH5EXTA	u9aw/9DSAI2:B36.PNT	FW Heater 5 Ext Press - Abs	PSIA
31	9PHRHT	u9aw/9RH:B15.PNT	Hot Reheat Pressure at Turbine	PSIG
32	9PIDFASUCT	u9aw/9AFAI1:B11.PNT	ID Fan A Suction Pressure	InH2O
33	9PIDFBSUCT	u9aw/9AFAI1:B19.PNT	ID Fan B Suction Pressure	InH2O
34	9PRHECON	u9aw/9AFAI1:B16.PNT	RH Furnace Press After Econ	InH2O
35	9PSHECON	u9aw/9AFAI1:B8.PNT	SH Furnace Press After Econ	InH2O
36	9PSHWB	u9aw/9WBS:B26.PNT	SH Windbox Pressure	InH2O
37	9PTHROTTLEA	u9aw/9TC:B2A.PNT	Throttle Pressure at Stop Vlv A	PSIG
38	9PTHROTTLLEB	u9aw/9TC:B2B.PNT	Throttle Pressure at Stop Vlv B	PSIG

Table 5: continued

39	9PTURBEXH	u9aw/9DSA12:B27.PNT	Turbine Exhaust Pressure	InHgAbs
40	9TAPHAAIRIN	u9aw/9DSA15:B31.PNT	Air Preheater A Air Inlet Temp	DegF
41	9TAPHAGASIN	u9aw/9DSGR:B3.OUT	Air Preheater A Gas Inlet Temp	DegF
42	9TAPHAGASOUT	u9aw/9DSGR:B4.OUT	Air Preheater A Gas Outlet Temp	DegF
43	9TAPHBAIRIN	u9aw/9DSA15:B39.PNT	Air Preheater B Air Inlet Temp	DegF
44	9TAPHBGASIN	u9aw/9DSGR:B5.OUT	Air Preheater B Gas Inlet Temp	DegF
45	9TAPHBGASOUT	u9aw/9DSGR:B6.OUT	Air Preheater B Gas Outlet Temp	DegF
46	9TBFPSUCT	u9aw/9DSA13:B7.PNT	BFP Suction Header Temperature	DegF
47	9TCRCWTRINW	u9aw/9DSA14:B50.PNT	Circ Water Inlet Temp, West	DegF
48	9TCRCWTRROUTE	u9aw/9DSGR:B7.OUT	Circ Water Outlet Temp, East	DegF
49	9TCRCWTROUTW	u9aw/9DSGR:B8.OUT	Circ Water Outlet Temp, West	DegF
50	9TCRHT	u9aw/9FWAI1:B9.PNT	Cold Reheat Temp at Turbine	DegF
51	9TFWH1DRAIN	u9aw/9FWAI1:B20.PNT	FW Heater 1 Drain Temperature	DegF
52	9TFWH1EXT	u9aw/9FWAI1:B23.PNT	FW Heater 1 Extraction Temp	DegF
53	9TFWH1FWIN	u9aw/9FWAI1:B17.PNT	FW Heater 1 Water In Temp	DegF
54	9TFWH1FWOUT	u9aw/9FWAI1:B18.PNT	FW Heater 1 Water Out Temp	DegF
55	9TFWH2DRAIN	u9aw/9FWAI1:B21.PNT	FW Heater 2 Drain Temperature	DegF
56	9TFWH2FWIN	u9aw/9FWAI1:B16.PNT	FW Heater 2 Water In Temp	DegF
57	9TFWH3DRAIN	u9aw/9FWAI1:B22.PNT	FW Heater 3 Drain Temperature	DegF
58	9TFWH3EXT	u9aw/9FWAI1:B24.PNT	FW Heater 3 Extraction Temp	DegF
59	9TFWH3FWIN	u9aw/9FWAI1:B15.PNT	FW Heater 3 Water In Temp	DegF
60	9TFWH4EXT	u9aw/9FWAI1:B25.PNT	FW Heater 4 Extraction Temp	DegF
61	9TFWH5DRAIN	u9aw/9DSA13:B36.PNT	FW Heater 5 Drain Temperature	DegF
62	9TFWH5EXT	u9aw/9DSA13:B35.PNT	FW Heater 5 Extraction Temp	DegF
63	9TFWH5FWOUT	u9aw/9DSA13:B6.PNT	FW Heater 5 Water Out Temp	DegF
64	9TFWH6DRAIN	u9aw/9DSA13:B38.PNT	FW Heater 6 Drain Temperature	DegF
65	9TFWH6EXTA	u9aw/9DSA13:B39.PNT	FW Heater 6 Extraction Temp A	DegF
66	9TFWH6EXTB	u9aw/9DSA13:B40.PNT	FW Heater 6 Extraction Temp B	DegF
67	9TFWH6FWOUT	u9aw/9DSA13:B5.PNT	FW Heater 6 Water Out Temp	DegF
68	9TFWH7DRAIN	u9aw/9DSA13:B43.PNT	FW Heater 7 Drain Temperature	DegF
69	9TFWH7EXTA	u9aw/9DSA13:B42.PNT	FW Heater 7 Extraction Temp A	DegF
70	9TFWH7FWOUT	u9aw/9DSA13:B4.PNT	FW Heater 7 Water Out Temp	DegF
71	9TFWH8DRAIN	u9aw/9DSA13:B44.PNT	FW Heater 8 Drain Temperature	DegF
72	9TFWH8EXTA	u9aw/9DSA13:B45.PNT	FW Heater 8 Extraction Temp A	DegF
73	9TFWH8FWIN	u9aw/9DSA13:B2.PNT	FW Heater 8 Water In Temp	DegF
74	9TFWH8FWOUT	u9aw/9DSA13:B3.PNT	FW Heater 8 Water Out Temp	DegF
75	9THRHTA	u9aw/9STAI1:B14.PNT	Hot Reheat Temp at Int Vlv A	DegF
76	9THRHTB	u9aw/9STAI1:B15.PNT	Hot Reheat Temp at Int Vlv B	DegF
77	9TPULVA	u9aw/9DS_CTRL:B1.PNT	Pulverizer A Outlet Temperature	DegF
78	9TPULVB	u9aw/9DS_CTRL:B2.PNT	Pulverizer B Outlet Temperature	DegF
79	9TPULVC	u9aw/9DS_CTRL:B3.PNT	Pulverizer C Outlet Temperature	DegF
80	9TPULVD	u9aw/9DS_CTRL:B4.PNT	Pulverizer D Outlet Temperature	DegF
81	9TPULVE	u9aw/9DS_CTRL:B5.PNT	Pulverizer E Outlet Temperature	DegF
82	9TPULVF	u9aw/9DS_CTRL:B6.PNT	Pulverizer F Outlet Temperature	DegF

Table 5: continued

83	9TRHABSPR	u9aw/9STAI1:B10.PNT	RH Attempt A Before Spray Temp	DegF
84	9TRHHDR	u9aw/9STAI1:B11.PNT	RH Outlet Header Temperature A	DegF
85	9TRHOUT1	u9aw/9RH:B2.PNT	Reheat Outlet Temperature #1	DegF
86	9TRHOUT2	u9aw/9RH:B3.PNT	Reheat Outlet Temperature #2	DegF
87	9TSHHDRA	u9aw/9STAI1:B7.PNT	SH Outlet Header Temperature A	DegF
88	9TSHOUT1	u9aw/9SH:B3.PNT	Superheat Outlet Temperature #1	DegF
89	9TSHOUT2	u9aw/9SH:B2.PNT	Superheat Outlet Temperature #2	DegF
90	9AMPBFPA	u9aw/9CBAI1:B11.PNT	BFP A Amps	Amps
91	9AMPBFPB1	u9aw/9CBAI1:B12.PNT	BFP B Amps from Bus 1	Amps
92	9AMPBFPB2	u9aw/9CBAI1:B13.PNT	BFP B Amps from Bus 2	Amps
93	9AMPBFPC	u9aw/9CBAI1:B14.PNT	BFP C Amps	Amps
94	9COMMCHK	u9aw/9DSA17:B26.RO01	Seconds since midnight	seconds
95	9FBFPA	u9aw/9DL:B11.PNT	BFP A Discharge Flow - Raw	KLB/HR
96	9FBFPB	u9aw/9DL:B12.PNT	BFP B Discharge Flow - Raw	KLB/HR
97	9FBFPC	u9aw/9DL:B13.PNT	BFP C Discharge Flow - Raw	KLB/HR
98	9FCOALA	u9aw/9PULVA:B1.PNT	Coal Flow Pulv A	LB/HR
99	9FCOALB	u9aw/9PULVB:B1.PNT	Coal Flow Pulv B	LB/HR
100	9FCOALC	u9aw/9PULVC:B1.PNT	Coal Flow Pulv C	LB/HR
101	9FCOALD	u9aw/9PULVD:B1.PNT	Coal Flow Pulv D	LB/HR
102	9FCOALE	u9aw/9PULVE:B1.PNT	Coal Flow Pulv E	LB/HR
103	9FCOALF	u9aw/9PULVF:B1.PNT	Coal Flow Pulv F	LB/HR
104	9FCWMU	u9aw/9DSA12:B18.PNT	Coldwell Tank Makeup Flow	KLB/HR
105	9FFWH3DR	u9aw/9DSA12:B19.PNT	FWH 3 Drain Flow - Raw	LB/HR
106	9FHEATSTM	u9aw/9DSA12:B17.PNT	Building Heating Steam Flow	KLB/HR
107	9FPULVA	u9aw/9PULVA:B2A.OUT	Pulv A PA Flow Rate	KLB/HR
108	9FPULVB	u9aw/9PULVB:B2A.OUT	Pulv B PA Flow Rate	KLB/HR
109	9FPULVC	u9aw/9PULVC:B2A.OUT	Pulv C PA Flow Rate	KLB/HR
110	9FPULVD	u9aw/9PULVD:B2A.OUT	Pulv D PA Flow Rate	KLB/HR
111	9FPULVE	u9aw/9PULVE:B2A.OUT	Pulv E PA Flow Rate	KLB/HR
112	9FPULVF	u9aw/9PULVF:B2A.OUT	Pulv F PA Flow Rate	KLB/HR
113	9FRHASPRAY	u9aw/9STAI1:B3.PNT	Reheat Spray A Flow	KLB/HR
114	9FRHBSPRAY	u9aw/9STAI1:B4.PNT	Reheat Spray B Flow	KLB/HR
115	9FSHASPRAY	u9aw/9STAI1:B1.PNT	Superheat Spray A Flow	KLB/HR
116	9FSHBSPRAY	u9aw/9STAI1:B2.PNT	Superheat Spray B Flow	KLB/HR
117	9FSOOTRH	u9aw/9SBP:B2.PNT	RH Furn Sootblowing Steam Flow	KLB/HR
118	9FSOOTSH	u9aw/9SBP:B1.PNT	SH Furn Sootblowing Steam Flow	KLB/HR
119	9LCWTANK	u9aw/9CWT:B1.PNT	Coldwell Tank Level	INCHES
120	9LDEA	u9aw/9DA:B4.PNT	Selected Deaerator Level	INCHES
121	9LFWH2	u9aw/9HP_HTRS:B1.PNT	FW Htr 2 Level	INCHES
122	9O2SHA	u9aw/9TA:B12.PNT	Excess O2 A in Superheat Furnace	%VOL
123	9O2SHB	u9aw/9TA:B13.PNT	Excess O2 B in Superheat Furnace	%VOL
124	9PFLYASHA	u9aw/9AFAI1:B10.PNT	Pressure after A Fly Ash Collect	INCHES H2O
125	9PFLYASHB	u9aw/9AFAI1:B18.PNT	Pressure after B Fly Ash Collect	INCHES H2O
126	9PRHFURNA	u9aw/9DRAFT:B4.PNT	RH Furnace Pressure A	INCHES H2O

Table 5: continued

127	9PSHAHT	u9aw/9AFAI1:B6.PNT	SH Furnace Press after HT SH	INCHES H2O
128	9PRHFURNB	u9aw/9DRAFT:B1.PNT	SH Furnace Pressure A	INCHES H2O
129	9PSOOT	u9aw/9SBP:B3.PNT	Sootblowing Steam Pressure	PSIG
130	9RHCO2PCT	u9aw/9CBAI1:B30.PNT	Unit CEMS RH CO2	PCT
131	9RHCOPPM	u9aw/9CBAI1:B28.PNT	Unit CEMS RH CO	PPM
132	9RHCOPPM1	u9aw/9RH_CO:B1.PNT	Bailey CO Monitor - Reheat Furnace	PPM
133	9RHNOXLB	u9aw/9CBAI1:B29.PNT	Unit CEMS RH NOX	LB/MMBTU
134	9RHNOXPPM	u9aw/9CBAI1:B27.PNT	Unit CEMS RH NOX	PPM
135	9SHCO2PCT	u9aw/9CBAI1:B26.PNT	Unit CEMS SH CO2	PCT
136	9SHCOPPM	u9aw/9CBAI1:B24.PNT	Unit CEMS SH CO	PPM
137	9SHCOPPM1	u9aw/9SH_CO:B1.PNT	Bailey CO Monitor - Superheat	PPM
138	9SHNOXLB	u9aw/9CBAI1:B25.PNT	Unit CEMS SH NOX	LB/MMBTU
139	9SHNOXPPM	u9aw/9CBAI1:B23.PNT	Unit CEMS SH NOX	PPM
140	9TAMBIENT	u9aw/9FWAI1:B32.PNT	Ambient Air Temperature	DegF

Table 6: List of signals recorded via the SQL connection interface (6/2001 –11/2001)

#	PMW Name	Foxboro ID	Description	Unit
1	9PFLYASHA	u9aw/9AFAI1:B10.PNT	Pressure after A Fly Ash Collect	INCHESH20
2	9PIDFASUCT	u9aw/9AFAI1:B11.PNT	ID Fan A Suction Pressure	InH2O
3	9PFLYASHB	u9aw/9AFAI1:B12.PNT	Pressure after B Fly Ash Collect	INCHESH20
4	9PRHECON	u9aw/9AFAI1:B16.PNT	RH Furnace Press After Econ	InH2O
5	9PIDFBSUCT	u9aw/9AFAI1:B19.PNT	ID Fan B Suction Pressure	InH2O
6	9PSHAHT	u9aw/9AFAI1:B6.PNT	SH Furnace Press after HT SH	INCHESH20
7	9PSHECON	u9aw/9AFAI1:B8.PNT	SH Furnace Press After Econ	InH2O
8	9P1STSTGA	u9aw/9BM:B0.PNT	First Stage Pressure A	PSIG
9	9P1STSTGB	u9aw/9BM:B0A.PNT	First Stage Pressure B	PSIG
10	9SHNOXPPM	u9aw/9CBAI1:B23.PNT	Unit CEMS SH NOX	PPM
11	9SHCOPPM	u9aw/9CBAI1:B24.PNT	Unit CEMS SH CO	PPM
12	9SHNOXLB	u9aw/9CBAI1:B25.PNT	Unit CEMS SH NOX	LB/MMBTU
13	9SHCO2PCT	u9aw/9CBAI1:B26.PNT	Unit CEMS SH CO2	PCT
14	9RHNOXPPM	u9aw/9CBAI1:B27.PNT	Unit CEMS RH NOX	PPM
15	9RHCOPPM	u9aw/9CBAI1:B28.PNT	Unit CEMS RH CO	PPM
16	9RHNOXLB	u9aw/9CBAI1:B29.PNT	Unit CEMS RH NOX	LB/MMBTU
17	9RHCO2PCT	u9aw/9CBAI1:B30.PNT	Unit CEMS RH CO2	PCT
18	9LCWTANK	u9aw/9CWT:B1.PNT	Coldwell Tank Level	INCHES
19	9FHWPPUMPS	u9aw/9DA:B13.PNT	Hotwell Pumps Discharge Flow	KLB/HR
20	9LDEA	u9aw/9DA:B4.OUT	Selected Deaerator Level	INCHES
21	9FBFPA	u9aw/9DL:B11.PNT	BFP A Discharge Flow - Raw	KLB/HR
22	9FBFPB	u9aw/9DL:B12.PNT	BFP B Discharge Flow - Raw	KLB/HR
23	9FBFPC	u9aw/9DL:B13.PNT	BFP C Discharge Flow - Raw	KLB/HR
24	9PRHFURNB	u9aw/9DRAFT:B1.PNT	SH Furnace Pressure A	INCHESH20
25	9PRHFURNA	u9aw/9DRAFT:B4.PNT	RH Furnace Pressure A	INCHESH20
26	9FSTEAM	u9aw/9DSAI2:B16.PNT	Measured Steam Flow - Raw	KLB/HR
27	9FHEATSTM	u9aw/9DSAI2:B17.PNT	Building Heating Steam Flow	KLB/HR
28	9FCWMU	u9aw/9DSAI2:B18.PNT	Coldwell Tank Makeup Flow	KLB/HR
29	9FFWH3DR	u9aw/9DSAI2:B19.PNT	FWH 3 Drain Flow - Raw	LB/HR
30	9FBWCPLEAK	u9aw/9DSAI2:B20.PNT	BCW Pump Leakoff Flow - Raw	KLB/HR
31	9PTURBEXH	u9aw/9DSAI2:B27.PNT	Turbine Exhaust Pressure	InHgAbs
32	9PFWH1EXT	u9aw/9DSAI2:B33.PNT	FW Heater 1 Extraction Pressure	PSIG
33	9PFWH3EXT	u9aw/9DSAI2:B34.PNT	FW Heater 3 Extraction Pressure	PSIG
34	9PFWH4EXT	u9aw/9DSAI2:B35.PNT	FW Heater 4 Extraction Pressure	PSIG
35	9PFWH5EXTA	u9aw/9DSAI2:B36.PNT	FW Heater 5 Ext Press - Abs	PSIA
36	9TFWH8FWIN	u9aw/9DSAI3:B2.PNT	FW Heater 8 Water In Temp	DegF
37	9TFWH8FWOUT	u9aw/9DSAI3:B3.PNT	FW Heater 8 Water Out Temp	DegF
38	9TFWH5EXT	u9aw/9DSAI3:B35.PNT	FW Heater 5 Extraction Temp	DegF
39	9TFWH5DRAIN	u9aw/9DSAI3:B36.PNT	FW Heater 5 Drain Temperature	DegF
40	9TFWH6DRAIN	u9aw/9DSAI3:B38.PNT	FW Heater 6 Drain Temperature	DegF

Table 6: continued

41	9TFWH6EXTA	u9aw/9DSAI3:B39.PNT	FW Heater 6 Extraction Temp A	DegF
42	9TFWH7FWOUT	u9aw/9DSAI3:B4.PNT	FW Heater 7 Water Out Temp	DegF
43	9TFWH6EXTB	u9aw/9DSAI3:B40.PNT	FW Heater 6 Extraction Temp B	DegF
44	9TFWH7EXTA	u9aw/9DSAI3:B42.PNT	FW Heater 7 Extraction Temp A	DegF
45	9TFWH7DRAIN	u9aw/9DSAI3:B43.PNT	FW Heater 7 Drain Temperature	DegF
46	9TFWH8DRAIN	u9aw/9DSAI3:B44.PNT	FW Heater 8 Drain Temperature	DegF
47	9TFWH8EXTA	u9aw/9DSAI3:B45.PNT	FW Heater 8 Extraction Temp A	DegF
48	9TFWH6FWOUT	u9aw/9DSAI3:B5.PNT	FW Heater 6 Water Out Temp	DegF
49	9TFWH5FWOUT	u9aw/9DSAI3:B6.PNT	FW Heater 5 Water Out Temp	DegF
50	9TBFPSUCT	u9aw/9DSAI3:B7.PNT	BFP Suction Header Temperature	DegF
51	9TCRCWTRINW	u9aw/9DSAI4:B50.PNT	Circ Water Inlet Temp, West	DegF
52	9TAPHAIRIN	u9aw/9DSAI5:B31.PNT	Air Preheater A Air Inlet Temp	DegF
53	9TAPHBAIRIN	u9aw/9DSAI5:B39.PNT	Air Preheater B Air Inlet Temp	DegF
54	9COMMCHK	u9aw/9DSAI7:B26.RO01	Seconds since midnight	seconds
55	9TAPHAGASIN	u9aw/9DSGR:B3.OUT	Air Preheater A Gas Inlet Temp	DegF
56	9TAPHAGASOUT	u9aw/9DSGR:B4.OUT	Air Preheater A Gas Outlet Temp	DegF
57	9TAPHBGASIN	u9aw/9DSGR:B5.OUT	Air Preheater B Gas Inlet Temp	DegF
58	9TAPHBGASOUT	u9aw/9DSGR:B6.OUT	Air Preheater B Gas Outlet Temp	DegF
59	9TCRCWTRROUTE	u9aw/9DSGR:B7.OUT	Circ Water Outlet Temp, East	DegF
60	9TCRCWTROUTW	u9aw/9DSGR:B8.OUT	Circ Water Outlet Temp, West	DegF
61	9TPULVA	u9aw/9DS_CTRL:B1.PNT	Pulverizer A Outlet Temperature	DegF
62	9TPULVB	u9aw/9DS_CTRL:B2.PNT	Pulverizer B Outlet Temperature	DegF
63	9TPULVC	u9aw/9DS_CTRL:B3.PNT	Pulverizer C Outlet Temperature	DegF
64	9TPULVD	u9aw/9DS_CTRL:B4.PNT	Pulverizer D Outlet Temperature	DegF
65	9TPULVE	u9aw/9DS_CTRL:B5.PNT	Pulverizer E Outlet Temperature	DegF
66	9TPULVF	u9aw/9DS_CTRL:B6.PNT	Pulverizer F Outlet Temperature	DegF
67	9FFW1	u9aw/9FW:B7.PNT	Feedwater Flow #1 - Raw	KLB/HR
68	9FFW2	u9aw/9FW:B8.PNT	Feedwater Flow #2 - Raw	KLB/HR
69	9PDRUM	u9aw/9FW:B9.OUT	Selected Drum Pressure	PSIG
70	9TFWH3FWIN	u9aw/9FWAI1:B15.PNT	FW Heater 3 Water In Temp	DegF
71	9TFWH2FWIN	u9aw/9FWAI1:B16.PNT	FW Heater 2 Water In Temp	DegF
72	9TFWH1FWIN	u9aw/9FWAI1:B17.PNT	FW Heater 1 Water In Temp	DegF
73	9TFWH1FWOUT	u9aw/9FWAI1:B18.PNT	FW Heater 1 Water Out Temp	DegF
74	9TFWH1DRAIN	u9aw/9FWAI1:B20.PNT	FW Heater 1 Drain Temperature	DegF
75	9TFWH2DRAIN	u9aw/9FWAI1:B21.PNT	FW Heater 2 Drain Temperature	DegF
76	9TFWH3DRAIN	u9aw/9FWAI1:B22.PNT	FW Heater 3 Drain Temperature	DegF
77	9TFWH1EXT	u9aw/9FWAI1:B23.PNT	FW Heater 1 Extraction Temp	DegF
78	9TFWH3EXT	u9aw/9FWAI1:B24.PNT	FW Heater 3 Extraction Temp	DegF
79	9TFWH4EXT	u9aw/9FWAI1:B25.PNT	FW Heater 4 Extraction Temp	DegF
80	9PCRHT	u9aw/9FWAI1:B3.PNT	Cold Reheat Pressure at Turbine	PSIG
81	9TAMBIENT	u9aw/9FWAI1:B32.PNT	Ambient Air Temperature	DegF
82	9PDEA	u9aw/9FWAI1:B4.PNT	Deaerator Pressure	PSIG
83	9PFW	u9aw/9FWAI1:B6.PNT	FW Entering Economizer Pressure	PSIG

Table 6: continued

84	9PFWH2EXT	u9aw/9FWAI1:B7.PNT	FW Heater 2 Extraction Pressure	PSIG
85	9TCRHT	u9aw/9FWAI1:B9.PNT	Cold Reheat Temp at Turbine	DegF
86	9LFWH2	u9aw/9HP_HTRS:B1.PNT	FW Htr 2 Level	INCHES
87	9LFWH1	u9aw/9HP_HTRS:B2.PNT	FW Htr 1 Level	Inches
88	9LFWH3	u9aw/9HP_HTRS:B3.PNT	FW Htr 3 Level	Inches
89	9FCOND	u9aw/9HW:B3.PNT	Condensate Flow - Raw	KLB/HR
90	9JSS	u9aw/9LCAI1:B1.PNT	Station Service Load	MW
91	9FCOALA	u9aw/9PULVA:B1.PNT	Coal Flow Pulv A	LB/HR
92	9FPULVA	u9aw/9PULVA:B2A.OUT	Pulv A PA Flow Rate	KLB/HR
93	9FCOALB	u9aw/9PULVB:B1.PNT	Coal Flow Pulv B	LB/HR
94	9FPULVB	u9aw/9PULVB:B2A.OUT	Pulv B PA Flow Rate	KLB/HR
95	9FCOALC	u9aw/9PULVC:B1.PNT	Coal Flow Pulv C	LB/HR
96	9FPULVC	u9aw/9PULVC:B2A.OUT	Pulv C PA Flow Rate	KLB/HR
97	9FCOALD	u9aw/9PULVD:B1.PNT	Coal Flow Pulv D	LB/HR
98	9FPULVD	u9aw/9PULVD:B2A.OUT	Pulv D PA Flow Rate	KLB/HR
99	9FCOALE	u9aw/9PULVE:B1.PNT	Coal Flow Pulv E	LB/HR
100	9FPULVE	u9aw/9PULVE:B2A.OUT	Pulv E PA Flow Rate	KLB/HR
101	9FCOALF	u9aw/9PULVF:B1.PNT	Coal Flow Pulv F	LB/HR
102	9FPULVF	u9aw/9PULVF:B2A.OUT	Pulv F PA Flow Rate	KLB/HR
103	9PHRHT	u9aw/9RH:B15.PNT	Hot Reheat Pressure at Turbine	PSIG
104	9TRHOUT1	u9aw/9RH:B2.PNT	Reheat Outlet Temperature #1	DegF
105	9TRHOUT2	u9aw/9RH:B3.PNT	Reheat Outlet Temperature #2	DegF
106	9RHCOPPM1	u9aw/9RH_CO:B1.PNT	Bailey CO Monitor - Reheat Furnace	PPM
107	9FSOOTSH	u9aw/9SBP:B1.PNT	SH Furn Sootblowing Steam Flow	KLB/HR
108	9FSOOTRH	u9aw/9SBP:B2.PNT	RH Furn Sootblowing Steam Flow	KLB/HR
109	9PSOOT	u9aw/9SBP:B3.PNT	Sootblowing Steam Pressure	PSIG
110	9TSHOUT2	u9aw/9SH:B2.PNT	Superheat Outlet Temperature #2	DegF
111	9TSHOUT1	u9aw/9SH:B3.PNT	Superheat Outlet Temperature #1	DegF
112	9SHCOPPM1	u9aw/9SH_CO:B1.PNT	Bailey CO Monitor - Superheat	PPM
113	9FSHASPRAY	u9aw/9STAI1:B1.PNT	Superheat Spray A Flow	KLB/HR
114	9TRHABSPR	u9aw/9STAI1:B10.PNT	RH Attemp A Before Spray Temp	DegF
115	9TRHHdra	u9aw/9STAI1:B11.PNT	RH Outlet Header Temperature A	DegF
116	9THRHTA	u9aw/9STAI1:B14.PNT	Hot Reheat Temp at Int Vlv A	DegF
117	9THRHTB	u9aw/9STAI1:B15.PNT	Hot Reheat Temp at Int Vlv B	DegF
118	9FSHBSPRAY	u9aw/9STAI1:B2.PNT	Superheat Spray B Flow	KLB/HR
119	9FRHASPRAY	u9aw/9STAI1:B3.PNT	Reheat Spray A Flow	KLB/HR
120	9FRHBSPRAY	u9aw/9STAI1:B4.PNT	Reheat Spray B Flow	KLB/HR
121	9TSHHDra	u9aw/9STAI1:B7.PNT	SH Outlet Header Temperature A	DegF
122	9FAIRA1	u9aw/9TA:B1.PNT	Combustion Air Flow A1	Pct
123	9O2SHA	u9aw/9TA:B12.PNT	Excess O2 A in Superheat Furnace	%VOL
124	9O2SHB	u9aw/9TA:B13.PNT	Excess O2 B in Superheat Furnace	%VOL
125	9O2RHA	u9aw/9TA:B14.PNT	Excess O2 A in Reheat Furnace	%Vol
126	9O2RHB	u9aw/9TA:B15.PNT	Excess O2 B in Reheat Furnace	%Vol

Table 6: continued

127	9FAIRB1	u9aw/9TA:B3.PNT	Combustion Air Flow B1	Pct
128	9JGROSS	u9aw/9TC:B1.PNT	Unit Gross Generation	MW
129	9PTHROTTLEA	u9aw/9TC:B2A.PNT	Throttle Pressure at Stop Vlv A	PSIG
130	9PTHROTTLEB	u9aw/9TC:B2B.PNT	Throttle Pressure at Stop Vlv B	PSIG
131	9PSHWB	u9aw/9WBS:B26.PNT	SH Windbox Pressure	InH2O

APPENDIX F

Matlab Code

```
% Matlab script function to retrieve data from the IAServer
% of the 9th unit at the Kingston plant.
% Data is written to a specific directory named according to the
current date.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p=1000;
count=0;
to=10000;
t1=10;t2=20;time=clock;oldday=time(3);

while p==1000;

if count==0;
    count=1;sample_count=count
    t1=clock;t1=t1(5);
    channel=ddeinit('iaserver','objectaccess');
    z=6;
    data(count,z)=ddereq(channel,'u9aw/9TA:B1.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TA:B3.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B20.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9HW:B3.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FW:B7.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FW:B8.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DA:B13.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B16.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TC:B1.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9LCAI1:B1.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9HP_HTRS:B2.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9HP_HTRS:B3.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TA:B14.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TA:B15.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9BM:B0.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9BM:B0A.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FWAI1:B3.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FWAI1:B4.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FW:B9.OUT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FWAI1:B6.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B33.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FWAI1:B7.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B34.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B35.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B36.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9RH:B15.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9AFAI1:B11.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9AFAI1:B19.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9AFAI1:B16.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9AFAI1:B8.PNT',[],to);z=z+1;
```

[illegible]

```

rc=ddeterm(channel);
c=clock;
data(count,1:5)=c(2:6);
t2=clock;t2=t2(5);
end;
if t1~=t2;
    count=count+1;
    sample_count=count
    channel=ddeinit('iaserver','objectaccess');
    z=6;
    data(count,z)=ddereq(channel,'u9aw/9TA:B1.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TA:B3.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B20.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9HW:B3.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FW:B7.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FW:B8.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DA:B13.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B16.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TC:B1.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9LCAI1:B1.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9HP_HTRS:B2.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9HP_HTRS:B3.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TA:B14.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TA:B15.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9BM:B0.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9BM:B0A.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FWAI1:B3.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FWAI1:B4.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FW:B9.OUT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FWAI1:B6.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B33.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9FWAI1:B7.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B34.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B35.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B36.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9RH:B15.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9AFAI1:B11.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9AFAI1:B19.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9AFAI1:B16.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9AFAI1:B8.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9WBS:B26.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TC:B2A.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9TC:B2B.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI2:B27.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI5:B31.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSGR:B3.OUT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSGR:B4.OUT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI5:B39.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSGR:B5.OUT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSGR:B6.OUT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI3:B7.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSAI4:B50.PNT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSGR:B7.OUT',[],to);z=z+1;
    data(count,z)=ddereq(channel,'u9aw/9DSGR:B8.OUT',[],to);z=z+1;

```

```

data(count,z)=ddereq(channel,'u9aw/9FWAI1:B9.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B20.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B23.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B17.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B18.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B21.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B16.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B22.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B24.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B15.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9FWAI1:B25.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B36.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B35.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B6.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B38.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B39.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B40.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B5.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B43.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B42.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B4.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B44.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B45.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B2.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DSAI3:B3.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9STAI1:B14.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9STAI1:B15.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DS_CTRL:B1.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DS_CTRL:B2.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DS_CTRL:B3.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DS_CTRL:B4.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DS_CTRL:B5.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9DS_CTRL:B6.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9STAI1:B10.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9STAI1:B11.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9RH:B2.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9RH:B3.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9STAI1:B7.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9SH:B3.PNT',[],to);z=z+1;
data(count,z)=ddereq(channel,'u9aw/9SH:B2.PNT',[],to);
rc=ddeterm(channel);
c=clock;
data(count,1:5)=c(2:6);
t1=clock;t1=t2;

time=clock;month=time(2);day=time(3);

if day~=oldday;
    cleardata=1;
    if ((month<10)&(day>10));
        fname=['tvadata0' eval('num2str(month)')
eval('num2str(day)')];
    elseif ((day<10)&(month>10));

```

```

        fname=['tvadata' eval('num2str(month)') '0'
eval('num2str(day)')];
        elseif ((day<10)&(month<10));
            fname=['tvadata0' eval('num2str(month)') '0'
eval('num2str(day)')];
        else
            fname=['tvadata' eval('num2str(month)') eval('num2str(day)')];
        end;

        oldday=day;
    else
        cleardata=0;
    end;
    if day==oldday;
        if ((month<10)&(day>10));
            fname=['tvadata0' eval('num2str(month)')
eval('num2str(day)')];
        elseif ((day<10)&(month>10));
            fname=['tvadata' eval('num2str(month)') '0'
eval('num2str(day)')];
        elseif ((day<10)&(month<10));
            fname=['tvadata0' eval('num2str(month)') '0'
eval('num2str(day)')];
        else
            fname=['tvadata' eval('num2str(month)') eval('num2str(day)')];
        end;
    end;

    save(eval('fname'),'data');
    if cleardata==1;
        clear data;
        count=0;
    end;

end;
t2=clock;t2=t2(5);

end;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NNPLS MODEL DEVELOPMENT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function
[model_parameters]=fullnnpls5(train,validate,maxfactor,filter_size,sig
);

```

```

% BRANDON RASMUSSEN

```

```

% The University of Tennessee - Knoxville
% Department of Nuclear Engineering

% The following function creates Non-linear partial least squares
models
% for all of the signals represented in the specified data file.

% train is an nxp matrix of data where n is the number of samples in
the
% data set and p is the number of signals. This data will be used to
% train the neural networks, calculate means, calculate standard
% deviations, and to derive the transformation weights for the
% orthogonal projections.
% validate is an mxp matrix of data where m is the number of samples
in the
% data set and p is the number of signals. This data will be used
for
% cross validation during neural network training.
% maxfactor=the maximum number of factors (latent vectors)
% to use in the neural network models.
% filter_size is the window size of the applied median filter.
% If filter_size is set to 0, no filtering will be applied.
% model_parameters is the returned structure which contains all of
% required parameters of the developed models, including:
% p (input loadings)
% w (transformation weights)
% wbs (neural network weights and biases)
% f (the optimal number of latent vectors for each of the p models)
% means (the mean values of the signals in the data set "train")
% stds (the standard deviations of the signals in the data set
"train")
% ns (the number of signals p)
% mf (the maximum number of latent vectors)
% fs (the median filter window size)

% Filter the data if necessary.
if filter_size>0;
    train=medfilt1(train,filter_size);
    validate=medfilt1(validate,filter_size);
end;

% h is a counter which refers to the response variable.
% Each iteration of the "h-loop" will result in 1 complete
% inferential module for the prediction of a single response
variable.

[num_signals]=size(train,2);

for h=sig;
fprintf('\nCreating Model For Signal Number %i\n', h);

% The nlnumfacx3 function is the primary function which creates the
model,
% and returns the corresponding weights, etc.
168

```

```

[ factor,p,w,wbs,ym,ys,xm,xs]=nlnumfacx33(train,validate,maxfactor,h);

% The following lines append the information for the hth model to
% a set of larger matrices for storage

all_wbs((maxfactor*h)-(maxfactor-1):(maxfactor*h),:)=wbs;
all_p((maxfactor*h)-(maxfactor-1):(maxfactor*h),:)=p;
all_w(:,(maxfactor*h)-(maxfactor-1):(maxfactor*h))=w;
all_factor(h,1)=factor;
all_ym(1,h)=ym;
all_ys(1,h)=ys;
j2= (num_signals-1)*h;
j1=j2-(num_signals-2);
all_xm(j1:j2,1)=xm;
all_xs(j1:j2,1)=xs;

end;

% Create and save a structure containing all of the model parameters.

model_parameters=struct('p', all_p, 'w', all_w,'wbs', all_wbs, 'f',
all_factor, 'xm', all_xm,'xs',all_xs, 'ym', all_ym,'ys', all_ys, 'ns',
num_signals, 'mf', maxfactor, 'fs', filter_size);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function
[ factor,p,w,wbs,ym,ys,xm,xs]=nlnumfacx33(train,validate,maxfactor,h);

% BRANDON RASMUSSEN
% The University of Tennessee - Knoxville
% Department of Nuclear Engineering

% This function creates a NNPLS model for the hth signal in the data
% set train. It returns the input loadings, transformation weights,
% neural network weights and biases, means and standard deviations.
% It calls the function nnplsx3 to develop the neural network models
% of the inner relationships.

%train is an nxp matrix of data where n is the number of samples in
the
% data set and p is the number of signals. This data will be used to
% train the neural networks, calculate means, calculate standard
% deviations, and to derive the transformation weights for the
% orthogonal projections.
%validate is an mxp matrix of data where m is the number of samples in
the
% data set and p is the number of signals. This data will be used
for
% cross validation during neural network training.
%maxfactor=the maximum number of factors (latent vectors)
% to use in the neural network models.

```

%h is the signal index specifying which of the p signals to develop a model for.

```
%factor = the optimal number of latent vectors
% (based on prediction error for validation data)
%p = input score vectors
%w = input transformation weights
%wbs = the neural network weights and biases
%ym = the mean value of signal h
%ys = the standard deviation of signal h
```

```
% Separate data into training inputs (tx), training output (ty),
% validation inputs(vx), and validation output (vy).
[tx,ty]=divide(train,h);
[vx,vy]=divide(validate,h);
```

```
% Scale the training data to mean center unit variance
[ntx,xm,xs,nty,ym,ys] = prestd(tx',ty');ntx=ntx';nty=nty';
```

```
% Scale the validation data based on the mean and standard deviations
% of the training data.
nvx=trastd(vx',xm,xs);nvx=nvx';
nvx=trastd(vy',ym,ys);nvx=nvx';
```

```
% This function creates the inner relationship models
[p,w,wbs] = nnpls3(ntx,nty,nvx,nvy,maxfactor);
```

```
% Calculate mean squared error for prediction of validation data for
% each possible number of latent vectors from 1 to maxfactor.
for n=1:maxfactor;
    [y] = nnplsprdx(nvx,p,w,wbs, n);
    val_error(n,1)=mean((vy-(poststd(y,ym,ys))).^2);
end;
```

```
% Define the optimal number of latent vectors as that which minimized
% the MSE for prediction on the validation data.
factor=find((val_error)==min(val_error));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [x,y]=divide(data,npred);
```

```
% This function just separates the data into the desired input matrix
and
```

```
% output matrix, where the output matrix contains a single parameter.
```

```
% data = entire data set
```

```
% npred = the number of the parameter in the data set you wish to
predict
```

```
y=data(:,npred);
```

```
if npred==1;
```

```
    x=data(:,2:size(data,2));
```

```
elseif npred==size(data,2);
```

```
    x=data(:,1:(npred-1));
```

```
else;
```

```
    x=[data(:,1:(npred-1)) data(:,(npred+1:size(data,2)))];
```

```

end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [p,w,wbs] = nnpls3(x,y,vx,vy,maxrank)
%NNPLS   Non Linear Partial Least Squares with one response variable %
%
%   Where:
%
%   x       is the matrix of predictor variables (inputs)
%   y       is the matrix of response variables (outputs)
%   vx      is the matrix of predictor variable validation data
%   vy      is the matrix of response variable validation data
%   maxrank is an optional limit to the number of factors calculated
%
%   * - vx and vy are used in cross-validation neural network training
%   - *
%
%   p       is the matrix of input score vectors
%   w       is the matrix of weights
%   t       is the matrix of input loadings
%   u       is the matrix of output loadings
%   wbs     is the vector containing the inner relationships (i.e.
neural network weights)

[m,n] = size(x);
[m,l] = size(y);
if nargin == 5, i = min([(n+1), (m+1), maxrank]); else i = min([n m]);
end

for h = 1:i,
    u(:,h)=y;
    w(:,h)=(x'*y)/(sqrt((y'*x)*(x'*y)));
    t(:,h)=x*w(:,h);
    p(h,:)=(t(:,h)'*x)/(t(:,h)'*t(:,h));
    vt(:,h) = vx*w(:,h);
    vu(:,h) = vy;
    vp(h,:) = vt(:,h)' * vx / (vt(:,h)' * vt(:,h));

    % This calls the function plsnet to train a single hidden layer
    network to map the
    % hth inner relationship. The vt and vu vectors are used for
    cross-validation training.
    % Adjustments to the training routine can be made within the plsnet
    function,
    % such as the training algorithm, number of hidden nodes, and
    stopping criterion.
    [net]=plsnet2(t(:,h),u(:,h),vt(:,h),vu(:,h));

    % This loop extracts the weight and bias values from the returned
    network structure "net".
    % size(net.IW{1,1},1) returns the number of neurons in the single
    hidden layer network

```

```

% The code is written to allow up to four hidden neurons, however
I usually use 2
% The loop extracts the weights into the "wbs" matrix.
% From the wbs matrix, the weight vectors are assigned as
follows:
%      w1 - input layer weights
%      b1 - input layer bias
%      w2 - output layer weights
%      b2 - output layer bias

if size(net.IW{1,1},1)==1;
    wbs(h,:)=[size(net.IW{1,1},1) net.IW{1,1}' 0 0 0 net.b{1}' 0
0 0 net.LW{2,1} 0 0 0 net.b{2}];
    w1=wbs(h,2)';b1=wbs(h,6)';w2=wbs(h,10);b2=wbs(h,14);
elseif size(net.IW{1,1},1)==2;
    wbs(h,:)=[size(net.IW{1,1},1) net.IW{1,1}' 0 0 net.b{1}' 0 0
net.LW{2,1} 0 0 net.b{2}];
    w1=wbs(h,2:3)';b1=wbs(h,6:7)';w2=wbs(h,10:11);b2=wbs(h,14);
elseif size(net.IW{1,1},1)==3;
    wbs(h,:)=[size(net.IW{1,1},1) net.IW{1,1}' 0 net.b{1}' 0
net.LW{2,1} 0 net.b{2}];
    w1=wbs(h,2:4)';b1=wbs(h,6:8)';w2=wbs(h,10:12);b2=wbs(h,14);
elseif size(net.IW{1,1},1)==4;
    wbs(h,:)=[size(net.IW{1,1},1) net.IW{1,1}' net.b{1}'
net.LW{2,1} net.b{2}];
    w1=wbs(h,2:5)';b1=wbs(h,6:9)';w2=wbs(h,10:13);b2=wbs(h,14);
else;
end;

% The response variable (y) is deflated by the information
contained in the hth neural network
% The validation data for the response variable is also deflated
in a similar way.
% The relation in parenthesis represents the neural network's
prediction of the hth
%      output loading transferred to the output space by q
y = y - ((w2*(tansig(w1*t(:,h)'+(b1*ones(1,size(t,1)))))+b2)');
vy = vy -
((w2*(tansig(w1*vt(:,h)'+(b1*ones(1,size(vt,1)))))+b2)');

% The predictor variables are also deflated by the information
in the hth component
x = x - (t(:,h) * p(h,:));
vx= vx - (vt(:,h) * vp(h,:));
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function [net] = plsnet2(t,u,vt,vu);
% plsnet2.m
%
% This function takes the inner relationship

```

```

%      in a PLS model and fits an ANN to it.
%
% t      is the vector of input scores
% u      is the vector of output scores
% vt     is the vector of input validation scores
% vu     is the vector of output validation scores
% net    is the network model for the PLS factor
%
%

net=newff(minmax(t'),[2 1],{'tansig','purelin'},'trainscg');
net.trainParam.show = 100;
net.trainParam.epochs=800;
net=init(net);
VV.P=vt';VV.T=vu';
net=train(net,t',u',[],[],VV);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [y,r] = nnplsprdx(x,p,w,wbs,n);
% nnplsprd.m
% Predicts unknown concentrations using NLPLS.
% [y, r] = plsprd(x, p, w, b, n)
%
% y is the matrix of response variables (outputs)% r is the
matrix of spectral residuals after decomposition
% x is the matrix of predictor variables (inputs)
% p is the matrix of input score vectors
% w is the matrix of weights
% b is a vector of objects containing the neural networks
% n is the number of factors to use
%
[maxrank, h] = size(p);
if nargin > 4
    maxrank = min([n, maxrank]);
end
y=0;
[j, i] = size(x);
for h=1:maxrank
    t(:,h) = x * w(:,h);

    x = x - t(:,h) * p(h,:);

    % Determine the number of hidden nodes used in the AANN for each
component
    % then extract the corresponding weight and bias values from the
wbs matrix
    if wbs(h,1)==1;
        w1=wbs(h,2)';b1=wbs(h,6)';w2=wbs(h,10);b2=wbs(h,14);
        %y = y + (((tansig(wbs(h,2)*t(:,h)+wbs(h,3))*wbs(h,4))+wbs(h,5))
* q(h,:));
        %y = y +
(((w2*(tansig(w1*t(:,h)'+(b1*ones(1,size(t,1)))))+b2)')*q(h,:));

```

```

elseif wbs(h,1)==2;
    w1=wbs(h,2:3)';b1=wbs(h,6:7)';w2=wbs(h,10:11);b2=wbs(h,14);
    %y = y +
    ((w2*(tansig(w1*t(:,h)'+(b1*ones(1,size(t,1)))))+b2)'*q(h,:));
    % y = y + ((
wbs(h,6:7)*([(tansig(wbs(h,2)*t(:,h)+wbs(h,4))';(tansig(wbs(h,3)*t(:,
h)+wbs(h,5))')]+wbs(h,8) )'*q(h,:))';
    elseif wbs(h,1)==3;
        w1=wbs(h,2:4)';b1=wbs(h,6:8)';w2=wbs(h,10:12);b2=wbs(h,14);
        %y = y +
        ((w2*(tansig(w1*t(:,h)'+(b1*ones(1,size(t,1)))))+b2)'*q(h,:));
    elseif wbs(h,1)==4;
        w1=wbs(h,2:5)';b1=wbs(h,6:9)';w2=wbs(h,10:13);b2=wbs(h,14);
        %y = y +
        ((w2*(tansig(w1*t(:,h)'+(b1*ones(1,size(t,1)))))+b2)'*q(h,:));

    else;
    end;

    % Update y by adding the Contribution from AANN h.
    y = y + ((w2*(tansig(w1*t(:,h)'+(b1*ones(1,size(t,1)))))+b2)');

end
r=x;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NNPLS MODEL VALIDATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Y] = nnpls_predict2(X,model_parameters);

% Creates NNPLS predictions for a full set of inferential models
% given the required parameters. Can take as input a matrix of data
% containing m observations, or a vector containing a single
observation.
% Note that the number of rows for X doesn't matter, but there must
be n
% columns, where n is the number of signals in the current model.

% INPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% X is a test data set of dimension m x n, where n is the number of
signals
% Note that the data here must have the same column structure
% as used to create the set of "model_parameters"
% model_parameters is a structure containing all of the necessary
parameters
% of a complete NNPLS model, it is output from the nnpls1 function.
% sig is a vector of signal indices for which predictions are desired.
% Usually this is set to 0, so that predictions for all signals are
performed.

```

```

% If sig is not set to zero, the structure of the output will still
be the same,
% however, the only nonzero entries in the output matrix will be for
the specified sig
% filt_option provides the option to median filter using a window size
of 'fs', where
% 'fs' is contained in the model_parameters structure. If
filt_option is set to 1
% median filtering will be applied, otherwise no filter will be
used.
% Set scale_option to 1 if MCUV scaling is to be applied, using the
means and
% standard deviations stored in the model_parameters file.
%

% OUTPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Y are the output predictions. The dimensions of Y are the same as X,
m x n.

% Extraction of Parameters from Structure
P=getfield(model_parameters,'p');
W=getfield(model_parameters,'w');
WBS=getfield(model_parameters,'wbs');
F=getfield(model_parameters,'f');
mf=getfield(model_parameters,'mf');

for h=1:size(X,2);

    [testx,testy]=divide(X,h);
    p=P(((mf*h)-(mf-1)):(mf*h),:);
    w=W(:,((mf*h)-(mf-1)):(mf*h));
    wbs=WBS(((mf*h)-(mf-1)):(mf*h),:);
    f=F(h);
    [Y(:,h)] = nnpls_predict8(testx,w,wbs,p,f);

end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [yp] = nnpls_predict8(x,w,wbs,p,fac);

% Creates an NNPLS predictionfor a single inferential model
% given the required parameters.

% INPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% x is the matrix of predictor variables
% y is the response vector

```

```

% w is the matrix of weights
% wbs is a vector of objects containing the neural networks
% fac is the number of latent vectors to try

% OUTPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% yp is the best prediction of the response variable,
% factor is the optimal number of latent vectors to use

% Initialize y
yp=zeros(size(x,1),fac);

for h=1:fac;

    % Calculate the latent vectors
    T(:,h)=x*w(:,h);

    % Determine the number of hidden nodes used in the AANN for
    each component
    %then extract the corresponding weight and bias values from the wbs
    matrix

    if wbs(h,1)==1;
        w1=wbs(h,2)';b1=wbs(h,6)';w2=wbs(h,10);b2=wbs(h,14);
    elseif wbs(h,1)==2;
        w1=wbs(h,2:3)';b1=wbs(h,6:7)';w2=wbs(h,10:11);b2=wbs(h,14);
    elseif wbs(h,1)==3;
        w1=wbs(h,2:4)';b1=wbs(h,6:8)';w2=wbs(h,10:12);b2=wbs(h,14);
    elseif wbs(h,1)==4;
        w1=wbs(h,2:5)';b1=wbs(h,6:9)';w2=wbs(h,10:13);b2=wbs(h,14);
    else;
    end;

    % Update y by adding the Contribution from ANN h.
    if h==1;

```

```

        yp(:,h) =
        ((w2*(tansig(w1*T(:,h)'+(b1*ones(1,size(T,1)))))+b2)');
    else
        yp(:,h) = yp(:,h-
1)+((w2*(tansig(w1*T(:,h)'+(b1*ones(1,size(T,1)))))+b2)');
    end;

    % Input Residual Matrix Deflation

    x=x-(T(:,h) * p(h,:));

end

yp=yp(:,h);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The Sequential Probability Ratio Test and Alarm Logic
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function
[ALARM]=sprt(R,sprt_mean,sprt_variance,var_mult,mean_mult,alpha,beta,n
a,inc):

% This function implements the SPRT and the alarm logic on a set of
% residuals
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INPUTS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% R is a matrix of residuals (n x p)
% sprt_mean= is a vector of the mean value parameters for the sprt
(1 x p)
% sprt_variance= is a vector of the variance parameters for the SPRT
(1 x p)
% var_mult= is an optional multiplicative factor applied to the
variance parameters.

```

```
% mean_mult= is the fraction of the mean value at which to set the
faulted mean parameters
% alpha= false alarm probability
% beta= missed alarm probability
% na= number of alarms necessary to occur in "inc" to produce a true
alarm
% inc= increment span in which the "na" numbr of alarms must be
observed to initiate an alarm report.
```

```
%%%%%%%%%
```

```
% OUTPUTS %
```

```
%%%%%%%%%
```

```
%ALARM is a matrix of the same dimension as R containing the binary
% alarm status for the residuals in R
```

```
sprt_m1=abs(mean_mult*sprt_mean)';
sprt_m2=-sprt_m1;
```

```
A = log(beta/(1-alpha));
B = log((1-beta)/alpha);
```

```
for i=1:size(R,1);
[num_points,num_sigs]=size(R);
```

```
for k=1:num_sigs;
sprt1(i,k)=((sprt_m1(k)/(sprt_variance(k)))*ones(num_points,1)).*(R(:,
k)-(0.5*sprt_m1(k)));
sprt2(i,k)=((sprt_m2(k)/(sprt_variance(k)))*ones(num_points,1)).*(R(:,
k)-(0.5*sprt_m2(k)));
end;
```

```
if i>1;
```

```
for k=1:num_sigs;
    sprt1(i,k)=sprt1(i-1,k)+sprt1(i,k);
    sprt2(i,k)=sprt2(i-1,k)+sprt2(i,k);
```

```

        if (sprt1(i,k)<=A);
            ssprt1(i,k)=0;
            alarm1(i,k)=0;
        elseif (sprt1(i,k)>B);
            ssprt1(i,k)=0;
            alarm1(i,k)=1;
        else
            alarm1(i,k)=0;
            ssprt1(i,k)=sprt1(i,k);
        end;
        sprt1(i,k)=ssprt1(i,k);

        if (sprt2(i,k)<=A);
            ssprt2(i,k)=0;
            alarm2(i,k)=0;
        elseif (sprt2(i,k)>B);
            ssprt2(i,k)=0;
            alarm2(i,k)=1;
        else
            alarm2(i,k)=0;
            ssprt2(i,k)=sprt2(i,k);
        end;
        sprt2(i,k)=ssprt2(i,k);

    end;

    alarm(i,:)=alarm1(i,:)|alarm2(i,:);
    if i<(inc+1);
        talarm(i,:)=sum(alarm(1:i,:));
    elseif i>(inc);
        talarm(i,:)=sum(alarm( (i-inc+1):i,:));
    end;
    ALARM(i,:)=talarm(i,:)>=na;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

VITA

Brandon Rasmussen was born on February 13, 1973 in West Islip, NY. He grew up in Copaigue, NY, moved to Evans City, PA at the age of 10, and graduated from Vincentian High School in Pittsburgh, PA in 1990. He began college at Kent State University, Kent, OH, transferred the following year to Gannon University, Erie, PA, and received a Bachelor of Science in Health Physics from Francis Marion University, Florence, SC, in August of 1995.

As a Health Physics intern, he spent the summer of 1993 at Carolina Power and Light's Shearon Harris Nuclear Power Plant, in New Hill, NC. He was a member of the Francis Marion University Health Physics club from 1993 through 1995, and was the active president from 1994 to 1995. He received a Francis Marion University Alumni Scholarship in 1993 and 1994, was awarded a National Academy for Nuclear Training Scholarship in 1994, and he received the Carolina Power and Light Environmental Health Physics award in 1995.

He entered the Nuclear Engineering graduate program at the University of Tennessee in 1997. In 1999, he and John P. Getty, under the direction of Dr. Arthur E. Ruggles, won first place honors in the American Nuclear Society's Graduate Design Competition for their design entitled "Pressure Vessel Diameter Reduction of a Modular Pebble Bed Reactor." The American Nuclear Society has awarded him Graduate Scholarships in 1999, and 2001.

He has completed two conference publications as the primary author, and has co-authored 3 other conference publications. Two other presentations are being prepared for May of 2002, and a submission has been made for publication in *Nuclear Technology*. All of these publications are related to the work presented in this thesis regarding instrument surveillance and calibration verification.

In February of 2002, he is traveled to Halden, Norway to collaborate with personnel, at the Halden Reactor Project. The purpose of the collaboration is to evaluate the utility of the NNPLS algorithm in their existing Plant Evaluation and Analysis by Neural Operators (PEANO) software. He will continue work on his Doctor of Philosophy in Nuclear Engineering at the University of Tennessee.

The author is a member of the American Nuclear Society and Phi Kappa Phi National Honor Society.