Dynamic Modelling of a Control Valve

A. Helling

Dissertation submitted in partial fulfilment of the requirements of the degree
Magister Ingeneriae in Electronic Engineering
at the North-West University

Supervisor: **Professor G. van Schoor**
Assistant Supervisor: **Professor A. S. J. Helberg**

2004

Potchefstroom

Summary

Process industries today enjoy significant benefits from advances made in the field of simulation as well as technologies that model normal plant operation. Plants however continue to suffer during abnormal operation such as startup, shutdown and equipment failures leading to production losses or personal injury.

One of the key elements that determines the operational safety of any plant is the training and technical knowledge of its personnel regarding plant behaviour. Simulators play a vital role in training personnel, preparing them for normal plant operation, abnormal plant operation as well as emergency and accident situations. This would not only enhance plant safety, but also decrease total downtime.

In order to create such a simulator, mathematical models are required for each of the numerous components within the plant. This dissertation focuses on the development of a mathematical model for a control valve; a component commonly found in various industries to control and manipulate processes. Different modelling methods are compared, taking into account applicable modelling criteria such as training data, algorithm complexity, oscillations near endpoints, degree of system integration and model limitations. Based on these criteria, fuzzy logic with the nearest neighbourhood clustering algorithm is chosen as an appropriate modelling technique especially due to its ability to deal with large quantities of data.

In order to meaningfully train the fuzzy logic system (FLS), a comprehensive set of physical operational data is required, covering all the different operational characteristics. To capture physical data, the development of a data acquisition (DAQ) system is introduced using two common DAQ systems to create a hybrid solution. Transducer signals are converted from mA to V, using custom developed signal conversion hardware. This will allow data to be sampled by a standard DAQ card and processed by accompanying software. Two post-processing software applications are created. The first application solves the governing equations (mass rate of flow, Reynolds number, expansibility factor and choke status) and the second application is used to graphically display the acquired and calculated data. A set of experiments are conducted, covering all relevant working areas, to capture the behaviour of the control valve. This is achieved using five initial pressures ranging from 200 kPa to 400 kPa in increments of 50 kPa. At each initial pressure a set of unit step responses with valve command signals ranging from 0 % to 100 % in increments of 5 % is acquired. 24 data files at each initial pressure set (200, 250, 300, 350 and 400 kPa) are acquired.

Before training the FLS, the optimal fuzzy logic parameters need to be determined e.g. radius (r), sigma (σ), the number of time delays, the time delay increments and the impact of the input signals. Determining these parameters is an iterative process. Only a single data set, with initial pressure of 300 kPa, is used to derive the optimal fuzzy logic parameters. Four performance criteria namely maximum error average (MEA), mean square error (MSE), root mean square error (RMSE) and coefficient of variation of the error residuals (CVRE) are used as benchmarks to obtain the optimal fuzzy parameters.

During both the search for the optimal fuzzy parameters and the training of the fuzzy models using these optimal fuzzy parameters, 70 % of the data are used for training and verification while the remaining 30 % of the data are used for validation.

Once the optimal fuzzy parameters are obtained using only the single data set, it is used to derive a number of fuzzy control valve models based on all the available data sets. All derived fuzzy models use the same parameters, except for a unique random file sequence associated with each of the models. The only prerequisite for the fuzzy models is that the generated file sequence be truly random. Irrespective of the random file sequence, fuzzy models with the same parameters, produce models with more or less the same performance. Therefore the performance criteria (MEA, MSE, RMSE and CVRE), for each data file, in the respective initial pressure sets, remains more or less the same.

This method is found to be very useful in deriving a dynamic fuzzy logic control valve model. Averaging the performance criteria of these five models, an overall modelling accuracy of 90 % is achieved.

It is recommended that a flow meter be installed to measure the mass rate of flow through the pipe network. This eliminates the need for an orifice, differential pressure transducer and the use of the first principle governing equation for the mass rate of flow. If a flow meter cannot be installed, a differential pressure transmitter with large range should be considered accompanied by a single orifice.

Opsomming

Vandag geniet vervaardiginsindustrieë aansienlike voordele as gevolg van vooruitgang in simulasies en tegnologieë wat die normale bedryf van aanlegte modelleer. Dit is egter so dat talle aanlegte steeds probleme ondervind tydens abnormale bedrywighede soos aanskakelings, afskakelings en die onklaar raak van toestelle wat lei tot produksieverliese en beserings.

Een van die belangrikste aspekte wat die operationele veiligheid van enige aanleg bepaal is die opleiding en tegniese kennis van personeel aangaande die gedrag van die aanleg. Simulators speel 'n belangrike rol in die opleiding van personeel wat hulle voorberei vir normale aanlegbedrywighede, abnormale bedrywighede en nood- en ongeluksituasies. Dit sal nie alleen aanleg veiligheidverbeter nie, maar ook verliese as gevolg van staantye beperk.

Om geskikte simulators te ontwikkel word wiskundige modelle benodig vir elk van die vele komponente in die aanleg. Hierdie studie fokus op die ontwikkeling van 'n wiskundige model vir 'n beheerklep; 'n onderdeel wat algemeen gebruik word in menige industrieë om prosesse te beheer en te manipuleer. Verskeie modelleringsprosesse word vergelyk, waar kriteria soos leerdata, algoritmekompleksiteit, ossillasies naby die eindpunte, modelintegrasie en modeltekortkominge in ag geneem word. Deur gebruik te maak van hierdie kriteria is wasige logika met die naaste omgewing groeperingsalgoritme 'n goeie modelleringstegniek, veral as gevolg van die vermoë om groot hoeveelhede data te verwerk.

Om die wasige logiese stelsel doeltreffend op te lei word 'n omvattende stel operasionele data benodig. Om díe operasionele data te meet, word 'n hibriede meetstelsel ontwerp, gebasseer op twee algemene meetstelsels. Omsetterseine word van mA na V omgeskakel deur doelontwikkelde seinomskakelingshardeware. Hierdie, saam met die meegaande sagteware, stel die datameetstelselkaart in staat om data te meet. Twee na-prosesserings sagtewareprogramme is ontwikkel. Die eerste program los beskrywende vergelykings op (massa vloei, Reynoldsgetal, uitsettingsfaktor en die smoorstatus). Die tweede program word gebruik om die data grafies voor te stel. 'n Stel eksperimente, wat alle relevante werkspunte insluit, is uitgevoer om die gedrag van die beheerklep te meet. Vyf aanvangs-drukke van 200 kPa tot 400 kPa in inkremente van 50 kPa is as werkspunte gekies. By elk van die aanvangsdrukke is 'n stel eenheidstrapresponse met klepbevelseine, wat wissel van 0 % tot 100 % in inkremente van 5 %, gemeet. 24 data leêrs by elk van die aanvangsdrukke (200, 250, 300, 350 en 400 kPa) is gemeet.

Voordat die wasige logiese stelsel (WLS) geleer word, moet die optimale wasige logiese parameters bepaal word nl. radius (r), sigma (σ), die aantal tydvertragings, die tydvertragings-inkremente en die impak van die insetseine. Om hierdie parameters te bepaal, word 'n iteratiewe proses benodig. 'n Enkele data stel met aanvangsdruk van 300 kPa word gebruik om die optimale wasige logiese parameters te bepaal. Vier prestasiekriteria genaamd maksimum gemiddelde fout (MEA), gemiddelde kwadraat fout (MSE), wortelgemiddelde kwadraat fout (RMSE) en koeffisiënt van variasie van die foutreste (CVRE) word as verwysing gebruik om die optimale wasige parameters te bepaal.

Met die bepaling van die optimale wasige parameters en die opleiding van die wasige modelle, word 70 % van die data vir opleiding en verifikasie gebruik en 30 % vir validasie.

Nadat die optimale wasige parameters bepaal is met behulp van die enkele datastel, word hierdie parameters gebruik om 'n aantal wasige beheerklepmodelle, gebasseer op al die beskikbare datastelle, te ontwikkel. Al hierdie ontwikkelde modelle maak gebruik van dieselfde parameters, behalwe vir 'n unieke willekeurige data leêr volgorde wat geassosieer word met elk van die modelle. Die enigste voorvereiste vir die wasige modelle is dat die data leêr volgorde willekeurig moet wees. Ongeag die data leêr volgorde, het wasige modelle met dieselfde parameters, modelle met min of meer dieselfde prestasies tot gevolg. Daarom bly die prestasiekreteria (MEA, MSE, RMSE en CVRE) vir elk van die data leêrs, in die onderskeidelike aanvangsdrukstelle, min of meer dieselfde.

Daar is bevind dat hierdie metode effektief is om dinamiese wasige logiese beheerklep-modelle te ontwikkel. Die gemiddelde prestasiekriteria van die vyf modelle het 'n 90% modelleringsakkuraatheid behaal.

Dit word aanbeveel dat 'n vloeimeter gebruik word om die massavloei in die pypnetwerk te bepaal. Dit skakel die gebruik van 'n smoorplaat, 'n drukverskilomsetter en die gebruik van die eerste beginsel vergelyking vir die massavloei uit. As 'n vloeimeter nie beskikbaar is nie, moet dit oorweeg word om 'n drukverskilomsetter met groter vermoë en 'n enkele meegaande smoorplaat te gebruik.

Acknowledgements

First and foremost, I would like to thank the PBMR for granting me the opportunity to further my studies and for funding this research.

There are a few people which I would like to thank. Without their help this study would not have been successful. They follow in no particular order.

My supervisor professor George van Schoor for his excellent guidance, support and most valued input. I cannot thank you enough.

Mr. Marius Jansen van Vuuren and Mr. Willem van Niekerk (Mechanical Advisors). Without their help and experience I would never have grasped all the mechanical aspects within this dissertation.

My father, Mr. A. L. Helling, for proofreading this dissertation.

And last but definitely not least, my family, that always supported, inspired and guided me. I am truly grateful for all the help and support that you have given me during this study.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
A/D	Analog to Digital
ADC	Analog Digital Converter
AEM	Abnormal Event Management
AI	Artificial Intelligent
AIGND	Analog Input Ground
AISENSE	Analog Input Sense
ASIC	Application Specific Integrated Circuit
ASME	American Standard for Mechanical Engineers
CD	Compact Disk
CMOS	Complementary Metal Oxide Semiconductor
CVRE	Coefficient of Variation of the Error Residuals
D/A	Digital to Analog
DAQ	Data Acquisition System
DAQ-STC	Data Acquisition System Time Control
DIFF	Differential
DMA	Direct Memory Access
FL	Fuzzy Logic
FLS	Fuzzy Logic System
GUI	Graphical User Interface
IEEE	Institute for Electrical and Electronic Engineers
ISO	International Organisation for Standardisation
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
MAX	Measure and Automation Explorer
MDF	Maximum Time Delay Factor
MEA	Maximum Error Average
MF	Membership Function
MSE	Mean Square Error
NASA	National Aeronautic and Space Administration
NI	National Instruments
NICS	Nitrogen Inventory Control System
NI-PGIA	National Instrument Programmable Gain Instrument Amplifier
NNC	Nearest Neighbourhood Clustering
NRSE	Nonreferenced Single Ended
PBMM	Pebble Bed Micro Model
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PLC	Programmable Logic Controller

Abbreviation	Description
RMSE	Root Mean Square Error
RSE	Referenced Single Ended
SCADA	Supervisory Control And Data Acquisition
SCSI	Small Computer System Interface
TTL	Transistor-Transistor Logic
USB	Universal Serial Bus
VI	Virtual Instrument
VM	Virtual Machine (Java)

CHAPTER 1

Introduction

This chapter aims to provide introductory information regarding the necessity of simulators and the modelling of control valves. The problem statement of this dissertation is given and the methodology followed is discussed. Thereafter a concise overview of the dissertation is given.

1.1 SIMULATORS AND SIMULATIONS

Process industries today enjoy significant benefits from advances made in the field of simulation as well as technologies that model normal plant operation. Plants however continue to suffer during abnormal operation such as startup, shutdown and equipment failures leading to production losses or personal injury.

One of the key elements that determines the operational safety of any plant is the training and technical knowledge of its personnel regarding plant behaviour. Simulators play a vital role in training personnel, preparing them for normal plant operation, abnormal plant operation as well as emergency and accident situations. This would not only enhance plant safety, but also decrease total downtime.

In order to create such a simulator, mathematical models are required for each of the numerous components within the plant. This dissertation focuses on the development of a mathematical model for a control valve; a component commonly found in various industries to control and manipulate processes. Different modelling methods are compared, taking into account applicable modelling criteria such as training data, algorithm complexity, oscillations near endpoints, degree of system integration and model limitations. Based on these criteria, fuzzy logic with the nearest neighbourhood clustering algorithm is chosen as an appropriate modelling technique especially due to its ability to deal with large quantities of data.

Designing such a training facility requires that the plant be sub-divided into smaller, more manageable sections, which in turn will then be modelled and added to the final training facility, which is known as the training simulator. This training simulator normally consists of both a software user environment and numerous hardware devices. The graphical user interface (GUI) simulation software package uses the derived models to present the model-based-generated data as a "virtual image" of the production facility and/or plant. The simulation software is

normally interpreted by a personal computer depending on the requirements and needs of the production facility and/or plant and communicates through a network connection to other hardware devices, which could possibly include and manage these hardware-based models.

The training simulator should be able to train operators, predict plant behaviour and detect early equipment failures. This dissertation aims to aid future companies and plant operators to derive artificial intelligent (AI) mathematical models for control valves with accuracy and ease. Using this method implies that the control valve is seen as either a grey-box or black-box device (refer to *Chapter 2*).

1.2 PROBLEM STATEMENT

A dynamic mathematical model of a control valve needs to be developed. In order to model such a control valve, incorporating both static and dynamic valve behaviour, control valve parameters such as upstream and downstream static pressures, temperature, valve relative opening and valve command signal need to be acquired. A block diagram illustrating the perceived input and output parameters is shown in Figure 1.1.

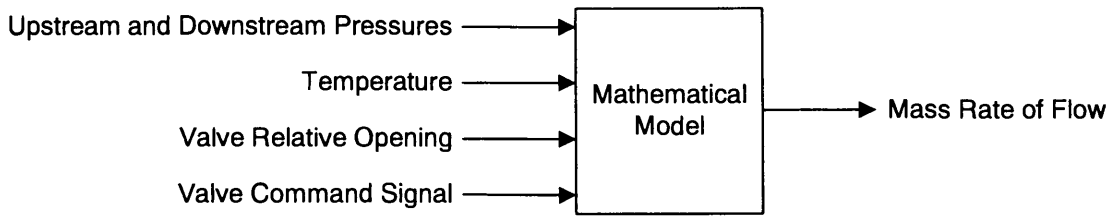


Figure 1.1. *Mathematical Model Block Diagram.*

The following sub-problems are identified from Figure 1.1, in order to derive and verify the dynamic mathematical model:

- Physical system
  - Transducers
  - Orifices
  - Pressure vessels
  - Valves (other than the control valve)
  - Compressor
  - Air dryer
- Data acquisition (DAQ) system
  - Develop a new or use an existing DAQ system
  - Develop custom DAQ and data management software
  - Conduct experiments to capture the control valve behaviour

- Finding an appropriate modelling method
  - First principle modelling methods
    - First principle modelling techniques
    - Curve fitting techniques
    - Experimental modelling techniques
  - Dynamic modelling methods
    - Artificial modelling techniques
- Develop the mathematical model
- Evaluate the mathematical model

The aim of this study is to design a generic procedure, which could be used in any production facility, to develop dynamic mathematical models for control valves. These models must predict plant behaviour in parallel with the real production facility and should therefore not be computationally demanding.

1.3 PROPOSED METHODOLOGY

Due to many uncertainties in the behaviour of a control valve, a first principle model might be difficult to derive as well as unnecessarily complex. Two relatively new methods, namely fuzzy logic (thinking patterns of the human brain) and artificial neural networks (physical construction of the human brain) may be used to derive such a dynamic control valve model.

A block and flow diagram, stipulating the proposed modelling procedure, used to derive a dynamic mathematical model, can be seen in Figure 1.2. In this case the physical system not only represents the control valve but also the measuring system components including: transducers, orifices, pressure vessels, a compressor and an air drier.

The DAQ system comprises hardware, custom software and a set of experiments. The signal conversion hardware, converts the industry standard 4 – 20 mA signals to 1 – 5 V signals, which can be sampled by the DAQ card.

The custom software is used to manage, process and log the acquired data. Three groups of applications are developed; a DAQ process control application, a post-processing application and a data plot application. These applications are respectively responsible for acquiring and managing the data, processing, re-sampling and re-formatting the acquired data and displaying the data.

To capture the behaviour of the control valve, numerous experiments need to be conducted over various working ranges. Such experiments include the unit step experiment, multi step experiment and many more.

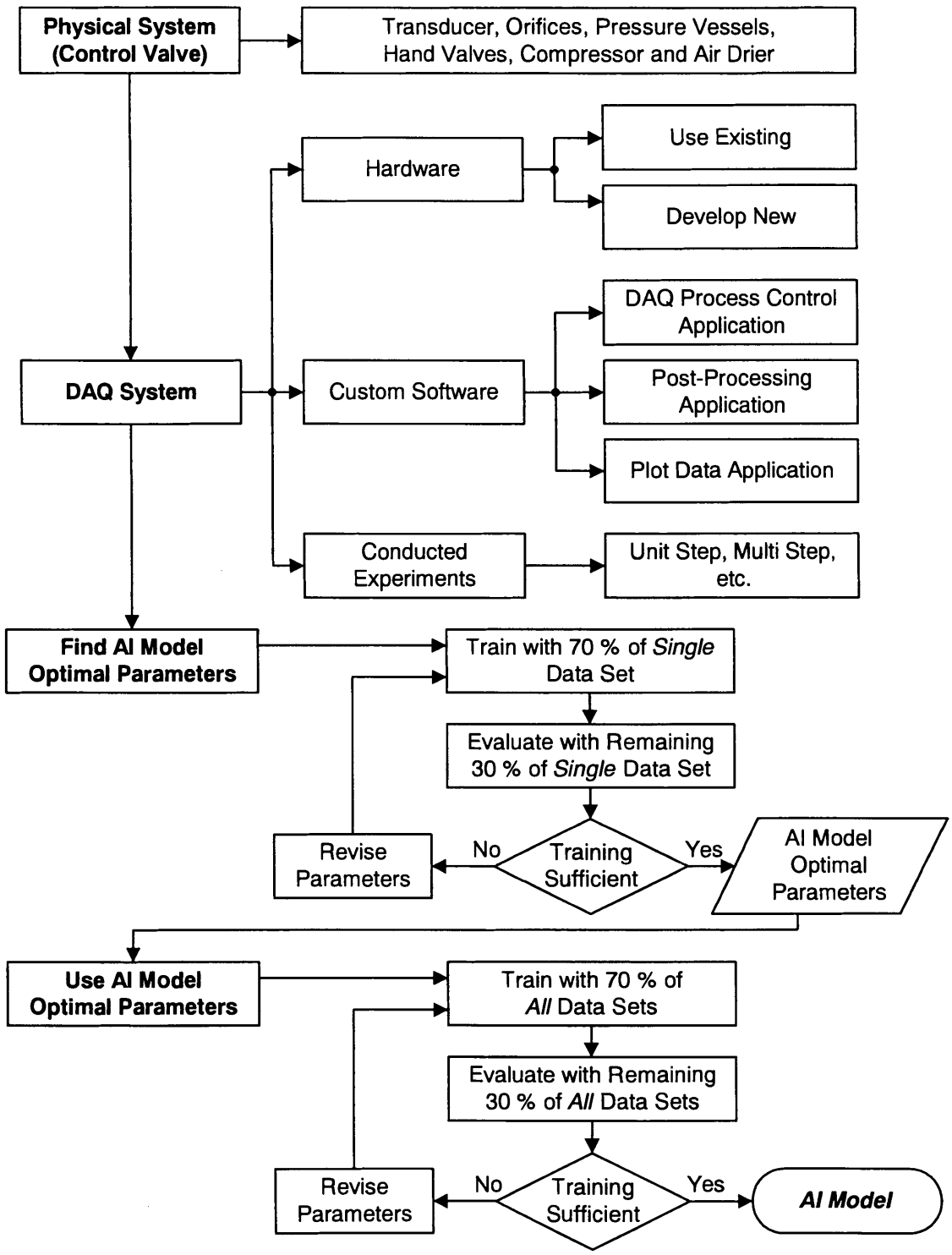


Figure 1.2. *Modelling Process Block and Flow Diagram.*

Finding the optimal AI model parameters, involves using a single data set. This is done to minimise training, verification and validation time. From Figure 1.2, it is clear that an iterative

process is used to find these optimal parameters. As usually the case with artificial neural networks (ANN) and fuzzy logic systems (FLS), 70 % of the data are used to train the system and the remaining 30 % are used to evaluate the system. If the training is sufficient, these parameters are used to similarly create an AI valve model by training an AI system using all the available data. If however training is not sufficient the parameters have to be revised, and another training and evaluation cycle commences.

1.4 OVERVIEW OF DISSERTATION

The main focus of *Chapter 2* is modelling methods. This chapter covers the modelling process, the construction of models and the various model types. Fundamental modelling methods are compared with dynamic modelling methods. Fundamental modelling methods include first principle models, model fitting and experimental modelling, whereas dynamic modelling methods include fuzzy logic and artificial neural networks. These modelling methods are compared. Fuzzy logic, with its nearest neighbourhood clustering algorithm, provides an adaptive framework specifically well suited for large sample problems.

Chapter 3 covers the dynamic capturing of data. Two commonly used DAQ systems are introduced on which the final hybrid DAQ system is based and developed. This DAQ system is capable of acquiring both the static and dynamic behaviour of the control valve. In order to capture, manipulate, log and display the acquired data, the DAQ system comprises signal conversion hardware (converting the mA signals to V signals), custom software (managing and displaying acquired data) and an experimental procedure (to capture the behaviour of the control valve). The pipe network setup comprising the gas supply system, the mass flow calculation system and the control valve itself, is also discussed. Typical control valve parameters include upstream and downstream static pressures, temperature and valve relative opening are captured. These are used to calculate the mass rate of flow through the pipe network and thus through the control valve.

In order to minimise the training, verification and validation times, a procedure to find the optimal fuzzy logic system (FLS) parameters, based on only a single data set, is introduced in *Chapter 4*. These parameters include the fuzzy radius (r) and sigma (σ), the time delay factors and the time delay increments. The optimum parameters are determined by comparing the respective performance measurement criteria namely maximum error amplitude (MEA), mean square error (MSE), root mean square error (RMSE) and the coefficient of variation of the error residuals (CVRE). Following the outlined procedure, fuzzy parameters resulting in a good overall performance can be obtained.

The results of the final control valve models, based on all the data sets, are given in *Chapter 5*. These models are derived, based on the optimum parameters obtained in *Chapter 4*. The performance measurement criteria, given in *Chapter 4*, are used to determine the effectiveness of the derived control valve models. The optimal parameters are firstly verified and then used to derive five fuzzy logic control valve models. These models are compared and an average system performance determined, based on the performance measurement criteria.

Conclusions are drawn and recommendations discussed in *Chapter 6*. Possibilities of system improvements are discussed, which may be implemented to improve future model performance.

The hardware user manual is covered in *Appendix A*. This includes the DAQ hardware signal converters, the DAQ card and the A/D process. The DAQ hardware signal converters include topics such as hardware system interface, converter layout and converter calibration. The features, signal connections and DAQ card pinouts are covered in the DAQ card section. A concise overview of the A/D process is also given.

Appendix B covers the software user manuals including a brief overview of the LabVIEW (Laboratory Virtual Instrument Engineering Workbench) programming environment and the software installation procedure. The software manuals include; the DAQ process control application, the data plot application and the post-processing application. The overview of LabVIEW covers main components of the front panel and the block diagram. The installation of the LabVIEW Run-time engine and the three data management software applications are covered in the installation procedure section.

Appendix C is dedicated to the fuzzy logic nearest neighbourhood clustering (NNC) Matlab graphical user interface (GUI) and the code. It covers the three main applications within the GUI namely; *FuzzyTrain*, *FuzzyRead* and *FuzzyValidate*. A program flow diagram, for each of the applications, is covered in which a number of modular functions are used such as: *NNCGetFile*, *NNCNormalise*, *NNCInit*, *NNCFLS* and *NNCDeNormalise*. The fuzzy Matlab GUI and its features are also covered as well as a quick reference guide.

The data CD contents are covered in *Appendix D*. Two main directories namely LabVIEW and Matlab and both its content are covered. The LabVIEW directory hosts the DAQ software, data, source code and extra library functions. The Matlab directory hosts a number of fuzzy control valve models, training and validation data and mass flow data. The various naming conventions, for the DAQ process, post-processing and fuzzy training, are also covered.

CHAPTER 2

Modelling Methods

Modelling of real-world systems, be that electronic components such as resistors, capacitors, inductors, integrated circuits or mechanical devices such as valves, thermocouples, pressure transducers or compressors, aim to construct a mathematical function relating variables in some way to serve as a model. Mathematical models are constructed to explain, predict, analyse and verify the dynamic and/or static behaviour of the real-world system [1].

Modelling is performed to obtain design equations, predict performance, investigate component tolerance, simulate plant behaviour, design and test concept control systems and identify and diagnose abnormal events [2].

Due to losses mounting to billions of dollars in the production industry in recent years, abnormal event management (AEM) is believed to be the number one problem that needs to be solved. The timely detection of abnormal events, diagnosing its causal origins and taking appropriate supervisory control decisions and actions to bring the process back to a normal, safe operating state is known as AEM. Although AEM does not fall within the scope of this dissertation, it forms an imperative part in modelling physical components. Adding this element into a mathematical model could produce an accurate dynamic model which predicts component failure and possible component dynamics when an abnormal event occurs.

Modern plants and production facilities use Supervisory Control And Data Acquisition (SCADA) systems to manage, control and log their behaviour over extended periods of time. The inputs to the plant are read from the SCADA database and applied to the model. The responses or outputs of the model are compared with that of the physical plant and used to infer the presence of faults. The complete reliance on human operators to cope with such abnormal events has become increasingly difficult due to the broad scope of the diagnostic activity that encompasses a variety of malfunctions such as process unit failures, process unit degradation and parameter drift. The size and complexity of modern process facilities, insufficient, incomplete and/or unreliable measurements make the diagnosis of faults even more difficult. It comes as no surprise that about three quarters of all industrial accidents are caused by human operators making erroneous decisions. These abnormal events have a significant economic and safety impact on the process industry [2], [3].

This chapter is dedicated to the modelling process, the different modelling methods used to derive and construct mathematical models and a comparison of the different modelling methods used within this dissertation.

2.1 THE MODELLING PROCESS

Constructing a mathematical model, aims to predict and analyse the effects various situations have on a real-world phenomenon. The model aids in understanding how a particular real-world system operates, what causes changes in the system and the sensitivity of the system to these changes.

The modelling process is presented in Figure 2.1 as a closed loop system. Given some real-world system, sufficient data is gathered to formulate a model. The model is formulated and then analysed, which leads to mathematical conclusions being drawn. The model with its conclusions is then interpreted and predictions are made as well as behavioural explanations are offered. Finally the conclusions are tested against new observations and data. This is an iterative process and it may be found that the model needs to be improved or even in some cases reformulated.

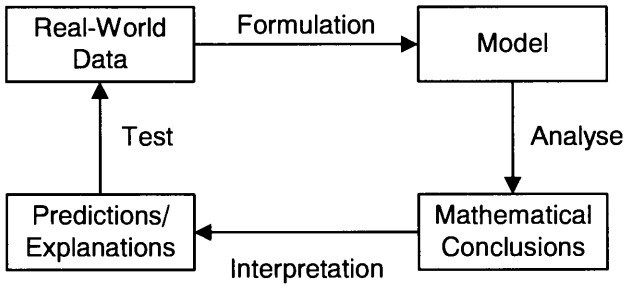


Figure 2.1 *The Modelling Process as a Closed System.*

Model, in its broad sense, has multiple meanings. A scaled duplicate of a production plant can be used to predict system behaviour under experimental conditions. An example of this is the Pebble Bed Micro Model (PBMM) situated at the Potchefstroom campus of the North-West University (South Africa). Another kind of model is a mathematical model where a particular real-world system or phenomenon is studied. This chapter is devoted to developing a mathematical model of a control valve which forms part of the PBMM. The model is derived within the system so as to predict in-system behaviour.

Mathematical models can be differentiated further. Existing mathematical models can be identified with a particular real-world phenomenon and used to study it, or new mathematical models may be constructed specifically to study a special phenomenon. Figure 2.2 depicts this

differentiation between models. The phenomenon of interest can be represented by a mathematical model by either constructing a new model or selecting an existing model. Another means of investigating the phenomenon is to construct a scaled replica on which experimental tests can be conducted or a simulation be validated.

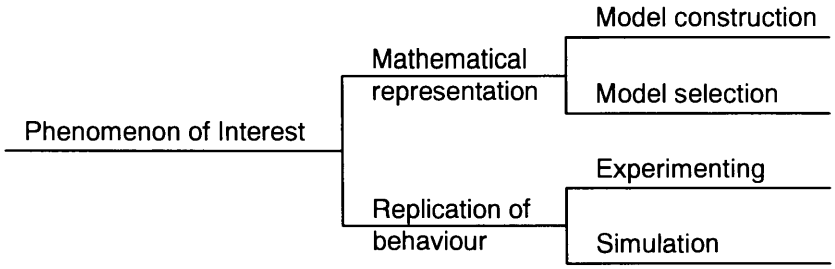


Figure 2.2. *Differentiation between Models*.

Real-world systems are often complex due to the fact that it is described by a number of partial differential equations or a system of nonlinear algebraic equations. The mathematics involved may be so complex that there is little hope of analysing or solving the model.

In this case it is possible to replicate the behaviour *directly* by conducting a number of experiments and collecting data which can possibly be analysed with statistical techniques or curve-fitting procedures. In other cases the behaviour can be replicated *indirectly*, by using a scaled-down model on which experimental procedures are then conducted. Often the behaviour is replicated by simulating the process on a computer.

2.1.1 Construction of Models

The modelling process has been covered and different techniques have been introduced when constructing mathematical models. The focus now shifts to an outline procedure which can be followed when constructing these models. The procedure is known as the mathematical modelling process and is as follows [1].

- *Identify the problem*: This is typically the most difficult part when modelling a real-world system. Usually large amounts of data need to be analysed and processed to identify a particular aspect
- *Make assumptions*: It is almost impossible to capture all the factors influencing the problem that has been identified. Reducing the number of factors under consideration simplifies the task. Modelling the problem involves finding relations between the remaining variables. Reducing the variables implies making assumptions, which fall into two main activities:

- *Classify the variables*: Variables can be classified into two main groups, the first being dependant variables and the second independent variables. The dependant variables are explained by the model whereas the independent variables are not. Each variable is classified as dependent, independent or neither. Independent variables are eliminated first and may possibly be incorporated later in the refined model
- *Determine interrelationships among the variables*: This implies studying variables independently and creating sub-models which can later be incorporated into the master model

- *Solve or Interpret the model*: Add all the sub-models together and evaluate the results predicted by the master model. In some cases the model consists of mathematical equations or inequalities that must be solved to find a solution for the model
- *Verify the Model*: Several questions should be asked when verifying the model:
  - Does the model address the problem identified?
  - Is the model usable in a practical sense?
  - Does the model make common sense? Once all these questions have been answered, results from the model may be compared with the real-world system provided the data is within the same range as that of the real-world system
- *Interpret the model*: Once the model has been verified it must be implemented into a system such as a simulator or prediction algorithm. Commissioning the model within a user friendly environment creates a simple (effortless to understand) model
- *Maintaining the model*: If at some stage the specifications of the model have changed after its implementation, some form of maintenance needs to be done on the model, possibly changing one or more features and/or variables. Once the changes have been made a new revised version of the model is released

As with any model, the outlined procedure is an approximation process and therefore has its limitations. The procedure seems to consist of discrete steps always leading to a usable model. This is rarely the case in practice and thus a disadvantage of the outlined procedure. A possible improvement to this method is to make use of an interactive modelling procedure [1].

The improved modelling procedure is known as a scientific method and is as follows:

- Make general observations of a phenomenon
- Formulate a hypothesis about the phenomenon
- Develop a method to test the hypothesis
- Gather data to use in the test

- Test the hypothesis using the data
- Confirm or deny the hypothesis

The mathematical modelling process and scientific method have similarities which include making assumptions or hypotheses, gathering real-world data and testing or verification using the data [1]. Modelling in general is an art with one fundamental rule; aim to be scientific and objective whenever possible.

2.1.2 Modelling Paradigms

Many physical systems are linear within some range of the variables but ultimately become nonlinear as the variables are increased without limit. A system is defined as being linear in terms of its excitation and response as well as the fact that its magnitude scale factor should be preserved. The behaviour of many mechanical and electrical elements can be assumed linear over a reasonably large range of the variables. This unfortunately is seldom the case with thermal and fluid elements which are more frequently nonlinear in character [4].

Deriving a mathematical model for a control valve, which relies on both thermal and fluid elements for proper operation, inevitably leads to the derivation of a *nonlinear* mathematical model. *Time delayed* inputs and outputs together with *feedback* are used to incorporate dynamics into the system. This implies that a *time varying*, *nonlinear*, *grey box* model (depicted by the bold line in Figure 2.3) is derived.

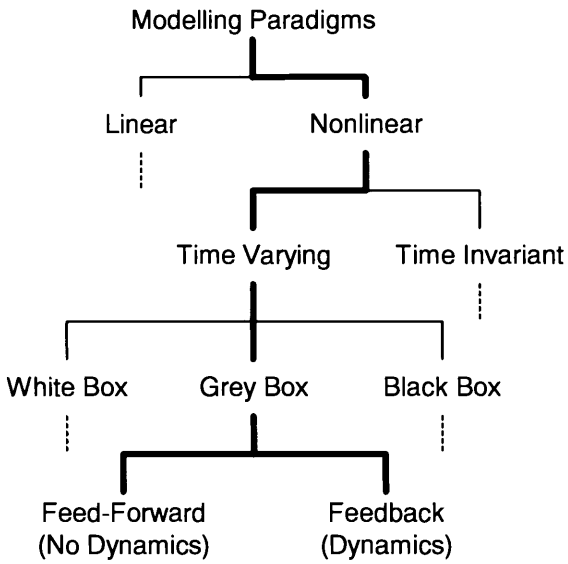


Figure 2.3. *Modelling Paradigms*.

2.1.2.1 Shades of Grey

Modelling paradigms aim at providing a degree of knowledge about the device being modelled. Given a mechanical device such as a control valve, compressor or turbine, developing a white box model for this device requires intricate knowledge about the device, its fundamental construction and operating limits. Deriving the white box model makes use of basic principles such as mechanical governing equations and fundamental physics equations and theories. Research has to be done to determine all the parameters necessary for successful modelling. Deriving a mathematical model of an element, by using the white box modelling method, can be a costly process. The modeller in this case should have extensive knowledge about the device and the engineering discipline and should therefore be adequately justified [2]. These model types are not very adaptable and any changes made to the device, be that mechanical or electrical, requires an entire revision of the modelling procedure.

Grey box models require less device specific knowledge, although the general structure or behaviour of the device, gathered from physical principles, may prove valuable. Fuzzy Logic (FL) is an example of a grey box modelling method. FL not only makes use of experimental data but it is also possible to add so called expert knowledge to the model. This enables the model to include both numerical information and human expert knowledge (linguistic information). Expert knowledge is usually not precise and is represented by terms like *cold*, *lukewarm*, *warm* and *hot*. Numerical and expert linguistic information have many fundamental differences. Numerical information obeys physical laws and mathematical axioms whereas with expert linguistic information no such laws and axioms exist. Two worlds exist within a man-machine system – the physical world and the human world. The physical world implies the *machine* part and the human world implies the *man* part. Analyzing these mixed world systems requires a framework which encapsulates both worlds. FL provides such a framework [5].

Black box models, as apposed to white box models, require no knowledge of the device or plant. Neural Networks fall into this category. Although it is accepted that no device or plant knowledge is required for a black box model, it is not entirely true. Some knowledge of the device or plant being modelled can significantly aid in selecting better topologies, delay factors and other modelling parameters that can result in a model which more closely resembles the behaviour of the real-world device or plant. Black box models also require experimental data for training, verifying and validating the model [2].

2.2 FUNDAMENTAL MODELLING METHODS

The term *fundamental* refers to modelling methods that have been used since the earliest days of mathematics. This not necessarily implies that these methods are invalid in this day

and age, but rather implies the *basic*, *primary* or *elementary* modelling methodologies used by mankind to explain real-world phenomenon in the mathematical domain. These modelling methods, due to its successes in the past, are often still being used today.

Fundamental modelling methods include first or basic principle models, model fitting and experimental modelling. The mentioned modelling methods have the ability to include static and dynamic aspects of the phenomenon or element under investigation, although, as stipulated, it requires intricate knowledge about the phenomenon or device as well as extensive knowledge in the specific engineering discipline.

First or basic principle models are usually associated with the so called white box models (2.1.2.1). These modelling methods make use of basic principles in the specific engineering discipline as well as fundamental physics. Model fitting, or curve fitting, and experimental modelling are closely related and tend to lean more toward the black box modelling paradigm due to experimental data being used to derive and/or fit a mathematical model. Even though they are closely related some differences do exist.

2.2.1 First Principle Modelling

This dissertation focuses on the development of a dynamic mathematical model for a control valve. The model is characterised by the valve relative opening, the differential pressure across the valve, its upstream temperature and the resultant mass rate of flow through the valve. This section is devoted to the first principle models, derived from fundamental mechanical and physics theories and laws, which calculate the mass rate of flow through the control valve given certain parameters.

The *flow rate calculations for compressible flow in pipes*, standardised by both the American Society of Mechanical Engineers (ASME) and the International Organisation for Standardisation (ISO), British standard, specify that flow rate measurement is based on the installation of a primary device such as an orifice plate into a pipeline in which fluid is flowing. The installed device causes a static pressure difference between the upstream side and the downstream side of the orifice. Knowing the geometry of the pipe and orifice, and the height (h) of the manometer[1], a theoretical value for mass flow ($\dot{m}$) can be calculated [6], [7]. The orifice plate inserted into the pipe can be seen in Figure 2.4.

[1] The simplest type of manometer is the U shaped tube. The tube is usually filled with mercury due to its high specific weight. The one end of the tube is connected to the tank, for which the pressure needs to be calculated, and the other end is left open (atmosphere). The fluids attain an equilibrium configuration from which it is relatively simple to deduce the tank pressure with reference to atmosphere [7]. This is called absolute pressure.

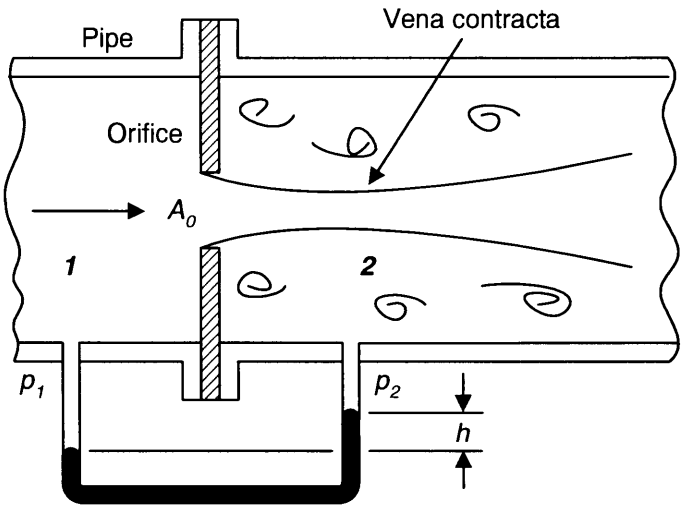


Figure 2.4. *Flow Rate Calculation by means of an Orifice Plate.*

Consider the first law of thermodynamics between points *1* and *2* within the figure. Neglecting potential energy of gravity, the steady, subsonic (not in choke), isentropic flow of a perfect gas, is given by:

$$c_p T_1 + \frac{V_1^2}{2} = c_p T_2 + \frac{V_2^2}{2} \tag{2.1}$$

with the continuity equation for the control volume given by:

$$\rho V_1 A_1 = \rho V_2 A_2 \tag{2.2}$$

V_1 in (2.2) is solved by replacing p_2/p_1 by $(p_2/p_1)^{1/k}$ for the isentropic pressure change of a prefect gas. From this the mass flow is:

$$\dot{m} = \rho_1 C_d A_2 \sqrt{\frac{2[(p_1 - p_2)/\rho_1]}{1-(A_2/A_1)^2}}\, Y \tag{2.3}$$

where the compressibility factor Y is:

$$Y = \sqrt{\frac{[k/(k-1)](p_2/p_1)^{2/k}\left[1-(p_2/p_1)^{(k-1)/k}\right]\left[1-(A_2/A_1)^2\right]}{\left[1-(A_2/A_1)^2(p_2/p_1)^{2/k}\right]\left[1-(p_2/p_1)\right]}} \tag{2.4}$$

The upstream and downstream temperatures are denoted by T_1 and T_2 respectively. The fluid upstream velocity is denoted by V_1 and its velocity at the vena contracta is V_2. The density of the fluid is ρ and the compressibility factor is Y. The cross sectional area of the orifice is denoted by A_1 and the area of the fluid at the vena contracta is A_2. The upstream and downstream pressures are denoted by p_1 and p_2 respectively. $\dot{m}$ is the mass flow through the pipe and C_d is the coefficient of discharge. The ratio of specific heats c_p and c_v is a useful dimensionless parameter given by $k = c_p/c_v$, where k for dry air is 1.4 [7].

First principle modelling is therefore not trivial even though the modeller has experience in the mechanical engineering discipline. If some of the parameters are to be added or removed from the model(s), the entire modelling procedure would have to be revised.

2.2.2 Model Fitting and Experimental Modelling

From section 2.2.1 it is clear that the first principle modelling methodology is an exact process not leaving much room for error or assumptions. Attempting to model a device without the necessary background knowledge often leaves most modellers discarding the process or trying to find an alternative method to accomplish the task. In this case the problem is so complex, having so many significant variables, that it prevents the formulation of a model explaining the situation. Experiments may be conducted to investigate the behaviour of the dependent variables within the range of the data points [1].

Three possible steps exist when analysing a collection of data points:

- Fitting a selected model type or types to the data
- Choosing the most appropriate model from the competing types that have been fitted
- Making predictions from the collection of data

In the first two steps, models exist that seem to explain the behaviour being observed. This is known as *model fitting*. In the third step however, a model does not exist to explain the behaviour but rather a collection of data points which can be used to predict the behaviour within the range of the data points. This is usually called an empirical model which *interpolates* between the collection of data points.

In the first step, the best model must be identified to solve the problem. However in the second step, a criterion is needed to compare models of different types and in step three a criterion must also be established to make predictions in between the observed data points. In the first two steps the modeller is willing to accept some deviation between the model and the collected data points, thereby creating a model that satisfactorily *explains* the behaviour under investigation. On the other hand, when interpolating, the modeller is guided by the carefully collected data points and a curve is sought that captures the trend of the data to *predict* in between the data points [1].

2.2.2.1 Modelling Pressure Drop Ratio Factor (X_T)

When a pressure difference is applied across a valve, fluid flows from high pressure to low pressure. If this pressure difference is further increased, beyond sonic velocity at the vena contracta, the vena contracta moves upstream towards the valve orifice and its cross sectional

area becomes larger resulting in an increase in fluid flow. When the vena contracta reaches the valve orifice the fluid flow becomes fully choked. A further increase in pressure difference across the control valve will in this case not result in an increase in fluid flow. This is known as the terminal pressure drop and is calculated by using the pressure drop ratio factor (X_T) [8].

From [9] the pressure drop ratio factor, within the coefficient tables, is a function of the control valve flow coefficient (C_v) which was introduced in 1944. This dimensionless parameter has become accepted as the universal gauge of valve capacity and is employed in numerous discussions of valve design, characteristics and flow behaviour.

By definition, the valve flow coefficient is: "the number of gallons per minute of water which will pass through a given flow restriction with a pressure drop of 1 psi". This capacity index can be used by the engineer to rapidly and accurately estimate the required size of a restriction in any fluid system [10].

The data can be seen in Table 2.1 and only applies to a Neles segmented ball valve with a nominal diameter of 65 mm (2.5").

Table 2.1. $X_T = f(C_v/d^2)$.

C_v/d^2	X_T
0.59	0.64
1.45	0.64
2.86	0.63
4.93	0.62
7.64	0.59
11.00	0.54
15.07	0.49
20.67	0.40
30.80	0.32
48.90	0.16

Making use of model fitting, neither the *straight line* fit nor the *power curve* fit results in a model sufficiently describing the pressure drop ratio factor. The model for a straight line fit is $y = ax + b$ and for a power curve is $y = ax^n$. Fitting these models implies selecting the best values for a, b and n. Both attempts can be seen in Figure 2.5 and Figure 2.6 respectively.

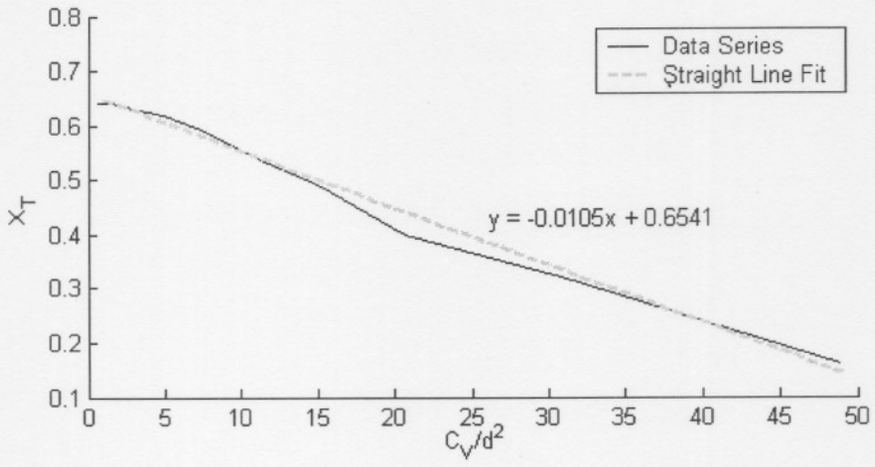


Figure 2.5. *Straight Line Fit*.

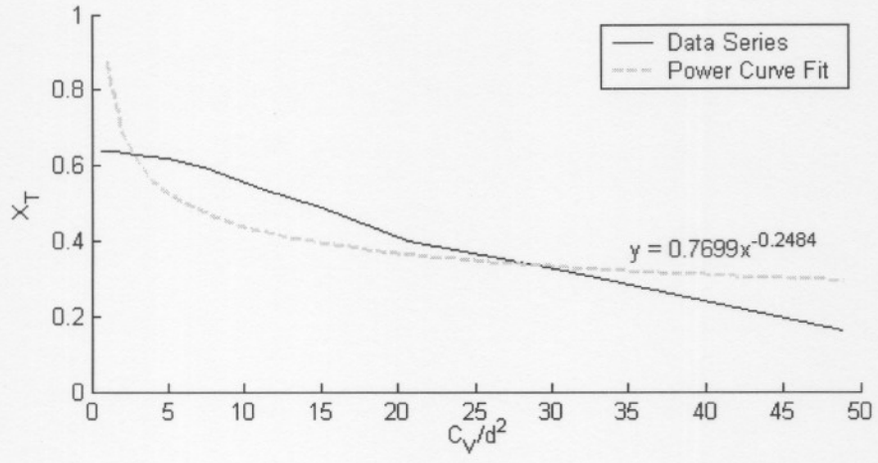


Figure 2.6. *Power Curve Fit*.

2.2.2.2 Experimental Modelling

When fitting a curve to previously collected data a particular model is selected for which parameters are calculated according to some criterion such as the least-squares criteria. Using this method the modeller expects some deviations between the fitted model and the collected data. The problem with this approach is that in many cases it is impossible to create a model that satisfactorily explains the system behaviour due to uncertainties as to what curve actually describes the behaviour.

By using the collected data samples an empirical model can be calculated. The data samples strongly influence the creation of such a model where a curve capturing the trend of the data is used to predict in between the data samples. This improvement is known as interpolation [1]. This section investigates multiterm models known as the polynomial.

Polynomial Models

Different methods for determining polynomial coefficients exist which include Lagrange interpolation methods and divided differences. These will however not be discussed as they are found throughout a number of modelling and calculus text books. These methods make use of theorems and algorithms to calculate the polynomial coefficients.

A fourth-order polynomial function (grey dotted line) used to estimate and interpolate between the data samples in Table 2.1 can be seen in Figure 2.7.

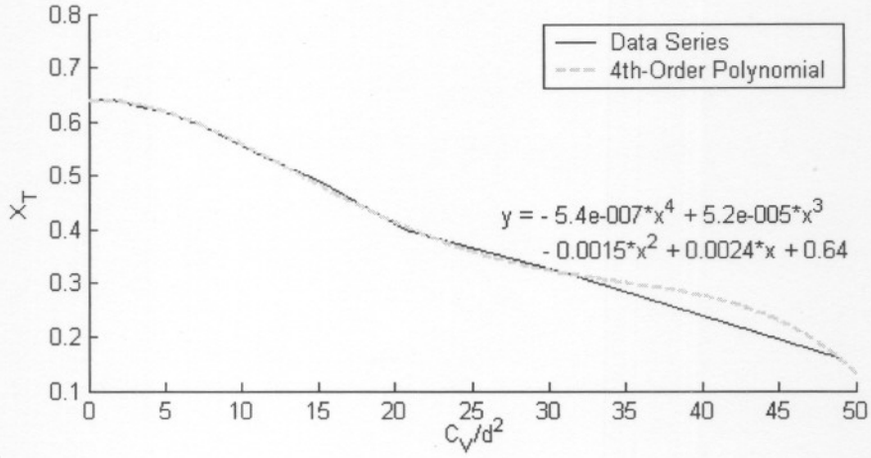


Figure 2.7. *4th-Order Polynomial Function*.

One of the drawbacks of high order polynomial functions is that it tends to oscillate near the endpoints of the data samples. This can be seen in Figure 2.7 where the oscillation causes a relatively large error near the end of the data samples. Fitting a higher order polynomial function, such as a 6th-order, will result in an even greater oscillation near the endpoints. A possible solution to this problem is to make use of a lower order function or to make use of a different interpolation method.

Often applications require a proportionality adjustment between two values read from a table of data samples. In order to approximate a value between two consecutive data samples it is often assumed that they have a linear variation from the first data sample to the second. This is known as *linear interpolation* and yields reasonable results for most applications especially when the data samples are closely spaced [1]. Partial data samples and their linear splines can be seen in Figure 2.8.

In this case (Figure 2.8), two linear spline formulae are derived between the consecutive intervals $x_1 \leq x \leq x_2$ [2.86, 4.93) and $x_2 \leq x \leq x_3$ [4.93, 7.64]. Solving for a_1, b_1, a_2 and b_2 yields results in the form of $S_1(x)$ and $S_2(x)$ respectively.

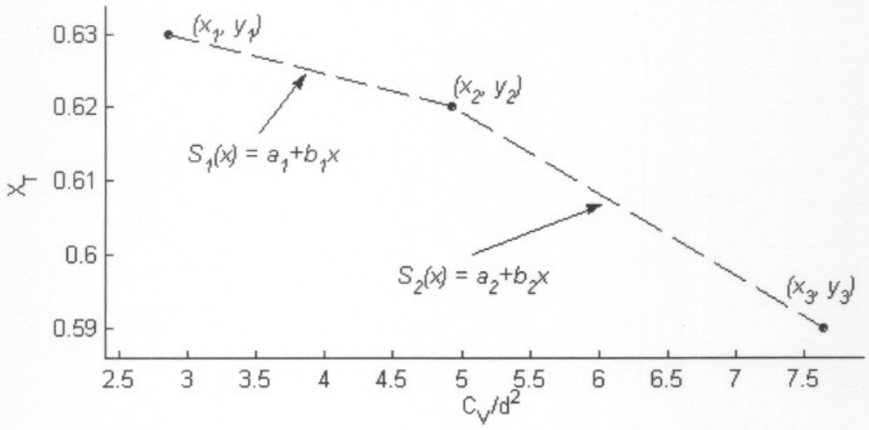


Figure 2.8. *Linear Splines*.

Even though the linear spline method is sufficient for most applications, it fails to capture the trend of the data due to unsmooth connections between the data samples. In Figure 2.8 a discontinuity at x = 4.93 is evident. A popular modern technique known as cubic splines interpolation, which uses different cubic polynomials between successive data samples to capture the trend of the data, incorporates smoothness into the empirical model. This is done by requiring that both the first and second derivative of consecutive splines agree at each data sample [1].

A typical cubic spline between two consecutive data samples are as follows:

$$\begin{aligned} S_i(x) &= a_i + b_i x + c_i x^2 + d_i x^3 \\ S_i'(x) &= b_i + 2c_i x + 3d_i x^2 \\ S_i''(x) &= 2c_i + 6d_i x \end{aligned} \tag{2.5}$$

where the first and second order derivative offers the possibility of matching up the slopes and curvatures of each interior data sample respectively. Due to these requirements, the first and second derivatives of both adjacent splines must be equal, therefore:

$$\begin{aligned} S_i'(x_{int}) &= S_{i+1}'(x_{int}) \\ S_i''(x_{int}) &= S_{i+1}''(x_{int}) \end{aligned} \tag{2.6}$$

where x_{int} is the interior data sample between the two consecutive data samples.

The endpoints of the data samples are defined by the second derivative which must be zero because the first derivative is a constant and therefore defined by:

$$\begin{aligned} S_1''(x_1) &= 0 \\ S_n''(x_n) &= 0 \end{aligned} \tag{2.7}$$

where x_1 defines the first data sample and x_n the last data sample.

Using (2.5), (2.6) and (2.7), obtain solutions for a_i, b_i, c_i and d_i. The results are valid within their respective intervals and the cubic splines solution for Table 2.1 can be seen in Figure 2.9.

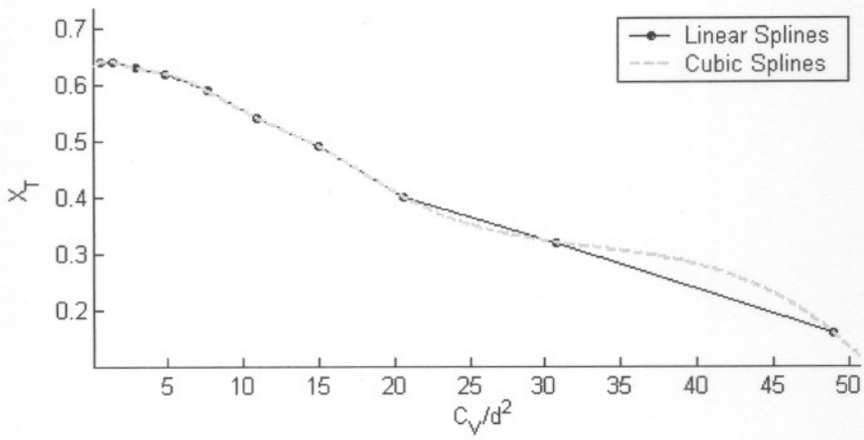


Figure 2.9. *Cubic Spline Interpolation for all Data Samples*.

By using cubic splines to interpolate between data samples, oscillations near the endpoints are eliminated. Due to the nature of the data samples and the fact that smoothness is a criterion in this model, the cubic spline model also seems to oscillate near the endpoints. This is not due to the high order of the fitted model as in the case of the polynomial function.

The advantages of cubic splines over polynomial functions are:

- Oscillation near the endpoints are eliminated
- Each cubic spline function passes through two data samples
- Cubic spline functions never exceed an order of three

2.3 DYNAMIC MODELLING METHODS

Modelling and identification are important steps in the design of a control system since most advanced approaches are based on a model of the process. Industrial systems exhibit many features that make the modelling tasks increasingly difficult. Once the entire operating range is considered, most devices exhibit nonlinear behaviour which cannot be described using conventional linear methods. The behaviour of an entire system is often determined more by the gross qualitative interaction of its components rather than by its complex quantitative behaviour. Therefore a mathematical formalism is needed that facilitates coherent integration of qualitative and quantitative information, symbolic and numeric data and computation with reasoning [11].

Modelling techniques based on fuzzy sets attempts to combine numerical and symbolic processing into one framework. On the one hand, fuzzy systems are knowledge-based systems

consisting of expert linguistic *If-Then* rules, and on the other hand fuzzy systems are also universal approximators that can realise nonlinear mappings. This duality allows qualitative knowledge as well as quantitative data to be combined in a complementary way. Compared to other nonlinear approximation techniques, fuzzy systems provide a more transparent representation of the nonlinear system in the sense that rules can be examined, analysed and validated by experts [11].

This dissertation focuses on creating a fuzzy logic model, based on measured data. The data is acquired from an installed control valve in a plant. The data acquisition process is described in detail in *Chapter 3*.

2.3.1 Fuzzy Logic

Despite the various successes of fuzzy logic, a certain degree of misunderstanding still exists in its use, how it compares with other systems and its strengths and limitations. In its narrow sense, fuzzy logic is a *logic* of approximate reasoning. In its broader sense it is related to the theory of fuzzy sets, which is classes of objects in which the transition from membership to nonmembership is gradual rather than abrupt. Fuzzy logic has many branches ranging from fuzzy arithmetic to fuzzy expert systems implying that any input space X can be fuzzified and be called *fuzzy X*.

All available information within an engineering application pertaining to the particular system should be used effectively. This implies that human expert information should also be used. Usually this human information is not precise and is represented by terms such as *low*, *medium*, *high* and so forth. In order to incorporate the information, so-called intelligent approaches have been emerging in the engineering community. Man-machine interaction has increased during the past few years and more engineering applications belong to a mixture of these two different worlds. Adaptive fuzzy systems provide a common framework for this environment.

An adaptive fuzzy system is a fuzzy system equipped with a training algorithm. This fuzzy logic system is constructed from a number of collections of *If-Then* rules and the training algorithm adjusts the parameters of the fuzzy logic system (FLS) based on numerical input-output pairs. Linguistic and numerical information are incorporated in the following way. Linguistic information can be incorporated directly into the FLS due to it being the main building block of a FLS. Numerical information, on the other hand, is incorporated by training the FLS to match input-output pairs [5], [12].

2.3.2 Fuzzy Logic Foundation

Fuzzy logic involves a number of concepts that need to be clarified. These include the following:

- Fuzzy sets
- Membership functions
- Logical operators
- If-Then rules

2.3.2.1 Fuzzy Sets

The starting point of fuzzy logic is the concept of a fuzzy set. A fuzzy set is defined as a set without a clearly defined boundary. This means it can contain elements with only a partial degree of membership. A classical set is a set that either wholly includes or wholly excludes any given element. Figure 2.10 can be seen as a classical set, where Mars, Venus, Jupiter, and Saturn are part of the *Solar System Planets* set [13], [14], [15] and Key, House, Apple and Computer are not part of this set. This is called a classical set because it has been around for a long time [12].

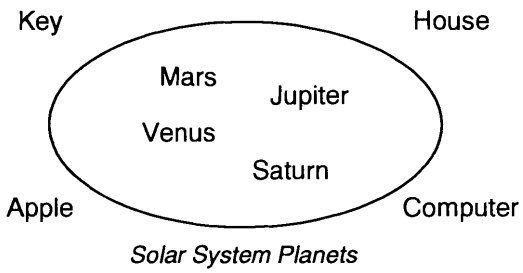


Figure 2.10. *Planets of our Solar System.*

In fuzzy logic, the truth of any statement becomes a matter of degree [12]. Reasoning in fuzzy logic implies making use of the undefined area between the Boolean True and False. If numerical values were to be linked to the Boolean operators, where true is equal to one and false equal to zero, then values such as 0.3 and 0.823 are also allowed. This is known as multivalued logic (Figure 2.11 *(b)*) as apposed to two-valued logic (Figure 2.11 *(a)*). In multivalued logic a statement such as "Is *X* a member of set *A*?" has many answers including *true* (yes), *false* (no) or any of the thousand intermediate values in between.

In Figure 2.11 *(a)*, notice that if any object does not fall within the classified nine planets then it is not even remotely considered a planet. The planet-ness truth value jumps discontinuously from 0 to 1 the moment any of the nine planets are considered.

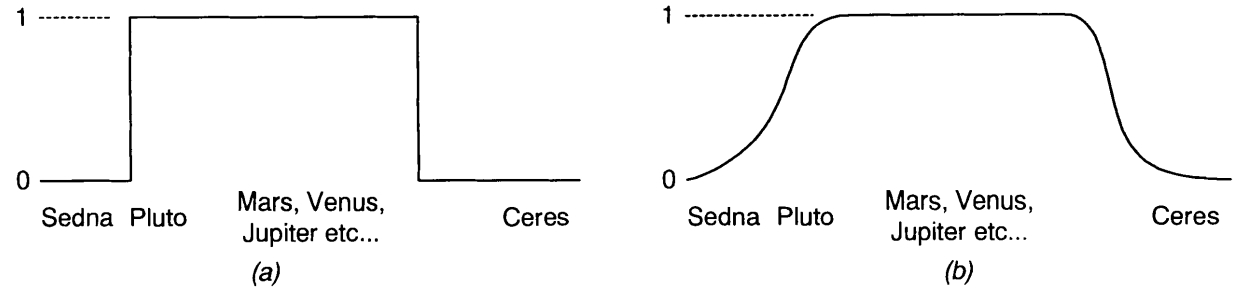


Figure 2.11. *Two-Valued Logic and Multivalued Logic.*

Using a smooth varying curve for planet-ness, gives a more human feeling as to what a planet is. From Figure 2.11 *(b)* Pluto is considered more of a planet than Sedna whereas Ceres has very little membership to planet-ness. The curve that defines planet-ness maps the input space to the output space and is known as a membership function.

2.3.2.2 Membership Functions

A membership function (MF) is a curve that defines how each point in the input space (universe of discourse) is mapped to the membership value (or degree of membership) between 0 and 1 [12].

Making use of the *historical plus* definition [14] to determine when an object is a planet or not can help clarify how a membership function is used. The universe of discourse is all possible object equatorial diameters e.g. from 0 to approximately twice the size of Jupiter (± 2 × 142870 km). Part of the definition states that any object larger than Pluto is a planet. This implies that if a new object were to be discovered and its equatorial diameter is e.g. 1 mm larger than that of Pluto it would be the tenth planet. If however its diameter is smaller than that of Pluto's by 1 mm it would not be considered a planet. This sharp-edged transition from non-membership to membership can be seen in Figure 2.12 *(a)*. A more human approach to this can be seen in Figure 2.12 *(b)*, where the transition from *not a planet* to a *planet* is represented by a smooth curve. It implies that both objects are planets to some degree, but one is significantly less of a planet than the other.

Each application can have its own unique MF curves and is not bounded to conform to specific MF curves. Triangular MFs exist as well as trapezoidal MFs but the most commonly used MF is the Gaussian MF.

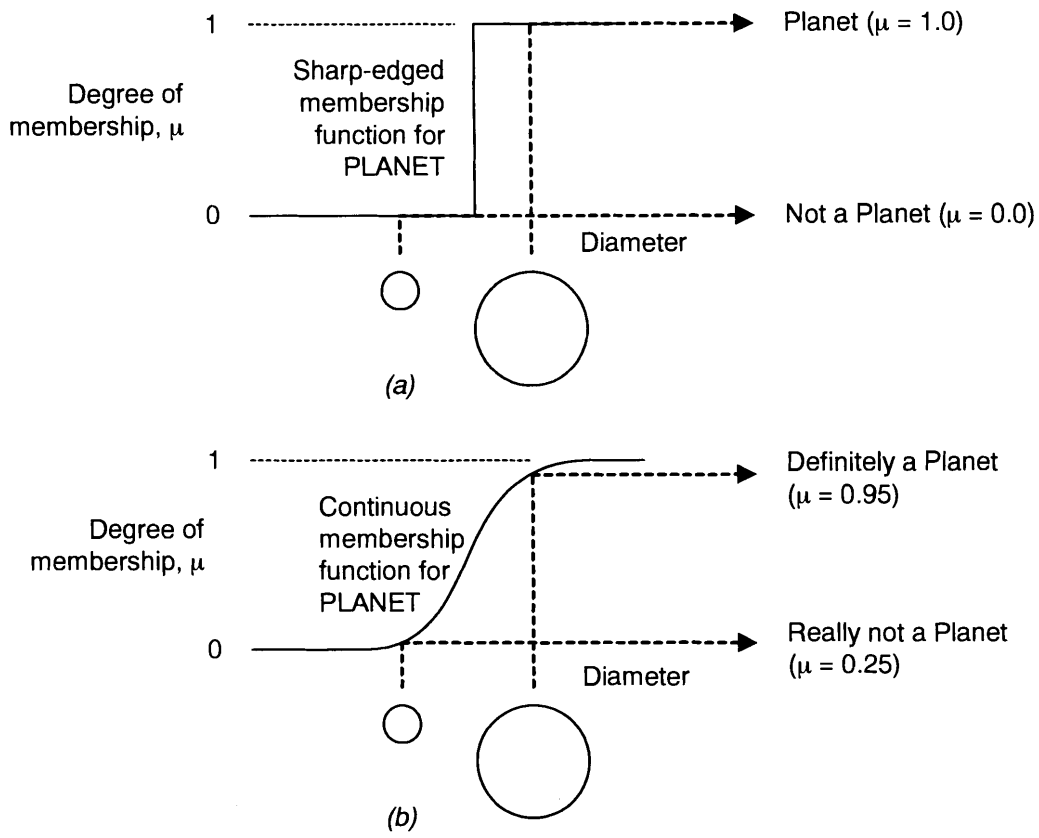


Figure 2.12. *Planet-ness According to Equatorial Diameter.*

2.3.2.3 Logical Operators

Fuzzy logic is a subset of standard Boolean logic which, as stated, implies that all possible values between the logic true and false can also be used as opposed to only the standard true and false values. The standard Boolean operations can be seen in Table 2.2.

Table 2.2. *Standard Boolean Operators.*

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

OR

A	not A
0	1
1	0

NOT

To make use of all the values between the extremes and still complying with the standard Boolean operations, the *AND* operation is replaced with a *min(A, B)*, the *OR* operation replaced with a *max(A, B)* and the *NOT* operations replaced with a *1-A*. This results in multivalued logic operations and are more generally known as fuzzy intersection or conjunction (*AND*), fuzzy union or disjunction (*OR*) and fuzzy compliment (*NOT*). Additional fuzzy operators for

intersections and unions exist but will not be covered in this dissertation. These include T-norm for a fuzzy intersection and T-conorm (or S-norm) for a fuzzy union.

2.3.2.4 If-Then Rules

Linguistic rules are the central part of a fuzzy logic system. The rules are more generally in the form: *IF x* is *A THEN y* is *B* where *A* and *B* are the linguistic values defined by fuzzy sets on the ranges *x* and *y* respectively. The *IF*-part (*x* is *A*) is known as the antecedent or premise and the *THEN*-part (*y* is *B*) is known as the consequent or conclusion.

When analysing the rule; *IF equatorial radius is large THEN object is a planet*, note that *large* (other values can be *small* or *medium*) represents a number between 0 and 1 and so the antecedent returns a single number between 0 and 1. *Planet* (other sets can be *Dust particle*, *Asteroid*, *Embryo Star*, *Supernova*, *Embryo Solar* or *Black Dwarf*) represents a fuzzy set and so the consequent is an assignment that assigns the entire fuzzy *B* to the output variable *y*.

Interpreting a fuzzy *IF-THEN* rule involves firstly evaluating the antecedent (fuzzyfying the input and applying the necessary fuzzy operators) and secondly applying that result to the consequent, known as implication.

The antecedent of a rule can have multiple parts as in; *IF equatorial radius is large AND object is round AND object orbits the sun THEN object is a planet*. As with the antecedent, the consequent can also have multiple parts as in; *IF surface temperature is cold THEN object is far from the sun AND object has no atmosphere*.

2.3.3 Fuzzy Inference System

Fuzzy logic involves the mapping of an input space to an output space. The process of formulating this mapping is known as fuzzy inference and provides a basis from which decisions can be made. The fuzzy inference process uses all the aspects briefly covered in section 2.3.2.

Fuzzy inference systems have been successfully applied in fields such as automatic control, data classification, decision analysis, expert systems and computer vision [11].

2.3.4 Fuzzy Logic Classification

Fuzzy logic systems can be classified into three basic types; pure fuzzy logic systems, Takagi and Sugeno's fuzzy system and fuzzy logic systems with fuzzifier and defuzzifier [5].

Pure fuzzy logic system consists of a collection of fuzzy *IF-THEN* rules which are used by the fuzzy inference engine to determine a mapping from the input fuzzy sets (input universe of discourse) to the output fuzzy sets (output universe of discourse) (refer to Figure 2.13).

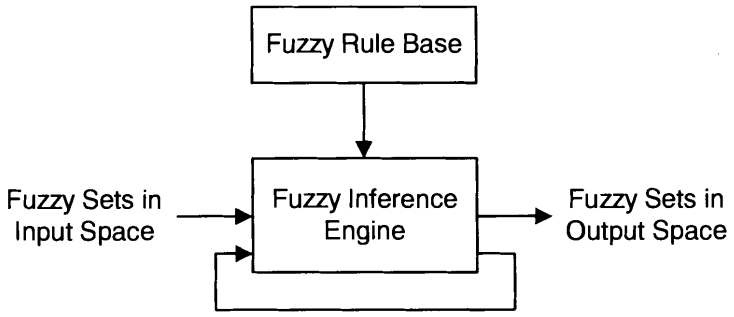


Figure 2.13. *Pure Fuzzy Logic System.*

Takagi Sugeno's fuzzy system as opposed to a pure fuzzy logic system considers rules with a fuzzy *IF*-part and with a crisp *THEN*-part. The advantage with this system is that it provides a compact equation in which it is simple to estimate parameters. A disadvantage of this system is that the *THEN*-part is not fuzzy and therefore does not provide a natural framework to incorporate rules from human experts.

Engineering applications make use of real-valued variables. In order to use the pure fuzzy logic system in Figure 2.13, a fuzzifier and defuzzifier are added to the input and output respectively as shown in Figure 2.14. The fuzzifier maps crisp points in the input space to fuzzy input sets and the defuzzifier maps fuzzy output sets to crisp points in the output space. The fuzzy rule base and fuzzy inference system are the same as in the pure fuzzy logic system.

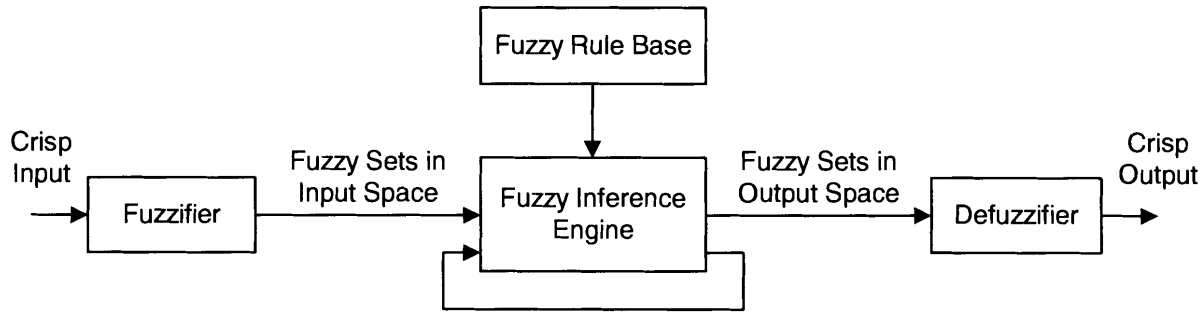


Figure 2.14. *Fuzzy Logic System with Fuzzifier and Defuzzifier.*

This fuzzy system has many advantages:

- Because of its real-valued inputs and outputs it is suitable for many engineering applications
- It provides a natural framework to incorporate *IF-THEN* expert rules

- There is much freedom in the choice of fuzzifier, fuzzy inference engine and defuzzifier, allowing an optimal selection for a suitable fuzzy logic system
- Different training algorithms can be developed for this fuzzy logic system

2.3.5 Nonlinear Dynamic System Identification

It has been proven extensively that neural networks can be used as identifiers for nonlinear components in dynamic systems. The neural network back-propagation algorithm makes it possible to train the neural network identifiers to match unknown nonlinear mappings. These are called neural identifiers [5].

Fuzzy logic also has a back-propagation algorithm which can be used in a similar manner. Although back-propagation has been proven to map nonlinear dynamic systems, other algorithms also exist with similar features such as nearest neighbourhood clustering. These are also called fuzzy identifiers. Two advantages of fuzzy identifiers over neural identifiers are:

- The parameters of the fuzzy identifiers have clear physical meanings whereas the weight matrices of a neural identifier can not be interpreted easily
- Human expert knowledge (linguistic information) can be incorporated into fuzzy identifiers due to its natural framework which is not possible with neural networks

The identification system constitutes three sections namely *(a)*, *(b)* and *(c)* which can be seen in Figure 2.15. Section *(a)* is used to acquire real-world data from the control valve. This is covered in more detail in *Chapter 3*. The acquired data is then used in section *(b)* to derive a fuzzy logic model. The model, derived in section *(b)*, is used in section *(c)* to predict an output parameter (massflow, Reynolds no., Expansibility or Valve choke status).

Section *(b)* uses direct as well as time delayed system inputs and time delayed system outputs. With these fuzzy system inputs an output is calculated. The training error is then equal to the difference between the *fuzzy system output* and the *desired output*.

Once training has been completed, the data are verified with a verification procedure. This is very similar to the validation process which in this case only uses data which have not been encountered before by the fuzzy logic system. Both procedures make use of direct as well as time delayed system inputs but differ from the training process in that time delayed *FLS Model* outputs are fed back into the *FLS Model*. This eliminates the possibility that the *FLS Model* might learn to follow the system output instead of effectively mapping input-output pairs. The verification and validation error is equal to the difference between the *FLS Model output* and the *desired output*.

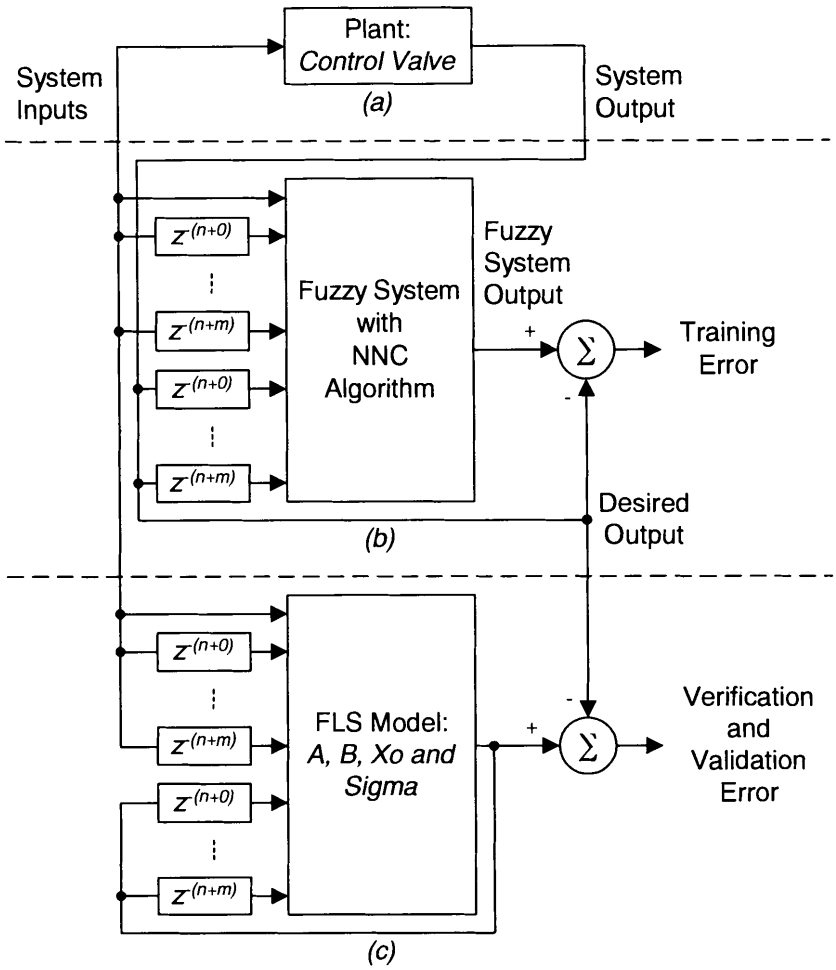


Figure 2.15. *Fuzzy Logic Identification System.*

The procedure to obtain the optimal fuzzy logic model parameters and settings are covered in *Chapter 4*. These optimal parameters and settings are obtained by using only a single set of data files and are thereafter applied to all data sets.

2.4 FUZZY LOGIC NEAREST NEIGHBOURHOOD CLUSTERING

Capturing a system's dynamic behaviour often requires expensive and lengthy experiments. A fuzzy logic system capable of matching all the input-output pairs to any given accuracy is required. By choosing the number of rules equal to the number of input-output pairs in the training set, this can be achieved. Each rule is therefore responsible for matching one input-output pair.

This becomes impractical when faced with a large-sample problem. Limiting the number of rules can however be achieved by first creating clusters of the input data. This is done by using the nearest neighbourhood clustering algorithm. The created clusters are then used by the optimal fuzzy logic system as input data where each data cluster is matched to an output

fuzzy set with one rule. This is called an adaptive fuzzy logic system due to its added training feature [5].

2.4.1 Optimal Fuzzy Logic System

Suppose a small number of input-output pairs $(\underline{x}^l, y^l)$ are given, $l = 1, 2, \ldots, N$ an optimum fuzzy logic system which can match all N input-output pairs to any given accuracy is denoted by:

$$f(\underline{x}) = \frac{\sum_{l=1}^{N} y^l \exp\left(-\frac{|\underline{x} - \underline{x}^l|^2}{\sigma^2}\right)}{\sum_{l=1}^{N} \exp\left(-\frac{|\underline{x} - \underline{x}^l|^2}{\sigma^2}\right)} \tag{2.8}$$

The σ denotes a smoothing parameter. A small σ results in a smaller matching error even though the fuzzy logic system $(f(\underline{x}))$ becomes less smooth. Once the fuzzy logic system is less smooth, it does not generalize well for input-output pairs outside of the training set. Therefore σ should be chosen so as to provide a balance between matching and generalization. As a general rule, a large σ can smooth out noisy data, while a small σ can make the fuzzy logic system as nonlinear as required to closely approximate the training data [5].

2.4.2 Adaptive Fuzzy Logic System

The optimal fuzzy logic system (2.8) uses one rule for each input-output pair in the training set. This is no longer practical when using large training sets. For these large-sample problems clustering techniques are used to group training pairs which are then represented by one rule. The algorithm is as follows:

- Establish a cluster centre $\underline{x}_0^1$ at the first input-output pair
- Assign $A^1(1) = y^1$, $B^1(1) = 1$ and select a radius r. A represents the sum of the outputs and B the number of input-output pairs in the current cluster
- Consider the k^{th} input-output pair $(\underline{x}^k, \underline{y}^k)$, $k = 2, 3, \ldots$ and suppose M cluster were created with centres at $\underline{x}_0^1, \underline{x}_0^2, \ldots, \underline{x}_0^M$. Compute the distances of the current fuzzy input data $\underline{x}^k$ to all M cluster centres, $|\underline{x}^k - \underline{x}_0^l|$, $l = 1,2, \ldots, M$, where $\underline{x}_0^{l^*}$ denotes the nearest cluster centre
  - If $|\underline{x}^k - \underline{x}_0^{l^*}| > r$, establish $\underline{x}^k$ as a new cluster centre: $\underline{x}_0^{M+1} = \underline{x}^k$. Assign $A^{M+1}(k) = y^k$ and $B^{M+1}(k) = 1$
  - If $|\underline{x}^k - \underline{x}_0^{l^*}| \le r$. Update $A^{l^*}(k) = A^{l^*}(k-1) + y^k$ and $B^{l^*}(k) = B^{l^*}(k-1) + 1$

- The adaptive fuzzy system at the k^{th} input-output data sample is computed as in (2.9) if $\underline{x}^k$ does not establish a new cluster. If however $\underline{x}^k$ does create a new cluster, increment M by 1.

$$f_k(x)\frac{\sum_{l=1}^{M} A^l(k)\exp\left(-\frac{\underline{x}-\underline{x}_0^l}{\sigma^2}\right)}{\sum_{l=1}^{M} B^l(k)\exp\left(-\frac{\underline{x}-\underline{x}_0^l}{\sigma^2}\right)} \tag{2.9}$$

The complexity of the adaptive fuzzy system is determined by the radius r. If the radius is small more clusters will be created, resulting in a more sophisticated nonlinear regression. This however also results in a fuzzy system which is more computationally intensive [5].

2.5 COMPARISON OF MODELLING TECHNIQUES

In any engineering discipline the best suited method should be sought after so as to obtain the best results. In this section a comparison of the methods covered in previous sections are discussed. Considering all the advantages and disadvantages it is possible to draw a conclusion as to which method would be best suited to model the behaviour of the control valve.

Table 2.3. *Modelling Methods Comparison*.

Criterion	First Principle	Model Fitting	Experimental Modelling	Fuzzy Logic System
Use real-world data in modelling process	No	Yes	Yes	Yes
Modelling process involves training	No	No	No	Yes
History based data	No	Yes	Yes	Yes
Training data as model	N/A	N/A	N/A	Yes
Algorithm complexity	Application specific	No	No	Intermediate
Oscillations near data sample endpoints	Possible	No	Yes	N/A
Mathematical model order	Application specific	Application specific	Application specific	Application specific
Degree of system integration	Simple	Simple	Intermediate	Intermediate

Criterion	First Principle	Model Fitting	Experimental Modelling	Fuzzy Logic System
Adaptability to system changes	Model needs revision to accommodate changes	Model needs revision to accommodate changes	Model needs revision to accommodate changes	Simple – provided data is available
Model data range limitations	No	Yes	Yes	Possibly not due to interpolation and extrapolation algorithm
Overall performance	Good	Medium	Medium	Good

Fuzzy logic uses previously acquired real-world data (history based data) to derive a mathematical model. This model is derived by training the system with the acquired data, which after training, together with the mathematical function (2.9), are used as the mathematical model. Model fitting, first principle and experimental modelling as apposed to fuzzy logic do not involve training algorithms even though model fitting and experimental modelling do make use of history based data.

Depending on the application, the algorithms' complexities vary from simple, in the case of model fitting and experimental modelling to application specific for a first principle model and intermediate for fuzzy logic.

Due to the algorithm, which leans towards a weighted average function, fuzzy logic does not tend to oscillate near the endpoints as in the case when using a high order experimental model. The first principle modelling method can also avoid oscillation by carefully choosing system parameters and using effective modelling techniques. Due to low order interpolation techniques, model fitting does not pose any such disadvantages.

Due to many application specific factors which have to be considered, it is difficult to estimate the order of such a model. Experimental modelling however easily tends to reach higher orders, when attempting to derive a model, producing a model difficult to interpret. Model fitting on the other hand have low order functions describing the application's behaviour, producing a number of simple functions within a specific range. All models can easily be integrated into a simulation environment.

Fuzzy logic, as apposed to the other modelling methods, provides an environment where a fuzzy logic model can be derived and validated with relative ease. This however depends mainly on the availability of the data as well as the format of the data. The mathematical model

depends on four parameters; the sum of the outputs in a cluster ($\underline{A}$), the number of input data samples in a cluster ($\underline{B}$), the cluster centres ($\underline{x_0}$) and the one dimensional smoothing parameter – sigma (σ).

The remaining three modelling methods have to undergo either a partial or an entire revision of the modelling procedure if any device behavioural changes are made. Thereafter the new revision of the model, as with all other methods, has to be updated in the simulation environment.

Model fitting and experimental modelling do not employ adaptive algorithms, which implies that the models are only valid within the specified data range. Data not within range cannot be accurately interpreted. First principle models in general make use of generic models and functions which are to cover an even larger range, as in the case with model fitting and experimental modelling. Fuzzy logic models use an adaptive algorithm which is capable of interpolating between data samples or extrapolating with historic data samples to some degree of accuracy.

Although first principle models require extensive knowledge about the specific engineering discipline, these models are still widely used and provide a good overall performance. Model fitting and experimental modelling provide models which are only valid within the specific data range. Within this data range these models' performance are good but are poorly defined outside of the data range. Fuzzy logic models generally provide good performance although finding the optimal parameters such as number of time delays, effective time delay increments, system radius and the smoothing parameter can be a daunting task. Its overall performance is therefore good.

2.6 SUMMARY

In this chapter numerous modelling methods were covered in relative detail. This provided a good basis to establish a number of comparison criteria. Taking all the criteria into consideration, fuzzy logic was found to be an excellent modelling technique, providing good overall performance within a wide data range. Fuzzy logic also provides an adaptive algorithm with the ability to be trained online.

Chapter 4 searches for the optimal fuzzy logic parameters given a single set of data files. These parameters will then be used in *Chapter 5* to derive the final fuzzy logic mathematical model which can be used within a simulation environment.

CHAPTER 3

Dynamic Data Capturing

This chapter presents the development of a Data Acquisition (DAQ) System to be used in conjunction with Fuzzy Modelling to derive a dynamic mathematical model of a control valve. Device specific data are obtained by using hardware signal converters, a DAQ-device (normally an Analog to Digitial (A/D) card) and a suite of software to manage the DAQ process. The software is programmed in a graphical programming development environment called LabVIEW™ (Laboratory Virtual Instrument Engineering Workbench) and is customised to meet the specific needs of the application.

Due to the nature of the measured signals, analog current loop signals (4 – 20 mA according to industry standards [16]) have to be converted to analog voltage signals with signal conditioning hardware. The DAQ device, which includes an A/D converter, converts the voltage signals into 12-bit digital signals, which are scaled, displayed and logged to file.

Measure and Automation Explorer (MAX) is a utility used by National Instruments™ (NI) to configure DAQ-devices, perform system diagnostics, create and edit virtual channels, tasks, interfaces, scaling functions and virtual instruments [17].

In this chapter common terms and concepts associated with DAQ systems are discussed. The pipe network setup, hardware design and software user interfaces for both data acquisition and post-processing applications are also covered.

3.1 COMPONENTS OF A DAQ SYSTEM

The DAQ system comprises sensors and transducers, signal conditioning hardware and a suite of software for acquiring, manipulating and displaying the acquired data on a graph as well as simultaneously saving it to file.

The sensors and transducers transfer information by means of 4 – 20 mA current loops, resulting in a system which is less susceptible to noise and immune to line impedance. A current loop conductor path can therefore stretch over several meters between a sensor and signal conditioning hardware or programmable logic controller (PLC) input/output device.

Adding an isolation amplifier to a current loop protects system electronics from electrical noise and transients, and allows sensors and transducers to be electrically isolated from high voltages which could damage these instruments [18].

Signal conditioning hardware is used to convert the current loop signal to a voltage signal which can then be sampled by the DAQ device. Voltage conductor paths should be kept as short as possible to minimise the effect of noise on the system. Sampling the analog voltage signal implies converting it to a sequence of 12-bit digital values (refer to *Appendix A*).

Developed software simplifies the DAQ process, not only making it accurate and consistent but also saving it to file. Once the raw data have been acquired, post-processing is done which calculates the values of the governing (standard) equations such as mass rate of flow (kg/s), expansibility (expansion) factor, the Reynolds number and valve choke status.

Two construction methods are mainly used when implementing DAQ solutions. Figure 3.1 (*a*) uses a DAQ plug-in card, which resides in the computer, be that a desktop PCI (Peripheral Component Interconnect) card, or a laptop PCMCIA (Personal Computer Memory Card International Association) card. The current loop passes through signal conditioning hardware, converting it to an ADC (Analog Digital Converter) interpretable voltage signal which is then sampled by the DAQ device. In Figure 3.1 (*b*) the DAQ board is external to the master computer, which measures, converts and conditions the signals outside of the master computer. This DAQ process is remotely controlled through various busses such as the serial port, parallel port, Universal Serial Bus (USB) port or IEEE 1394 (Firewire) port. These busses also provide data transmission once data have been acquired [19]. Use this method when real-time acquisition is required.

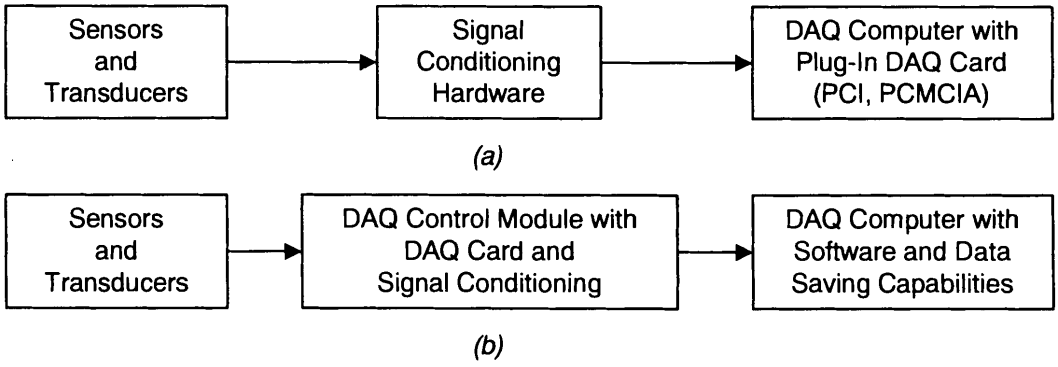


Figure 3.1. *DAQ System Setups*.

3.2 IMPLEMENTED DAQ SYSTEM

The implemented DAQ configuration is a hybrid of the two construction methods mentioned. A remote computer (terminal) in which a PCI DAQ-device (PCI-6023E) and DAQ-software is installed, is remotely controlled by a master via a 100-base TCP network. This configuration method enables simultaneous manipulation of both the plant and the DAQ process as illustrated in Figure 3.2.

Besides simultaneous control of both plant and DAQ process other advantages include:

- Reduced network latency due to local data saving capabilities of the terminal computer
- High DAQ rates due to limited terminal computer processes
- Effortless data transfer from terminal computer to master computer via the established network

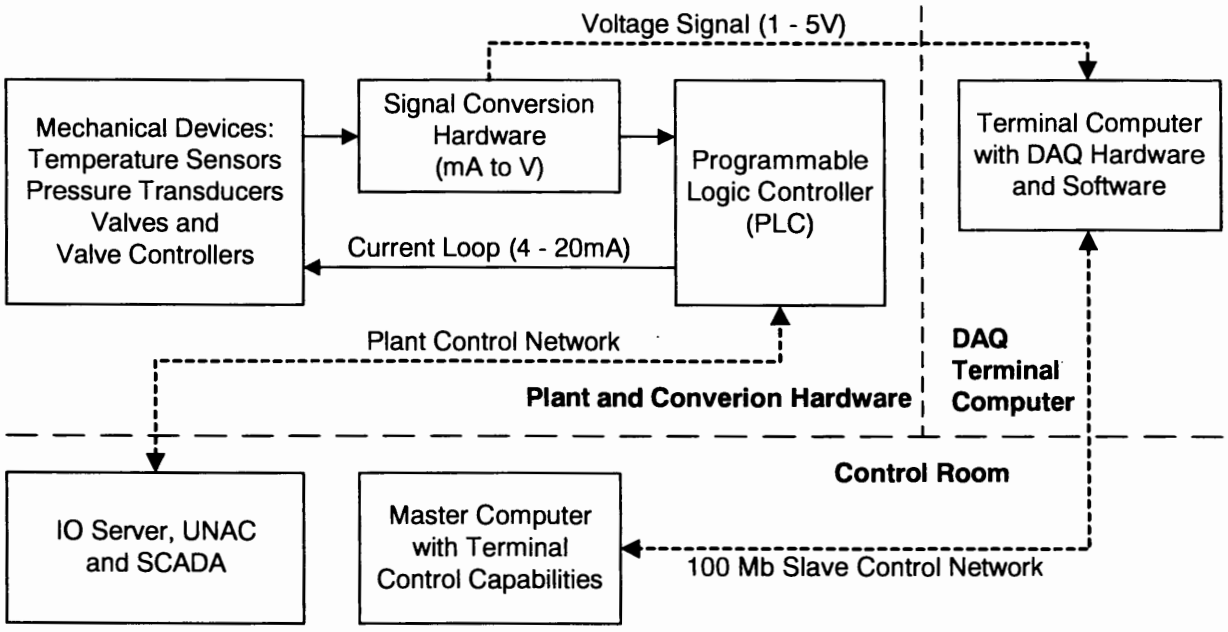


Figure 3.2. *DAQ Setup*.

The terminal computer features both master and slave capabilities. It is a master in the sense that the entire DAQ process can be managed and controlled from the terminal. This implies managing the virtual channels, setting up the hardware, facilitating data saving and displaying the sampled data on a graph. It is a slave when another master takes control of the terminal computer by means of remote desktop applications such as Netmeeting© from Microsoft™. This allows the new master to control the remote terminal (previous master) thus the DAQ process, via an established TCP network.

3.3 PIPE NETWORK SETUP

To derive the mathematical model of the control valve, physical data need to be acquired in order to train the Fuzzy Logic System (FLS). The pipe network setup used to acquire the training and validation/verification data can be seen in Figure 3.3. It is clear from this figure that the pipe network setup is divided into three sections:

- Gas supply and manifold (*W-X*)
- Mass flow calculation (*X'-Y*)
- Control valve (*Y'-Z*)

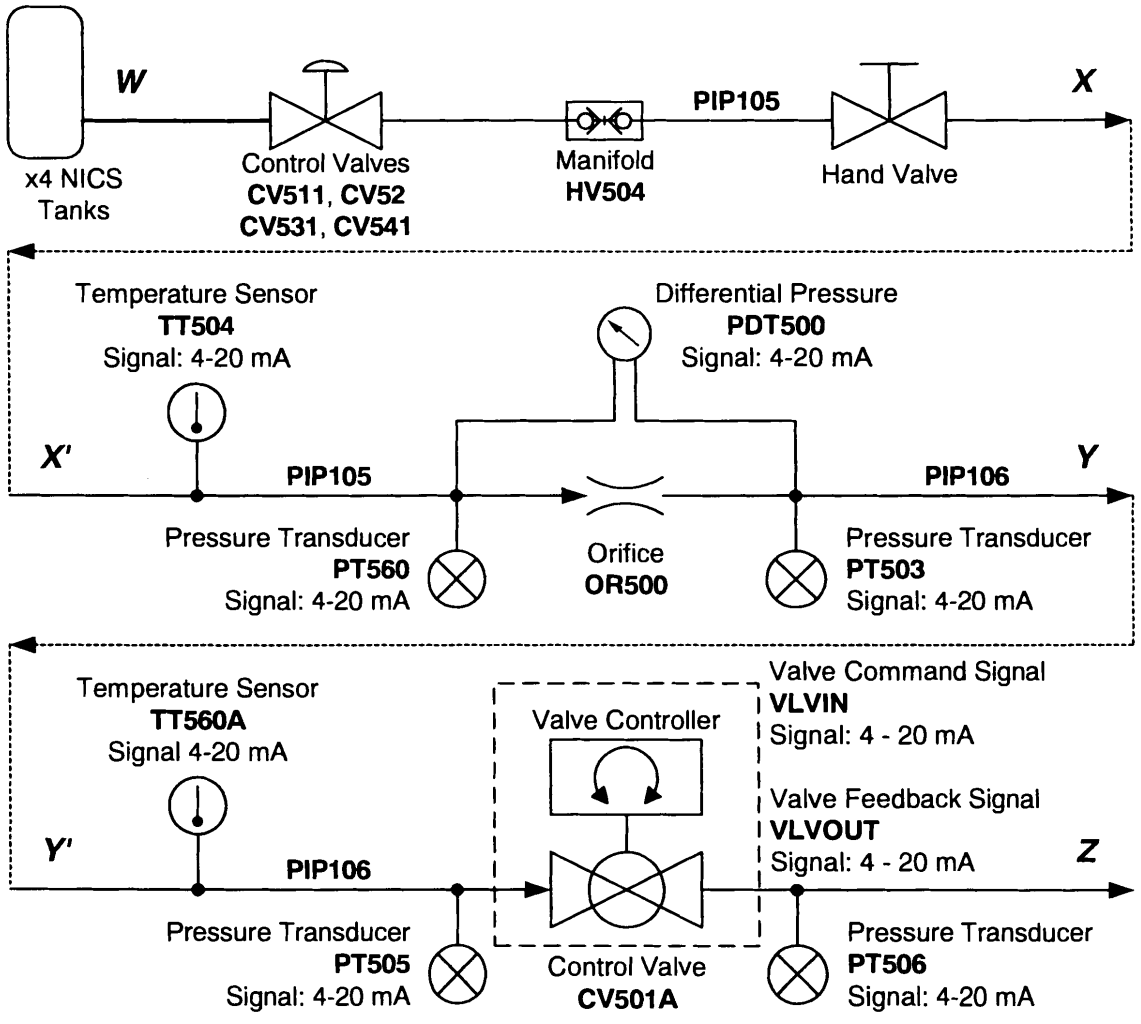


Figure 3.3. *Pipe Network Setup and Transducers.*

3.3.1 Gas Supply and Manifold

The Nitrogen Inventory Control System (NICS) tanks are inflated with dry air to a specified pressure (determined by the conducted experiment but not more than 5 bar). The compressed

dry air is then released through the pipe network while measuring the temperatures, pressures, differential pressure and control valve opening.

The choice of dry air involves a number of factors:

- There is an abundant supply
- No additional costs are presented; therefore no limit to the number of experiments that can be conducted
- No health risks involved, which are present when using nitrogen or any other gases

3.3.2 Mass Flow Calculations

While controlling the behaviour of the control valve, dry air is being released through the pipe-line network simultaneously acquiring the temperatures, pressures and differential pressures thereof. These measurements are then used to calculate the mass flow through the orifice, effectively indicating the mass flow through the control valve.

Since the Reynolds number is a dimensionless indication directly related to the mass rate of flow through a specific pipe it is also calculated. Both the mass rate of flow and Reynolds number are used in the FLS to derive the control valve mathematical model.

By means of the differential pressure across, as well as the mass rate of flow through the orifice and control valve, their choke status is also calculated. This parameter could also be used in the derivation of the control valve mathematical model, enabling the characterisation of the control valve in both choke and normal (non-choke) operation conditions.

3.3.3 Control Valve

The control valve, being the main element in the pipe network setup, is opened and closed according to the predetermined experiments, which enables an accurate characterisation of the control valve. Measurements of the temperature, pressures and valve opening are used to characterise the control valve. Refer to section 3.8 for more information on the conducted experiments pertaining to the characterisation of the control valve.

3.4 DAQ HARDWARE CONVERTERS

Bi-directional communication is established between PLC and transducer via analog current loops. This industry standard implies that, in order to extract information from such transducers, i.e. to be sampled by a DAQ-device, the current loop signals need to be converted to voltage signals. The signal converters linearly convert the 4 – 20 mA current loop signals into 1 – 5 V voltage signals, which could then be sampled by the DAQ device and saved to file.

Detailed specifications and a user manual of the hardware signal converters can be found in *Appendix A*. Features, signal connections and DAQ device pinouts pertaining to the DAQ PCI card can also be seen in the appendix as well as the analog to digital conversion process.

3.5 DAQ SOFTWARE

LabVIEW is a powerful and flexible instrumentation and analysis software development environment. Applications range from NASA projects (subdivisions of the Mars Pathfinder Sojourner) to determining the interaction between molecules in thin films used in chemical sensors and optical devices [19].

LabVIEW "programs" are called Virtual Instruments (VI) and differ from text based programming languages such as C++, Pascal, Java and Basic in that programming is done by means of a graphical programming environment called *G* programming language. VIs are created using graphical symbols to describe programming actions and are easily understood due to its graphical nature.

Once such VIs have been created it may be necessary to debug if undesirable results are obtained. This is done graphically with a number of actions normally used in any other programming environment. These include: setting breakpoints, single-stepping through the program, inspecting (watch variables) and manipulating values.

Once a complete graphical user interface (GUI) has been programmed the VI together with its subVIs can be packaged into an executable file, ready for distribution. Much like Java, an interpreted language, which requires a Java Virtual Machine (VM) to execute Java applets, a packaged (executable) VI requires the installation of the LabVIEW Run-Time Engine [20]. Together with the Run-Time Engine, MAX (which includes DAQ device drivers) also needs to be installed in order to set up virtual channels, manage channel scaling and conduct device testing to name a few. If the DAQ system however is not used for development purposes, but purely for acquiring data from a physical system, creating an executable of the VI could drastically reduce the total installation size. This will eliminate unnecessary overheads by incorporating only the necessary components.

3.5.1 Configuring the Channels

A DAQ virtual channel in MAX sets up the behaviour of the channel and enables communication with the physical channel on the DAQ device. LabVIEW together with MAX creates an environment with which these virtual channels can be set up with great ease in whatever mode the user desires. These channel modes could be anything from; nonreferenced

single ended (NRSE), referenced single ended (RSE) to differential (DIFF) mode (refer to *Appendix A*). Creating these virtual channels in MAX will not be covered due to a self explanatory channel-setup-wizard.

Together with selectable channel modes, users can also implement a scaling formula. The aim of the formula is to scale raw voltage data into more interpretable values, be that temperatures (°C), pressures (kPa) or valve opening (%), while the system acquires the physical data. Due to the transducers' and DAQ Hardware converters' input/output linear characteristics, it is simple to derive such scaling formulae. The MAX implemented scaling functions can be seen in Table 3.1.

Table 3.1. *System Implemented Scales*.

Transducer Plant Tag	Transducer Range	Scale
TT504 and TT560A	0 – 150 °C	$y = 37.5x - 37.5$
PT560, PT505 and PT506	0 – 1000 kPa [2]	$y = 250x - 163.5$
PDT500	-10 – 40 kPa [2]	$y = 12.5x - 22.5$
VLVIN and VLVOUT	0 – 100 %	$y = 25x - 25$

For this application the channels are set up in differential mode to minimise noise interference. Figure 3.3 indicates that only eight channels are used to adequately capture the transient characteristics of the control valve.

3.5.2 DAQ Process Control GUI

Controlling the DAQ process involves a GUI which simplifies acquiring the experiment data, making it effortless, accurate and consistent. The GUI controls both DAQ device and process, allowing the user to select system settings pertaining to the DAQ process i.e. sampling rates, buffer sizes, channels, experiment-startup and shutdown status as well as data logging file settings. The DAQ process control GUI does not control the transducers and control valve, it only monitors their conditions and logs it to file. The GUI is shown in Figure 3.4 where the pipe network setup (constituting transducers, an orifice and a control valve) can be seen at the top. The DAQ system control can be seen on the left (middle to bottom), the trend graph in the middle and the quick reference help system in the bottom right hand corner. Appendix B covers the DAQ software user manual in detail.

[2] Gauge pressures are converted to Absolute pressures (86.5 kPa).

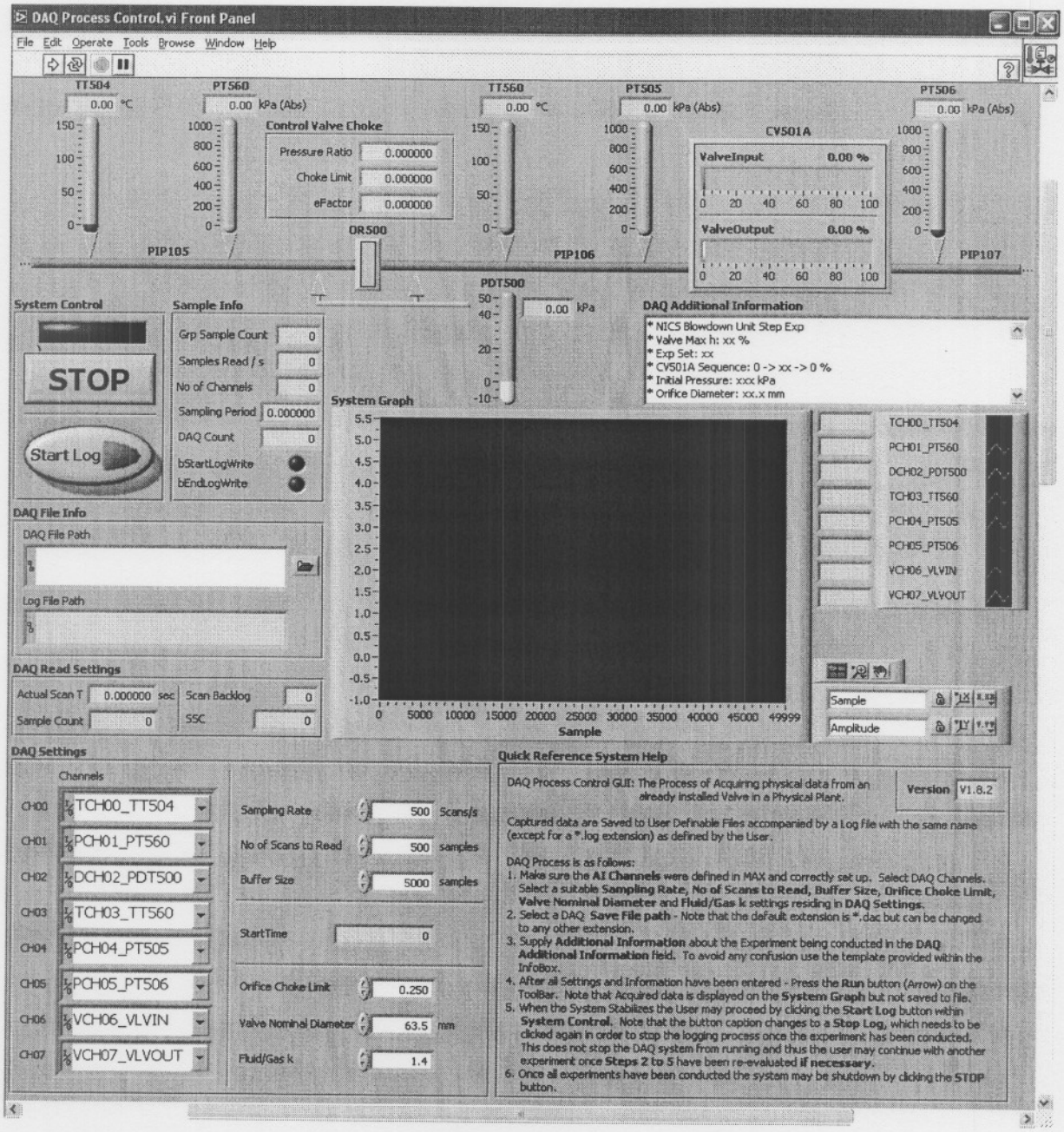


Figure 3.4. *DAQ Process Control GUI*.

The GUI features are as follows:

- DAQ process control – managing the initial system startup, experiment startup and shutdown status and file save settings
- Graphical representation of the experimental pipe network setup indicating all system temperatures, pressures and valve opening
- Data file and log file control
- Additional experiment user specifiable information
- DAQ channel settings
- Quick reference help system and

- System feedback – orifice/valve choke status, experiment sampling information, graphical transducer indicators and DAQ trend displaying

3.5.2.1 Acquisition Process

Once all settings have been set up (channels, sampling rates, filename and additional information) the acquisition process can be started by clicking the *run* arrow on the taskbar. This initiates the acquisition process by displaying the acquired values on the pipe network indicators (refer to Figure 3.4). Data will not be logged to file until the *Start Log* button has been clicked. The software flow diagram can be seen in Figure 3.5.

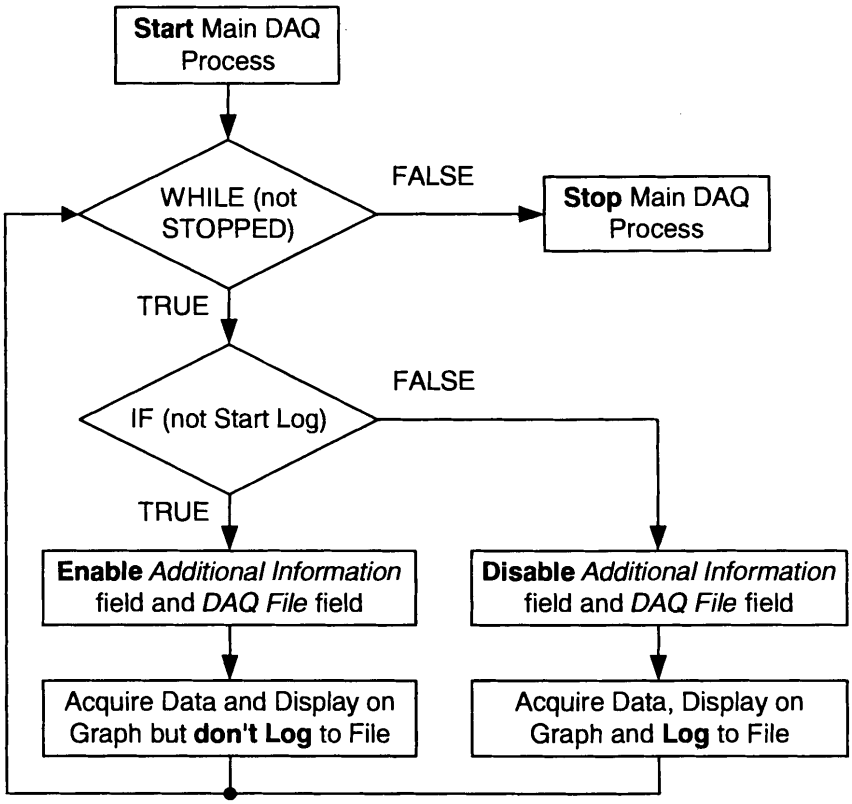


Figure 3.5. *DAQ System Flowchart.*

Clicking the *Start Log* button will initiate the data saving process which logs data to the specified file. The button's caption changes to *Stop Log* with which the data saving process can be stopped at anytime without stopping the main acquisition process. This allows the user to visually decide when the actual data logging should start and stop. When the data logging process has been stopped the user can once again modify the applicable settings needed for another experiment.

3.6 POST PROCESSING

Once all data have been acquired post-processing is required in order to enable proper training of the FLS thus enabling the development of a truly dynamic mathematical model characterising the behaviour of the control valve.

The acquired data include both dynamic and static control valve characteristics. Calculating the values of the governing equations for fluid-flow dynamics such as mass rate of flow, Reynolds number, expansibility factor and choke status will aid in the development of an accurate mathematical model.

The calculations of the governing equations are based on the British International Standard (ISO 5167-1) [6].

3.6.1 Mass Rate of Flow Calculation

According to the British International Standard [6] the method of measuring flow in a pipe-line is based on the installation of a primary device such as an orifice plate, nozzle or a venturi tube. The installation of this primary device causes a static pressure difference between the upstream side and the downstream side of the device. The rate of flow can be determined from the measured value of this pressure difference and from the knowledge of the characteristics of the flowing fluid/gas as well as the circumstances under which the device is being used.

The mass rate of flow can be determined with the following formula provided the differential pressure is within the uncertainty limits stated in ISO 5167 [6]:

$$q_m = \frac{C}{\sqrt{1-\beta^4}} \varepsilon \frac{\pi}{4} d^2 \sqrt{2\Delta p \rho} \tag{3.1}$$

Table 3.2. *Symbols and Subscripts.*

Symbol	Quantity	SI Unit
q_m or $\dot{m}$	Mass rate of flow	kg/s
β	Diameter ratio: $\beta = d/D$	-
C	Coefficient of discharge	-
d	Diameter of orifice of primary device at working conditions	m
D	Upstream internal pipe diameter at working conditions	m
p	Absolute static pressure of the fluid	Pa
Δp	Differential pressure	Pa
ε	Expansibility (expansion) factor (upstream or downstream)	-
ρ	Density of fluid (upstream or downstream)	kg/m^3

Symbol	Quantity	SI Unit
VelOfAp	Velocity of approach factor: $\frac{1}{\sqrt{1-\beta^4}}$	-
R_{gas}	Gas constant from $pv = RT$	kJ/kg-K
T_o	Temperature in Kelvin	K
P_{up}	Upstream absolute pressure	Pa
P_{down}	Downstream absolute pressure	Pa
μ	Dynamic viscosity of fluid	Pa.s
R_{ep}	Reynolds number with respect to pipe	-

The calculation of the mass rate of flow is purely an arithmetic process and usually involves a complex numeric method to solve the rate of flow. This is due to the fact that q_m is a function of C which in turn is a function of R_{ep} which in turn is again a function of q_m.

This method for calculating the mass rate of flow could be time consuming and requires a numerical method to solve for q_m. The complexity of the calculation is simplified by fixing C to an average found within the tables of the British Standards (ISO 5167) [6].

Once the simplification, concerning C, has been done the governing equations for the expansibility factor, mass rate of flow and Reynolds number found within the British Standards no longer require a complex and time consuming numerical method to obtain a solution. The governing equations are simplified to (3.2) and any programming language such as LabVIEW, C++ or Matlab can be used to quickly obtain solutions.

3.6.2 Choke Flow Calculations

3.6.2.1 Choke

Choked flow occurs when the fluid velocity reaches sonic values at any point in the valve body, trim or pipe. As the differential pressure across the valve is increased with constant inlet (upstream) pressure, the specific volume flow rate increases to a point where sonic velocity is reached. The velocity at any point in the valve or downstream piping is limited to sonic flow (Mach = 1). As a result, the rate of flow will be limited to an amount which yields a sonic velocity in the valve under the specific upstream and downstream pressure conditions, thus implying a constant flow rate even though differential pressure across the valve or orifice is increased [8], [21].

$$\beta = {}^{d}/_{D}, \quad D = 0.0635$$

$$VelOfAp = \frac{1}{\sqrt{1-\beta^4}}$$

$$T_o = T_{degC} + 273.15$$

$$\rho = \frac{P_{up}}{R_{gas}T_o}, \quad R_{gas} = 0.2968 \tag{3.2}$$

$$\varepsilon = 1 - \left[\frac{0.41 + 0.35\beta^4}{1.4}\right]\frac{\Delta P}{P_{up}}$$

$$\dot{m} = C \times VelOfAp \times \varepsilon \times \frac{\pi}{4} \times d^2 \times \sqrt{2.\Delta P.\rho.1000}, \quad C = 0.61$$

$$R_{ep} = \frac{4\dot{m}}{\pi\mu D}, \quad \mu = 2\times10^{-5}$$

3.6.2.2 Calculate Choke Flow Conditions

From the British Standards [6] it is clear that choked flow through the orifice can be estimated by evaluating the ratio of the downstream to upstream pressure. The ratio should be smaller than 0.75.

Due to the fact that an orifice is used to calculate the mass rate of flow through the valve and pipe network, a differential pressure transducer (*PDT500*) was installed across the orifice with a maximum range of +/-500 mbar (+/-50 kPa) (also see Figure 3.3). The above mentioned ratio could be formulated to incorporate the differential pressure instead of the downstream pressure by substituting $P_{down} = P_{up} - \Delta P$ in $\frac{P_{down}}{P_{up}} < 0.75$ to obtain (3.3) describing the flow through the orifice by means of differential pressure and upstream pressure.

$$\frac{\Delta P}{P_{up}} > 0.25 \tag{3.3}$$

To calculate whether the flow through the valve is in choke is not as trivial as in the case for an orifice. The estimation for choked flow through a valve is divided into three parts.

- Calculate a Valve flow coefficient, C_v
- Calculate a Terminal pressure drop ratio, x_T
- Compare the Pressure ratio with the Choke limit

From the valve specific datasheets the function describing the valve flow coefficient is:

$$C_v = \left(1.9837 \times 10^{-9}\right)h^6 - \left(3.7298 \times 10^{-7}\right)h^5 + \left(2.3493 \times 10^{-5}\right)h^4 - 0.0004453h^3 + 0.009669h^2 + 0.25542h - 0.0004731 \tag{3.4}$$

This approximated sixth-order function describes the valve flow coefficient and is thus used to find any values, given a valve relative opening (h) in percentage.

Once the C_v value has been calculated, together with the valve relative opening (now in inches) can then be used to calculate the terminal pressure drop ratio (x_T). Again the valve specific manufacturing datasheets are used to calculate a function describing x_T. The function describing x_T can be seen in (3.5). Note the entire function is described by five individual first-order equations which are only valid within the specified ranges.

From the Terminal pressure ratio it is possible to calculate the choke limit. The choke limit is then compared with the pressure ratio to determine if the flow through the valve is in choke. This can be seen in (3.6). If the Pressure Ratio (*PRatio*) is greater than/or equal to the Choke Limit (*ChokeLimit*) the flow through the valve is said to be in choke.

$$\begin{aligned}
X_T &= -0.00563\,C_v/d^2 + 0.6471; && 0 \le C_v/d^2 \le 4.93 \\
X_T &= -0.013\,C_v/d^2 + 0.68595; && 4.93 < C_v/d^2 \le 15.07 \\
X_T &= -0.0161\,C_v/d^2 + 0.7322; && 15.07 < C_v/d^2 \le 20.67 \\
X_T &= -0.00854\,C_v/d^2 + 0.5791; && 20.67 < C_v/d^2 \le 48.90 \\
X_T &= -0.000355\,C_v/d^2 + 0.17734; && 48.90 < C_v/d^2 \le 500
\end{aligned} \tag{3.5}$$

$$\begin{aligned}
PRatio &= \Delta P / P_{up} \\
ChokeLimit &= F_k \times X_T
\end{aligned} \tag{3.6}$$

3.6.3 Expansibility Factor (ε)

This coefficient takes into account the compressibility of the fluid/gas. The expansion factor is equal to one if the fluid is incompressible and less than one if compressible. The expansion factor accounts for the variation of specific weight as the gas passes from the valve inlet to the vena contracta where the flow area is the smallest. It also accounts for the change in cross-sectional area of the vena contracta as the pressure drop is varied [6], [8], [21].

The expansibility factor is calculated with the following formula:

$$\varepsilon = 1 - \left(0.41 + 0.35\beta^4\right)\frac{\Delta P}{\kappa P_{up}} \tag{3.7}$$

where κ is the isentropic exponent ratio which is the ratio of the relative variation in pressure to the corresponding relative variation in density under elementary reversible isentropic transformation conditions [6]. The value for κ can be read from various ideal gases' property tables but in this case equals 1.4 [7].

3.6.4 Post-Processing GUI

The post-processing application in Figure 3.6 makes use of (3.2) through to (3.7). The governing equations together with the original data are saved to a new file with the same name, except for a *.pr* extension. The processed file is used to train the FLS and to verify and validate the behaviour of the dynamic control valve model.

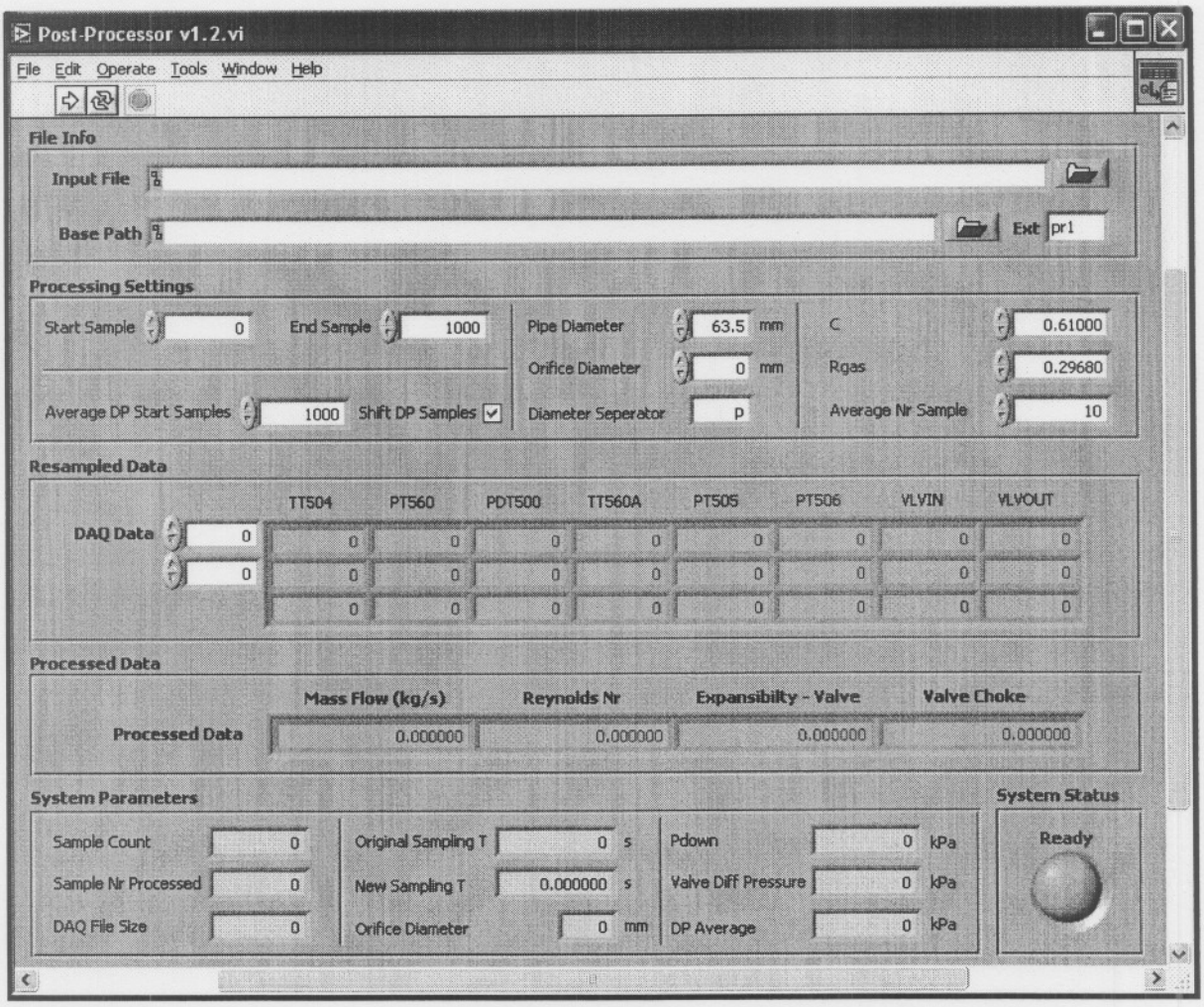


Figure 3.6. *Post-Processing GUI*.

Processing original data entails selecting the original input file from the *File Info* section and setting up the *Processing Settings*. Press the *run* button on the toolbar to commence processing. Refer to *Appendix B* for the post-processing GUI user manual.

3.7 DATA INTERPRETATION

From Figure 3.7 it is clear that during the acquisition process, transducer data is graphed on the trend chart. These trend graphs assist in identifying the status of the transducers at any point in time. Once an experiment has been conducted it is often necessary to view the acquired data. The *Plot Data* application, which was specifically designed to view the acquired and processed data, assists with this task.

The data plot application is divided into an *original data* section and a *processed data* section which graphs the originally acquired data and the processed data respectively. The *log file* section reads the log file information. To view acquired and/or processed data, supply a valid file (created with either the *DAQ Process Control* application producing a *.dac file or the *Post-Processing* application producing a *.pr* file) and press the *run* button on the toolbar.

The data plot GUI's *original data* section can be seen in Figure 3.7. Due to the ambient temperature the two sensor trends (TCH01_TT504 and TCH03_TT560) do not differ significantly and are almost indistinguishable. Once the control valve is opened, the orifice upstream absolute pressure (PCH01_PT560) decreases. Due to the reduction in the pipe network, caused by the orifice, the control valve absolute upstream pressure (PCH04_PT505) is always less than the orifice upstream absolute pressure (PCH01_PT560). This causes a pressure difference across the orifice which allows the calculation of the mass rate of flow through the orifice and thus through the control valve. A more accurate version of the pressure difference across the orifice is acquired by the differential pressure transducer (DCH02_PDT500). The control valve downstream absolute pressure decreases due to the creation of a vacuum caused by the vena contracta.

The virtual channels set up in MAX have a fixed naming convention which allows instant identification. The convention is as follows: *aCHb_c*, where *a* represents the transducer type, *b* the differential channel number and *c* the transducer's plant identification name. Four different transducer types are introduced; *T* the abbreviation for temperature, *P* the abbreviation for pressure, *D* the abbreviation for differential pressure and *V* the abbreviation for control valve. Differential channel numbers, represented by *b*, range from *00* to *07* due to eight channels being used to derive the model. The plant identification names, represented by *c*, can be seen

in Figure 3.3. The eight channels used to describe the dynamic behaviour of the control valve are listed, in sequence, in Table 3.3.

Table 3.3. *Differential DAQ Channels*.

	Channel Name	Description
1	TCH00_TT504	Orifice upstream absolute temperature
2	PCH01_PT560	Orifice upstream absolute pressure
3	DCH02_PDT500	Orifice differential pressure
4	TCH03_TT560	Valve upstream absolute temperature
5	PCH04_PT505	Valve upstream absolute pressure
6	PCH05_PT506	Valve downstream absolute pressure
7	VCH06_VLVIN	Control valve input command signal from control room
8	VCH07_VLVOUT	Control valve output signal

Typical data acquired from the pipe-line and processed by the post-processor can be seen in Figure 3.7.

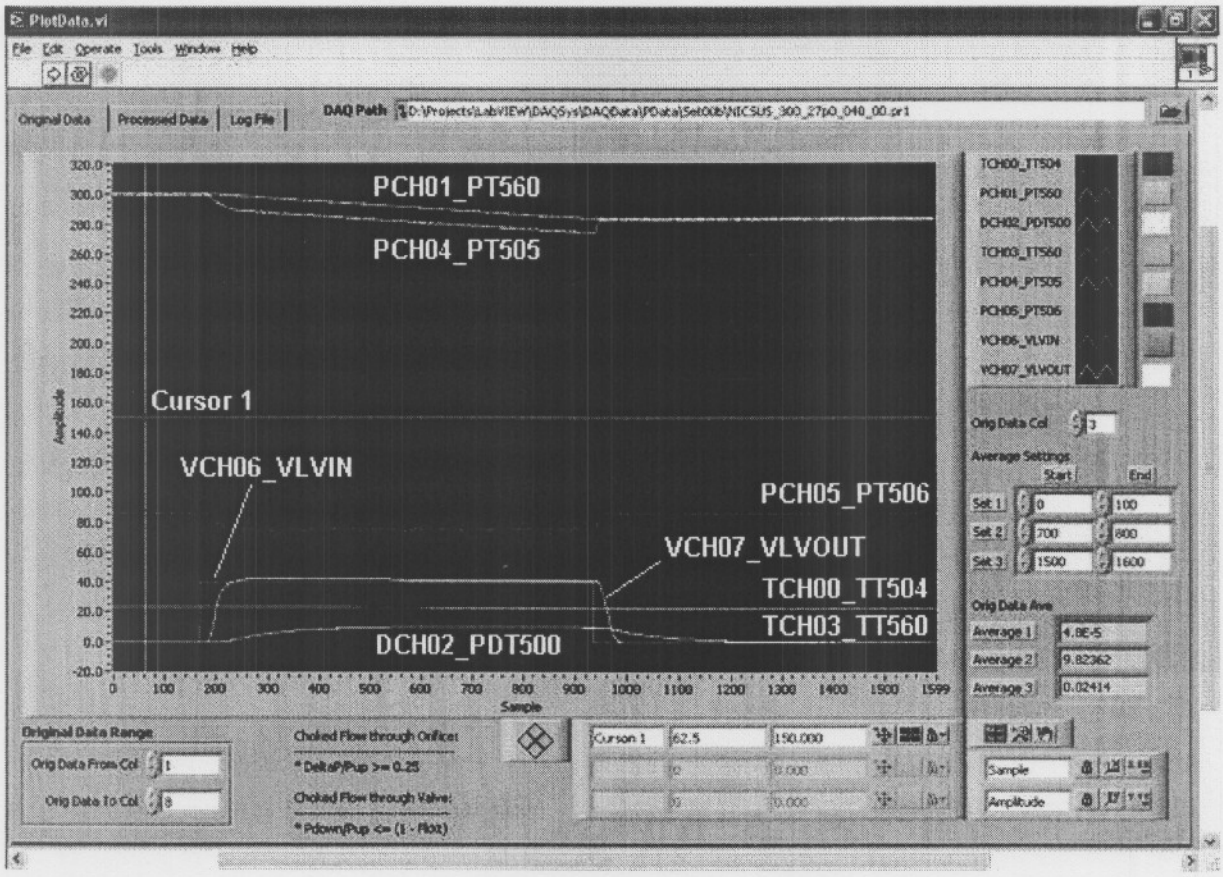


Figure 3.7. *Originally Acquired Data*. [3]

[3] The channel sequence portrayed in the *plot data* application also applies to the *pipe network setup*.

Due to multiple trends on the same graph, the amplitude dimension depends on the specific trend examined. A temperature trend has a °C amplitude dimension, a pressure trend has a kPa amplitude dimension and a control valve relative opening has a % amplitude dimension. This is due to the scaling functions implemented in MAX (refer to Table 3.1).

The processed data, calculated by the governing equations, can be seen in Figure 3.8 (*processed data* tab). Due to the large values of the Reynolds number the other calculated values (mass rate of flow, expansibility and choke status) can not be seen without enlarging the specific area. The mass rate of flow, expansibility and choke status can be seen in the inset added to Figure 3.8. The white box indicates the original position on the graph. In this instance the maximum mass rate of flow through the pipe network is about 0.0892556 kg/s and the flow through the control valve remains in choke throughout the entire experiment (indicated by the boolean value of 1 within the inset).

A maximum of three cursors per graph may be used to query trend values. Averages, with settings specified in the *original data* section, can also be used to estimate trend values.

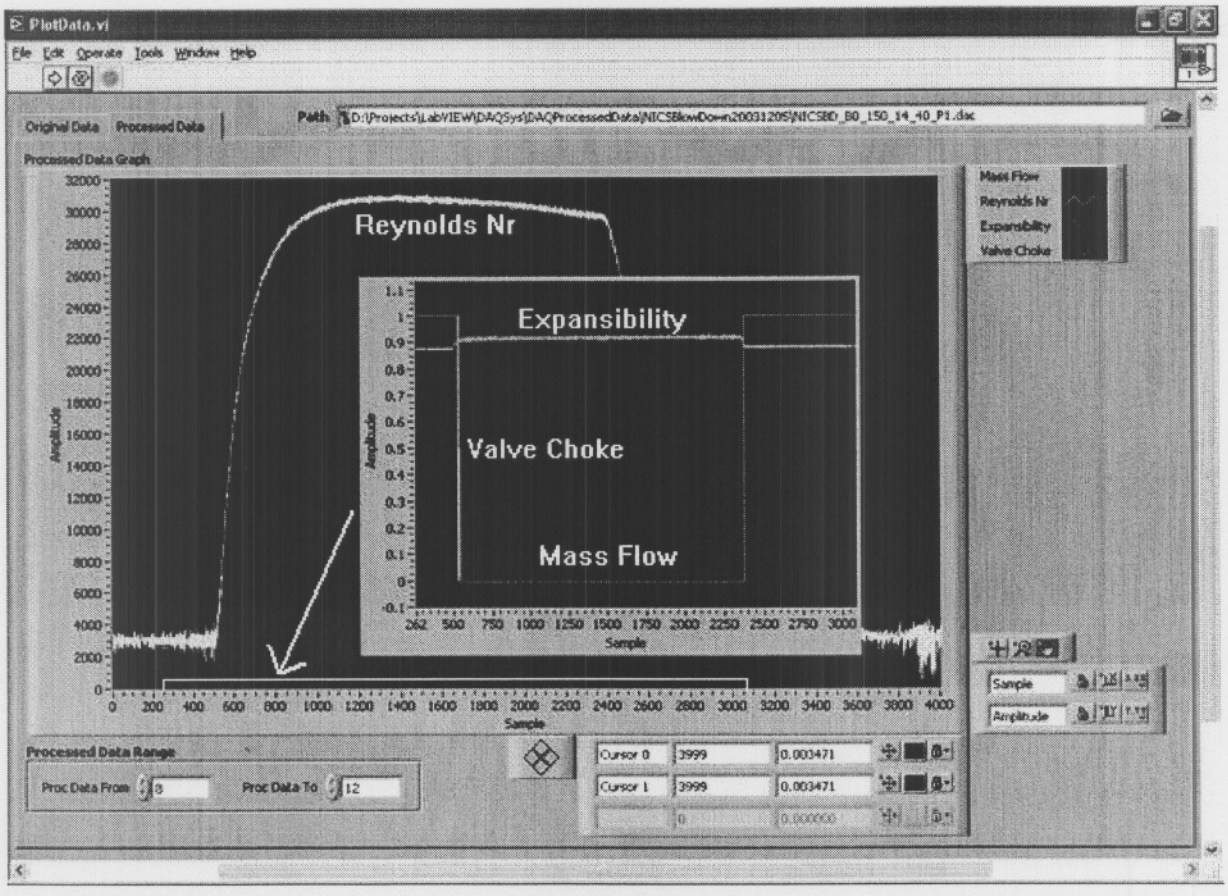


Figure 3.8. *Processed Data*.

From the log file (bearing the same name as the original data file except for the extension) the sampling period could be used to calculate the time which passed between two samples. This enables a time-stamp to be connected to each sample.

Refer to *Appendix B* for the in depth *plot data* application user manual.

3.8 EXPERIMENTAL PROCEDURES

To characterise the dynamic behaviour of the control valve a number of experiments need to be conducted on the control valve. These experiments are used to acquire control valve specific parameters such as valve temperature, valve upstream and downstream pressures and valve relative opening.

The valve is controlled via the control room and a SCADA system. The current implemented SCADA system cannot generate any mathematical signals such as sine-waves, chirp-signals or even ramp-signals. It only allows unit step signals to control the system and thus limits the number of experiments that can be conducted. The experiment conducted is called the unit step experiment and its main objective is to characterise the control valve during the opening and closing phases. Other experiments may also be conducted but will possibly require the use of extra instrumentation such as compressors, flow meters and pressure vessels.

3.8.1 Unit Step Experiment

The advantage of using unit step commands is that these signals contain high frequency components which introduce transient components to the control valve. The captured control valve response is then used to derive the dynamic mathematical model. The unit step experiment always commences at a relative control valve opening of zero percent (0 %). The command then steps to a maximum experimentally determined relative opening of x % before returning back to zero percent relative opening. The step commands are as follows:

- The valve is fully closed (0 % open)
- A command signal is issued to open the valve x %
- The valve responds by opening to $\pm x$ % with possible overshoot and ringing but eventually stabilising at about x % relative opening
- Wait for temperatures and pressures to stabilise before another command is issued
- Another command is issued to fully close the valve
- The valve responds by closing to 0 % relative opening
- Wait for temperatures and pressures to stabilise before the DAQ process is stopped

The mentioned command sequence can be seen in Figure 3.7 where *VCH06_VLVIN* represents the command signal and *VCH07_VLVOUT* the valve response. Notice that *DCH02_PDT500* reaches steady state before introducing any changes to the command signal.

The unit step experimental sequences for an initial pressure range of 200 kPa to 400 kPa with increments of 50 kPa can be seen in Table 3.4.

Table 3.4. *Unit Step Sequence.*

Orifice Size (mm)	Unit Step Sequence (%)			
14	0 → 5 → 0,	0 → 10 → 0,	0 → 15 → 0,	0 → 20 → 0,
	0 → 25 → 0			
20	0 → 25 → 0,	0 → 30 → 0,	0 → 35 → 0,	
27	0 → 35 → 0,	0 → 40 → 0,	0 → 45 → 0,	0 → 50 → 0
47.5	0 → 50 → 0,	0 → 55 → 0,	0 → 60 → 0,	0 → 65 → 0,
	0 → 70 → 0,	0 → 75 → 0,	0 → 80 → 0,	0 → 85 → 0,
	0 → 90 → 0,	0 → 95 → 0,	0 → 100 → 0	

3.8.2 Naming Convention

A naming convention was chosen when selecting experiment DAQ file names. This enables instant identification of an experiment by looking at its file name. The filename structure is as follows: *Name_A_B_C_D_E.dac*. The filename convention reveals the following:

Table 3.5. *Filename Convention Information.*

Component	Name	Description
Name	Experiment name	Could be one of the following tests: Manifold Blow Down (MANBD), NICS Unit Step (NICSUS) or NICS Multi Step (NICSMS)
A	Experiment number	The experiment number from 000 to 999. *A* is often omitted
B	Initial pressure	Pressure range from 200 to 400 kPa with 50 kPa increments
C	Orifice diameter (*d*)	$14 \leq d \leq 47.5$ (mm)
D	Maximum valve opening (*h*)	$0 \leq h \leq 100$ (%)
E	Experiment set number	Experiment set number from 00 to 99
dac	DAQ file name extension	Default extension (**d**ata **ac**quisition)

3.8.3 Orifice Ranges

When conducting the mentioned experiments it should be noted that the differential pressure transducer installed across the orifice only has a range of +/-500 mbar (+/-50 kPa). The installed differential pressure transmitter will exceed its maximum limit e.g. given a 14 mm orifice, 350 kPa upstream pressure with the valve stroked to a 100 %. Exceeding the differential pressure transducer limits can permanently damage the device. To make sure the differential pressure is within range an orifice with a larger inner diameter (d) should be installed e.g. d = 47.5 mm. The orifice ranges are given in Table 3.6.

Table 3.6. *Approximate Orifice Ranges*.

Upstream Pressure Range (kPa)	Orifice Size (mm)	Valve Stroke Range (%)
200 to 400	14	$0 \leq h \leq 25$
200 to 400	20	$25 \leq h \leq 35$
200 to 400	27	$35 \leq h \leq 50$
200 to 400	47.5	$50 \leq h \leq 100$

3.9 SUMMARY

This chapter covered all of the aspects involved when acquiring physical data from the pipe network for the development of a dynamic mathematical model of a control valve. It is necessary to have a good understanding of all these aspects in order to make good and informed decisions concerning design variables.

Two DAQ systems, commonly used in industry, were introduced from which a hybrid DAQ system was developed. It incorporated components of both solutions. The pipe network setup was discussed comprising sections such as; the gas supply system, the mass flow calculation (using an orifice) and the control valve. Hardware signal converters were used to convert the industry standard current signals into voltage signals.

The developed applications were also covered which included the DAQ process control GUI, used to acquire the real-world data and the post-processing applications used to respectively calculate the values for the governing equations and graph the raw and processed data.

Experimental procedures were introduced with the aim of fully characterising the behaviour of the control valve. The governing equations and original transducer data will be used to derive the dynamic mathematical model.

CHAPTER 4

Fuzzy Logic Model Parameters

Five data sets, each consisting of twenty-four (24) data files which in turn average about one-thousand (1000) data samples, are used to describe the behaviour of the control valve. This implies that about eighty-four thousand (84000) data samples have to be processed if seventy percent (70 %) of the data files are used as the training set. An additional thirty-six thousand (36000) data samples (the remaining 30 %) have to be validated and compared in order to find the effective fuzzy logic system (FLS).

This chapter explores the possibility of finding an optimal fuzzy logic model based on only a single data set. About seventeen thousand (17000), as opposed to eighty-four thousand (84000), data samples have to be processed to find an estimation of an optimal FLS. About seven thousand (7000), as opposed to thirty-six thousand (36000) data samples have to be validated and compared. This is a significant reduction in data samples and therefore considerably reduces the training, verification and validation times.

Once the optimal FLS has been derived, all available (five) data sets are used to derive the final FLS based on the optimal FLS parameters. These results will be explored and compared in *Chapter 5*.

4.1 PROCEDURE FOR OPTIMAL PARAMETERS

Finding optimal parameters for a FLS can be a daunting task. Unfortunately no predefined method can be used to obtain these parameters. It all depends on the system, the application and the modeller's intuition. There is no right or wrong method although it is highly recommended that some procedure be used which can, to a certain extent, be justified with mathematics and common sense.

The following procedure is used:

- Use a single data set e.g. with initial pressure starting at 300 kPa [4]
- Determine a random file sequence and use it as the basis for training, verifying and validating

[4] It is recommended to use this data set because it in the middle of the data sets (200, 250, *300*, 350 and 400) and could result in a good estimate of the behaviour of the control valve at the other initial pressures.

- Train the model identification FLS with numerous parameters and repeat the process until the optimal parameters have been found. Use the following procedure for the optimal parameters:
  - Use various maximum time delay factors (MDF) e.g. 100, 80, 60, 40, 20 and 0. Each time delay factor defines a unique section comprising a number of settings for radius (r) and sigma (σ) e.g. 5, 3, 1, 0.5, 0.3 and 0.1
  - Within each section, defined by the unique MDF, determine the optimal settings for r and σ
  - Compare the optimal r-s and σ-s with each other and determine the optimal time delay factors
  - Once the optimal time delay factors and its r and σ settings have been selected, determine the impact of each input signal on the model identification FLS
    - If any of the input signals do not have a significant impact on the FLS, remove all its time delayed versions except for the original input signal
  - Determine the effect when all the input signals, which do not have a significant impact on the FLS, have been removed
  - If need be, gradually add time delayed versions of these input signals to enhance the system performance
- Use the parameters from the optimal solution to derive a fuzzy logic model for the entire system (*Chapter 5*)
  - Repeat this process a number of times each time with a unique random file sequence to determine the effectiveness of the fuzzy logic model
  - Compare the fuzzy models

From the outlined procedure it is evident that four key variables are introduced which need to be optimally selected in order to effectively derive the fuzzy control valve model. The four variables are *input time delays*, *maximum time delay factor* (MDF) and the fuzzy parameters radius (r) and sigma (σ). Another fixed variable is the *time delay increments*.

To capture the dynamic behaviour of the control valve a number of time delays for each channel need to be added to the fuzzy system. The number of time delays depends on the impact the specific input signal has on the fuzzy system. Each fuzzy input signal has its own related number of time delayed inputs and does not necessarily depend on another input signal.

Maximum time delay is a factor describing the upper limit of the time delays used as inputs to the FLS. This parameter initially remains equal for all the fuzzy input signals but is eventually optimised depending on the impact the input signals have on the fuzzy system.

Optimising the input signals involves removing some of the time delayed inputs, thus making the fuzzy system less complex. The maximum time delay parameter ranges from 0 to 100 in increments of 20 whereas the time delay increment parameter is fixed at 5 samples.

The fuzzy parameters r and σ are optimally selected by keeping them equal [5] thus effectively eliminating excessive possibilities when finding individual optimal values. During the investigation of the optimal delay factors, different values for r and σ ($r = \sigma$) are also investigated. Typical values include: 5, 3, 1, 0.5, 0.3 and 0.1.

4.2 PERFORMANCE MEASUREMENT

Finding the optimal parameters for the FLS some form of performance measurement criterion needs to be defined. In this dissertation four such criteria are used which are as follows: maximum error amplitude (MEA), mean square error (MSE), root mean square error (RMSE) and coefficient of variation of the error residuals (CVRE) [22].

The MEA defines the maximum error within a data file whereas the CVRE defines the variability of the errors on a relative, unit-less basis [22]. The MSE and RMSE are classical error criteria often used as performance measures for neural networks and fuzzy logic systems.

The MEA is defined as follows:

$$MEA = \frac{\max(|\overline{x} - \overline{y}|)}{\max(\overline{y})} \tag{4.1}$$

where x is the desired (target) output vector and y the predicted (simulated) output vector.

The MSE is defined as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{4.2}$$

where n is the amount of data samples, x_i is the desired output value and y_i is the predicted output value.

The RMSE is defined as follows:

$$RMSE = \frac{\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}}{\sqrt{\sum_{i=1}^{n}x_i^2}} \tag{4.3}$$

where n, x_i and y_i have the same meaning as defined above.

The CVRE is defined as follows:

$$CVRE = \frac{1}{\bar{x}}\sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}} \qquad (4.4)$$

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

where n, x_i and y_i have the same meaning as defined above.

4.3 OPTIMAL PARAMETER SELECTION

Following the outlined procedure stipulated in section 4.1, when using the initial pressure set of 300 kPa and a randomly generated file sequence (24, 8, 21, 15, 2, 22, 16, 19, 10, 7, 4, 3, 11, 14, 6, 12, 23, 9, 5, 20, 18, 13, 17 and 1) the following results are obtained. It is generally accepted that seventy percent (70 %) of all data, be used for the training set and the remaining thirty percent (30 %), be used as the validation data set.

4.3.1 Finding the Optimal Delay Factors

Finding the optimal delay factors involves six sections, each of which investigates the impact the maximum delay factor has on the system. During the investigation of the optimal delay factors all input signals have the same upper limit time delay values specified by the maximum time delay factor (MDF). These sections have maximum time delay factors of 100, 80, 60, 40, 20 and 0.

For all five fuzzy input parameters, each with maximum time delay factor of 100 and time delay increments fixed at 5 samples, 20 additional time delayed input signals are generated (100/5 = 20). One fuzzy input signal thus results in a total of 21 input signals comprising of 1 fuzzy input signal and 20 time delayed versions thereof ranging from 5 to 100 (the MDF). The five input parameters therefore altogether result in 105 input signals. Adding feedback to the system by means of time delayed versions of the output parameter, results in a fuzzy system with a total of 120 input signals and 1 output parameter. This procedure is repeated for all the maximum time delay factor sections. A summary of the fuzzy input signals can be seen in Table 4.1 with a fixed time delay increment of 5 samples, a fixed number of fuzzy input parameters (5) and a single fuzzy output parameter (Massflow).

Table 4.1. *Breakdown of Fuzzy Input Signals.*

Maximum Time Delay Factor	No. of Time Delayed Fuzzy Inputs	Subtotal Fuzzy Inputs per Input Parameter	Subtotal of Input Signals	No. of Output Time Delays	Total No. of Fuzzy Input Signals
100	20	21	105	20	125
80	16	17	85	16	101
60	12	13	65	12	77
40	8	9	45	8	53
20	4	5	25	4	29
0	0	1	5	0	5

For each delay factor section a number of experiments are conducted with r and σ ranging from 0.1 to 5. Each of these experiments results in a FL mathematical model. Based on the derived FL model the mentioned four error criteria are calculated for each of the remaining 30 % data files, which have not been encountered before by the FLS. Table 4.2 shows the four error criteria for each of the six data files as well as the model's average error values. These average errors represent the effectiveness of the derived model.

Table 4.2. *Model Validation Criteria Errors and Average Errors.*

Filename	File No.	MAE	MSE	RMSE	CVRE
NICSUS_300_20p0_035_00.pr3	9	0.148653	0.000756	0.044150	0.059745
NICSUS_300_14p0_020_00.pr3	5	0.137508	0.001465	0.061664	0.082797
NICSUS_300_47p5_080_00.pr3	20	0.067061	0.000503	0.039847	0.056876
NICSUS_300_47p5_070_00.pr3	18	0.096004	0.000735	0.049364	0.071276
NICSUS_300_27p0_050_00.pr3	13	0.092868	0.000481	0.036309	0.050202
NICSUS_300_47p5_065_00.pr3	17	0.110338	0.001287	0.060080	0.081304
NICSUS_300_14p0_000_00.pr3	1	0.799832	0.092546	0.643462	0.714405
Average Errors		**0.207466**	**0.013968**	**0.133554**	**0.159515**

The MDF for this example is 40 samples with time delay increments of 5 samples and an r and σ of 0.1 which results in 13295 clusters being created.

The error averages for each of these fuzzy models as portrayed in Table 4.2 are graphed and compared to find the optimal parameters.

4.3.1.1 Maximum Delay Factors, r and σ

The average errors for a maximum delay factor of 100 and various r and σ settings can be seen in Table 4.3. The graphic representation of the results can be seen in Figure 4.1.

Table 4.3. *Average Errors for Various r-s and σ-s with MDF 100.*

r and σ	MAE	MSE	RMSE	CVRE	Average
5	0.425366	0.048712	0.383109	0.509775	0.341740
3	0.342472	0.016577	0.209641	0.269058	0.209437
1	0.230891	0.011670	0.137252	0.167717	0.136882
0.5	**0.212006**	**0.010923**	**0.118787**	**0.142371**	**0.121022**
0.3	0.210980	0.013458	0.122833	0.145152	0.123106
0.1	0.246862	0.030182	0.170732	0.198607	0.161596

The minimum value in the average column of Table 4.3 specifies the optimum settings for *r* and σ for this specific maximum delay factor. The average values are graphed in Figure 4.1 where it can be seen that the four error criteria reach a minimum if *r* and σ are both set to 0.5.

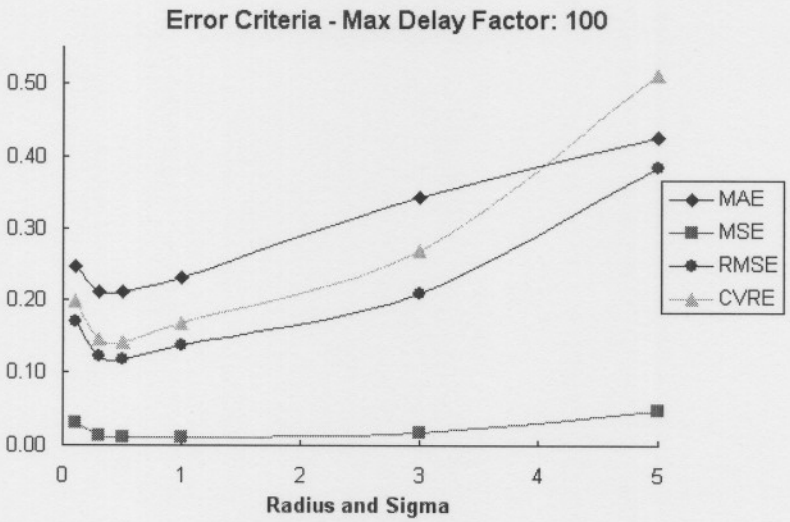


Figure 4.1. *MDF 100.*

Repeat the process for the remaining maximum delay factors i.e. 80, 60, 40, 20 and 0 as stipulated above to obtain the following results: The average errors for a MDF of 80 can be seen in Table 4.4 and its related graph seen in Figure 4.2.

Table 4.4. *Average Errors for Various r-s and σ-s with MDF 80.*

r and σ	MAE	MSE	RMSE	CVRE	Average
5	0.440780	0.061460	0.427796	0.572405	0.375610
3	0.402917	0.019518	0.235671	0.305455	0.240890
1	0.268560	0.012659	0.155539	0.193011	0.157442
0.5	0.201333	0.010545	0.124795	0.151453	0.122032
0.3	**0.195463**	**0.012532**	**0.122348**	**0.145498**	**0.118960**
0.1	0.233124	0.030178	0.170472	0.198158	0.157983

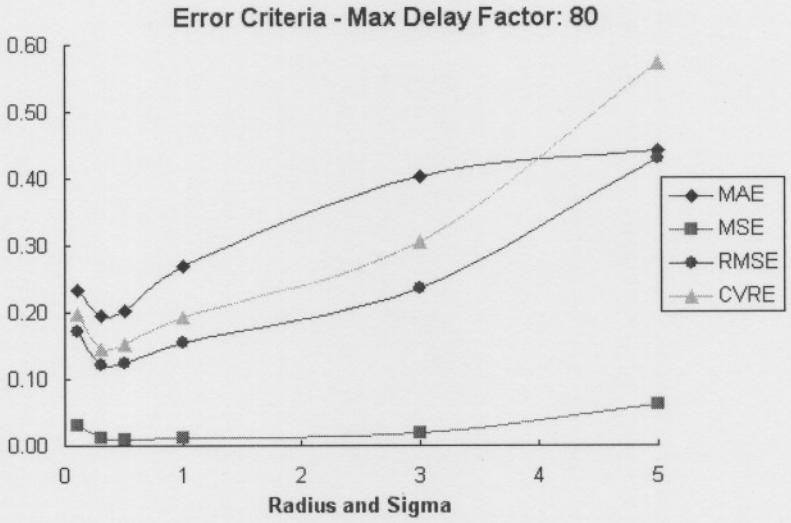


Figure 4.2. *MDF 80*.

The average errors for a MDF of 60 can be seen in Table 4.5 and its related graph in Figure 4.3.

Table 4.5. *Average Errors for Various r-s and σ-s with MDF 60*.

r and σ	MAE	MSE	RMSE	CVRE	Average
5	0.491738	0.076594	0.476772	0.640969	0.421518
3	0.466225	0.024965	0.274984	0.361289	0.281866
1	0.343453	0.015489	0.189644	0.239775	0.197090
0.5	0.242961	0.011520	0.143804	0.177635	0.143980
0.3	**0.200897**	**0.013225**	**0.133665**	**0.160910**	**0.127174**
0.1	0.209157	0.018675	0.142846	0.168072	0.134687

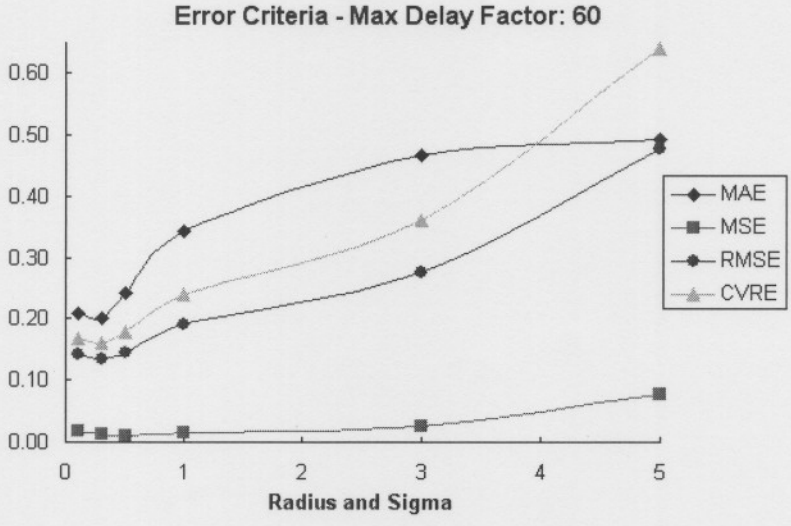


Figure 4.3. *MDF 60*.

The average errors for a MDF of 40 can be seen in Table 4.6 and its related graph in Figure 4.4.

Table 4.6. *Average Errors for Various r-s and σ-s with MDF 40.*

r and σ	MAE	MSE	RMSE	CVRE	Average
5	0.552120	0.100468	0.541494	0.731135	0.481304
3	0.524077	0.037682	0.340370	0.451990	0.338529
1	0.449978	0.020779	0.236898	0.304723	0.253094
0.5	0.320677	0.013259	0.175896	0.222606	0.183109
0.3	0.248210	0.013367	0.153220	0.189137	0.150984
0.1	**0.207466**	**0.013968**	**0.133554**	**0.159515**	**0.128626**

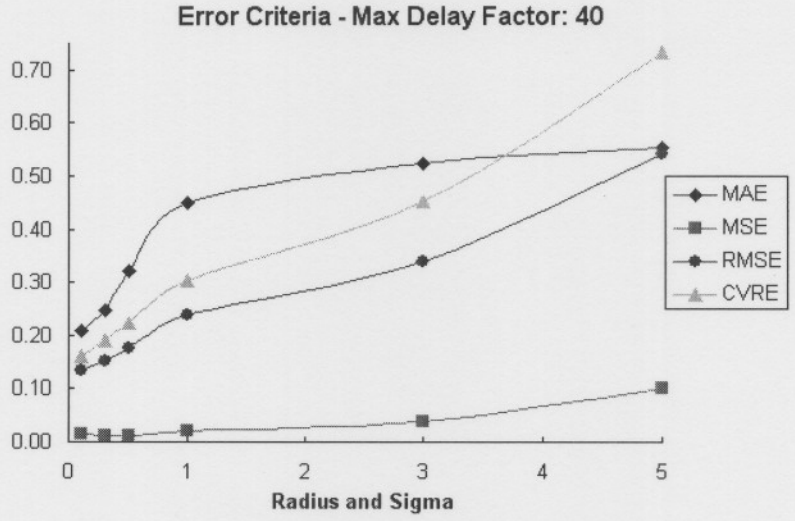


Figure 4.4. *MDF 40.*

The average errors for a MDF of 20 can be seen in Table 4.7 and its related graph in Figure 4.5.

Table 4.7. *Average Errors for Various r-s and σ-s with MDF 20.*

r and σ	MAE	MSE	RMSE	CVRE	Average
5	0.632658	0.153780	0.664072	0.901799	0.588077
3	0.581107	0.087404	0.508183	0.685369	0.465516
1	0.631409	0.031129	0.307051	0.403409	0.343249
0.5	0.491369	0.020480	0.242939	0.315738	0.267632
0.3	0.405505	0.016987	0.212578	0.273159	0.227057
0.1	**0.293670**	**0.015563**	**0.166987**	**0.206038**	**0.170565**

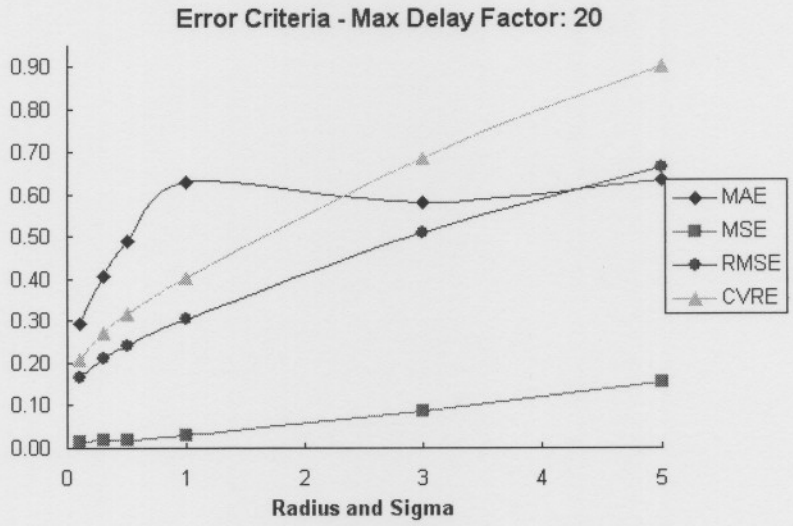


Figure 4.5. *MDF 20*.

The average errors for a MDF of 0 can be seen in Table 4.8 and its related graph in Figure 4.6.

Table 4.8. *Average Errors for Various r-s and σ-s with MDF 0*.

r and σ	MAE	MSE	RMSE	CVRE	Average
5	0.632658	0.153780	0.664072	0.901799	0.588077
3	0.632658	0.153780	0.664072	0.901799	0.588077
1	0.667991	0.070853	0.461075	0.619965	0.454971
0.5	0.731359	0.049578	0.390849	0.518920	0.422676
0.3	0.730136	0.045512	0.371891	0.489693	0.409308
0.1	**0.727862**	**0.041266**	**0.345550**	**0.450173**	**0.391213**

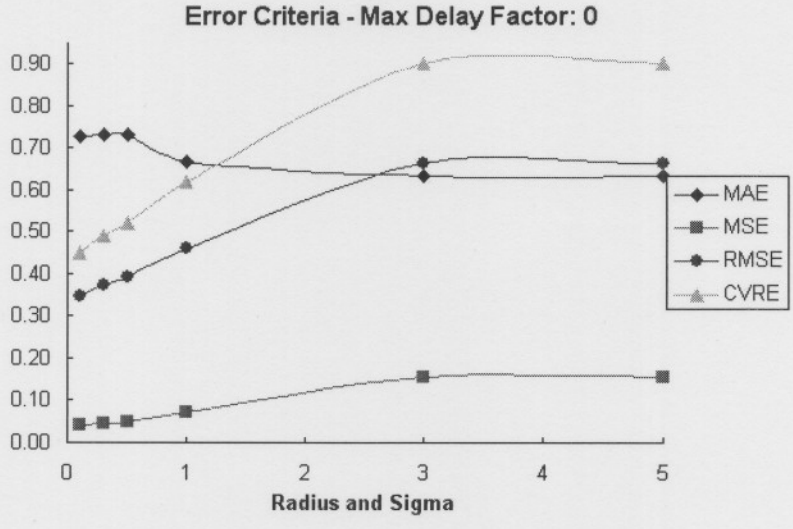


Figure 4.6. *MDF 0*.

From the first entry of Table 4.7 and the first two entries of Table 4.8 it is evident that only one cluster was created during the training process. This is evident from the error criteria, which are calculated and remain the same for all three cases and therefore constitute the upper limit of the fuzzy r and σ parameters. Smaller settings for r and σ should therefore be selected.

In order to find the most effective MDF and fuzzy r and σ settings, the smallest error criteria averages are compared with one another to find the optimal results. This can be seen in Figure 4.7 where each trend represents a MDF setting and its minimum r and σ settings.

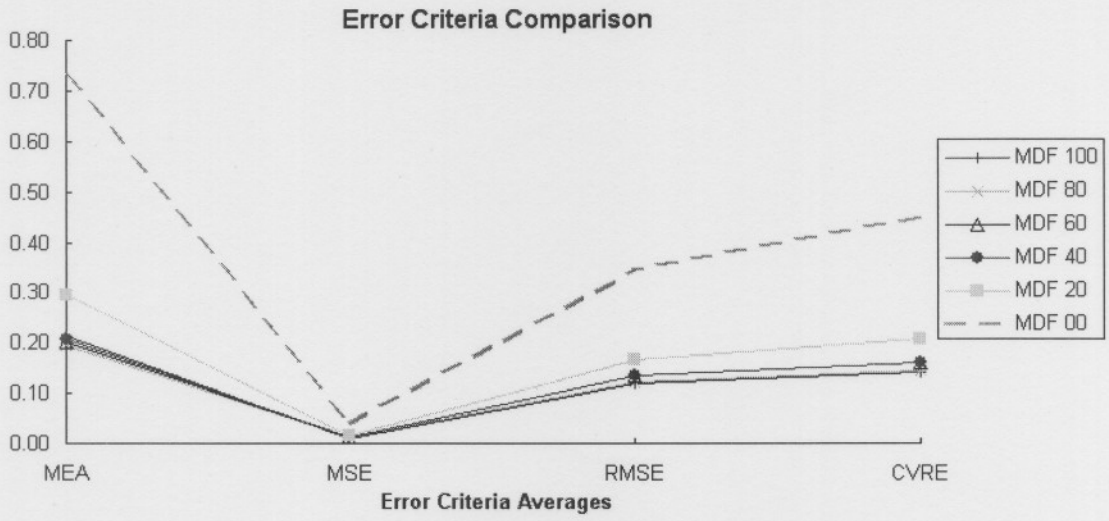


Figure 4.7. *Comparison of Minimum Error Criteria*.

MDF 0 does not predict the behaviour of the control valve effectively due to system dynamics not being incorporated and therefore may be used as a benchmark to compare the other MDFs. The comparison can be seen in Figure 4.8.

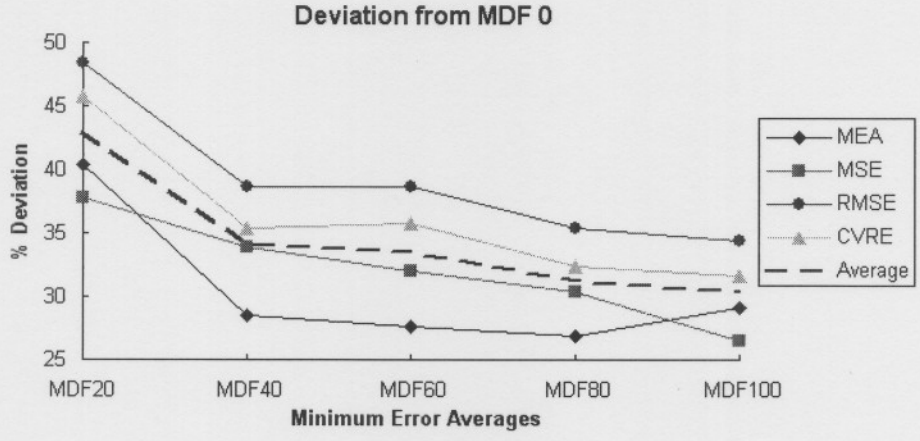


Figure 4.8. *Comparison of MDF 20 to MDF 100 with MDF 0.*

Moving from MDF 20 to MDF 40 a significant decrease in the deviation can be seen whereas from MDF 40 to MDF 60, MDF 60 to MDF 80 and MDF 80 to MDF 100 only a slight decrease is noticeable. The average values of the four error criteria also show a downward trend.

The most effective MDF setting seems to be MDF 100 although MDF 40 already produces sufficient results. Using MDF 100 will result in a much more complex system than that of MDF 40 due to a vast number of fuzzy inputs (125) to predict a single output. Therefore all calculations and common sense point to MDF 40 which produces good results with the lease number of fuzzy inputs (less than 53).

4.3.1.2 Optimal Time Delays

From the minimum average values of MDF 40 it can be seen that the fuzzy parameters r and σ should be set to 0.1. This results in a fuzzy system with at most 53 input signals from which some can be eliminated depending on their influence on the fuzzy system. Once each input signal's influence order has been determined, the time delayed versions of the input signals with the least impact on the fuzzy system are reduced. This results in a fuzzy system with less than 53 input signals and therefore reduces the complexity of the system without significantly reducing the accuracy of the fuzzy model.

Using the optimal settings of MDF 40, the impact of each channel on the fuzzy system can be compared and its influence order be established. The deviation from the optimal settings of MDF 40 for each channel is summarised in Table 4.9.

Table 4.9. *Deviation from MDF 40.*

Input Signal	Ch No.	MEA	MSE	RMSE	CVRE	Influence Order
Original	-	**0.207466**	**0.013968**	**0.133554**	**0.159515**	-
TT560A	4	0.100543	0.003272	0.032325	0.044155	2
PT505	5	0.007312	0.000749	0.000359	0.000553	5
PT506	6	0.023234	0.000166	0.001246	0.001608	4
VLVIN	7	0.000145	0.000322	0.001168	0.001308	6
VLVOUT	8	0.012927	0.002987	0.011837	0.013740	3
MassFlow	9	0.306738	0.010118	0.106161	0.144964	1

The input signal deviations are graphed in Figure 4.9. From this figure it is evident that the input time delayed versions of the Massflow output signal has the most significant impact on the fuzzy system whereas the valve input command (VLVIN) has the least significant impact. The influence order of each input/output signal is also portrayed in Table 4.9.

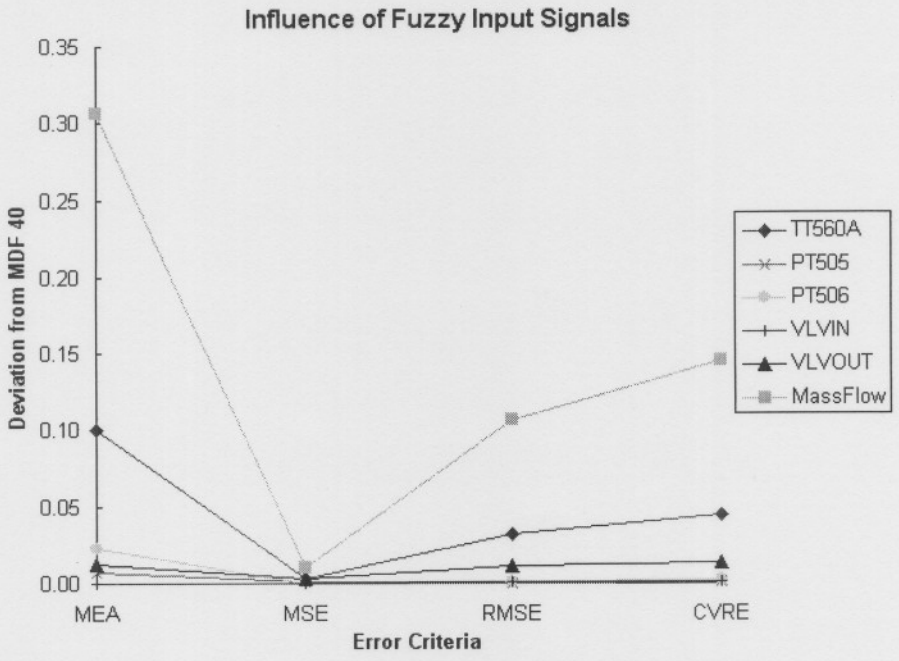


Figure 4.9. *Influence Order and Deviation from MDF 40.*

The upstream and downstream static pressures and the valve input command do not have a significant impact on the fuzzy system, as portrayed in Figure 4.9, and their time delayed versions can therefore be reduced. The number of possibilities can be reduced by keeping all time delayed versions of these input signals equal. Four experiments are conducted; the first experiment uses no time delayed versions of the mentioned three signals, the second, third and fourth uses 1, 2 and 3 time delayed versions respectively each with a fixed time delay increment of 5 samples. The impact of the least significant fuzzy inputs, according to the average error criteria, can be seen in Table 4.10.

Table 4.10. *Impact of Least Significant Fuzzy Inputs.*

Max Time Delay Factor	No. of Time Delays	MEA	MSE	RMSE	CVRE
0	0	0.027466	0.000044	0.000843	0.001195
5	1	0.058384	0.000747	0.003495	0.004177
10	2	0.010753	0.00053	0.001812	0.002033
15	3	0.000256	0.000247	0.001109	0.001283

The average error values in Table 4.10 represent the deviation from the MDF 40 settings and give an understanding of the impact of these input signals on the fuzzy system.

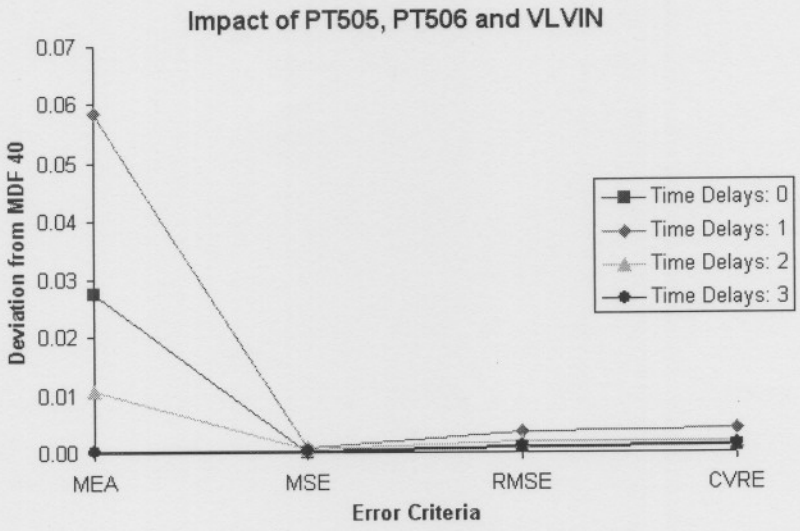


Figure 4.10. *Impact of Least Significant Fuzzy Input Signals.*

Due to the fact that a fuzzy system with good approximation, good results, capturing the dynamic valve behaviour and with a minimum number of inputs are required, the final fuzzy system settings are as follows. The fuzzy radius (r) and sigma (σ) parameters are set to 0.1, a constant time delay increment factor of 5 samples and maximum delay factors for TT506A, VLVOUT and Massflow set to 40 (8 time delayed versions per channel) whereas PT505, PT506 and VLVIN are set to 5 (1 time delayed version per channel). This implies that the final fuzzy system has 32 input signals consisting of 5 input signals and 27 time delayed versions thereof.

4.4 SUMMARY

This chapter highlighted a procedure which can be used to find the optimal settings for the fuzzy logic system. The impact of each input signal and its time delayed versions were also investigated. Four performance measurement criteria were introduced which were used as a benchmark throughout the chapter.

The final settings will be used in *Chapter 5* to derive the final fuzzy mathematical models for the control valve.

CHAPTER 5

Fuzzy Control Valve Model Results

Chapter 4 was dedicated to finding the optimal fuzzy logic parameters based on a single data set. These parameters are now applied to all the data consisting of five data sets with initial pressures ranging from 200 to 400 kPa. In this chapter, these parameters are used to derive a complete fuzzy logic mathematical model for the control valve.

The parameters obtained in *Chapter 4* should first be re-evaluated in order to ensure proper training, verification, validation and successful estimation of fuzzy parameters using all the available data sets. This is done by deriving various models each with a slight variation in a specific fuzzy parameter and establishing the effect of the parameter on the fuzzy system. A summary of the various models and their slight parameter deviations can be seen in Table 5.1. The variations in Table 5.1 are marked in bold. After the evaluation of the fuzzy parameters, various models are derived using the final optimal fuzzy parameters.

Table 5.1. *Various Models with Parameter Deviations*.

Model Name	Sequence No.	Time Delay Range	Fuzzy Parameters: r and σ
CVMod00	1	**0 – 40**	0.1
CVMod01	1	**15 – 40**	0.1
CVMod02	1	5 – 40	**0.3**
CVMod03	1	5 – 40	0.1
CVMod04	1	5 – 40	**0.08**
CVMod05	2	5 – 40	0.1
CVMod06	3	5 – 40	0.1
CVMod07	4	5 – 40	0.1
CVMod08	5	5 – 40	0.1

The first two models (CVMod00 and CVMod01) investigate the impact of the time delay range on the fuzzy system. The following three models (CVMod02, CVMod03 and CVMod04) investigate the effect of the fuzzy parameters radius (r) and sigma (σ). Due to the fact that the mentioned fuzzy control valve models make use of the same random file sequence (evident from the *Sequence No.* column in Table 5.1), the results can be compared and the final fuzzy parameters be verified. Once this is done four additional models (CVMod05, CVMod06, CVMod07 and CVMod08) are derived each with a unique random file sequence. These models will then be compared.

5.1 EVALUATION OF FUZZY PARAMETERS

Using the first two models in Table 5.1, the impact of a variation in the time delay range is investigated. This variation can be seen in Figure 5.1 (for the error criteria: MEA and MSE) and Figure 5.2 (for the error criteria: RMSE and CVRE).

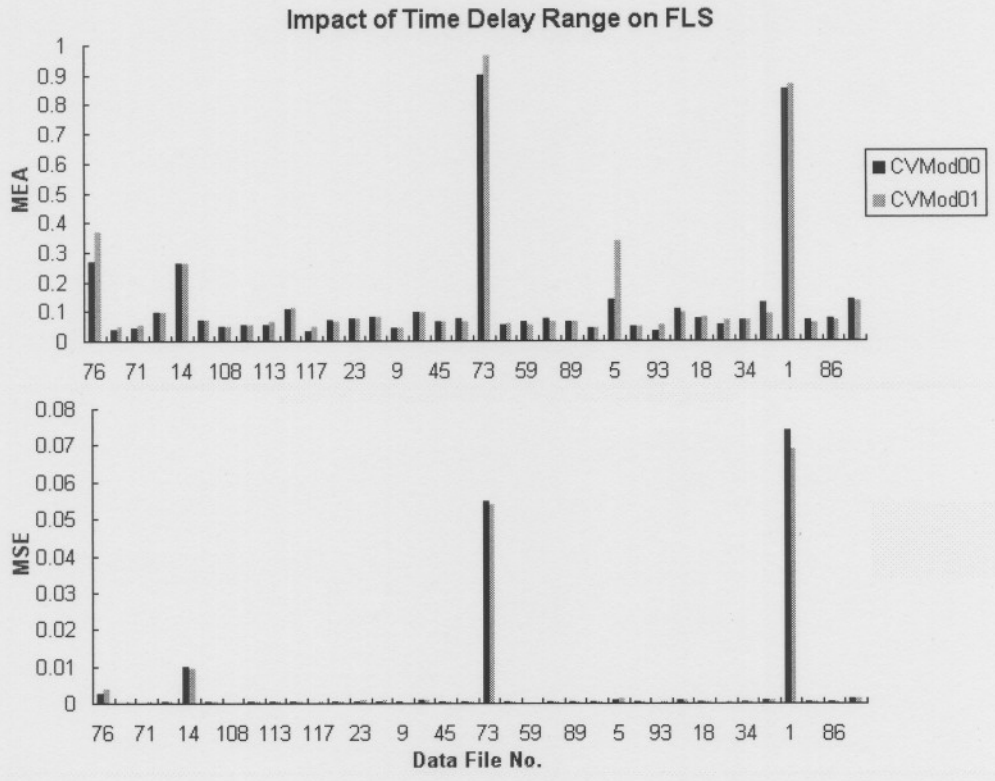


Figure 5.1. *Impact of the Time Delay Range on FLS – MEA and MSE.*

Very slight differences are noticeable when adding time delayed versions of the least significant fuzzy input signals. Therefore only one time delayed version for each of these input signals is added as an additional input and has a fixed time delay increment of 5 samples as mentioned in *Chapter 4*.

It has been verified that the least significant fuzzy input signals (PT505, PT506 and VLVIN) should each have a single time delayed version, at the 5-th sample. It has also been verified that the most significant fuzzy input signals (TT560A, VLVOUT and Massflow) should each have a time delayed version at samples 5, 10, 15, 20, 25, 30, 35 and 40. Now that the optimal number of time delays for each channel have been verified the impact of the fuzzy parameters r and σ should be verified. These parameters are verified with the control valve models CVMod02, CVMod03 and CVMod04 and the results can be seen in Figure 5.3 (for the error criteria: MEA and MSE) and Figure 5.4 (for the error criteria: RMSE and CVRE).

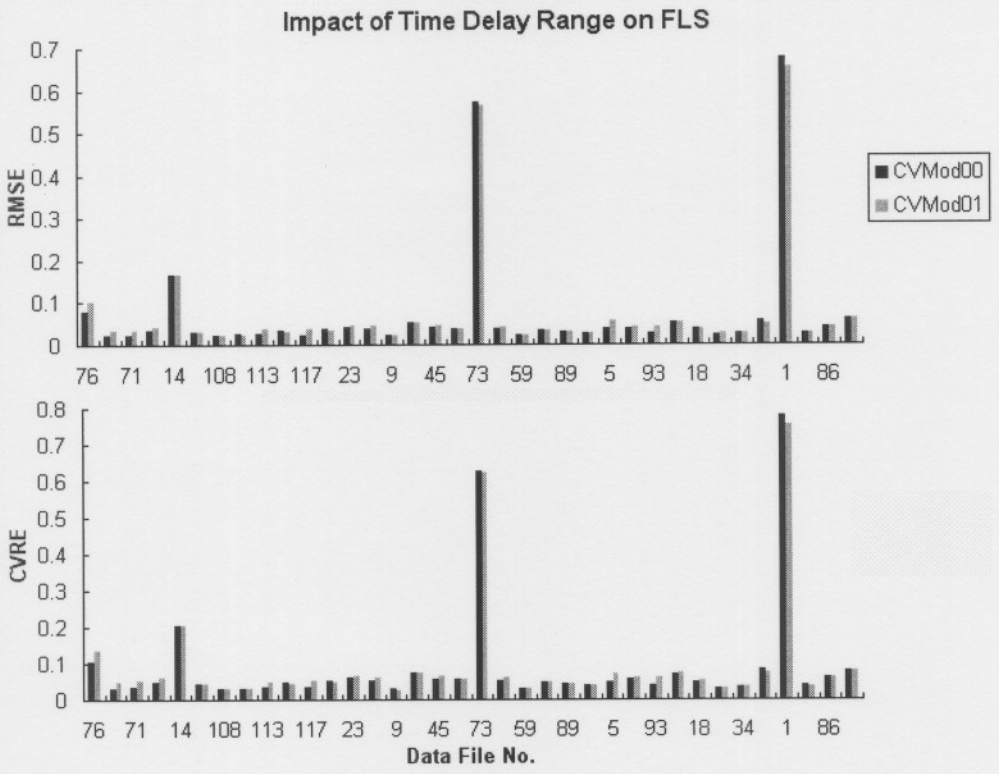


Figure 5.2. *Impact of the Time Delay Range on FLS – RMSE and CVRE.*

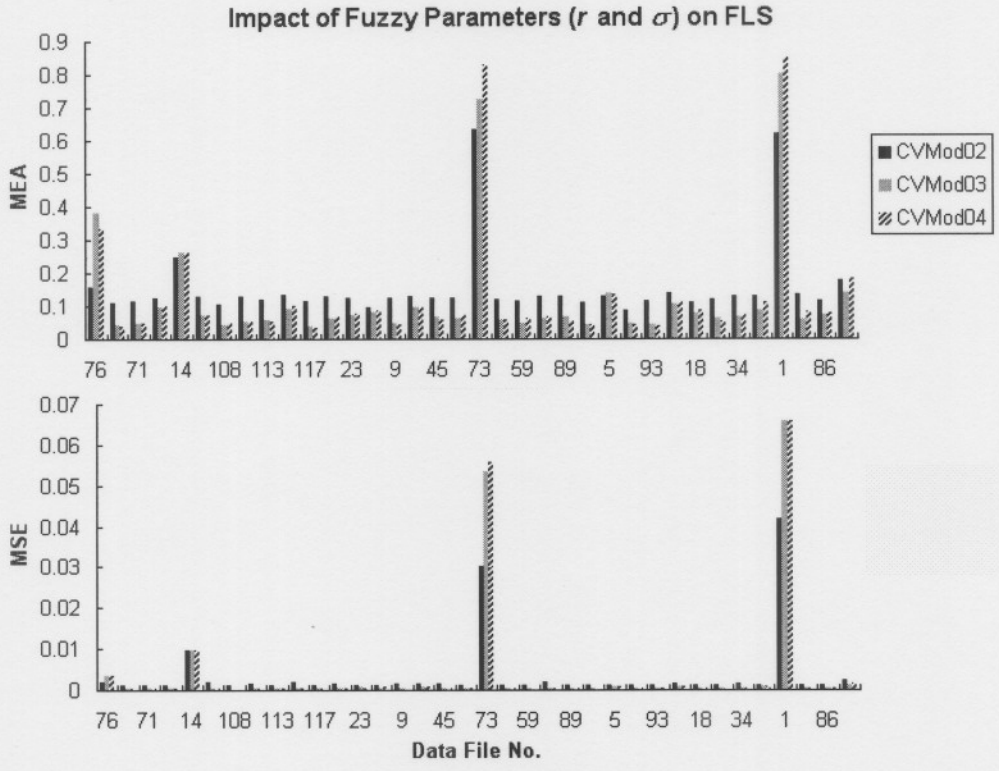


Figure 5.3. *Fuzzy Parameters r and σ Evaluation Results – MAE and MSE.*

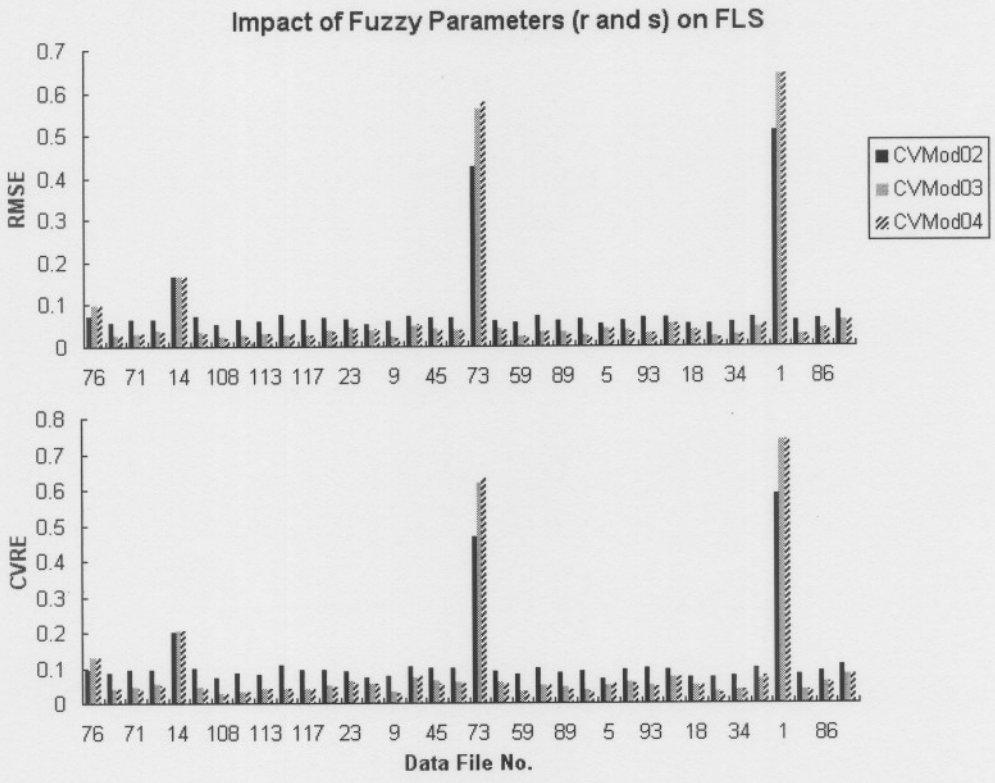


Figure 5.4. *Fuzzy Parameters r and σ Evaluation Results – RMSE and CVRE.*

From Figure 5.3 and Figure 5.4 it is evident that CVMod04, with a r and σ of 0.08, produces better results than the other two derived models (CVMod02 and CVMod03). Due to the fact that the derived model should have good generalising capabilities, CVMod03 with a r and σ of 0.1 is used as the final fuzzy settings for these parameters. Therefore the final fuzzy control valve model parameters as used by CVMod03 (stipulated in Table 5.1) are used to derive four additional models namely CVMod05, CVMod06, CVMod07 and CVMod08.

5.2 MODEL RESULTS

The results of the mentioned five fuzzy control valve models (CVMod03, CVMod05, CVMod06, CVMod07 and CVMod08) are discussed in this section. Possible explanations as to the fuzzy model results and their behaviour are also discussed.

The 30 % validation order of the random file sequences for each of the five models can be seen in Table 5.2. The bold numbers, within this table, indicate a file which has three or more occurrences within the five respective models. As an example; data file 76 occurs in models: CVMod03, CVMod07 and CVMod08 whereas data file 117 occurs in models: CVMod03, CVMod05, CVMod07 and CVMod08.

Table 5.2. *Control Valve Models 30 % Validation Random Order.*

No.	CVMod03	CVMod05	CVMod06	CVMod07	CVMod08
1	**76**	**117**	56	**9**	29
2	**47**	43	**37**	11	**69**
3	71	35	21	66	31
4	90	**14**	106	45	**52**
5	**14**	75	50	**62**	24
6	32	4	57	98	**47**
7	108	64	**63**	18	**91**
8	6	**63**	49	57	**94**
9	113	8	15	85	**117**
10	12	92	**58**	88	74
11	**117**	**37**	105	100	109
12	**81**	93	17	61	7
13	23	**9**	**5**	10	**9**
14	65	**67**	8	**52**	**73**
15	**9**	42	**69**	87	72
16	**63**	115	114	96	33
17	45	**94**	102	**67**	30
18	41	50	23	109	44
19	**73**	111	77	48	80
20	**37**	30	112	15	2
21	59	11	**67**	**117**	46
22	**58**	**81**	90	**47**	**76**
23	89	**73**	99	114	65
24	84	**91**	**94**	95	35
25	**5**	16	3	1	60
26	20	105	104	97	116
27	93	70	64	116	110
28	**62**	10	119	**58**	38
29	18	21	70	**76**	115
30	13	**69**	20	54	**5**
31	34	112	**91**	4	49
32	17	77	**14**	40	**81**
33	1	53	97	43	102
34	82	**62**	6	13	59
35	86	41	7	119	32
36	100	**52**	80	26	113

As mentioned, four error criteria are calculated for each data file. Using these error criteria an average error is calculated which is used as an estimate as to the effectiveness of the

fuzzy logic model on the data files. Each data file therefore has a single error value used as a benchmark.

5.2.1 Control Valve Model 03

The results for CVMod03 can be seen in Figure 5.5 *(a)*, *(b)*, *(c)* and *(d)*. From this figure, data files 1, 73, 14 and 76 produce large average errors. Data files 82, 71, 93 and 113 produce medium average errors whereas data files 108, 9, 117 and 59 produce small average errors. The data files with the large average errors do not produce adequate results and do not visually fit the desired output values accurately enough. This is evident from the CVRE error criteria graph in Figure 5.5 *(d)*, which gives an estimate of visual data fitting.

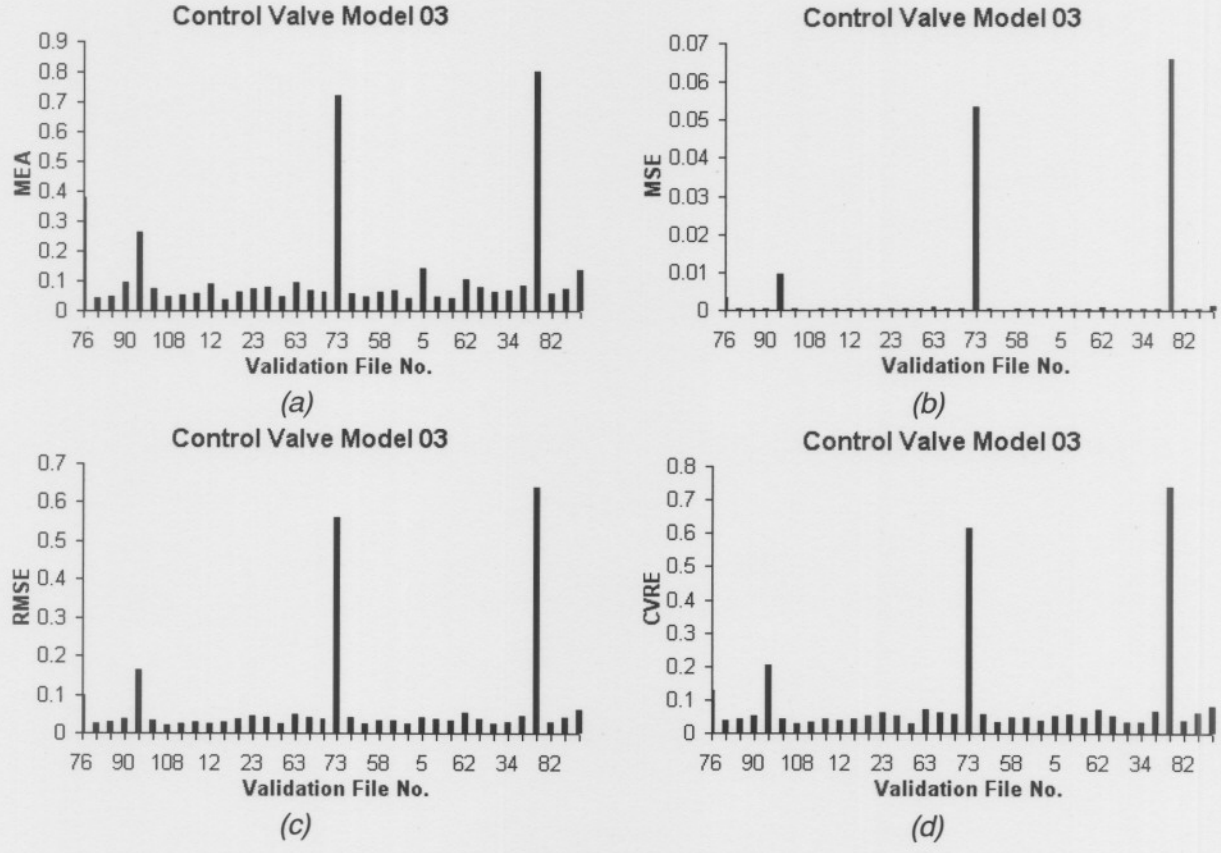


Figure 5.5. *CVMod03 Results*.

The reason for the poor performance of data files 73 and 1 is because the valve was never opened during the experiment. This is evident from their file names: NICSUS_350_14p0_**000**_00 and NICSUS_200_14p0_**000**_00 (refer to *Appendix D* for the naming convention) for data file 73 and 1 respectively. During the training, verification and validation stages, all the data are normalised between 0 and 1. The acquired data (VLVIN, VLVOUT and Massflow), consist mainly of noisy signals, and once normalised have a large variation from 0 to 1. A small deviation from the actual values causes a large deviation from the

desired value. This implies that the noise in the signal has a huge impact when the control valve is not opened (0 %) as apposed to when it is opened to e.g. 70 %. Noise therefore normally has a greater effect on the signals if the control valve is opened slightly, or not opened at all. The greater the control valve is opened, the more the effect of the noise diminishes. This can be seen in Figure 5.6 and Figure 5.7. From these figures it is evident that the average errors become smaller as the control valve is opened more.

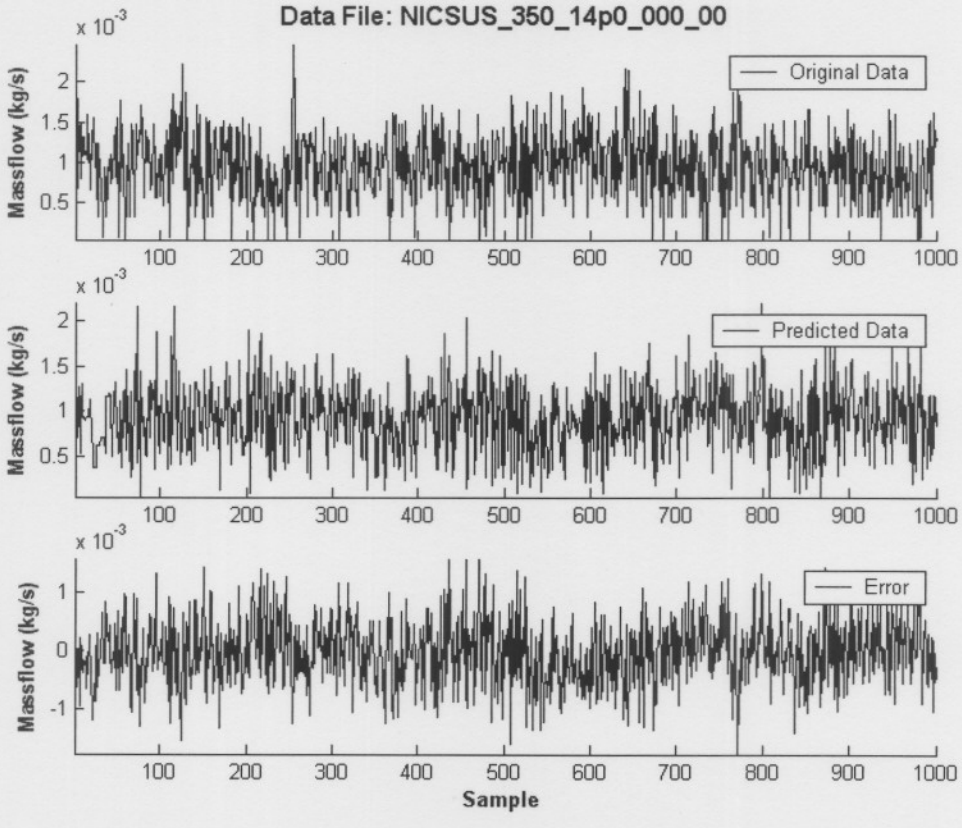


Figure 5.6. *Data File #73, Control Valve Not Opened, Large Average Error: 0.488620.*

Data file 82 in Figure 5.5 produces medium errors. A possible cause for this can be the change over from a smaller orifice (d = 20 mm) to a larger orifice (d = 27 mm). This was done not to exceed the maximum limit of the differential pressure transducer across the orifice as discussed in *Chapter 3*.

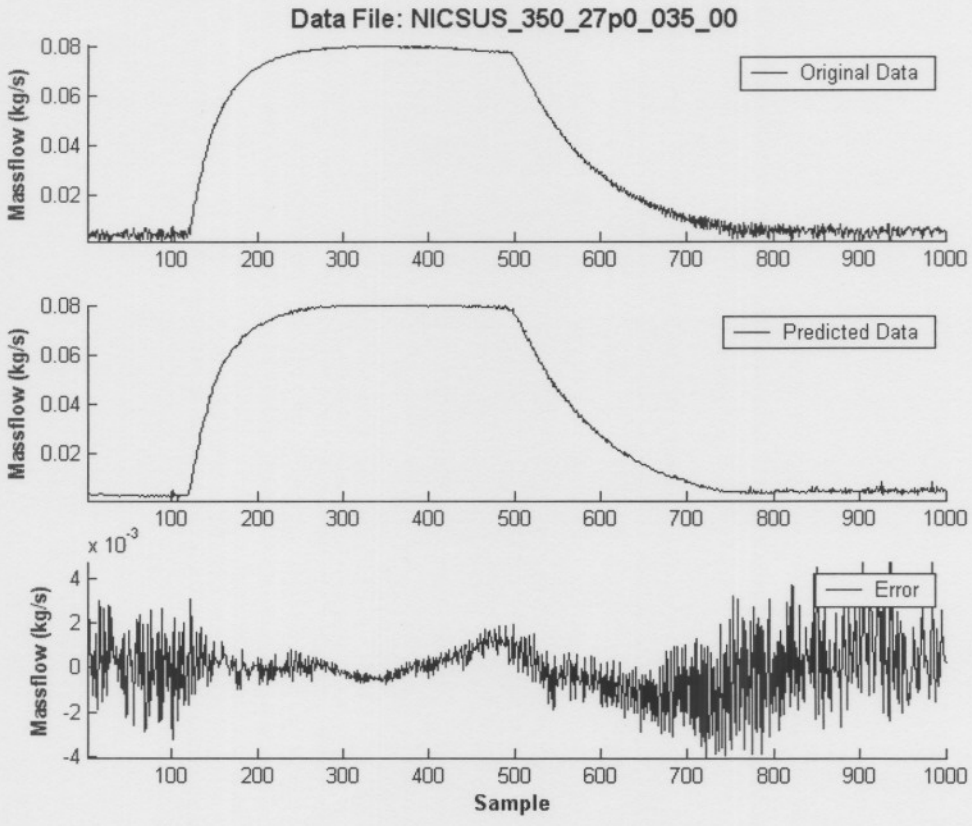


Figure 5.7. *Data File #82, Control Valve Opened 35 %, Medium Average Error: 0.03087.*

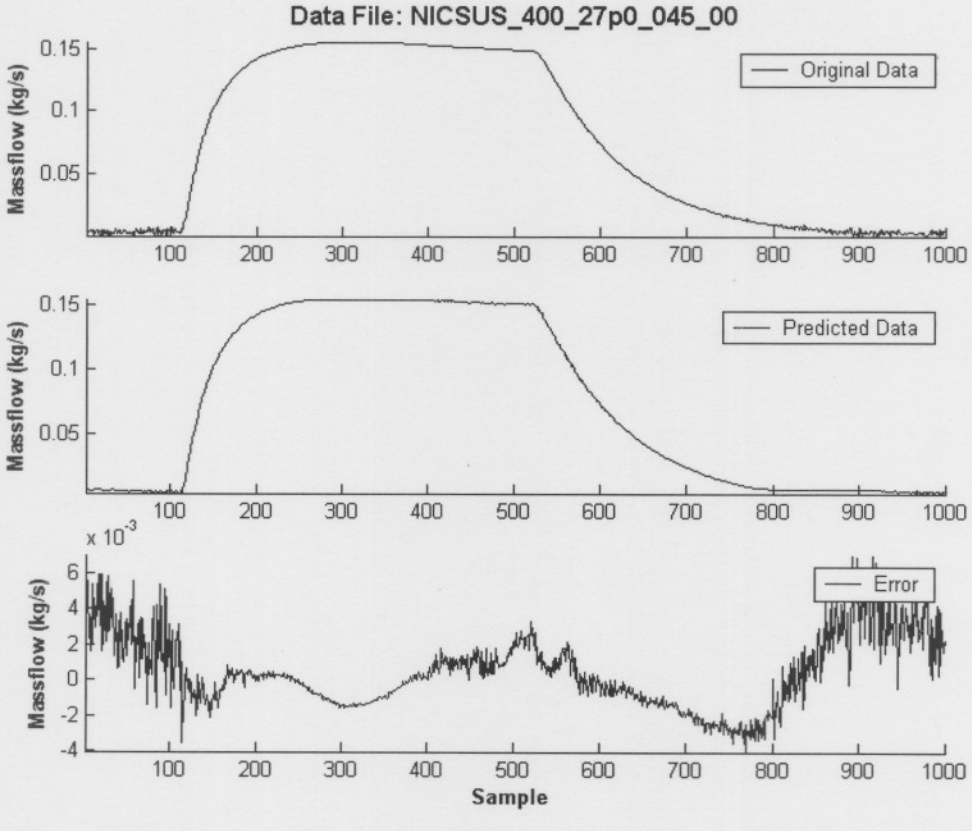


Figure 5.8. *Data File #108, Control Valve Opened 45 %, Small Average Error: 0.023745.*

Visually the predicted results in Figure 5.6 do not fit the original values very accurately as opposed to Figure 5.7 and Figure 5.8 where a certain amount of noise filtering is evident. Even though the latter two figures may seem to produce large errors their resultant average errors are acceptable.

5.2.2 Control Valve Model 05

Similar results are obtained for each of the remaining control valve models. CVMod05 produces the following results and can be seen in Figure 5.9 *(a)*, *(b)*, *(c)* and *(d)*.

From Figure 5.9, data files 50, 73, 75 and 4 produce large average errors which can again be the result of a small valve opening usually from 0 % to 15 %. Data files 42, 52 and 14 produce medium average errors and the remaining data files produce small average errors. Typical data file results can be seen in Figure 5.10 to Figure 5.12.

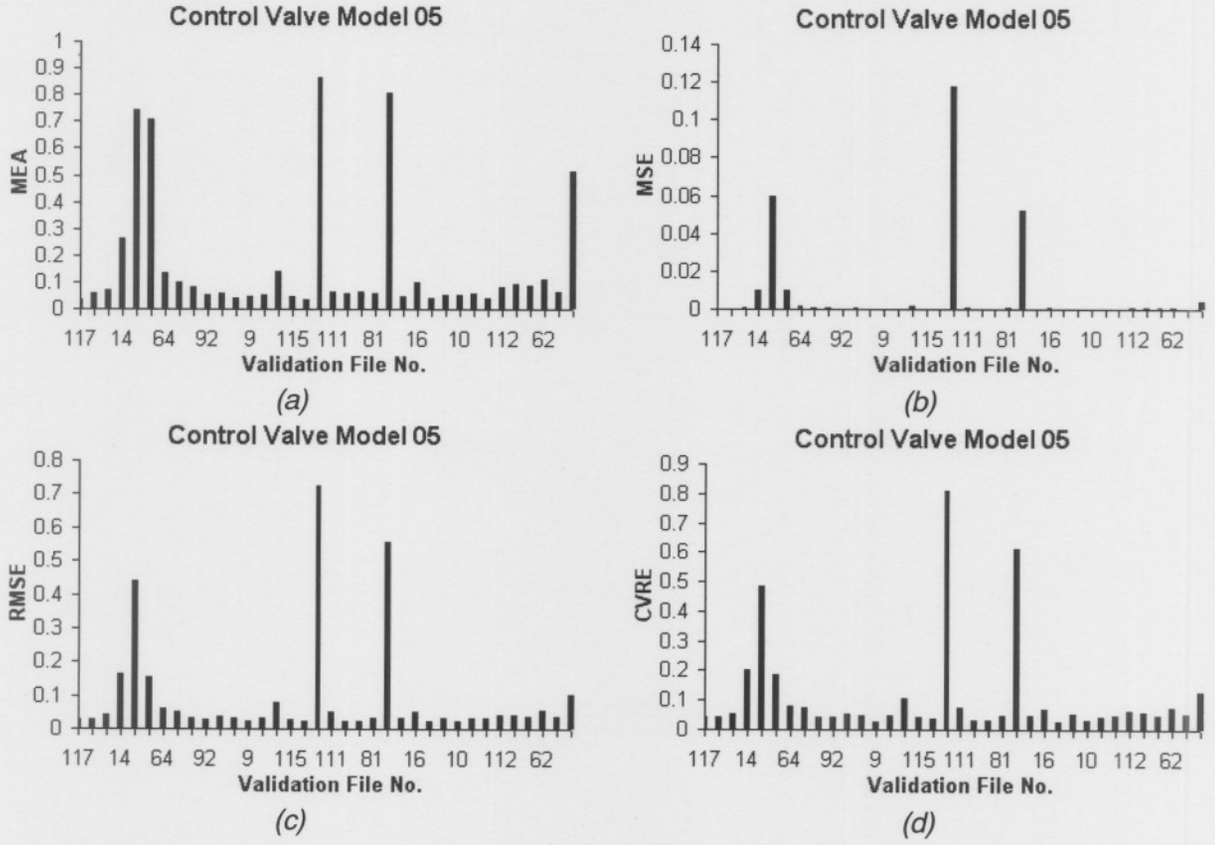


Figure 5.9. *CVMod05 Results*.

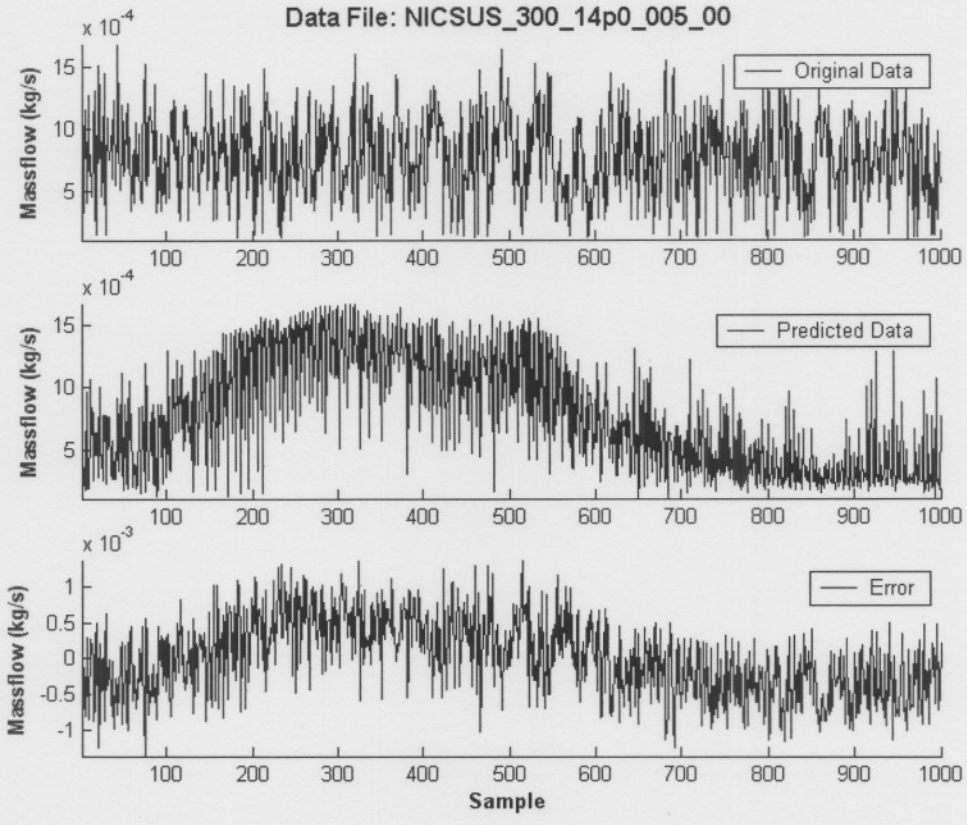


Figure 5.10. *Data File #50, Control Valve Opened 5 %, Large Average Error: 0.629498.*

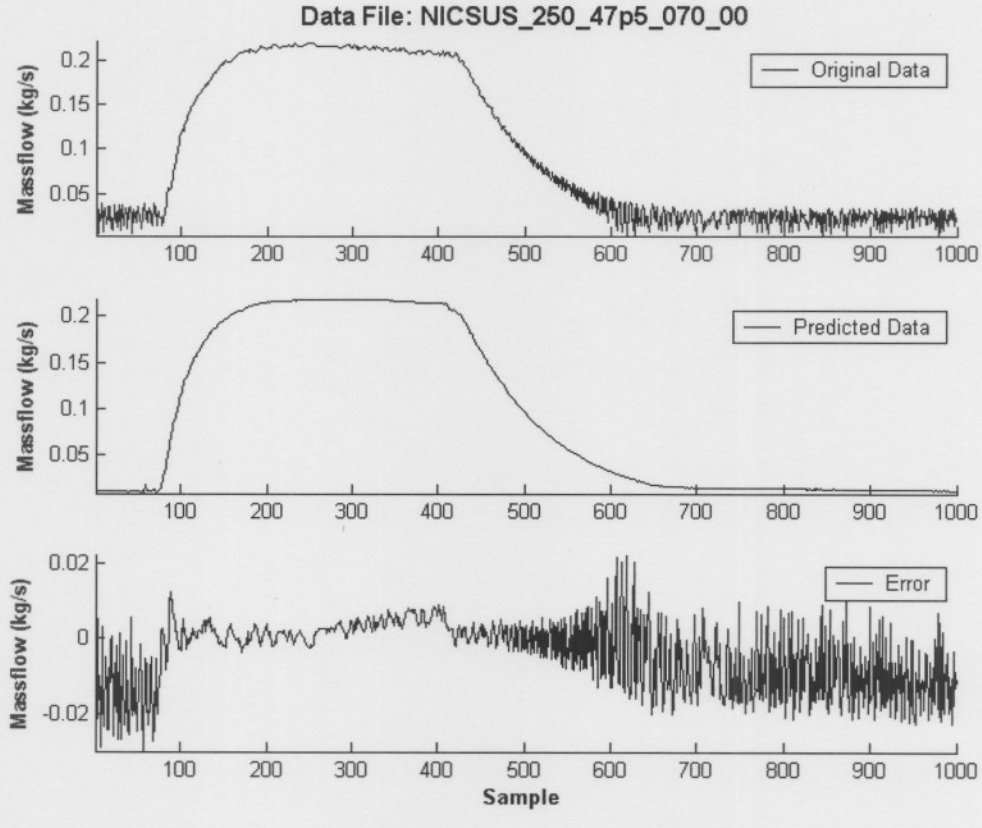


Figure 5.11. *Data File #42, Control Valve Opened 70 %, Medium Average Error: 0.081281.*

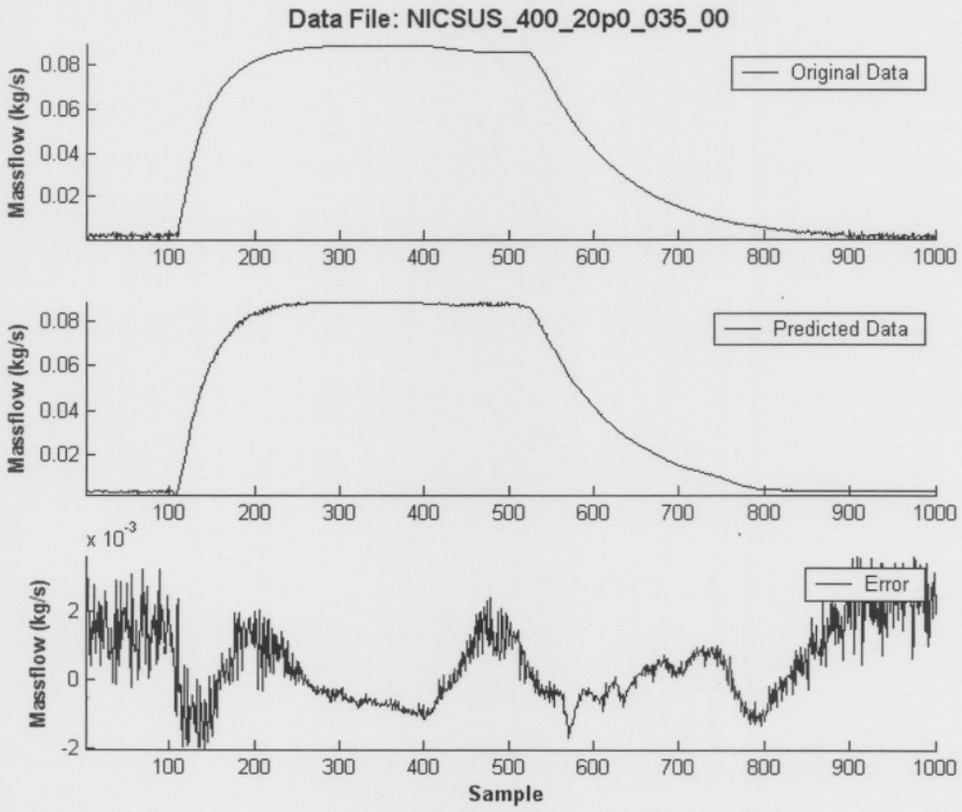


Figure 5.12. *Data File #105, Control Valve Opened 35 %, Small Average Error: 0.022740.*

Notice that even though the original massflow value in Figure 5.10 does not show any signs of an increase when the control valve is opened, the predicted data does indicate an increase in the massflow.

Figure 5.11 and Figure 5.12 produces a medium average error and small average error respectively. A certain amount of filtering is present in both results and the predicted results fit the original data adequately enough.

5.2.3 Control Valve Model 06

The results for CVMod06 can be seen in Figure 5.13 *(a)*, *(b)*, *(c)* and *(d)*. From this figure data files 50, 49, 3 and 97 produce large average errors, data files 15 and 14 produce medium average errors and the remaining data files including 94, 105, 119 and 6 produce small average errors. Typical results can be seen in Figure 5.14 to Figure 5.16.

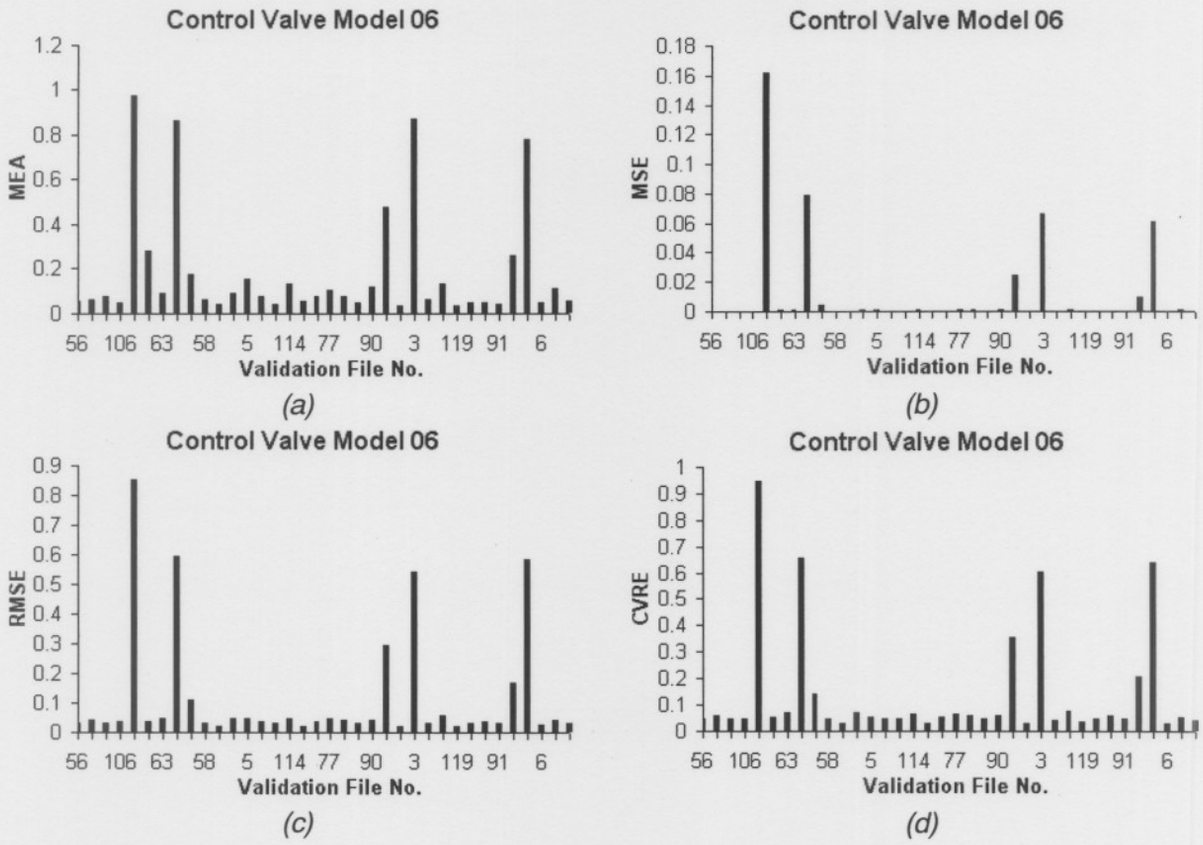


Figure 5.13. *CVMod06 Results*.

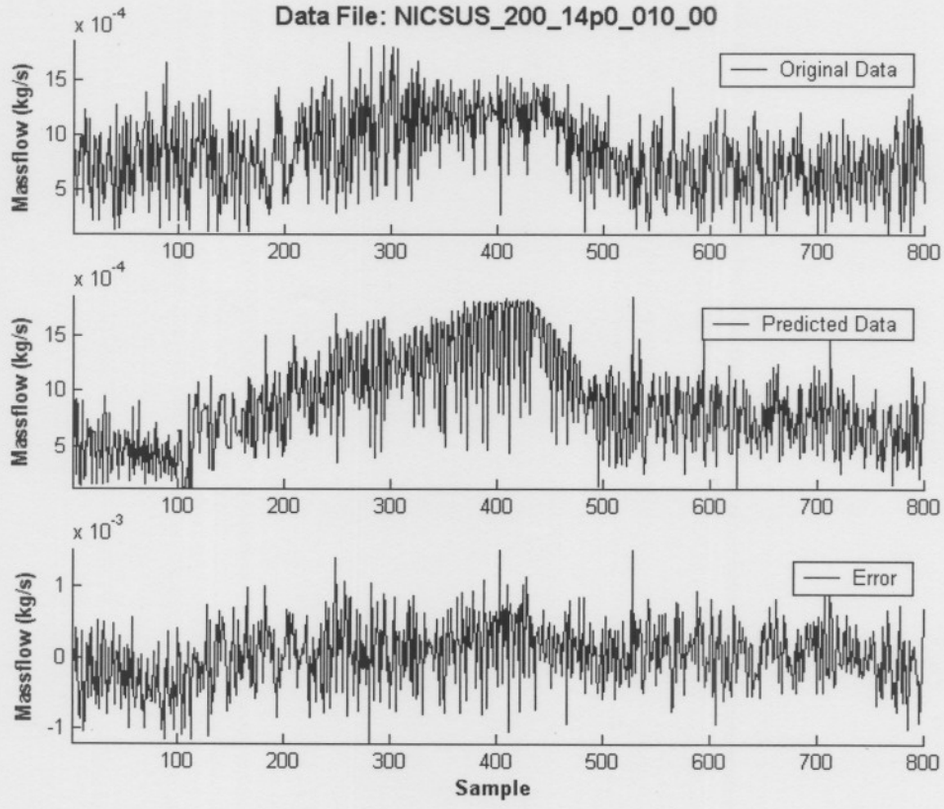


Figure 5.14. *Data File #3, Control Valve Opened 10 %, Large Average Error: 0.519672.*

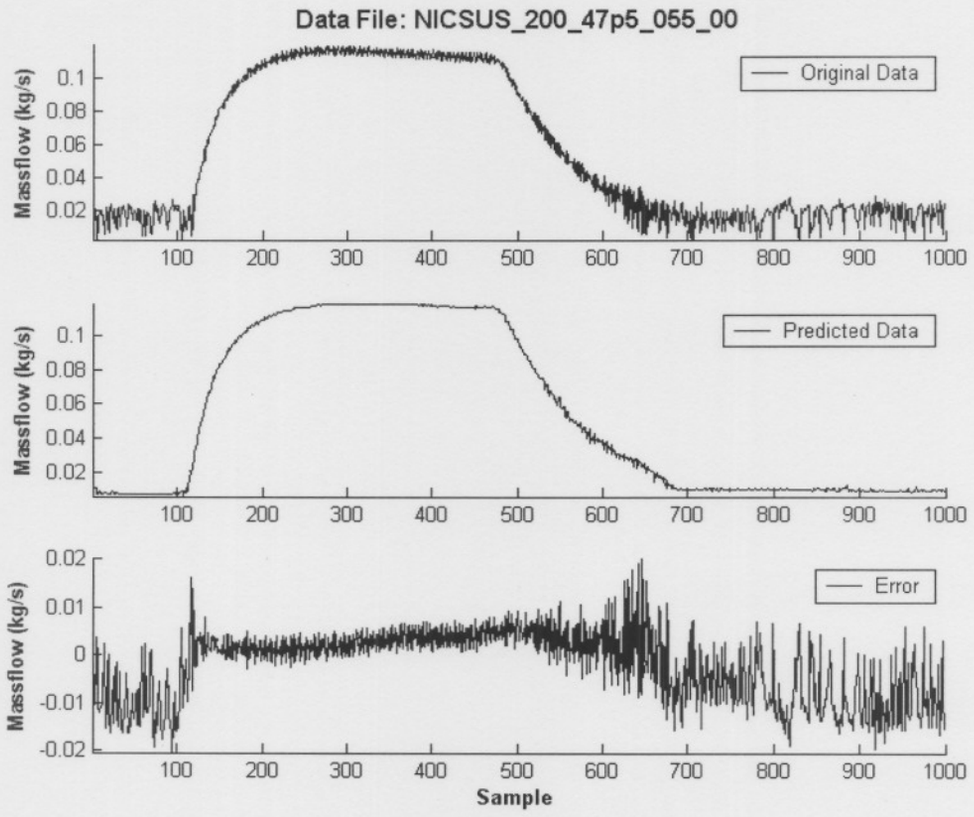


Figure 5.15. *Data File #15, Control Valve Opened 55 %, Medium Average Error: 0.108512.*

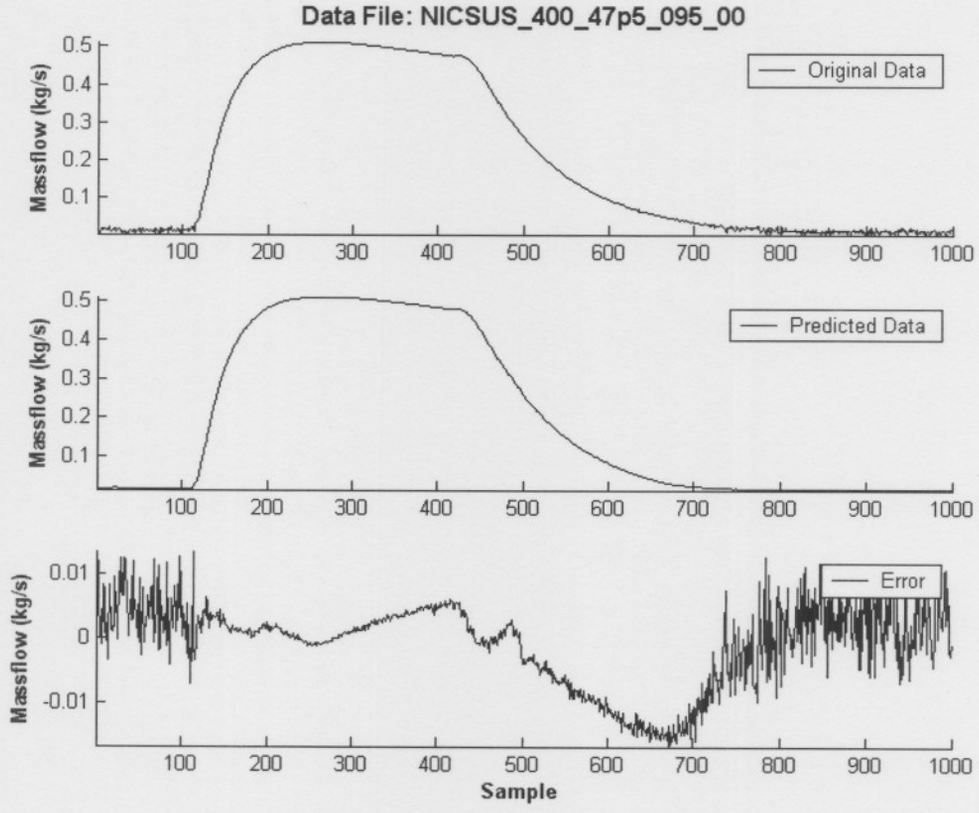


Figure 5.16. *Data File #119, Control Valve Opened 95 %, Small Average Error: 0.022444.*

5.2.4 Control Valve Model 07

The results for CVMod07 can be seen in Figure 5.17 *(a)*, *(b)*, *(c)* and *(d)*. From this figure the following data files produce large average errors which include data files 98, 1, 26 and 97. The data files which produce medium average errors are 15, 76, 52 and 57 whereas the remaining data files including 54, 9, 116 and 61 produce small average errors. These results can be seen in Figure 5.18 to Figure 5.20.

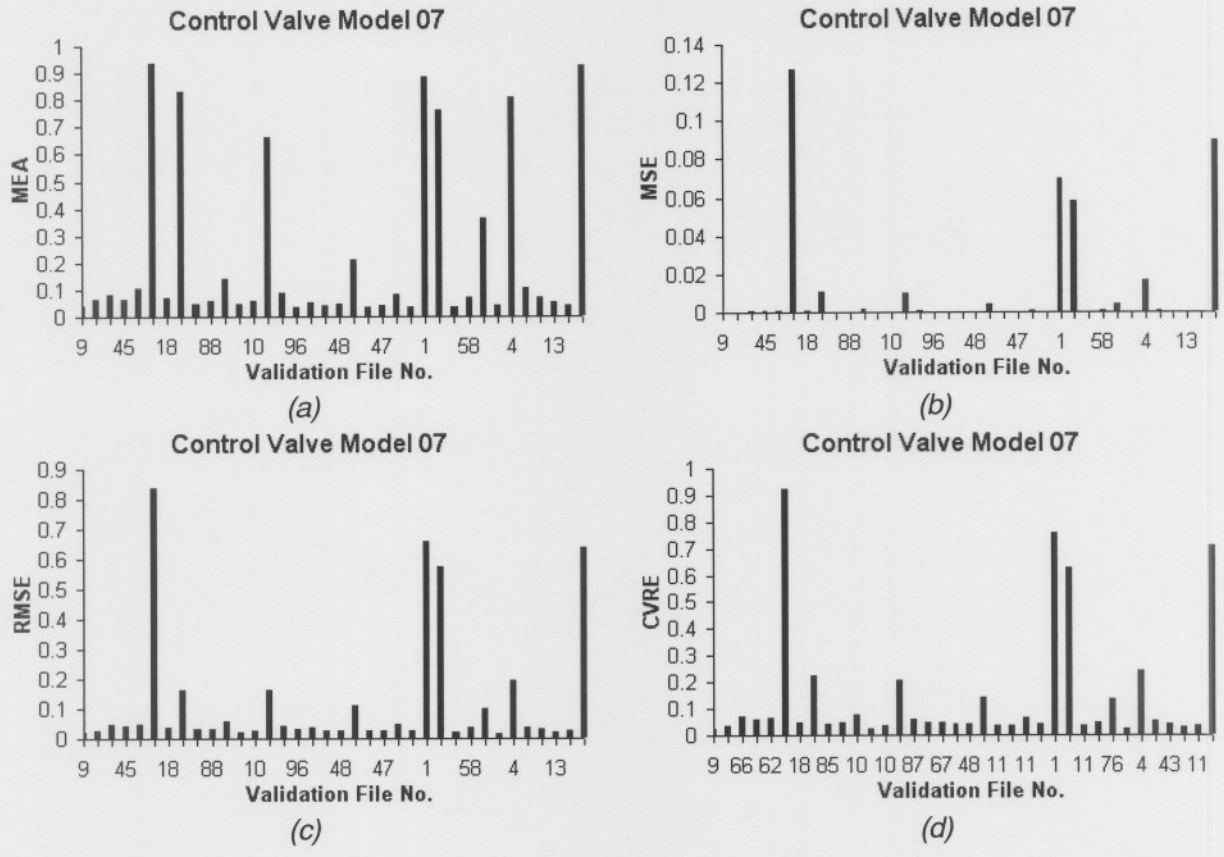


Figure 5.17. *CVMod07 Results*.

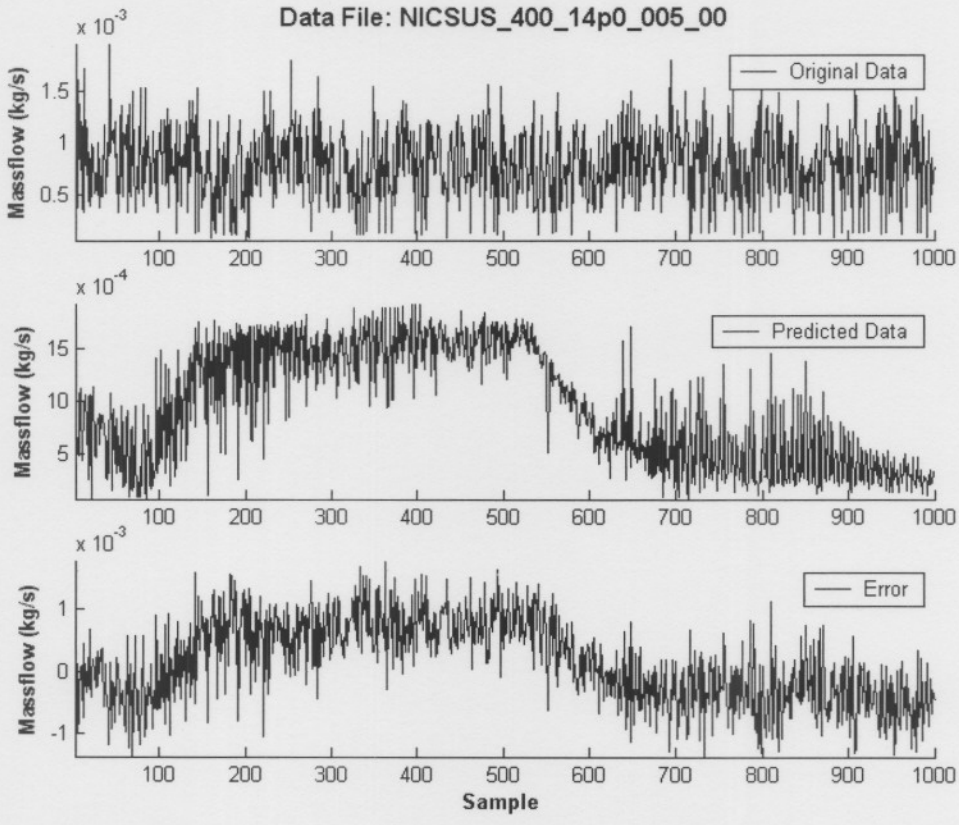


Figure 5.18. *Data File # 98, Control Valve Opened 5 %, Large Average Error: 0.705824.*

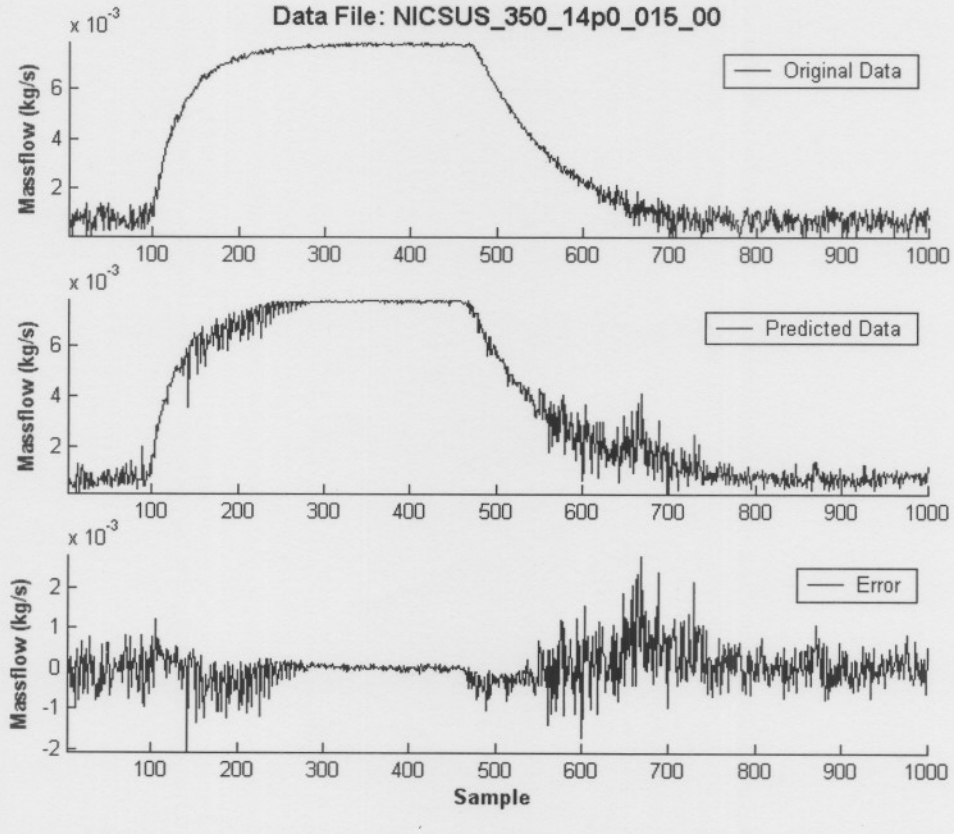


Figure 5.19. *Data File #76, Control Valve Opened 15 %, Medium Average Error: 0.150293.*

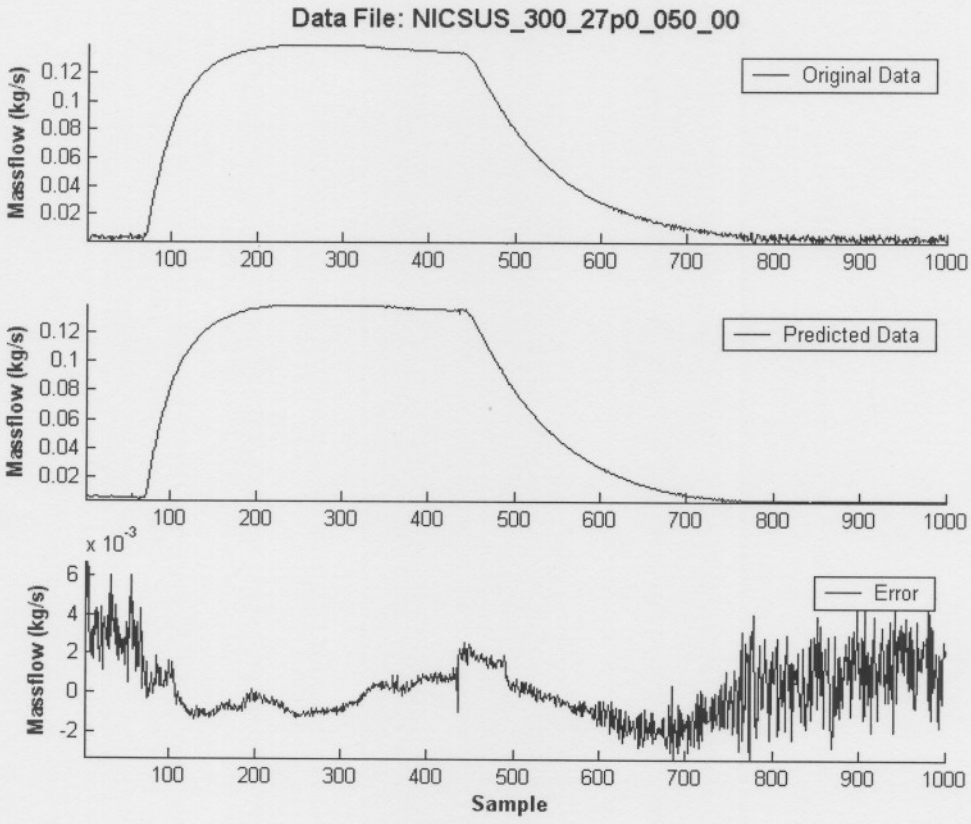
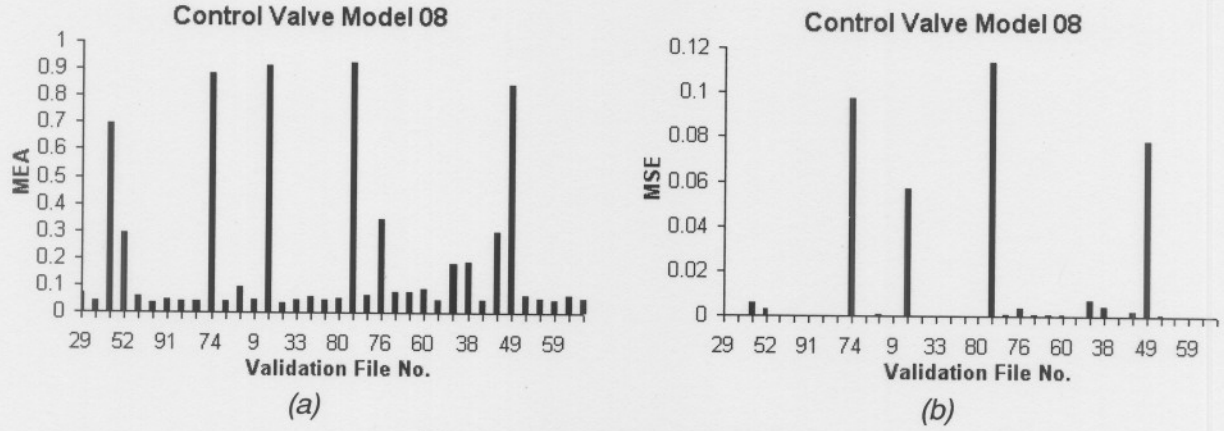


Figure 5.20. *Data File #61, Control Valve Opened 50 %, Small Average Error: 0.023431.*

5.2.5 Control Valve Model 08

The results for CVMod08 can be seen in Figure 5.21 *(a)*, *(b)*, *(c)* and *(d)*. From this figure it is evident that data files 2, 74, 73 and 49 produce large average errors, data files 110. 31, 38, 76 and 52 produce medium average errors whereas the remaining data files including 47 and 72 produce small average errors. Typical results can be seen in Figure 5.22 to Figure 5.24.



(a) *(b)*

Figure 5.21. *CVMod08 Results.*

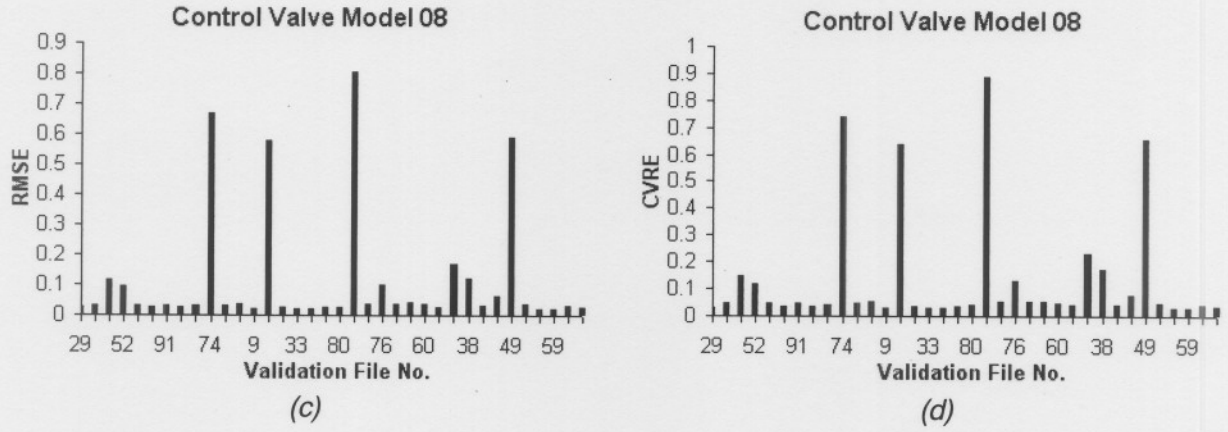


Figure 5.21. *CVMod08 Results*.

The control valve in Figure 5.22 was opened 5 %. The original data does not show any change in massflow due to the dead band of the control valve. The dead band is defined as the overlapping area of the control valve body and the actual inner moving part of the valve. In this case when the valve is opened 5 %, it remains within the dead band area and therefore does not allow air to flow through the control valve. The predicted data on the other hand does show an increase in massflow and therefore the error also increases.

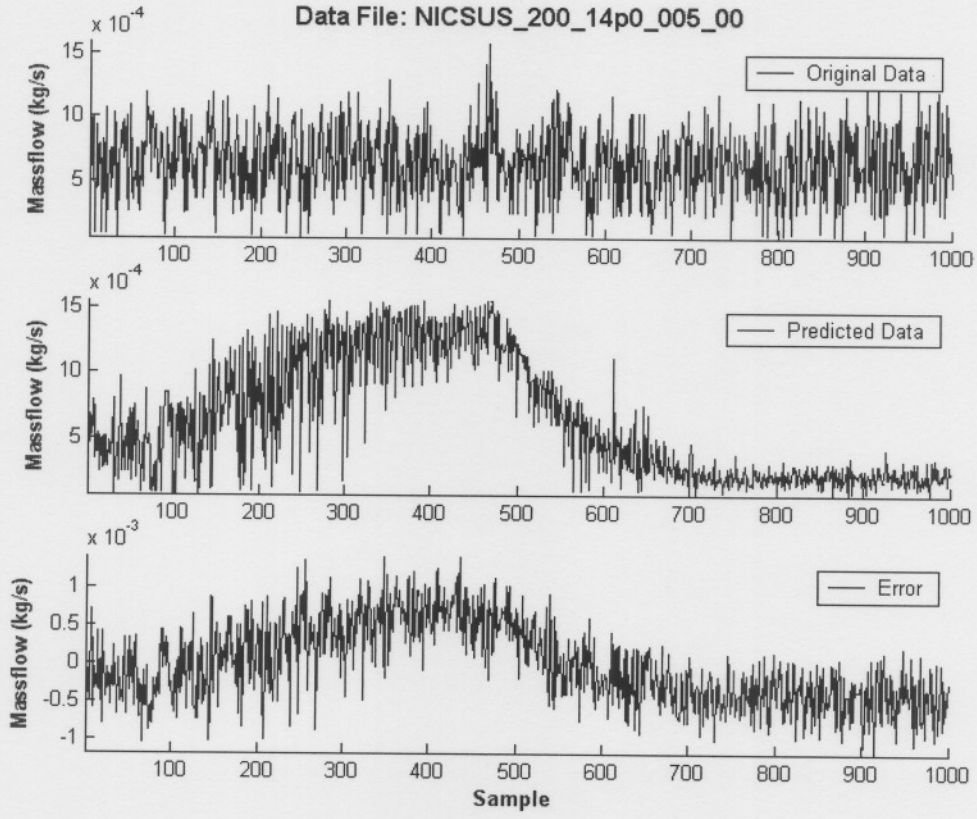


Figure 5.22. *Data File #2, Control Valve Opened 5 %, Large Average Error: 0.681248*.

Figure 5.23 and Figure 5.24 produce medium and small average errors respectively and a certain amount of filtering is again evident. Although the errors may seem large the predicted values produce good results.

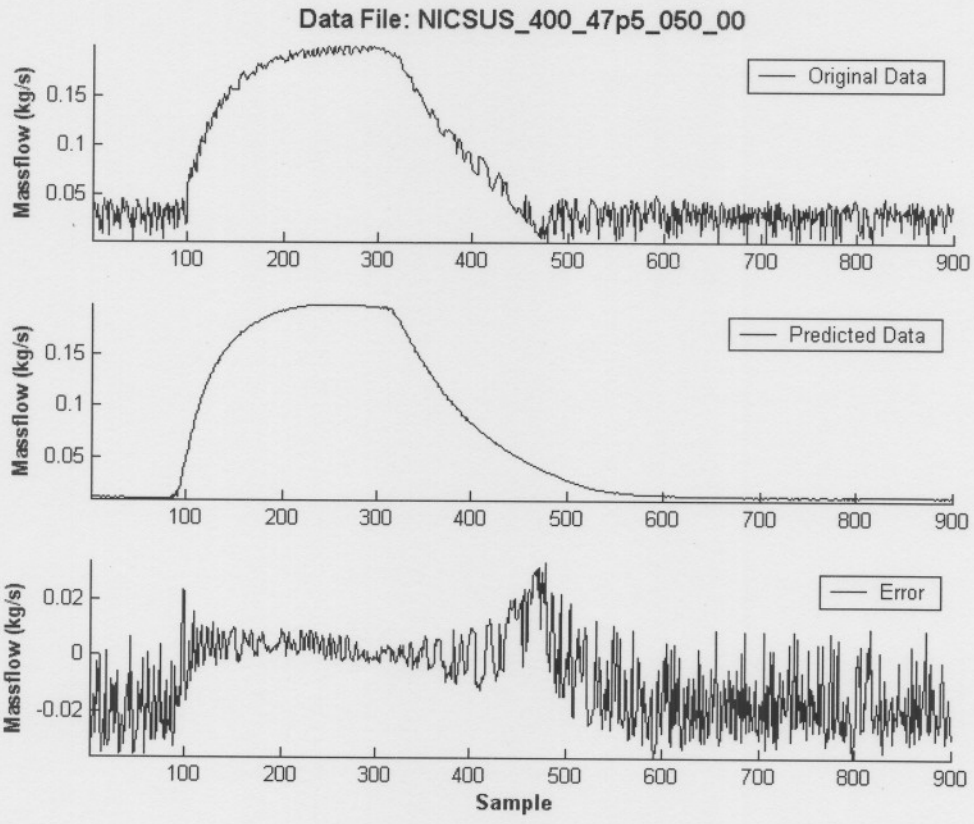


Figure 5.23. *Data File #110, Control Valve Opened 50 %, Medium Average Error: 0.147165.*

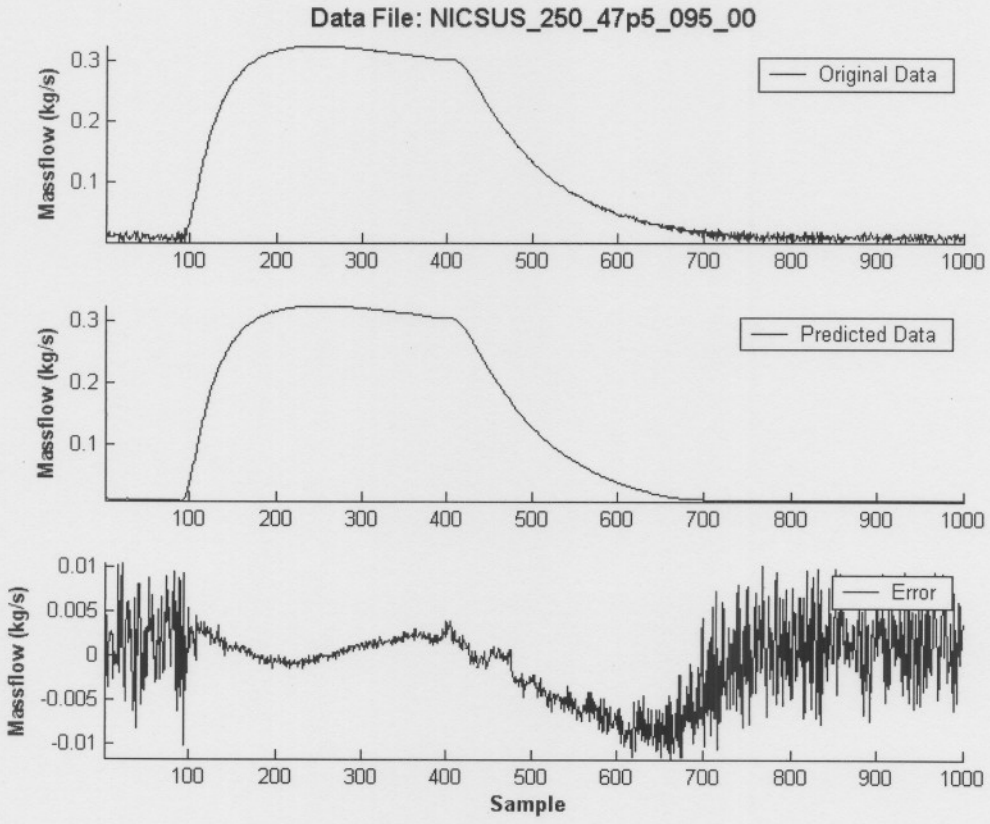


Figure 5.24. *Data File #47, Control Valve Opened 95 %, Small Average Error: 0.024254.*

5.3 COMPARISON OF FUZZY MODELS

As stipulated in 5.1, five models are derived. These models are compared to find their effectiveness as well as to estimate the effect a different random file sequence has on the fuzzy training (nearest neighbourhood clustering) algorithm.

Each of the models uses a unique random file sequence for training, verification and validation. A summary of the 30 % validation files, from the unique random file sequences, can be seen in Table 5.2. Each data file within a fuzzy model has an average error used as a benchmark. Only the average errors with at least three common occurrences in three different fuzzy models are compared. From Table 5.2 seventeen files have been identified and will be compared.

The impact of the random file sequence, used for training, verification and validation, needs to be investigated. If the data file order does not have an impact on the fuzzy system the common average error for each data file should not deviate significantly and should therefore remain constant. If however these common average errors are not constant the data file order does have an impact.

Table 5.3 summarises the seventeen data files with common average errors and are compared in Figure 5.25 to Figure 5.29.

Table 5.3. *Common Average Errors of Fuzzy Models*.

File No.	CVMod03 Average	CVMod05 Average	CVMod06 Average	CVMod07 Average	CVMod08 Average
5	0.057882	-	0.063621	-	0.110189
9	0.025229	0.023694	-	0.021377	0.025108
14	0.160113	0.160311	0.161569	-	-
37	0.038950	0.038227	0.039437	-	-
47	0.026346	-	-	0.026283	0.024254
52	-	0.187506	-	0.260235	0.125973
58	0.036246	-	0.034441	0.039636	-
62	0.058057	0.059471	-	0.055618	-
63	0.054794	0.056549	0.053948	-	-
67	-	0.033444	0.031614	0.034988	-
69	-	0.030457	0.029241	-	0.028833
73	0.488620	0.508256	-	-	0.546091
76	0.152358	-	-	0.150293	0.146051
81	0.037792	0.035645	-	-	0.036635
91	-	0.031541	0.0316553	-	0.030868
94	-	0.023699	0.021644	-	0.025157
117	0.026154	0.027208	-	0.023649	0.029389

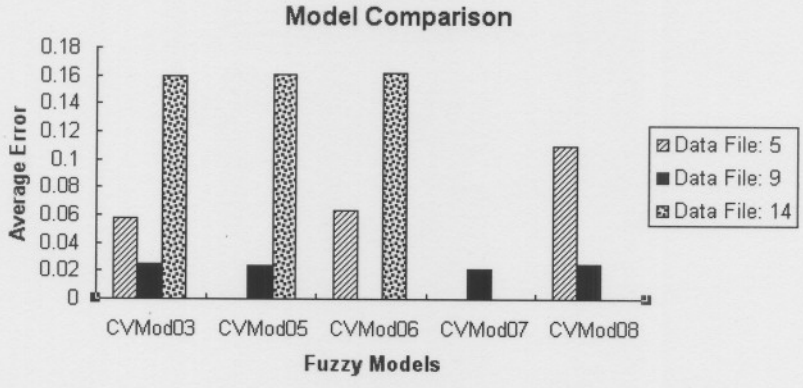


Figure 5.25. *Comparison of Data Files: 5, 9 and 14*.

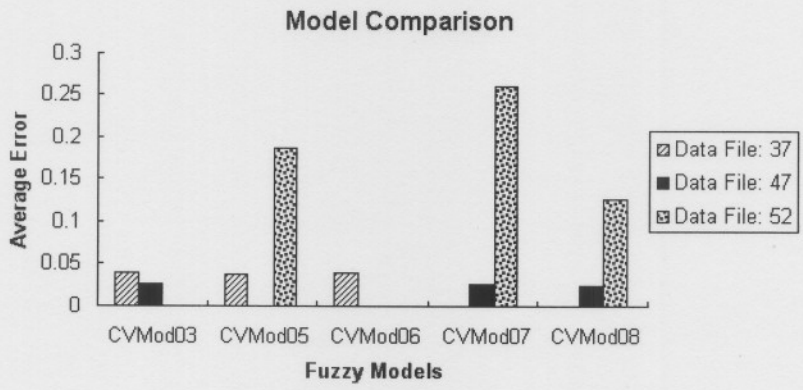


Figure 5.26. *Comparison of Data Files: 37, 47 and 52.*

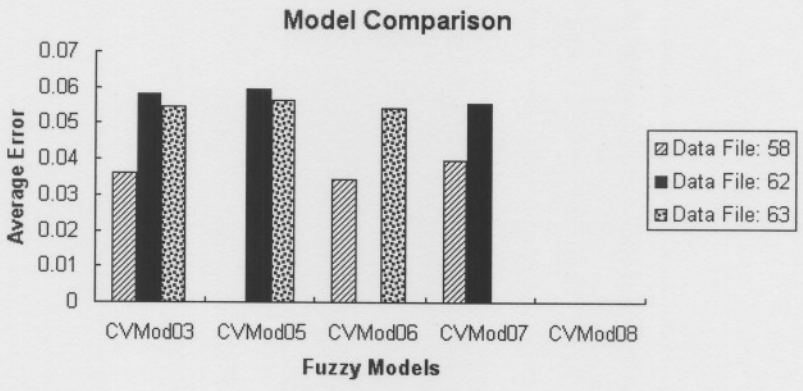


Figure 5.27. *Comparison of Data Files: 58, 62 and 63.*

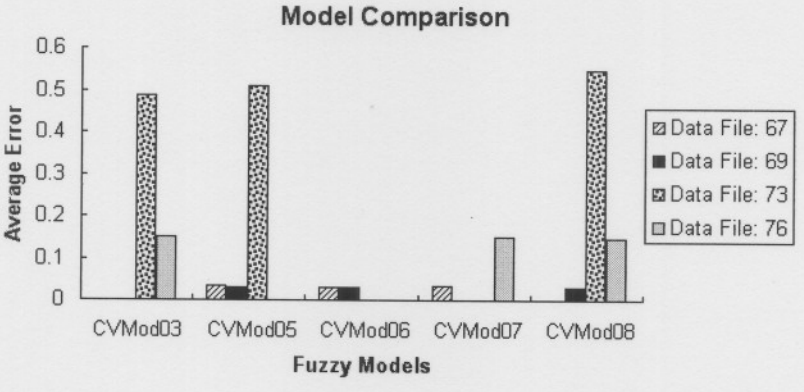


Figure 5.28. *Comparison of Data Files: 67, 69, 73 and 76.*

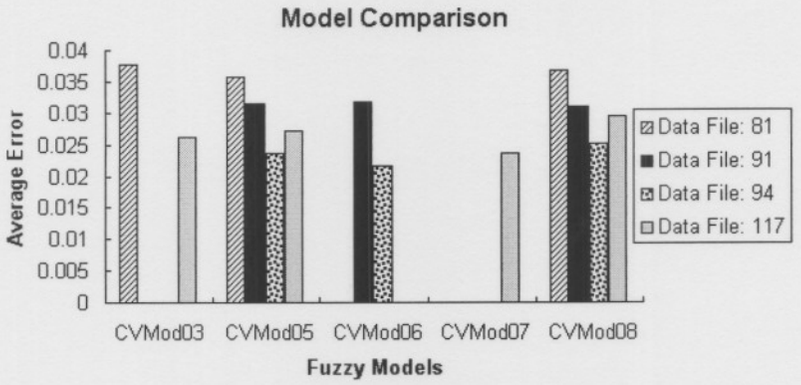


Figure 5.29. *Comparison of Data Files: 81, 91. 94 and 117.*

From these figures, the general tendency of the average errors remains constant although some data files e.g. 5, 52 and 117, tend to fluctuate. From this observation the conclusion can be drawn that the random file sequences used for training, validation and verification do not have an impact on the derived fuzzy mathematical model. Therefore any of the five derived models (CVMod03, CVMod05, CVMod06, CVMod07 and CVMod08) resembles the true dynamic behaviour of the control valve regardless of the random file sequence used for training, verification and validation.

Due to the fuzzy system's performance, regardless of the random file sequence, average values of the four error criteria of the five models can therefore be calculated. This can be seen in Table 5.4.

Table 5.4. *Fuzzy System Overall Performance*.

Model Name	MEA	MSE	RMSE	CVRE
CVMod03	0.12131485	0.004062157	0.071526289	0.089916086
CVMod05	0.164723617	0.007464772	0.090417861	0.110527206
CVMod06	0.186545697	0.011642319	0.116374869	0.139301903
CVMod07	0.222634347	0.01107495	0.119532728	0.143751311
CVMod08	0.194155197	0.010567091	0.112992	0.134940039
Average Errors	**0.177874742**	**0.008962258**	**0.102168749**	**0.123687309**

The averages for MEA, MSE, RMSE and CVRE are 0.177875, 0.008962, 0.102169 and 0.123687 respectively. From these averages the first three error criteria specify that the fuzzy system in general has an accuracy of 82.21 % for MEA, 99.10 % for MSE and 89.78 % for RMSE. The final error criterion, CVRE, specifies that, on average, the fuzzy model's predictions (data fitting) are good. Taking the average of the first three error criteria, results in a FLS with an effectiveness of about 90 %.

5.4 SUMMARY

This chapter used the parameter settings obtained in *Chapter 4* and derived five unique fuzzy mathematical models of the control valve. It was discovered that the fuzzy system is independent of the specific random file sequence used for training, validation and verification. Any of the derived fuzzy logic control valve models resembles the true dynamic behaviour of the control valve regardless of the random file sequence used for training, verification and validation.

The results of each of the five models were covered as well as compared. Using the results of the five models the final fuzzy system performance was calculated to be about 90 % when averaging the three error criteria i.e. MEA, MSE and RMSE. The error criterion CVRE indicated that the overall fuzzy predictions were accurate enough and visually produced sufficient results.

CHAPTER 6

Conclusions and Recommendations

This chapter concludes the dissertation and recommends a number of improvements which may be applied to the fuzzy mathematical models in order to increase its accuracy and effectiveness. Most of these improvements introduce additional components into the system and may drastically increase the system costs. On the other hand some of the improvements involve software changes or algorithm modifications which do not necessarily increase system costs as such.

6.1 CONCLUSIONS

The mass rate of flow, in a pipe network, can be successfully calculated by using a number of orifice plates together with a differential pressure transducer and the ISO 5167 British standards. Although this method is used in industry, a flow meter may yield more accurate results over a wider range.

The signal conversion hardware forms an integral part of the DAQ system in that, current loop signals are converted to voltage signals, which in turn can be sampled by the DAQ card. The signal converters have been successfully used in the DAQ system, resulting in a DAQ system with high accuracy and low noise.

The DAQ software, used to acquire, manage and process the data, have been successfully integrated into the DAQ system. These software applications include the DAQ process control application (acquire data), the post-processing application (calculate governing equation values) and the data plot application (display acquired and processed data on graphs).

The procedure stipulated in *Chapter 4*, resulted in a set of parameters which accurately captured the behaviour of the control valve. These optimal parameters were derived by using a predetermined procedure and only a single data set. Using these optimal parameters, while deriving control valve models using all the data sets, resulted in fuzzy logic mathematical models with an accuracy of ±90 %. Therefore it can be concluded that fuzzy logic with the nearest neighbourhood clustering algorithm can be successfully applied to model nonlinear dynamic engineering processes.

These fuzzy mathematical models consist of the fuzzy cluster centre vectors ($\underline{x}_0$), the sum of the output parameter in each fuzzy cluster (A) and the number of data samples in each fuzzy cluster (B).

6.2 RECOMMENDATIONS

Although the fuzzy algorithm was successfully applied to the engineering application, a number of recommendations are made which may improve the fuzzy logic model accuracy.

According to section 2.2.1, [6] and [7] the measurement of the mass rate of flow in a pipeline is based on the installation of a primary device such as an orifice plate. The orifice causes a static pressure difference in the pipeline from which the mass rate of flow can then be calculated. The mass rate of flow calculation formula is covered in section 3.6.1.

Instead of using an orifice to measure the mass rate of flow through the pipeline, a flow meter may be installed. This can drastically improve the accuracy and range of the measurements (in both non-choke and choke situations) but will increase the total system costs dramatically. It will also eliminate the need to use the ISO formula (which is valid only under certain conditions), orifice plates and a differential pressure transmitter. Note that the differential pressure transmitter only has a valid range of +/-500 mbar and therefore it is inevitable that various orifice plates with different sizes need to be used not to exceed its range.

If no other alternative is available and an orifice must be used in order to measure the mass rate of flow, the installation of a differential pressure transmitter with larger range should be considered. This can ensure the use of only one orifice plate to conduct all the experiments and therefore eliminates yet another variable. The sizing of the orifice plate should be carefully calculated to ensure that both low and high differential pressures can be measured with great accuracy. Care should also be taken to ensure that the orifice is never used in its choke region as this will result in serious modelling errors. This is due to the limiting factor the orifice plate has on the pipeline during these circumstances.

According to section 3.6.1, the calculation of the mass rate of flow is purely an arithmetic process and involves a complex numeric method. Fixing the coefficient of discharge (C), simplifies the calculations and thus eliminates the need for such a complex numeric method. Using such a numeric method can increase the accuracy of the mass rate of flow ($\dot{m}$), Reynolds number (R_{ep}) and the coefficient of discharge (C) and will thus increase the overall accuracy of the derived fuzzy mathematical model.

The experiments conducted in this dissertation only allowed the release of compressed dry air to ambient pressure (86.5 kPa). Therefore the characterisation of the control valve is limited. If the pressure on the downstream side of the control valve can also be regulated it may allow characterisation of the control valve over a wider range. This inevitably increases the number of experiments to be conducted but may prove very useful.

A regulated constant pressure source can be used as opposed to a pressure vessel, in order to eliminate another fluctuating variable. This effectively implies that the pressures on both upstream and downstream sides of the control valve should be regulated and kept constant throughout an experiment. It will enable the characterisation of the control valve in both choke and non-choke situations at much higher static pressures.

Developing and/or adding analog or digital filters to the data acquisition (DAQ) system, in order to eliminate noise, may improve the reliability of the data and thus the accuracy of the derived fuzzy control valve model.

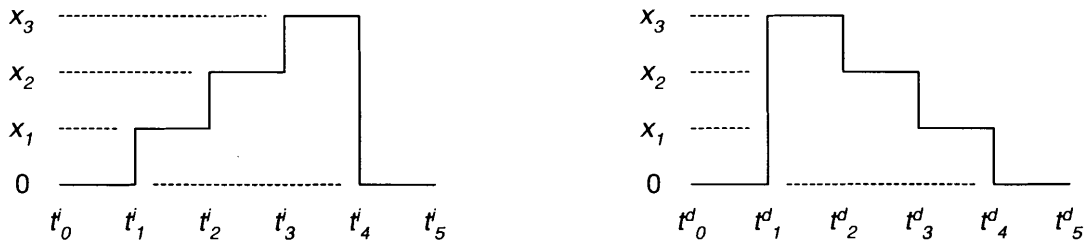
The fuzzy logic nearest neighbourhood clustering algorithm was used to derive the mathematical models. This algorithm is particularly useful when faced with large sample problems in that clusters are created during the training phase. These clusters thereby decrease the number of input-output data pairs where a cluster centre vector represents the entire cluster. During the one-pass training phase, it is capable of matching all these input-output data cluster pairs to any given accuracy [5].

Other fuzzy logic systems should also be explored such as the nearest neighbourhood algorithm with a non-singleton fuzzifier and defuzzifier (this implies the derivation of a new formula from first principles) which is to a certain extent immune to noise. Another option such as the orthogonal least squares algorithm can also be investigated as this again is a one-pass regression algorithm which is not sensitive to noise in the input signals. Linguistic fuzzy *IF-THEN* rules, obtained from a human expert, can also be used to derive an expert fuzzy model as yet another option [5].

As stated in section 4.1, no predetermined method can be used to obtain the optimal settings for the fuzzy logic system (FLS). However it is recommended to use a method which can be explained with mathematics and common sense. Therefore the method used in this dissertation is only one of many and by no means the ultimate procedure. It therefore implies that the method may be revised, if not totally re-evaluated, in order to justify the needs of the particular application.

During normal system operation and control it is seldom the case to fully close the control valve while controlling the process, except perhaps during startup and shutdown sequences. System control takes place around a certain working point often only slightly altering the control valve opening, in which case the working point is the control valve relative opening (*h*). With this in mind, experiments should be constructed so as to investigate the behaviour of the control valve in its natural working environment. This no doubt will increase the number of experiments but again will improve the fuzzy mathematical model behaviour during its natural operation. Other experiments can also be conducted but will most likely introduce additional mechanical components into the system and will therefore again increase the system costs.

Using the same procedure as stipulated in section 3.8.1 (for the unit step experiments), except for multiple increment or decrement steps, the control valve behaviour can be investigated around these working points. Typical increment and decrement sequences can be seen in Figure 6.1 *(a)* and *(b)* respectively. From these figures it is evident that the working points are 0, x_1, x_2 and x_3.



(a) Increment Multi Step *(b) Decrement Multi Step*

Figure 6.1. *Multi Step Experiment.*

In this dissertation the conducted experiments only covered the range from 200 kPa to 400 kPa with increments of 50 kPa. It is recommended to extend the range of operation (e.g 0 – 1000 kPa) and to decrease the increments (e.g. 20 kPa) to ensure that even more possibilities are covered.

Real world data, from a range of control valves, should be acquired as apposed to only data from a single control valve as in this case. This will allow the characterisation of an entire range of control valves (e.g. eccentric plug valves, segment ball valves, ball valves, butterfly valves and globe valves [9]) and will therefore not be limited to only a single control valve.

The derived fuzzy models ($\underline{x}_0$, *A* and *B* as stated in section 6.1) need to be incorporated into a simulator or an easy to use graphical user interface (GIU). It can therefore operate in parallel with an existing plant or production facility as well as simplify its general usage.

APPENDIX A

Hardware User Manual

Industry standards specify that command signals be transmitted by means of so called current loops due to its insensitivity to noise and immunity to line impedance. This appendix is dedicated to the hardware signal converter's user manual, the PCI-6023E DAQ board's (referred to as the ADC) features and signal connections.

A.1 DAQ HARDWARE SIGNAL CONVERTER

The ADC has a maximum input voltage range of ±10 V, which implies that the current loop signals (4 – 20 mA) be converted to voltage signals, typically in the range 1 – 5 V. This converted voltage signal may then be sampled by the ADC.

A.1.1 Hardware System Interface

From Figure 3.2 it can be seen that the current loop signal passes through the converter hardware, causing a small negligible voltage drop across the converters' internal sense resistor. The voltage across the sense resistor reaches a maximum of 1.5 V at full scale (20 mA loop current) making it very useful in current loops containing additional instrumentation [23]. Due to the innovative hardware design the only added load to the current loop is that of the sense resistor. This has an added advantage in that it does not cause any ground loops within the system, which may possibly cause system instability and/or even system failure. The converted current signal (voltage signal) is sampled by the ADC and locally managed, processed and saved by the terminal computer.

A.1.2 Converter Block Diagram

The hardware signal converter consists of two main sections; the first being the current to voltage conversion and the second the scaling and offset control. The scaling and offset control allows calibration of the converter if undesirable results are obtained.

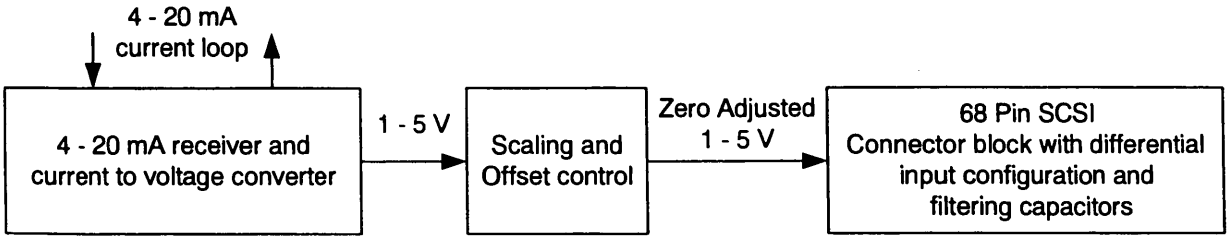


Figure A.1. *Converter Block Diagram.*

The current to voltage conversion is done by a current loop receiver from Burr-Brown (RCV420). The linear relation between input current signals and output voltage signals simplifies calculations as well as scaling functions. The scaling and offset circuit also has a linear response which therefore implies that the total converter response is linear.

The RCV420's linear response can be seen in Figure A.2. Notice that the input voltage across the internal sense resistor and the external load increases linearly as the input loop current increases. Extending both trends to the left, it can be seen that they intersect the (0, 0) point on the graph.

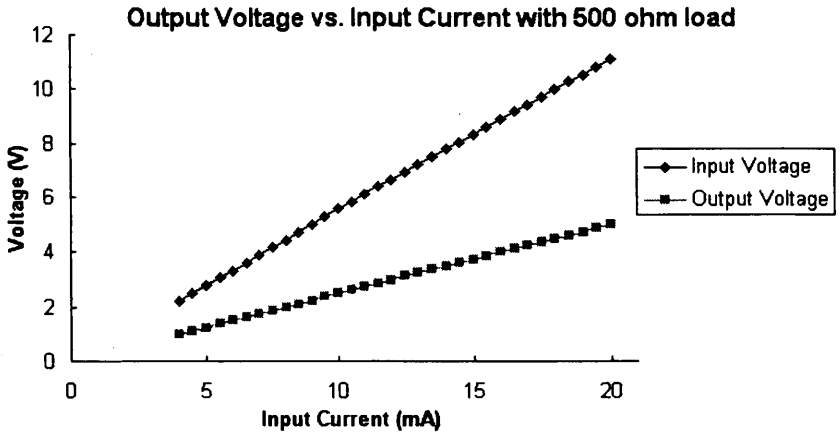


Figure A.2. *Linear Response of RCV420.*

A.1.3 Signal Converter User Manual

Eight differential mode channels are used to capture the dynamic characteristics of the control valve, therefore eight signal converters (one per channel), are used to convert all the current loop signals which need to be acquired. The converters have the following features:

A.1.3.1 Features

- Selectable zero-adjust reference
- Simple current to voltage conversion with internal sense resistor
- Low voltage drop across converter (1.5 V at max 20 mA current loop)
- Possible fault detection due to system set up
- ±40 V Common mode input range

A.1.3.2 DAQ Hardware Signal Converter Layout

The signal converter's component layout is shown in Figure A.3. Its dimensions are; 68 x 71 mm. The component description and specifications can be seen in Table A.1.

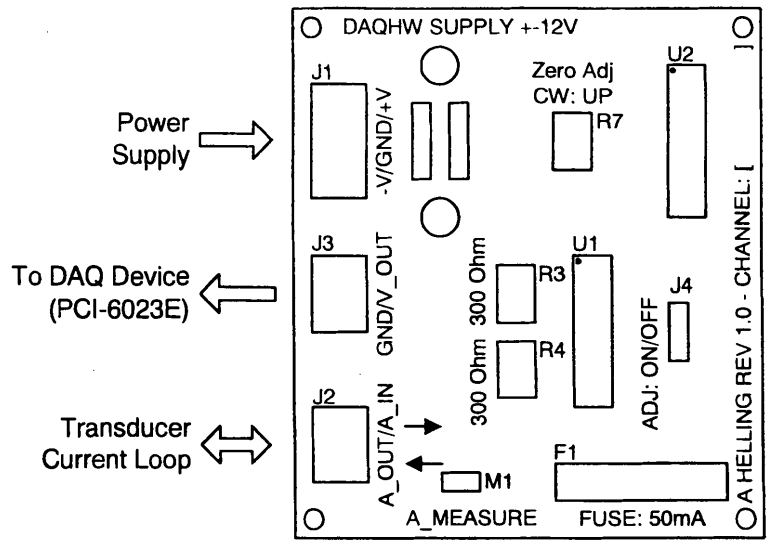


Figure A.3. *Converter Quick Reference Map.*

Table A.1. *Signal Converter Component Description.*

Component	Description	Specification
F1	Over-current system fuse	Quick blow 50 mA
J1	Power connector	±12 V battery power supply
J2	Current loop connector	4 – 20 mA transducer current loop
J3	Voltage output connector	1 – 5 V output to ADC
J4	Reference jumper	Zero or selectable adjustment reference
M1	Current measurement jumper	Male current loop inspection jumper
U1	Current to voltage converter	RCV420 current loop receiver
U2	Voltage offset generating IC	LM324 dual op-amp
R7	Offset adjustment selector resistor	50 kΩ multi-trim Pot

A.1.3.3 DAQ Converter System Integration

Integrating the signal converters into the system is very simple. Three connectors (*J1*, *J2* and *J3*) are respectively connected to power, loop current and voltage output (A/D Card). The converters use a ±12 V dual power supply which consists of two 12 V, 7 A/h rechargeable batteries. Batteries are used to supply the signal converters with a power supply which is stable, mobile and not susceptible to external interference and noise. Using another type of power supply could introduce these interferences and thus degrade the measured and converted signals. The power supply is connected to the hardware converter via connector *J1*. The positive, negative and ground terminals are shown in Figure A.3 denoted by *-V*, *GND* and *+V*.

Each transducer's current loop passes through a signal converter by breaking the loop and connecting it to connector *J2*, more specifically the current input terminal is connected to *A_IN* of *J2* and the current output terminal connected to *A_OUT* of *J2*. This in effect reconnects the current loop except that it now passes through the signal converter's sense resistor. It causes a maximum voltage drop of 1.5 V at full scale (20 mA loop current) across the sense resistor of the signal converter.

The PCI-6023E features sixteen analog input channels when configured in referenced single ended (RSE) and eight when configured in differential mode. From Figure 3.3 it is clear that the control valve's dynamic characteristics and behaviour are captured with eight transducers, therefore eight channels are used. These channels are set up in differential mode to limit the interference of noise. Each signal converter's output voltage (converted from input current) at connector *J3* is connected to an ADC differential mode channel.

After the signal converter has been installed and powered, removing jumper *M1* makes it possible to measure the loop current without having to disconnect any of connector *J2's* terminals. Notice that once jumper *M1* is removed from the hardware converter, the current loop is broken and all communication to the transducer is lost. Connecting an ammeter to jumper *M1* will however re-establish communication.

A protection fuse (*F1*) is added to the system to protect the sense resistor from over-current. Due to the loop current receiver's maximum electrical specifications (40 mA continuous and 250 mA momentary – 0.1 s) a quick blow 50 mA fuse is used as protection [23]. Other protection methods also exist but this is the most cost-effective technique.

The converter IC could be referenced to either ground or a selectable reference voltage. This allows accurate adjustment of the output voltage. The reference or zero adjustment is done via *R7* provided jumper *J4* is connected in the *ON* position.

A.1.3.4 Signal Converter Hardware Calibration

Calibrating the hardware is done in two steps. One by calibrating the resistor network, and step two by calibrating the signal offset.

Use an ohm meter and measure the resistance between pins *U1.1* and *U1.2*. This should measure a resistance according to (A.1).

$$R_x = R_s \Big/ \left(\frac{I_{L\,max}}{16\ mA} - 1 \right) \qquad \text{(A.1)}$$

where R_x is the external resistance, R_s the internal sense resistance and I_{LMAX} the maximum loop current. In this case R_x, which is *R3* and *R4*, should both be adjusted to a value of 300 Ω due to R_s being 75 Ω and I_{LMAX} being 20 mA. *R3* is measured across pins *U1.1* and *U1.2* and *R4* across pins *U1.2* and *U1.3*. Replace *U1* once both these resistors have been calibrated.

Step two, an optional step, needs adjustment only if *J4* is set to the *ON* position in which case the signal offset can be calibrated via *R7*. If it is set to the *OFF* position, no adjustment is needed and thus will have no effect.

Remove the signal converter from the system and disconnect it from the DAQ device. While supplying it with a constant 4 mA current loop, which could be generated with a simple resistor network and a small power supply, measure the output voltage across *J3*. The output voltage should be adjusted to reflect the values specified in Table A.2.

Remove the 4 mA loop current and again measure the output voltage. This should have dropped from 1.00 V to 0.00 V due to the change in loop current. This should be repeated with the remaining input currents within Table A.2 in order to increase the accuracy of the conversion hardware. Typical values are specified in the table below which can be used as a guideline.

Table A.2. *Current to Voltage Conversion.*

Input Current (mA)	Output Voltage (V)	Status
0	0	Error of Calibration
4	1	Normal Operation or Calibration
8	2	Normal Operation or Calibration
12	3	Normal Operation or Calibration
16	4	Normal Operation or Calibration
20	5	Normal Operation or Calibration

Once the signal converter is used in normal operating conditions and the output voltage has dropped to 0.00 V instead of the minimum 1.00 V, it may be due to an error which occurred either in the device or the converter. This may either indicate a faulty current loop or transducer. The device status is shown in Table A.2.

A.2 DAQ PCI-6023E CARD

The PCI-6023E series DAQ device is a high-performance multifunction analog, digital and timing input/output (I/O) board used in a broad variety of applications. This device uses *E Series* technology to deliver high performance and reliable data acquisition capabilities [24].

A.2.1 Features

The 6023E features 16 single ended referenced analog input channels (eight differential), eight digital I/O lines (compatible with 5 V TTL/CMOS) but no analog output channels. All these channels connect to sensors and transducers via a 68-pin male SCSI-II type connector. The *E series* features a 12-bit, 200 kS/s ADC with an input range of ±0.05 to ±10 V.

A.2.1.1 *E Series* Block Diagram

The block diagram of the *E Series* DAQ device can be seen in Figure A.4. The following paragraphs discuss its basic building blocks and features [24].

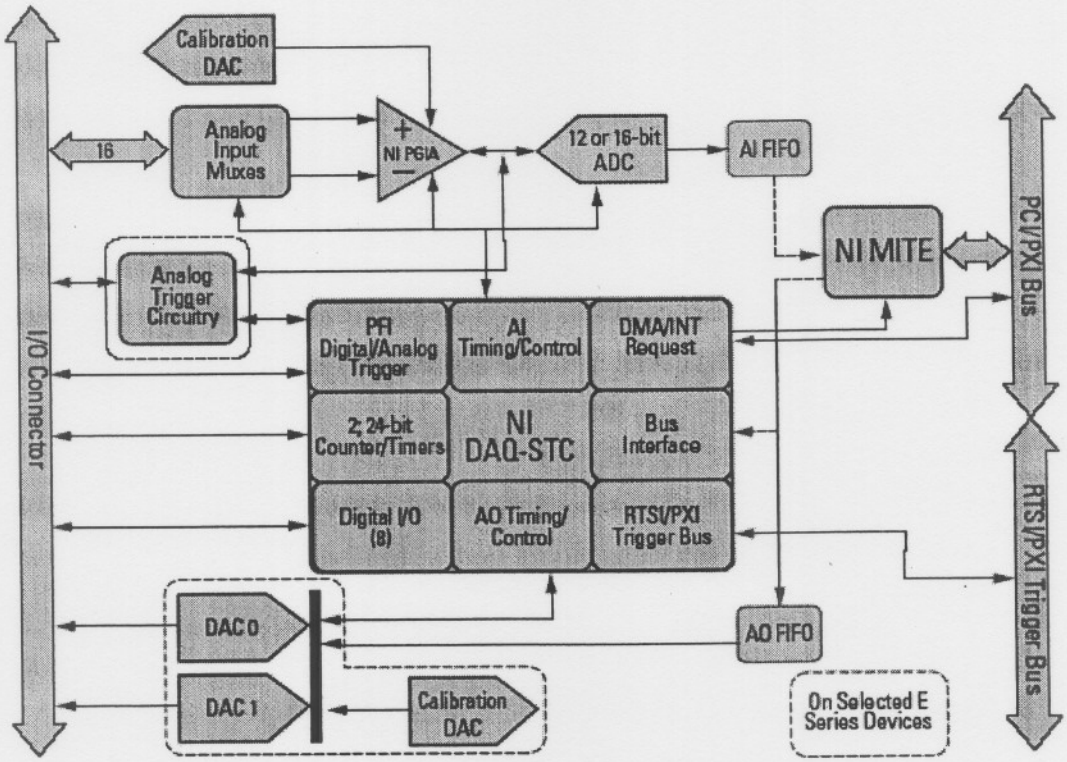


Figure A.4. *E Series Block Diagram*.

The *E series* multifunction DAQ devices are designed with components which reduce the amount of total temperature drift error to less than 0.001 % per °C. This creates a stable device which acquires data without additionally introducing temperature drift components.

The DAQ device also features resolution improvement technologies which introduces a four times improvement in resolution due to dithering and a carefully designed noise floor. Averaging a number of the real-world data samples can also improve the resolution. Such a data averaging function should either be added to the DAQ process GUI or to a post-processor.

The *E series* features a highly precise voltage reference, used during self-calibration. The device is calibrated at a single gain and relies on a precision resistor to reflect the calibration at the other gains. Self-calibration is initiated by a simple software call. The device should be calibrated often by using the onboard calibration circuit to minimise errors caused by temperature drift and time.

The National Instruments Programmable Gain Instrument Amplifier (NI-PGIA), which is an instrumentation-class amplifier, guarantees settling times at all gains. Typical off-the-shelf amplifier components might not meet the settling time requirements of high-gain measurement applications. If the amplifier does not settle between analog to digital (A/D) conversions, the full resolution of the digitiser is not achieved.

National Instruments (NI) multifunction DAQ devices are designed from the latest off-the-shelf A/D and digital to analog (D/A) converters. The NI-PGIA and data acquisition system time control (DAQ-STC) are both custom application specific integrated circuits (ASIC) developed by NI to optimise and control the A/D and D/A.

The DAQ-STC timing control ASIC was designed to provide more flexibility, lower power consumption and a higher immunity to noise and jitter, than off-the-shelf counter/timer products. The DAQ-STC provides two 24-bit counters/timers, interval and round-robin analog input channel scanning, onboard synchronisation of analog I/O and counter/timer operations, multiboard synchronisation with the real-time system integration (RTSI) trigger bus and better than 0.01% clock stability.

The NI-MITE application specific integrated circuit (ASIC) was designed to optimise data transfer for simultaneous multiple operations. It is capable of transferring data with Bus mastering using three scatter-gather direct memory access (DMA) channels, transferring interrupts and programmable I/O.

A.2.2 Signal Connections

E series acquisition devices can be configured in three input modes; nonreferenced single-ended (NRSE), referenced single-ended (RSE) and differential (DIFF). Single-ended mode configuration provides up to 16 analog input channels whereas differential mode features up to 8 input channels. Input modes are programmed on a per channel basis implying that some channels can be configured in single-ended mode and others in differential mode.

The NRSE configuration mode uses only one analog input line, which connects to the positive input of the channel, and the negative input connects to the analog input sense (AISENSE) line. The AISENSE line is used as the common ground when multiple channels are configured. ACH*xx* and AISENSE are referenced to ground via resistors (Refer to Figure A.5).

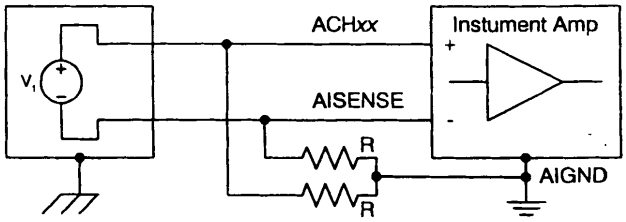


Figure A.5. *Nonreferenced Single-Ended Mode*.

RSE configuration mode also uses one analog input line, which connects to the positive input of the channel. The negative input of the channel is internally connected to analog input ground (AIGND). All AIGNDs are internally connected to each other (refer to Figure A.6).

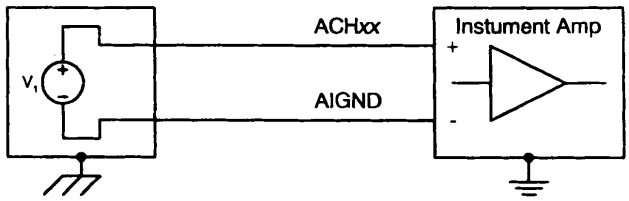


Figure A.6. *Reference Single-Ended Mode*.

Differential configuration mode uses two analog input lines. One connects to the positive input of the channel and the other to the negative input of the channel. It is strongly adviced that some form of common reference be created between multiple channels. This could be achieved by connecting high value resistors as seen in Figure A.7.

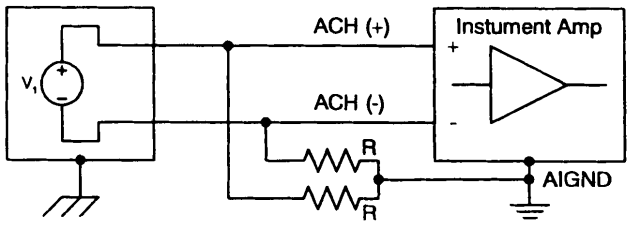


Figure A.7. *Differential Mode*.

A.2.2.1 DAQ Card Pinouts

For stability reasons it is decided to set up the channels in the differential mode as depicted in Figure A.7, which allowed the use of a maximum of eight channels on the specific DAQ device. This implies that *ACH00* together with *ACH08* produces differential channel 0, *ACH01* together with *ACH09* produces differential channel 1, *ACH02* together with *ACH10* produces differential channel 2, etc... Note that *ACH00* to *ACH07* are the positive channel terminals and *ACH08* to *ACH15* the negative channel terminals.

Due to the nature of the specific application's signals, only the analog input channels are used. Other applications may require the use of channels in different modes. Some applications might even make use of the digital inputs and outputs or an external strobe pin to latch data or trigger external devices. The PCI-6023E DAQ device SCSI pinouts are shown in Figure A.8.

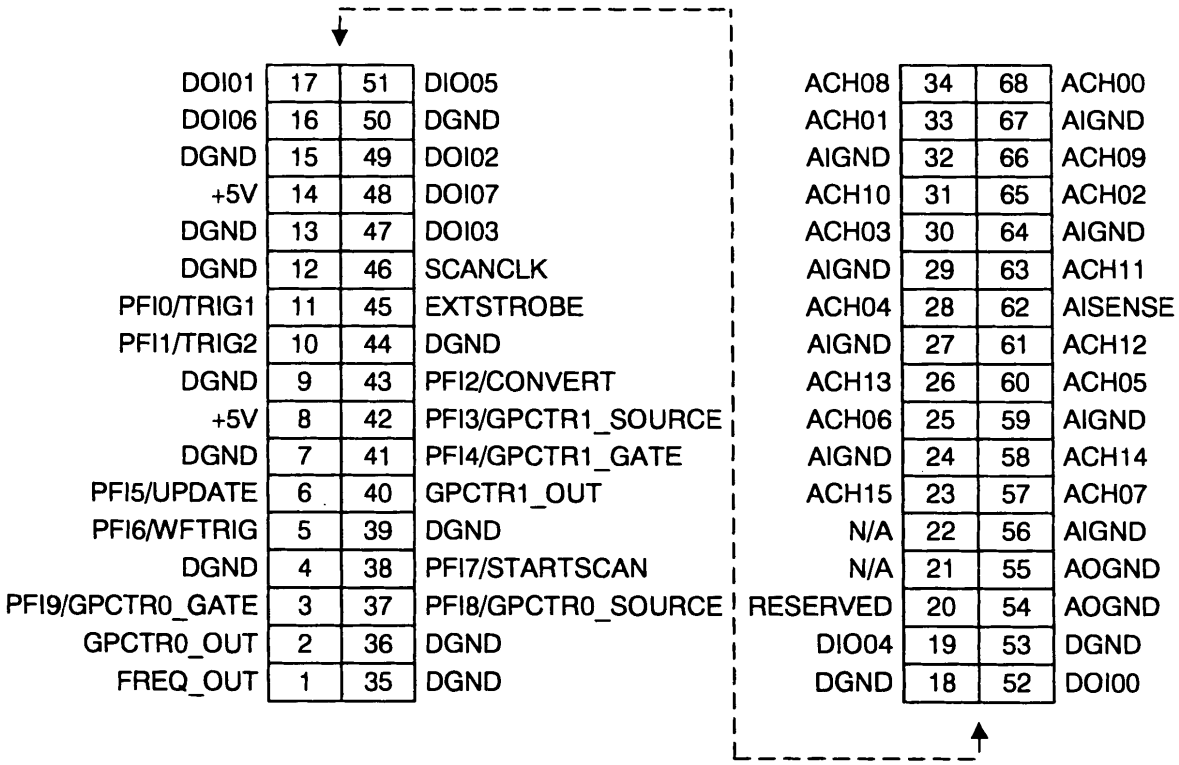


Figure A.8. *PCI-6023E Pinouts.*

A.3 A/D CONVERSION PROCESS

Modern computers operate with digital signals. Analog signals need to be converted to computer interpretable digital signals. This is done by the DAQ device's 12-bit ADC. When converting these analog signals to digital signals certain factors must be considered [19].

- Resolution
- Range
- Signal Limit Settings and
- Sampling Rate

Depending on the DAQ device used, these parameters will be set either physically on the hardware, by using DIP Switches, or using software such as Measurement and Automation Explorer (MAX) from National Instruments.

The number of bits used to represent an analog signal in the digital domain, determines the resolution of an ADC. The more bits used to represent such a signal the better it is due to the increase in resolution and thus produces a better representation thereof. An ADC using 3-bits has 2^3 (8) divisions (represented by 000 to 111 in Figure A.9) and an ADC using 5-bits has 2^5 (32) divisions (represented by 00000 to 11111). Each of these divisions is represented by a binary or digital code between 0 and 2^x. The ADC translates the measured analog signal into one of these digital codes which are then used by the computer for processing and storage. If a fast changing signal has to be converted to a digital signal a 3-bit ADC will not be able to represent the signal effectively. In this case an ADC with higher bit-rate needs to be used e.g. 5-bit, 10-bit or even 16-bit ADC to adequately represent the signal in the digital domain. The significant increase in the representation of the analog signal from 3-bits to 5-bits can be seen in Figure A.9.

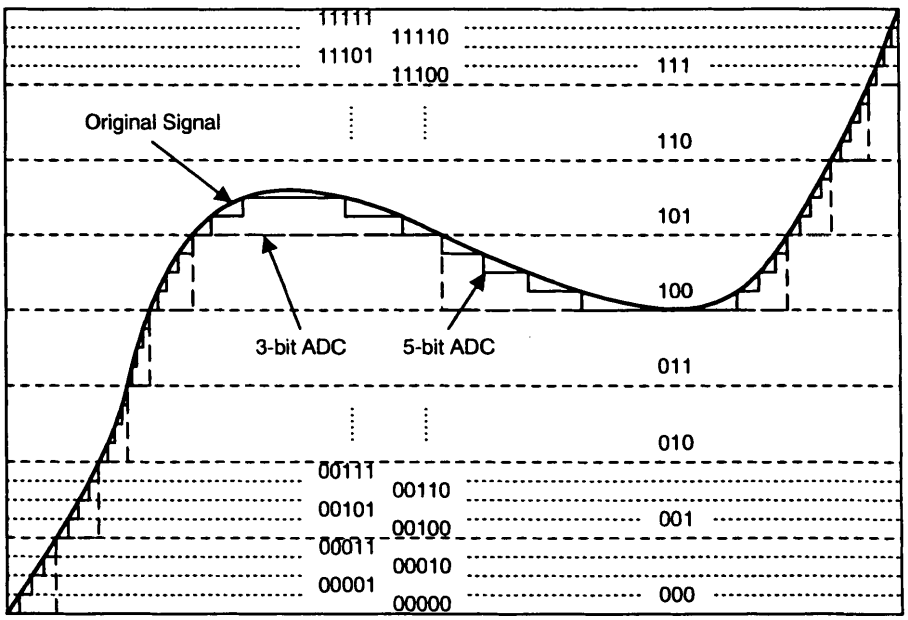


Figure A.9. *3-bit ADC versus 5-bit ADC.*

The minimum and maximum levels that an ADC can handle refer to its range. To take the best advantage of an ADC it should be attempted to match the range of the input signal to the range of the ADC. If a 3-bit ADC is used and the selected range is 0 to 10 V a total of 10 V will have to be divided into 8 (2^3) divisions. This implies that the smallest detectable voltage is 1.25 V. If the same ADC is used but the selected range increases to -10 to +10 V, 20 V will have to be divided into 8 divisions to represent the signal. The smallest detectable voltage increases to 2.5 V thus representing the signal with much less accuracy than before.

The limit settings are the maximum and minimum values of the signal that is measured. The closer the limit setting is to the incoming analog signal maximum and minimum, the more digital divisions will be available to the ADC to represent the signal.

The three above mentioned factors determine the smallest detectable change in the input voltage. A detected change in input voltage represents one least significant bit (1 LSB) of the digital value and is called the code width. The smallest code width (detectable voltage) is calculated with the formula:

$$V_{cw} = \frac{range}{2^{resolution}} \qquad \text{(A.2)}$$

where the range is in bits. For a 3-bit ADC with a range of 0 to 10 V the code width is 1.25 V with range 10 V and resolution of 3-bits. To increase the code width an ADC with higher resolution should be use, e.g. 12-bit with the same range. This results in a code width of 24.41 mV which has drastically increased from 1.25 V.

The sampling rate is the rate at which the ADC samples an incoming analog signal, effectively determining how often an analog-to-digital conversion takes place. The sampling rate is directly proportional to the maximum frequency of the incoming signal. According to the Nyquist criterion when converting an analog signal to a digital signal the signal should be sampled at least twice its maximum frequency. This will ensure that the digital signal is representative of the analog signal and no information is lost during the A/D process however general rule dictates that a signal should be sampled ten times faster than the maximum frequency to obtain a very good and visually accurate representation of the analog signal.

Not complying with the set standard will result in a misrepresentation of the original analog signal which is called an alias of the signal. This phenomenon is discussed in many books about signal processing and will therefore not be discussed further in this dissertation.

APPENDIX B

Software User Manual

This appendix is dedicated to the data acquisition (DAQ) software used to acquire physical data from an installed control valve. All DAQ and management software were programmed in a graphical programming environment called Laboratory Virtual Instrument Engineering Workbench (LabVIEW).

The appendix focuses on the LabVIEW programming environment, presenting a very basic overview of the programming environment used to program the acquisition and signal analysis software. The developed acquisition and analysis software include the DAQ process control application, the data plot application and the post-processing application.

These developed software applications allow any plant manager, plant engineer or operator to effectively obtain physical data from installed components within the plant. The software makes the acquisition process accurate, consistent and simple.

B.1 LABVIEW PROGRAMMING ENVIRONMENT

LabVIEW is a revolutionary graphical programming development environment based on the G programming language for data acquisition and control, data analysis and data presentation.

With LabVIEW, graphical programs called virtual instruments (VI) are *built* instead of *written* as required by text-based programming languages. Programming VIs is a twofold process. The front panel provides an interface with which the graphical user interface (GUI) is created and the block diagram provides a means of connecting and adding functionality to the GUI [19].

Programming actions (such as mathematical functions, string manipulation, matrix arithmetic, file input/output, port communication, program structures and waveform generation) rely on graphic symbols and icons to describe their behaviour. The programming actions can easily be identified by visual inspection. Graphic symbols are interconnected to each other by means of wires.

Once a VI has been programmed, conventional program debugging tools are used to set breakpoints, single-step through the program and animate its execution, making it possible to observe the flow of data.

B.1.1 Panel and Diagram Window

As mentioned each VI comprises two windows. The front panel is the interface to the VI code and the block diagram contains program code that exists in graphical form. Front panels contain various types of inputs and outputs commonly known as controls and indicators. All VIs have an icon and an associated connector residing in the front panel and the block diagram respectively. Controls can be manipulated whereas indicators mostly provide information. Figure B.1 shows a simple VI which graphs previously acquired data on the waveform graph once it has been re-sampled as specified by the *Samples to Read* knob control.

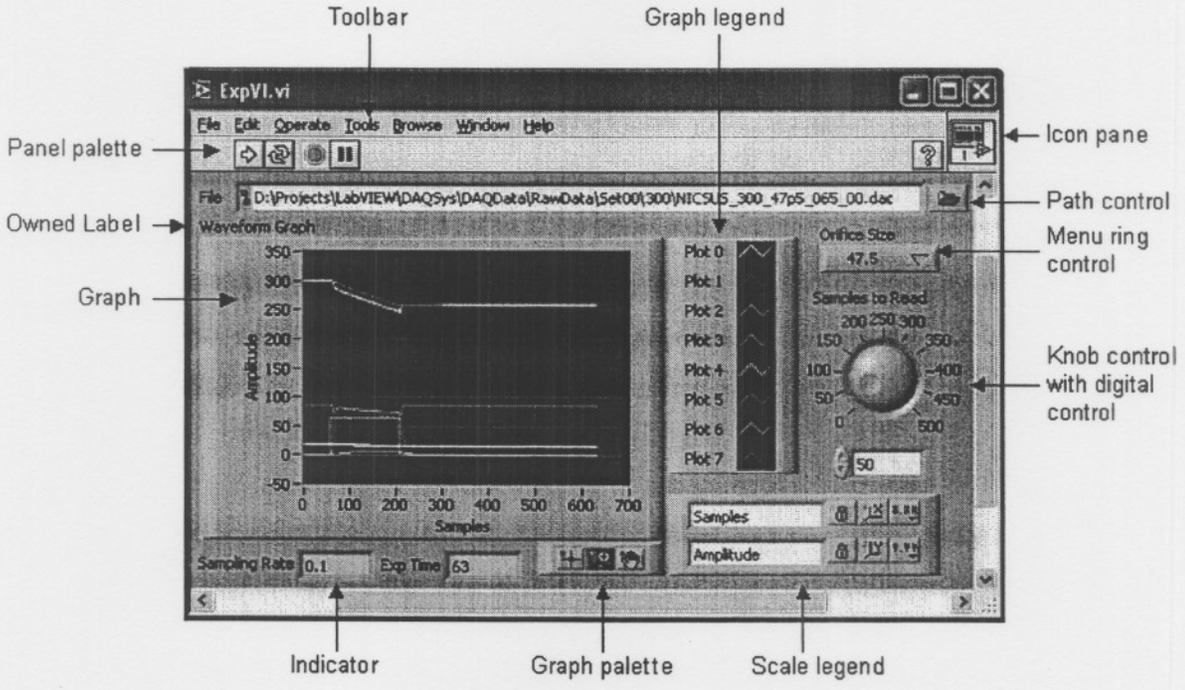


Figure B.1. *VI Front Panel*.

Block diagrams contain terminals corresponding to the front panel controls and indicators as well as constants, functions, subVIs, structures and wires which carry data from one object to another.

Executable icons (called nodes) are connected (or wired) together to create a program flow sequence which defines the operation of the VI. This concept of connecting icons to each other via pathways is known as data flow programming.

Although the block diagram in Figure B.2 might look intimidating at first, its functionality is very simple. A file path is supplied in the *File* field. The data are read from the spreadsheet file into a buffer from which the number of sample sets (size) is obtained. The re-sampling process averages *Samples to Read* samples, effectively reducing the original number of samples by the factor stipulated in the field *Samples to Read*. The sampling rate is calculated by multiplying *Samples to Read* with the original sampling period (0.002).

The valve choke status and mass rate of flow through the valve is calculated each by its own subVI within the *For loop* structure. The calculated data is added to the re-sampled data which is displayed on the *waveform graph*.

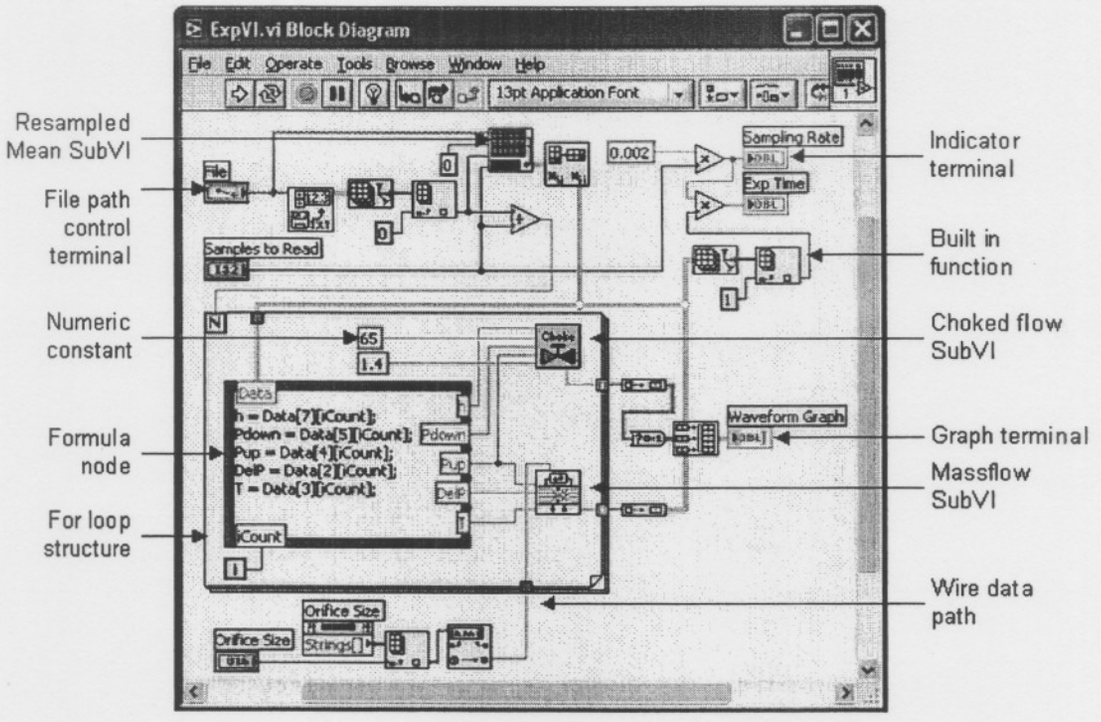


Figure B.2. *VI Block Diagram*.

For more information on the LabVIEW development environment refer to LabVIEW documentation and the LabVIEW student edition textbook [19].

B.2 INSTALLATION PROCEDURE

First and foremost, the developed DAQ software needs to be installed on the terminal computer. The software applications are installed in the following order.

- Process control application (*DAQ Process Control*)
- Data plot application (*Plot Data*)
- Post-processing application (*Post-Processor*)

The installation of the process control application consists of a self explanatory installation-wizard. Run *setup.exe* from the install path (refer to *Appendix D*) and follow the prompts. This entails selecting an installation path and the desired features. It is important to note that with the DAQ Process Control application the much needed LabVIEW Run-time engine must also be installed. This is a compact version of the LabVIEW interpreter and a prerequisite for any other executable VIs.

Once the process control application and the LabVIEW Run-time engine have been installed, copy the data plot application (*PlotData.exe*) and the post-processing application (*Post-Processor vx.x.x.exe*) into the same install directory to form the complete DAQ system. This DAQ system is capable of acquiring, processing and displaying data.

All three applications, used to acquire, process and display the data, are covered in the following sections. A user manual for each of the applications is discussed.

B.3 DAQ PROCESS CONTROL APPLICATION

The Process control application is the fundamental utility used in the DAQ process. This application enables quick assessment of the system status, streamlines the acquisition of physical data and enables diverse settings to be applied to experiments conducted.

Execute *DAQ Process Control.exe* (or a shortcut pointing to this executable) from the command line, start menu or desktop. A screen similar to Figure 3.5 is displayed.

The DAQ process control GUI is divided into four main sections:

- Pipe network setup
- DAQ system control
- System graph
- Quick reference help system

The DAQ process control application generates two experiment files (*.dac and *.log) which are used to save experiment data and settings.

B.3.1 Pipe Network Setup

The pipe network setup in Figure B.3 is a graphical representation of the physical pipe network used to acquire real-world data of the control valve.

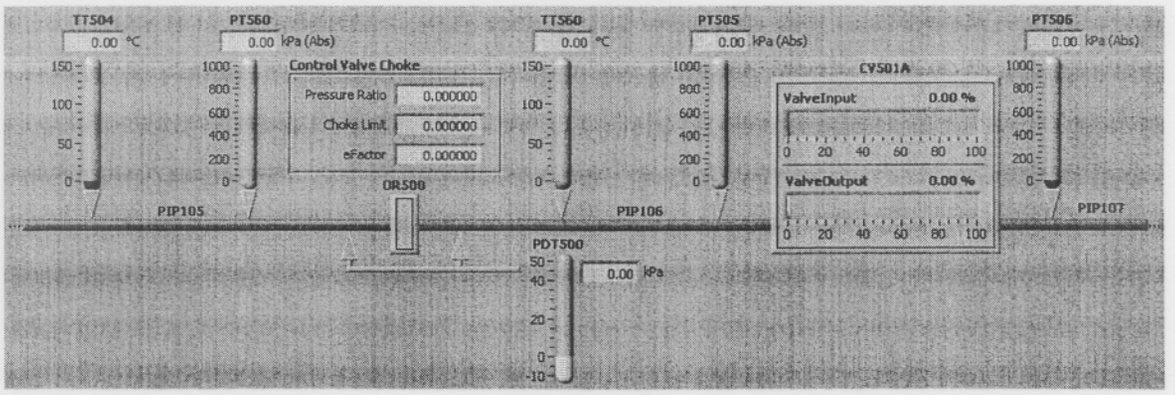


Figure B.3. *Pipe Network Setup*.

Each indicator represents a transducer be it a pressure transducer, temperature transducer or a control valve relative opening transmitter. The indicator scales are set to the maximum of the transducer limits but may be changed to any other value. These default limits will re-initialize each time the *DAQ Process Control* application is executed.

Above the orifice (*OR500*), the control valve (*CV501A*) choke status calculation values can be seen. These indicate when the flow through the control valve is in choke. Once the *Choke Limit* is smaller than the *Pressure Ratio* (refer to section 3.6.2) the flow through the control valve (*CV501A*) is in choke. Both orifice and control valve have the ability to indicate when its flow is in choke. The orifice and control valve changes from grey to red to indicate that the flow through the device is in choke. During non-choke operation, the colour of the orifice and control valve remains grey.

B.3.2 DAQ Experiment Control

The DAQ experiment control is divided into the following six sections as shown in Figure B.4 (*a*) and (*b*).

- System control
- Sample info
- DAQ file info
- DAQ read info
- DAQ read settings
- DAQ additional info

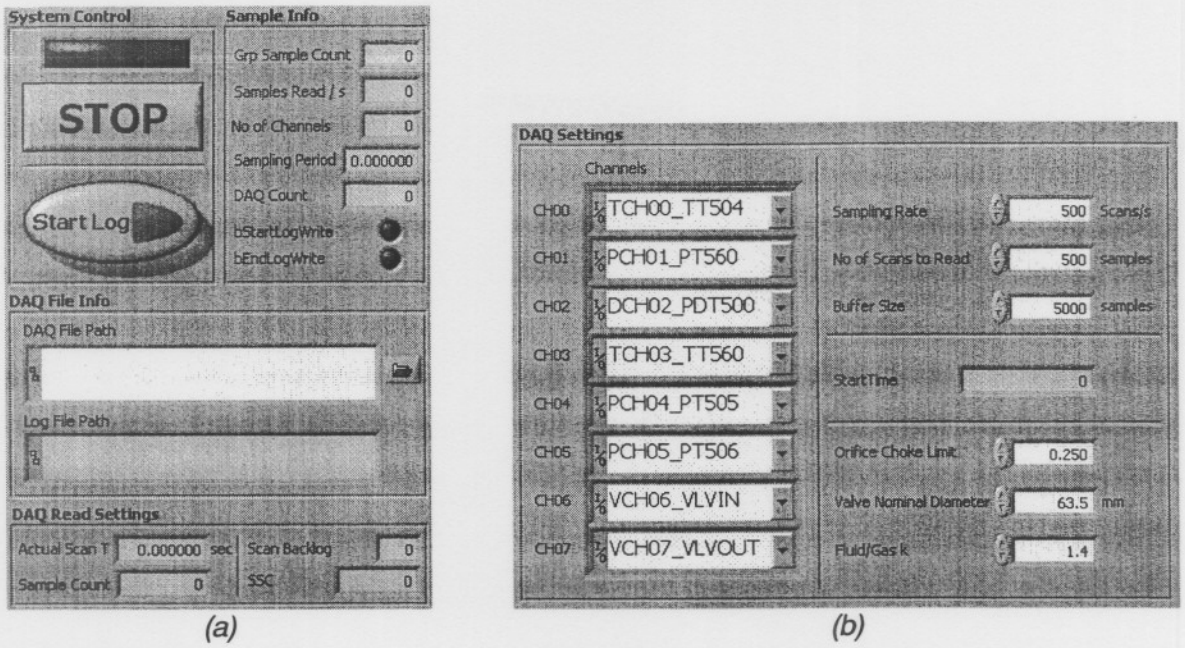


Figure B.4. *DAQ System Control.*

B.3.2.1 System Control

The process control application commences once the *run* button on the toolbar has been pressed. The green light above the stop button illuminates to indicate the process has been started. Data logging can be started by pressing the *Start Log* button provided the necessary settings have been correctly set up. Once the logging process has been started it can be stopped by pressing the *Stop Log* button.

B.3.2.2 Sample Info

The *Sample Info* section indicates DAQ process status. *Grp Sample Count* (Group sample count) indicates the number of times all channels have been acquired or scanned. *Samples Read / s* indicates the number of samples read per second and in effect resembles the *Sampling Rate* selected by the user for the specific experiment. *No. of Channels* indicates the number of channels which is currently used for data acquisition (the default is eight channels). *Sampling Period* is the effective system sampling period and also inversely proportional to the *Sampling Rate*. *DAQ Count* is the number of times the channels have been sampled during a data logging session. *bStartLogWrite* and *bEndLogWrite* are status flags used to determine the state of the logging process and has no significance other than indicating the completion of writing the log file.

B.3.2.3 DAQ File Info

DAQ File Info specifies the current data logging file and its accompanying experiment log file. The default extension for the DAQ file is ∗.dac and the log file is ∗.log. Only the DAQ file extension can be changed according to user requirements.

B.3.2.4 DAQ Read Settings

Actual Scan T indicates the actual system sampling period per channel. The number of samples logged to file is indicated by *Sample Count*. *Scan Backlog* is the amount of data remaining in the scan buffer. If *Scan Backlog* increases steadily, channel scanning is too fast and buffer reading of the scanned data too slow. Increasing the *No. of Scans to Read*, decreasing the *Sampling Rate* or increasing the *Buffer Size* will correct this problem. The group sample count at which the logging process is started is shown in the *SSC* field (Start Sample Count).

B.3.2.5 DAQ Settings

Provided that all virtual channels have been created in MAX (refer to section 3.5.1) and are connected to the appropriate channels on the DAQ device, it is strongly advised to check and/or adjust the experiment settings to ensure accurate results.

Select the appropriate channel connections from the dropdown list as determined by the physical connection on the DAQ device and virtual channel settings. Repeat the process for all channels from *CH00* to *CH07*.

The selected order of the channels (*CH00* being first and *CH07* being last) determines the order in which the channels are scanned and saved to file during the acquisition process. *CH00* is scanned and saved first and *CH07* scanned and saved last, which implies that *CH00* is the first column and *CH07* the last column within the acquisition file.

The *Sampling Rate* determines the number of samples to acquire per second for each channel. The user selects the *No. of Scans to Read* which has to be less or equal than the *Sampling Rate* to ensure that only valid data will be read. The *Buffer Size* is used to temporarily store the acquired data before reading and saving it to file. A circular buffer is used which implies that its head and tail are virtually connected. *Buffer Size* selection should be done so as to avoid overflow of old data with newly acquired data due to too low selections of the *No. of Scans to Read*.

StartTime refers to the time the DAQ process is started. This is used in conjunction with an *EndTime* variable to calculate the total acquisition process time in milliseconds.

According to the British Standards [6], an orifice is in choke when the ratio of its differential pressure to its upstream pressure is greater or equal to the *Orifice Choke Limit*. This choke limit is normally 0.25. The *Valve Nominal Diameter* and *Fluid/Gas k* together with other acquired data are used to calculate whether the flow through the valve, at a certain relative valve opening, is in choke or not. Refer to section 3.6.2 for more detail on choked flow through orifices and valves.

B.3.2.6 DAQ Additional Info

This field allows the user to supply additional information about the experiment being conducted. To avoid confusion about obtaining physical data of the control valve a template was created. This should be altered according to the experiment being conducted (refer to section 3.8). The template, in Figure B.5, supplies the experiment name (*NICS Blowdown Unit Step Exp*), maximum valve relative opening (*Valve Max h*), experimental set number (*Exp Set*), control valve sequence (*CV501A Sequence*), experiment initial pressure (*Initial Pressure*) and orifice diameter (*Orifice Diameter*).

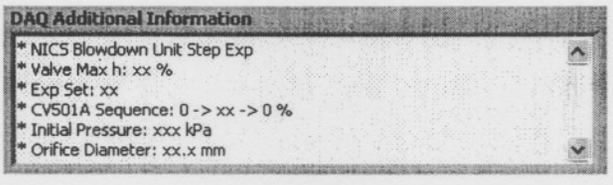


Figure B.5. *DAQ Additional Information*.

B.3.3 System Graph

While the DAQ process control application is executing, whether or not data are being logged to file, the acquired real-time data trends are graphed on the *System Graph*. This enables visual inspection of the system's performance, status and experimental procedures.

Refer to Figure B.6 for the layout of the *System Graph*. Notice that all eight channels have a digital display on the right side of the graph, which is used to indicate the current status of the transducer. Next to the digital displays is the graph legend, which indicates the channel names, their trend colours and line styles. In the bottom right hand corner is the graph palette and scale legend. The graph palette is used to zoom and pan the trend data whereas the scale legend is used to scale, lock and format axes.

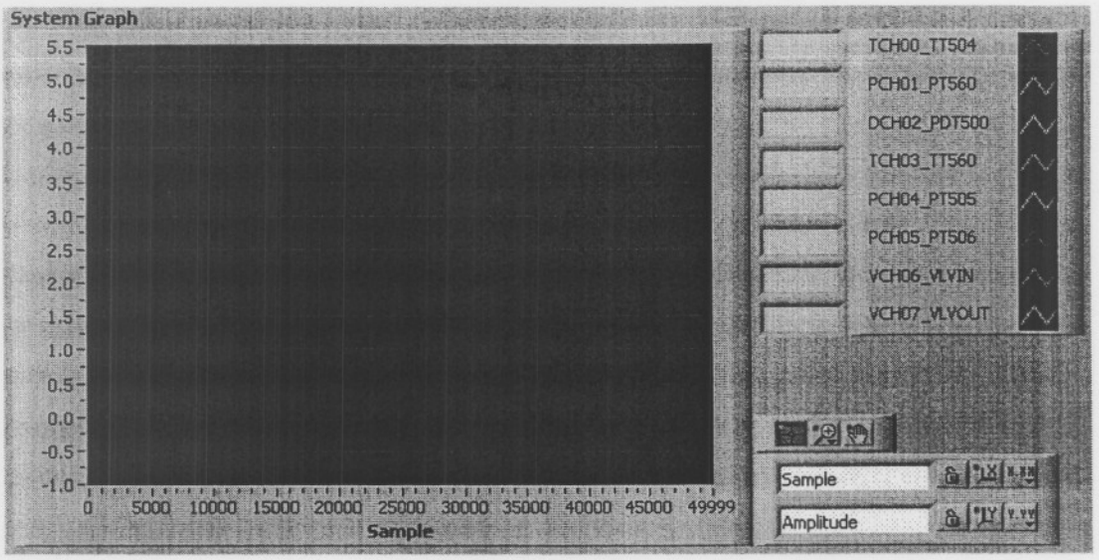


Figure B.6. *System Graph*.

B.3.4 Quick Reference Help System

The reference guide helps in setting up system parameters without having to know the entire system in detail. By following the instructions the operator can set up the system parameters with relative ease and start acquiring data within a very short period of time.

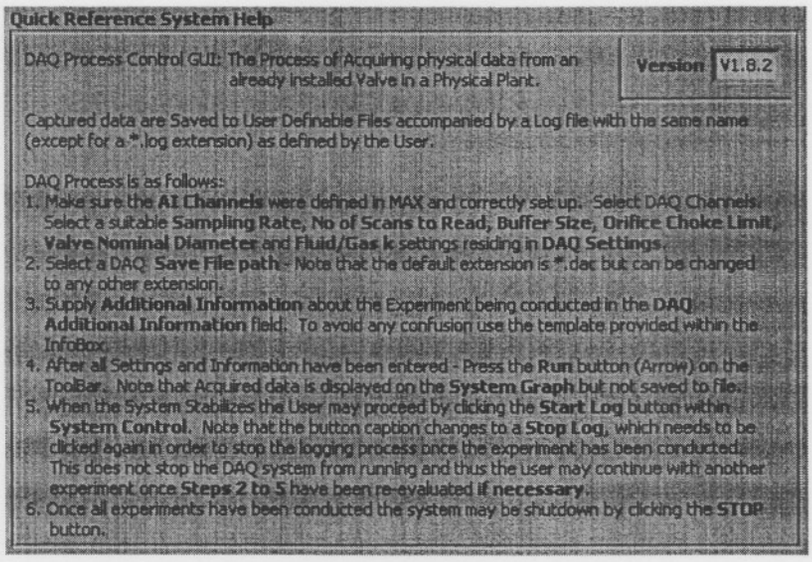


Figure B.7. *Quick Reference Help System*.

B.3.5 Experiment Files

During the data logging process, two files are created. The first is a *dac* file, which is used to save acquired data and the second is the *log* file, used to save experiment settings and data.

The *dac* file has eight columns, each representing an acquired channel (refer to B.3.2.5). A typical log file can be seen in Figure B.8 where experiment data include the date and time of the acquisition process, the initial pressure used in the experiment, the valve sequence, sampling rate, number of channels and number of samples logged to file.

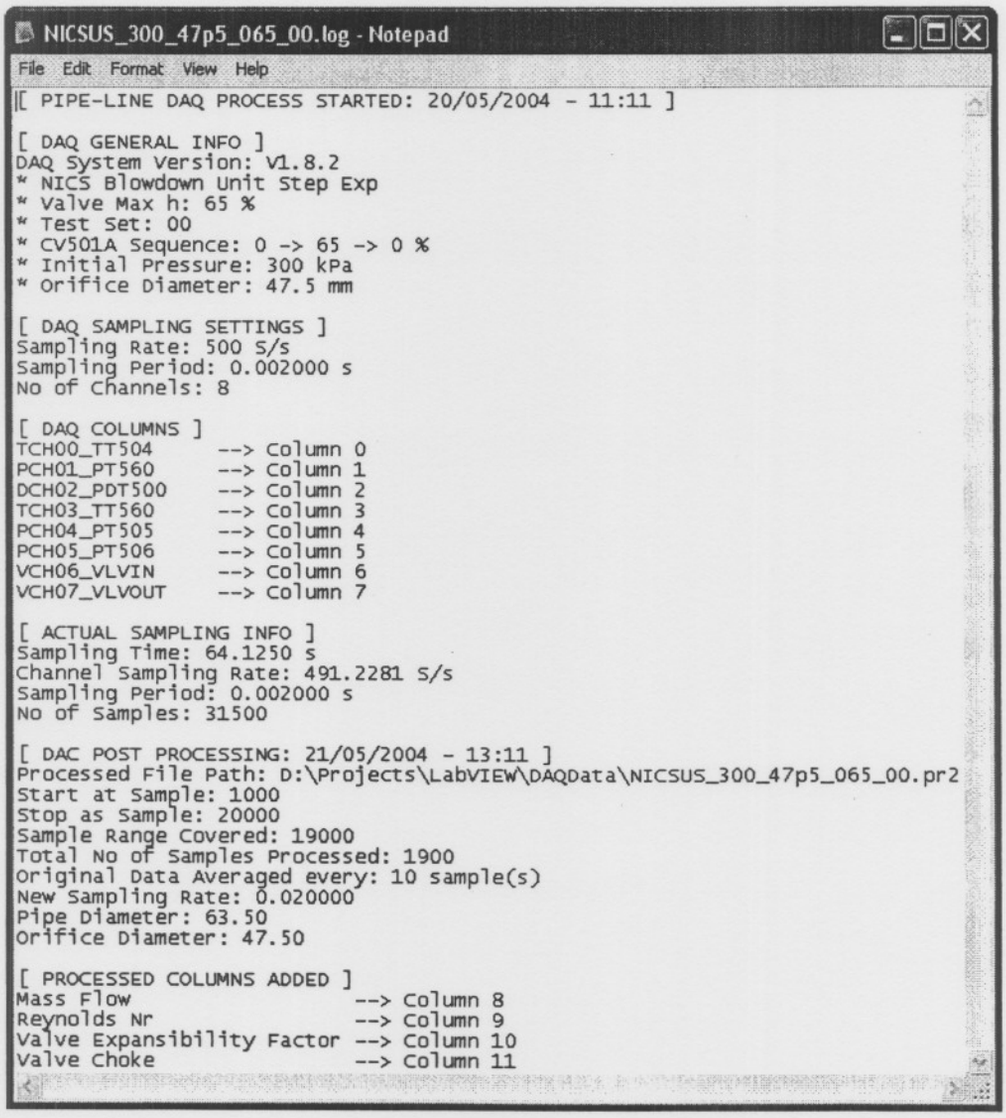


Figure B.8. *Experiment Log File*.

The *DAQ General Info* section supplies the current DAQ process control version number and the information given by the user in the *DAQ Additional Information* field (refer to section B.3.2.6). The sampling rate and period as well as the number of channels used in the experiment can be seen in the *DAQ Sampling Settings* section. The experiment channels, their names and column position in the *dac* file can be seen in the *DAQ columns* section. The *Actual Sampling Info* section supplies the acquisition time, the channel sampling rate, sampling period and the number of samples acquired.

The post-processing application also appends information to the *log* file. User specified settings are added, including the processing file path, the start and stop samples, the number of samples to average, pipe diameter and the orifice size. Calculated parameters are also added which include the sample range covered, total number of samples processed and new sampling rate. The *Processed Columns Added* section supplies the newly added processed channel names and their positions in the *dac* file (refer to section B.5).

B.4 DATA PLOT APPLICATION

During the acquisition process, the data at all the nodes are shown on the trend graph Figure B.6. The data plot application enables viewing of the acquired data and its associated log file once the experiments have been conducted. The application assists in deciding which sequence of samples are best suited for processing via the post-processor, eliminating unnecessary data samples which could lead to prolonged fuzzy logic system (FLS) training.

Execute *PlotData.exe* (or a shortcut pointing to this executable) from the command line, start menu, or desktop. A screen similar to Figure 3.7 is shown. Three sections exist within the data plot application. These include the original data section, the processed data section and the log file section. Due to many similarities between the original and processed data sections, these are covered together.

B.4.1 Original and Processed Data Sections

Supply a valid file name in the *DAQ Path* field and click the *run* button on the toolbar. Select one of the data section tabs to display the required data. The basic building blocks of both original and processed data sections are the waveform graph, data range sections and the data averaging sections.

B.4.1.1 Waveform Graph

These graphs are very similar to the system graph of the DAQ process control application except for data now read from the acquired file instead of from the DAQ device buffers as in the case of the process control application.

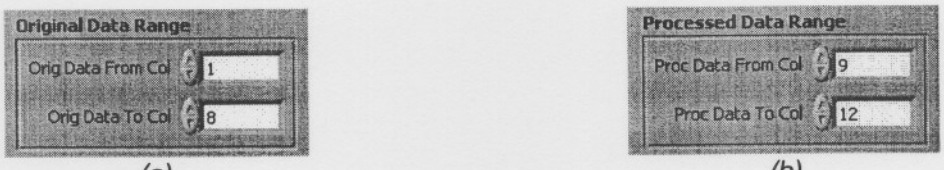
Each waveform graph consists of a viewable area, graph legend, scale legend, graph palette and cursor legend. The viewable area is used to plot the acquired data. The graph legend in the top right corner indicates the channel names, trend colours and line styles. All trends have the ability to be invisible by setting their trend colour to transparent. Revert back to the default trend colour by using the channel colours supplied next to the graph legend.

The graph palette, in the bottom right corner, is used to zoom and pan the trend data whereas the scale legend, below the graph palette, is used to scale, lock and format axes.

The cursor legend, below the viewable area to the right, is used to query trend values. A maximum of three cursors can be used to examine the trends. The cursor legend indicates the cursor name and its *x* and *y* coordinates. It also has the ability to select a cursor or multiple cursors, move the selected cursor(s) up, down, left or right, format cursor appearance and lock to a specific trend.

B.4.1.2 Data Range Sections

The original and processed data ranges respectively, select which columns (trends) within the acquisition file are original data and processed data. Figure B.9 (*a*) indicates that columns 1 to 8 are original data and (*b*) that columns 9 to 12 are processed data. This effectively selects which trends are viewed in which data section.



(a) *(b)*

Figure B.9. *Data Ranges*.

B.4.1.3 Data Average Sections

Along with the cursor legend, which queries trend values, three trend averages can also be used to query trend average values. These averages and their settings are to the right of the viewable area between the graph legend and scale palette (refer to Figure B.10 *(a)* for the original data section and *(b)* for the processed data section).

Within the original data section the settings can be set up by supplying all three sets with a *Start* and *End* sample. Select a trend, for which these averages are to be calculated, in the *Orig Data Trend* field for the original data section and *Proc Data Trend* field in the processed data section respectively. Click on the *run* button to calculate the averages which is then shown in the *Orig Data Ave* and *Proc Data Ave* for the original data section and processed data section respectively.

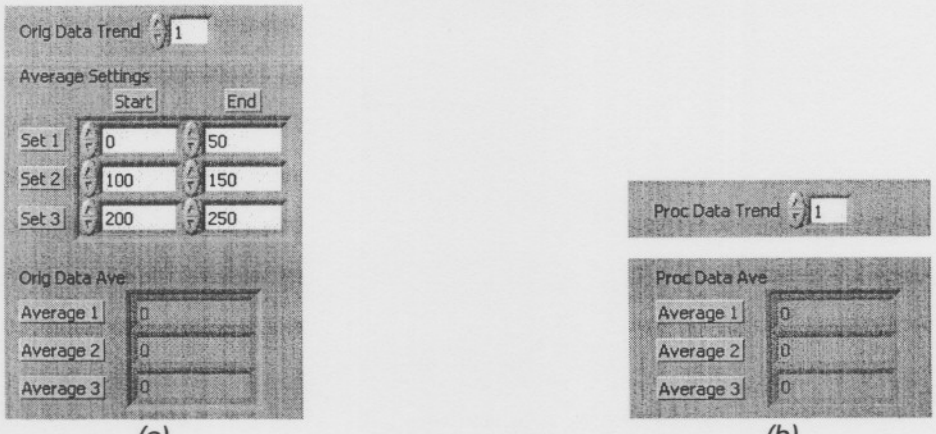


Figure B.10. *Data Average Settings and Values*.

B.4.2 Log File Section

The log file section displays the current log file associated with the *dac* file. The current log file path is shown in the *Log File Path* field. If no *log* file is available this section remains empty. Refer to section B.3.5 for the *log* file structure and entries.

B.5 POST-PROCESSING APPLICATION

The post-processing application is used to resample the raw acquired data, to obtain more compact files which are used for training purposes. Re-sampling the raw data entails averaging a number of samples and saving it to a new file. The re-sampled data are used to calculate the values of the governing equations (mass rate of flow through pipe-network, Reynolds number, expansibility factor and valve and orifice choke status) which are also saved to the file.

Execute *Post-processor.exe* or a shortcut pointing to the executable, to show a screen similar to Figure 3.6. The *Post-processor* application is divided into six sections:

- File information
- Processing settings
- Re-sampled data
- Processed data
- System parameters
- System status

B.5.1 File Information Section

This section controls the file settings. The original input file, containing the raw acquired data, is supplied in this section as well as the new base path and processed file extension. Refer to Figure B.11.

Supply a valid original raw acquired file in the *Input File* field, a new base path (without a file name) in the *Base Path* field and processing file extension in the *Ext* field. The processed file has the same name as the original input file except for a different file extension as specified in the *Ext* field. Use the default extension with different end-numbers to distinguish between consecutive processed files.

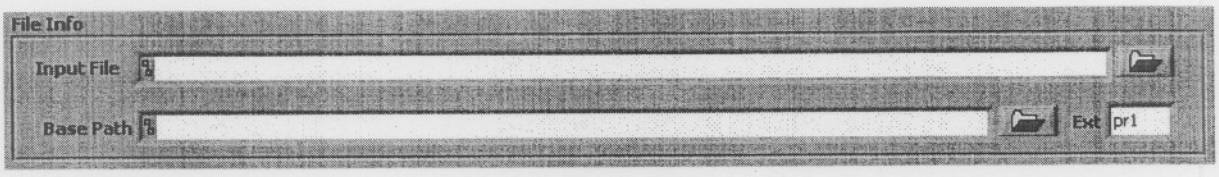


Figure B.11. *Post-Processing File Information*.

B.5.2 Processing Settings Section

The settings used to determine the sequence of samples to process, whether to shift the differential pressure trend around zero, the pipe and orifice diameter and the number of samples to average are set up in this section (refer to Figure B.12). These settings determine the behaviour of the post-processor during sample processing.

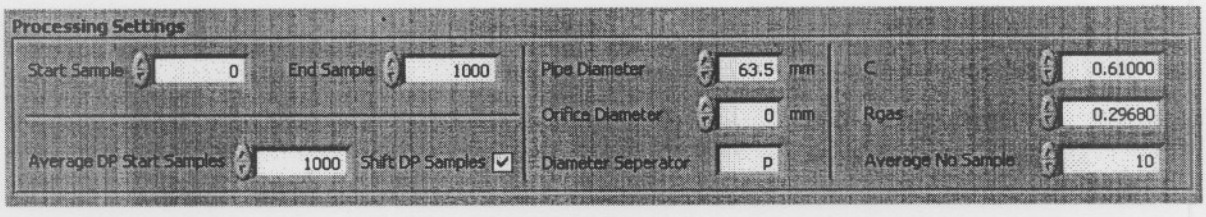


Figure B.12. *Processing Settings*.

Select the start and end sample from *Start Sample* and *End Sample* respectively. All the samples between and including the start and end samples are processed. Due to an offset in the differential pressure sensor output, a selectable number of samples (*Average DP Start Samples*) are used to calculate an average with which the data can be shifted about the zero point. This feature can be selected by checking the *Shift DP Sample* check box. Due to the occurrence of this offset *Shift DP Samples* is checked by default.

The pipe inner diameter is entered into the *Pipe Diameter* field. The default is a 63.5 mm (2.5 inch) pipe. Select the system orifice diameter in the *Orifice Diameter* field. If the *Orifice Diameter* is changed to 0, the system will try to locate the diameter within the input file name. The orifice diameter should be separated with a *Diameter Separator* e.g. 47p5, where *p* is the separator and the abbreviation for *point*. If the orifice diameter is greater than zero the entered value will be used instead of the value embedded within the filename.

Select system parameters; discharge coefficient (*C*), gas constant (R_{gas}) and the number of samples to average (*Average No Samples*). A selectable number of samples (*Average No Samples*) are used to calculate the solutions for the governing equations. This reduces the number of training samples by the factor: *Average No Samples*.

B.5.3 Re-Sampled Data Section

The re-sampled (averaged and/or shifted) data are shown in the *DAQ Data* field and can be seen in Figure B.13. The data columns (each representing a trend) are a one-to-one representation of the processed file layout, which implies that column one (*CH00*) is dedicated to *TT504* and column eight (*CH07*) dedicated to *VLVOUT*.

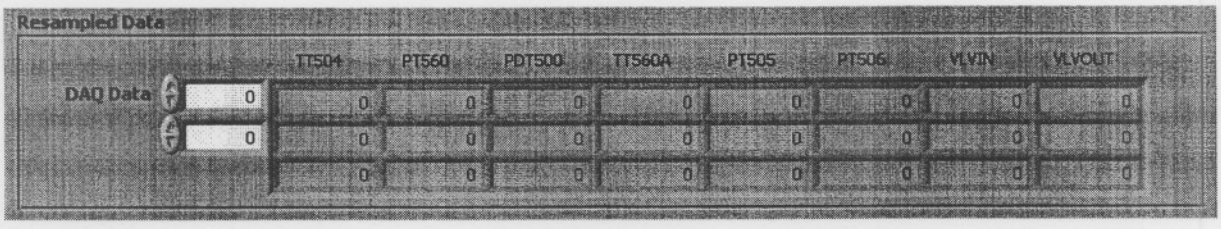


Figure B.13. *Re-sampled Data*.

B.5.4 Processed Data Section

This section indicates the current processed data which are saved to the file. This together with the originally averaged data comprises the processed file which is used to train, validate and verify the FLS. *Mass flow* is saved to column 9 and *Valve Choke* to column 12. Refer to Figure B.14 for the processed data added to processed file.

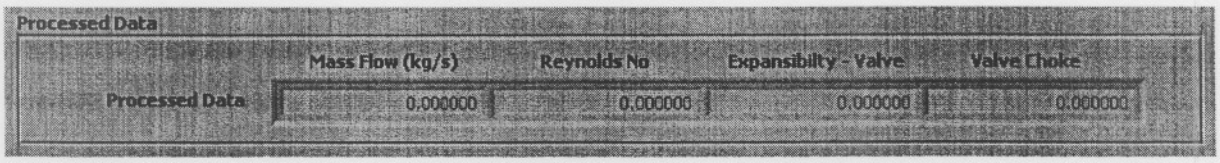


Figure B.14. *Processed Data*.

B.5.5 System Parameters Section

The system parameters section (Figure B.15) indicates the current experiment settings either read from the *log* file or calculated from processing settings. *Sample Count* indicates the current number of samples processed once the post-processor has been started. *Sample No Processed* indicates the current sample being processed. *DAQ File Size* indicates the input file size, thus the number of samples acquired during the acquisition process. The sampling rate at which the physical data were initially captured is read from the *log* file and displayed in the original sampling period (*Original Sampling T*) field. The new sampling period (*New Sampling T*) is calculated by multiplying the *Original Sampling T* by *Average No Samples*. The re-sampling of the original data creates a file with less data samples whilst still retaining all relevant information concerning valve behaviour. These processed files are used to train, verify and validate the FLS. The *Orifice Diameter* indicates the currently installed orifice diameter used in the governing equations calculations. The orifice downstream pressure is shown in *Pdown* and the valve differential pressure, which indicates the differential pressure across the valve, can be seen in the *Valve Diff Pressure* field. Both the latter parameters are calculated for completeness but not saved to file. The differential pressure average (*DP Average*) across the orifice is calculated from the first number of samples, specified by *Average DP Start Samples*, found in the *Input File*. This is used to shift the differential pressure about zero if so desired.

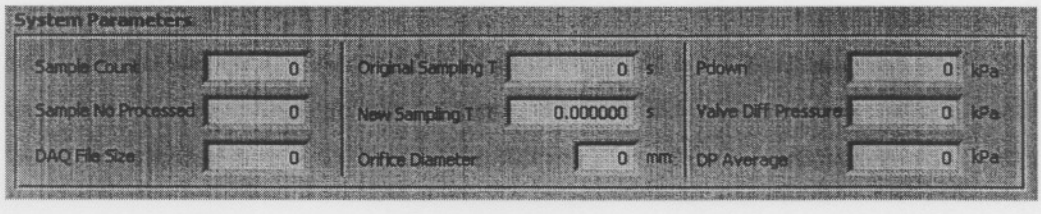


Figure B.15. *System Parameters*.

B.5.6 System Status Section

Once the post-processor has been started the *System Status* indicator turns red and the boolean value reads *Busy*. If the processing has been completed it returns to green and reads *Ready* as indicated in Figure B.16.

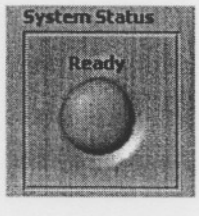


Figure B.16. *System Status*.

APPENDIX C

Fuzzy Logic NNC Matlab GUI and Code

The nearest neighbourhood clustering (NNC) algorithm is covered in *Chapter 2*. This algorithm is capable of creating a number of clusters based on the set parameters. These clusters are then used as training data for the fuzzy logic system (FLS), where all cluster input-output pairs are matched to any given accuracy. This is called an adaptive fuzzy logic system (refer to section 2.4).

The adaptive fuzzy logic system is divided into three main fuzzy applications i.e. *FuzzyTrain*, *FuzzyRead* and *FuzzyValidate*. Each of these fuzzy applications comprises a number of modular functions which will be covered in this appendix. A graphical user interface (GUI) is used to simplify the usage of all three fuzzy applications.

According to [25], the definition of *validation* is defined as the evidence that demonstrates that the code or calculation method is fit for its purpose. On the other hand *verification* is the process of ensuring that the controlling physical equations have been correctly translated into computer code.

Irrespective of these definitions for validation and verification, in this dissertation verification implies that previous data, which the system trained with, are verified to obtain a final training verification error. This can significantly differ from the training error which was calculated for the fuzzy system during the training process. Validate on the other hand implies that data, not previously encountered by the fuzzy logic system (FLS), are used to obtain a validation error. This validation error is used to determine the efficiency of the trained FLS.

C.1 FUZZYTRAIN

The *FuzzyTrain* application consists of two main parts. In the first part (*training process*) the acquired data files are used to train the FLS, using the nearest neighbourhood clustering algorithm. The second part re-evaluates the trained data, in what is called the *verification process*, to calculate the training errors. These training errors are used to determine the effectiveness of the fuzzy system and to determine whether training parameters need to be adjusted in order to achieve a more accurate system.

C.1.1 Training Process

The first part of the *FuzzyTrain* software flow diagram can be seen in Figure C.1. Firstly the software starts by obtaining the number of files within the specified *Base Path* as well as the user selectable percentage of training files (*TrainNum*). A decision is made as to whether the random files sequence is to be created automatically or initialised from the user definable number string. This will be covered in more detail in section C.4.2. Thereafter the training file sequence (*TrainFileSeq*) is determined from the random file sequence (*RandFileSeq*).

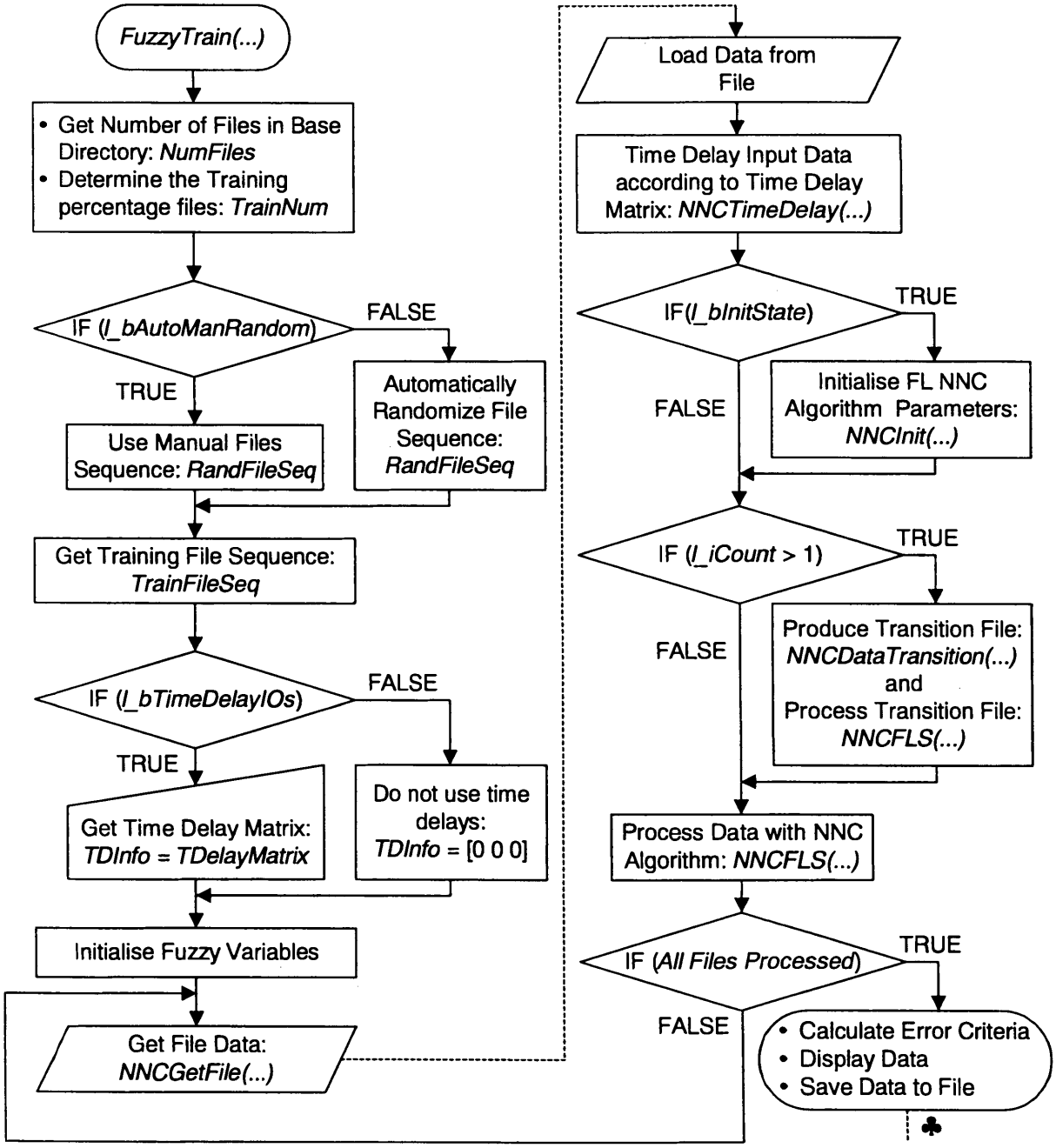


Figure C.1. *FuzzyTrain – Training Process Flow Diagram.*

The decision on whether or not to implement time delays within the system is then evaluated. If so, a user selectable time delay matrix is used to calculate the time delay inputs and outputs. Thereafter the initial fuzzy logic parameters are initialised. The specific file path is obtained, with the *NNCGetFile* function (refer to section C.4.1), from which the data is then read and loaded into the system variables. The *NNCTimeDelay* function (refer to section C.4.2) in conjunction with the applied time delay matrix, is then used to create the time delayed inputs and outputs. If the first file is to be processed, the nearest neighbourhood algorithm parameters are initialised. As soon as the current file has been processed, but prior to the next file being processed, a transition file is generated (refer to section C.4.5) and processed. The aim of the transition file is to obtain a smooth transition from one file to the next instead of an abrupt changeover which can cause the FLS to behave inappropriately.

The training process loop continues until all files have been processed after which the training data and FLS errors are saved to file.

C.1.2 Verification Process

The second part of the *FuzzyTrain* application is the verification process which follows the training process. The aim of the verification process is to verify the previously trained files and thus to calculate the final training errors. These errors will be used to define the effectiveness of the FLS. The verification process software flow diagram can be seen in Figure C.2 and continues from the flow diagram in Figure C.1.

Firstly the saved data from the training process is read from the file specified in the *Save Filename* field. Thereafter the first file name is composed from the first file within the *Base Path* directory. The data from this file is loaded into the system variables. The loaded data are then delayed with the *NNCTimeDelay* function and the time delay matrix. The first file is verified with the *NNCVerify* function and the process is repeated until all files have been verified. Between two consecutive data files a transition file is created and verified to allow for a smooth transition from one data file to the next. This transition file is also verified with the *NNCVerify* function.

The training process errors (section C.1.1) cannot be used to determine the effectiveness of the FLS because these errors were calculated during the training process and thus does not reflect the true nature of the trained fuzzy system. Therefore once all files have been verified the FLS verification errors are calculated and along with the verification data saved to the same file as that of the training process. These verification errors are known as the final training errors and used to determine the effectiveness of the FLS.

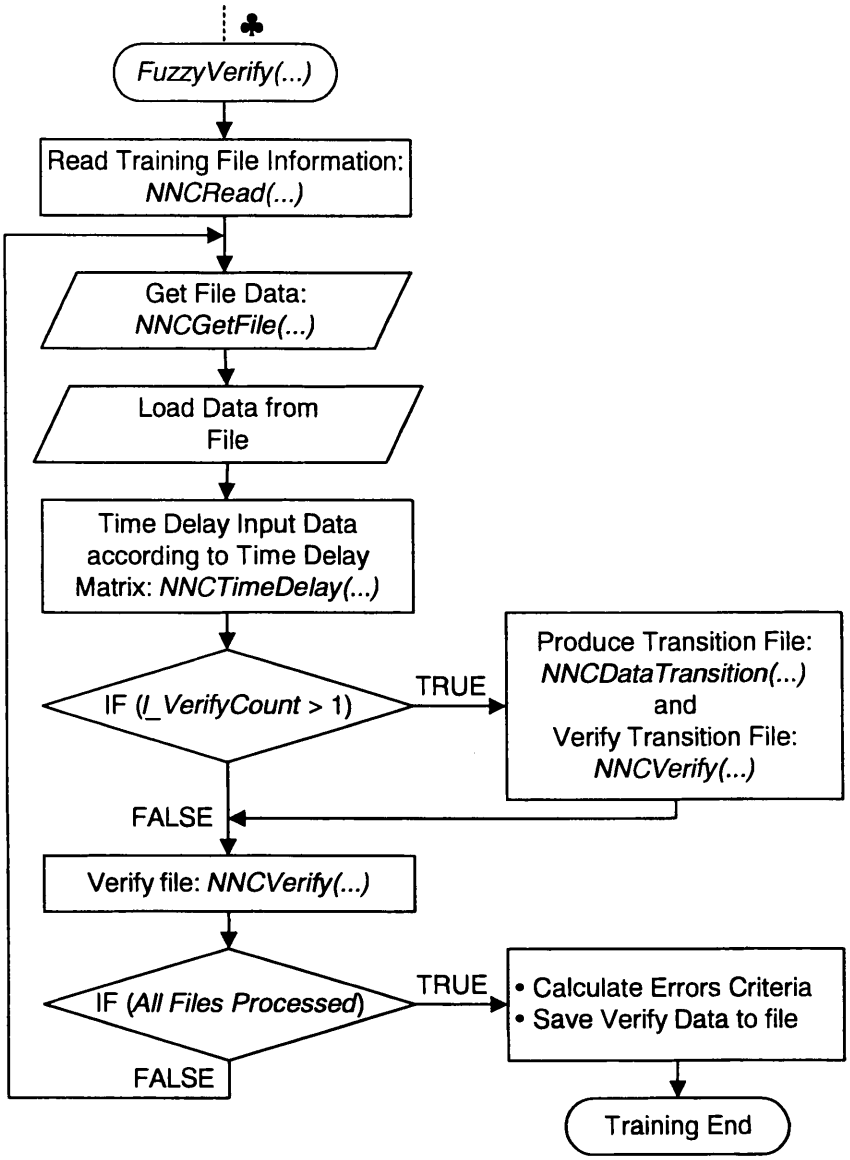


Figure C.2. *FuzzyTrain – Verification Process Flow Diagram.*

C.2 FUZZYREAD

The *FuzzyRead* application reads the FLS training settings, training results and verification results from the specified file. The *FuzzyRead* application calls the *NNCRead* function to obtain the saved information regarding the specific FLS. The *NNCRead* function will be covered in sections C.4.7 and C.5.2.

C.3 FUZZYVALIDATE

The *FuzzyValidate* application validates files by using a pre-trained FLS. In the verification process, error criteria of the entire system are calculated. The *FuzzyValidate* application calculates the error criteria on a file to file basis. This implies that each file has a set

of error criteria which will be used to validate the FLS using data which have not been encountered by the FLS in the training process. The *FuzzyValidate* application software flow diagram can be seen in Figure C.3.

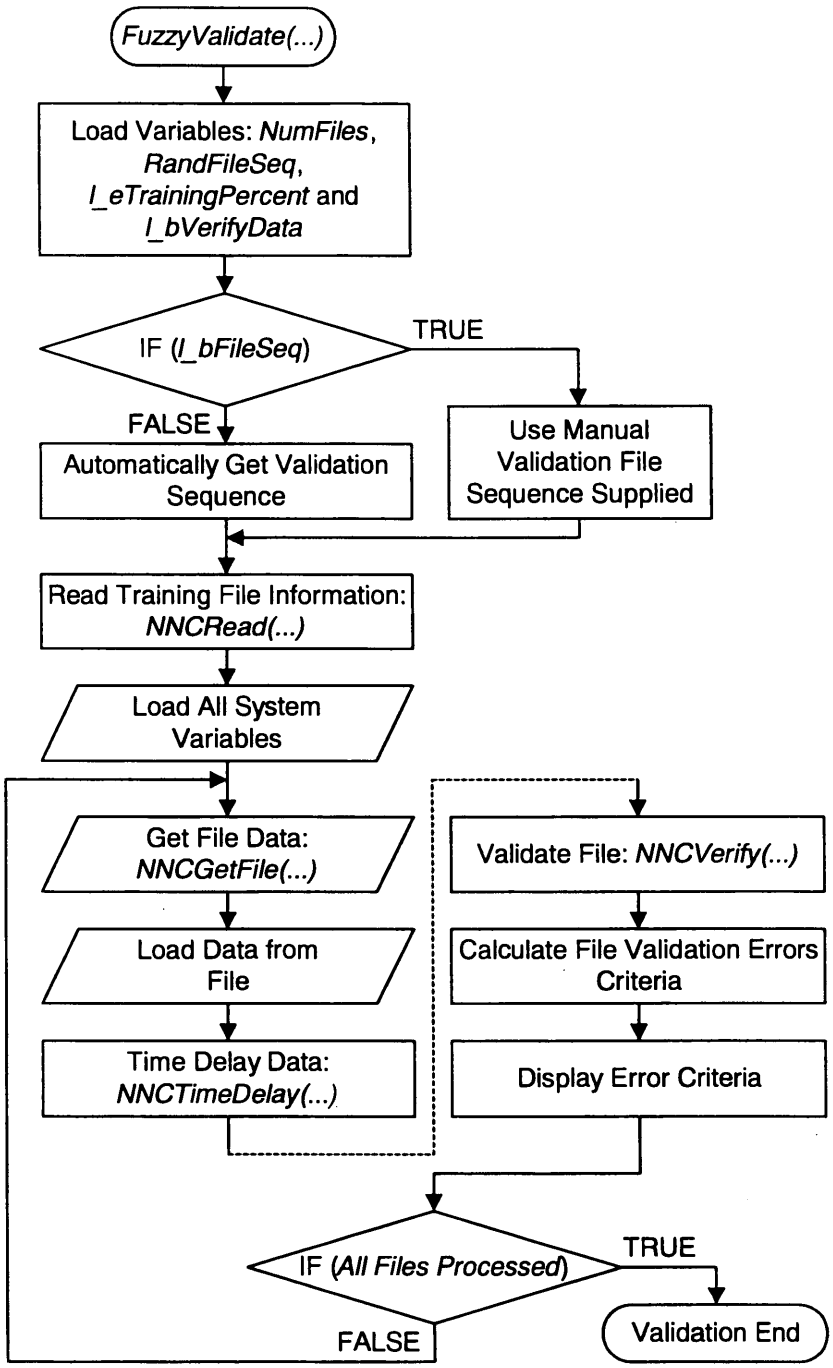


Figure C.3. *FuzzyValidate Application Flow Diagram.*

The fuzzy validation application starts by loading a number of system variables. This is followed by a decision on whether to use a manual validation file sequence or the previously generated validation file sequence. This will be covered in the fuzzy (Graphical User Interface) GUI section C.5.3. The FLS information is read by the *NNCRead* function followed by loading

the FLS variables. The first file data are loaded into the system variables and time delayed versions are calculated by the *NNCTimeDelay* function. The time delayed data are then verified by the *NNCVerify* function followed by the calculation of the error criteria which are then displayed. The process continues until all the specified files have been validated.

C.4 MODULAR FUNCTIONS

Each of the fuzzy applications makes use of a number of modular functions. These modular functions are generically developed and can be applied to any fuzzy function with relative ease. The functions are as follows: *NNCGetFile*, *NNCNormalise*, *NNCInit*, *NNCInTDVec*, *NNCDataTransition*, *NNCFLS*, *NNCRead*, *NNCVerify* and *NNCDeNormalise*.

C.4.1 NNCGetFile

The *NNCGetFile* function is used to obtain a full file path by supplying the *Base Path* and the *index* of the file. The function returns a valid file path if the *index* is within range and an error if the *index* is out of range. The *index* should therefore never exceed the number of files within the base directory.

C.4.2 NNCNormalise

The *NNCNormalise* function normalises each input and output signal individually. This is done by subtracting the minimum value of the specified input or output vector to eliminate the signal offset. The input or output vector is then divided by the maximum value to scale (normalise) the signal between zero and one. This constitutes the final normalised vector which is used to train, validate and verify the FLS.

C.4.3 NNCInit

According to the NNC algorithm, if the first input-output pair is encountered, the algorithm parameters should first be initialised. This is done by initialising the first *cluster centre* to the first input-output pair and setting the *number of clusters* (*M*) equal to one. Thereafter set the *sum of the outputs* (*A*) equal to the first output data sample and initialise the *number of data samples* (*B*), within the first cluster, to one. Refer to section 2.4.2 for the NNC algorithm.

C.4.4 NNCInTDVec

The *NNCINTDVec* function generates a number of time delayed inputs, according to the specified time delay matrix. These time delayed versions of the input and/or output signals are added column wise to the original input signals.

A typical example can be seen in Figure C.4 where the time delay matrix in *(a)* is applied to the input signals. The result can be seen in Figure C.4 *(b)*. The light grey column indicates

an output channel and the grey block section indicate the preceding zeros. Notice that when a time delayed vector is added to the input matrix, the preceding values are always set to zero.

I/O	Channel	Factor
1	1	1
1	3	2
0	1	2

(a)

1	2	3	4	0	0	0
5	6	7	8	1	0	0
9	10	11	12	5	3	4
13	14	15	16	9	7	8
17	18	19	20	13	11	12

(b)

Figure C.4. *NNCTimeDelay Function Flow Diagram and Time Delay Example.*

C.4.5 NNCDataTransition

Due to the nature of the acquired data the first entry of the current file almost never accurately follows the last entry of the previous file. The data transition function allows for a relatively smooth transition from one file to the next by connecting two consecutive files with a straight line. A one-to-one connection method is used which implies that each input channel after being time delayed, is connected to the following file by means of a transition file.

Using the example in Figure C.4, the transition file is generated from user definable parameters which include the *Transition Time* in seconds (s) and the file *Sampling Period* in samples/second (S/s). If the transition time is e.g. 10 s and the sampling period is 0.04 S/s, 250 transition data samples by 7 columns wide are generated. This increases training time but prevents a sudden changeover between two consecutive files. A graphic representation of the one-to-one transition method can be seen in Figure C.5.

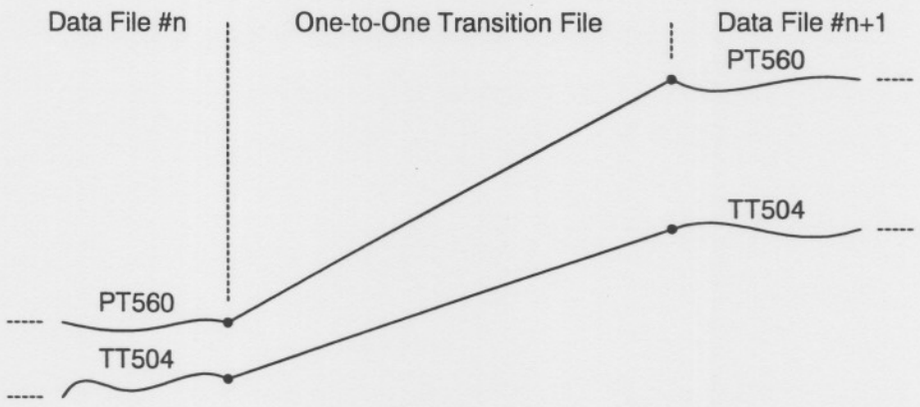


Figure C.5. *One-to-One Connection Method.*

C.4.6 NNCFLS

The nearest neighbourhood clustering fuzzy logic system (*NNCFLS*) function is the core of the FLS. This function implements the NNC algorithm where clusters are defined according to the one-dimensional parameter namely the cluster radius (r). More detail about this algorithm can be seen in section 2.4.2.

C.4.7 NNCRead

The *NNCRead* function reads saved data from the *.mat file. This file contains all the relevant information concerning the FLS. The *NNCRead* function loads the training settings, training results and verification result sections from the specified file. These sections are displayed in the Matlab command window and the FLS training result is shown on a graph.

C.4.8 NNCVerify

The *NNCVerify* function is used for both verification and validation procedures. This function uses the trained FLS to either verify previously trained files or to validate files which have not been encountered before.

By using the training values for both A and B in (2.9), a value for the fuzzy logic system, pertaining to the current input data sample, is predicted. Note however that if the input data sample is very far from all the cluster centres an infinite output is evident due to the *exp(.)* part in both numerator and denominator of (2.9) resulting in a division by zero. In this case the resultant output is set to zero and it is recommended to increase the system radius.

C.4.9 NNCDeNormalise

De-normalising the normalised data involves multiplying the normalised data with the maximum value. Thereafter the minimum value is added to readjust the original signal offset.

C.5 FUZZY LOGIC NNC GUI

The *FuzzyTrain*, *FuzzyRead* and *FuzzyValidate* applications have been incorporated into a GUI. This simplifies the training, verification, reading and validation processes. It also allows experiment settings to be saved and reloaded.

The GUI (*Fuzzy Nearest Neighbour GUI*) is divided into three modes, the first is the *Train FLS* and the second and third modes are *Read FLS* and *Validate FLS* respectively. Each of these modes (applications) has different capabilities, features and settings. The default GUI can be seen in Figure C.6 and is initialised when the *FuzzyGUI.m* file is executed.

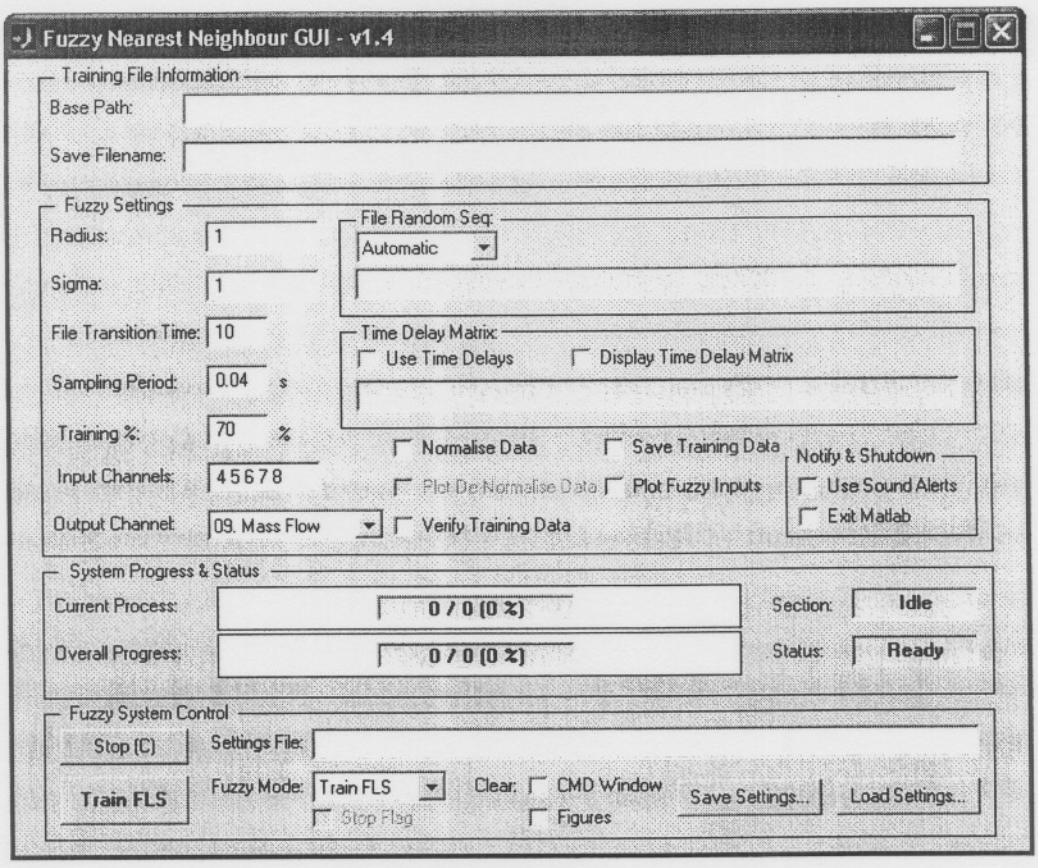


Figure C.6. *Fuzzy Nearest Neighbour GUI – Train FLS.*

The GUI is divided into four sections, *Training File Information*, *Fuzzy Settings*, *System Process and Status* and *Fuzzy System Control* each with its own tasks. The *Training File Information* section allows the set up of the data path and fuzzy results filename. The *Fuzzy Settings* section allows the manipulation of the fuzzy training, verification and validation settings. The *System Progress and Status* section provides feedback of the current executing process. This section allows the user to assess exactly what the status of the particular process is. The *Fuzzy System Control* section enables control of the system before, during and after training, verification and validation processes. It also allows the current settings to be saved to a settings file which can be re-loaded at a later stage.

C.5.1 Train FLS

The application in Figure C.6 is used to train the FLS with the provided settings. Each of the applicable sections, their features and capabilities are discussed.

C.5.1.1 Training File Information

Within this section the *Base Path* is supplied which points to a directory containing the data files used for training, verification and validation procedures. This path should not be supplied with a terminating backslash (\).

The *Save Filename* field is used to supply a *.mat file used to save the training and verification data once these procedures have been completed. Matlab returns an error if the supplied base path does not exist and should therefore be created beforehand.

C.5.1.2 Fuzzy Settings

This section is used to set up a particular FLS's settings. The system *Radius* and *Sigma* are set up in their respective fields and can be anything greater than zero. Both these system parameters are dimensionless.

The *File Transition Time* depends on the processed data *Sampling Period*. These parameters are used to determine the number of samples to generate comprising the transition data file. The acquired files' sampling period should be supplied in the *Sampling Period* field.

The percentage of random files used to train the FLS is supplied in the *Training* % field. This parameter is also used to determine the percentage of files which will be used to verify and validate the FLS. When verifying data files, the trained system uses *Training* % number of files. On the other hand when validating the FLS, against files which have not been encountered during the training procedure, the remaining (100 % - *Training* %) number of files are used for validation purposes.

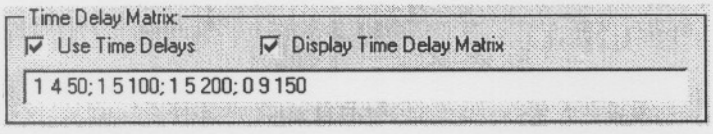
The system input channels are supplied in the *Input Channels* field. The input channels can either be delimited with a comma (,) or a space. The channel number entered in this field can be obtained from the input data file. In this case the channels (column) 4 to 8 are used. From the input files it can be seen that these channels represent the following acquired channels: *TT560A*, *PT505*, *PT506*, *VLVIN* and *VLVOUT*.

The FLS only has one output channel. A total of four output modes can be selected which includes the prediction of the system *Mass Flow*, *Reynolds Number*, *Expansibility factor* and *Valve Choke Status*. These four output channels also have an associated channel within the data file. The channel (column) number within the data file is supplied next to the *Output Channel* entry which in this case is nine.

Two methods exist for obtaining a random file sequence which can be selected within the *File Random Seq* section. This random file sequence is used in both training, validation and verification processes. If *Automatic* is selected from the drop-down box, a random file sequence is automatically generated. If *Manual* is selected from the drop-down box, the file sequence must be manually entered in the edit field that becomes available below the drop-down box. If a

manual file sequence is entered in the field, care should be taken not to repeat any file indices and the number of file indices should be the same as the number of files within the specified *Base Path*.

Within the *Time Delay Matrix* section, the possibility of including time delays within the system can be selected by checking the *Use Time Delays* checkbox. Once this checkbox has been checked, an edit field becomes available in which the time delays can be specified. Specifying the time delays involves three parameters. These include specifying if an input or an output channel is to be delayed. A one (1) in this first parameter signifies an input channel, whereas a zero (0) signifies an output channel. The second parameter specifies the channel number to be delayed which can be obtained from either *Input Channels* field or *Output Channel* drop down list, depending if it is an input channel or an output channel respectively. The third parameter is the delay factor which indicated the factor by which the channel must be delayed. All three parameters should be separated by either a comma or a space and delimited by a semi-colon. An example of the time delay field converted to the time delay matrix can be seen in Figure C.7. When the *Display Time Delay Matrix* checkbox has been checked, the specified time delay matrix is displayed in the command window.



(a)

I/O	Ch	Factor
1	4	50
1	5	100
1	5	200
0	9	150

(b)

Figure C.7. *Input and Output Time Delay Field and Matrix.*

The input data are normalised if the *Normalise Data* checkbox is checked. Once checked the *Plot De-Normalised Data* checkbox is available which allows the data to be de-normalised before it is graphed. In order to verify the trained data, the *Verify Training Data* checkbox need to be checked. The training data are saved to the specified file (*Save Filename*) if the *Save Training Data* checkbox is checked. To plot the input data on a graph, the *Plot Fuzzy Inputs* checkbox need to be checked.

The *Notify and Shutdown* subsection allows Matlab to used Sound Alerts, by checking the *Use Sound Alert* checkbox. This enables audible alerts if the fuzzy system has completed its training and verification. The option of exiting Matlab is enabled once the *Exit Matlab* checkbox

has been checked. Together with an external application (*ShutdownEx*) the computer can be shutdown completely.

C.5.1.3 System Progress and Status

The system progress and status section provide feedback during training, verification and validation procedures. Two progress bars represent the *Current Process* progress and the other represents the *Overall Progress*. The *Section* field supplies information regarding the current process being performed. These include *Training*, *Validation* and *Verification*. The *Status* field indicates if the Fuzzy Nearest Neighbour GUI is *Ready* or *Busy*. An example can be seen in Figure C.8.

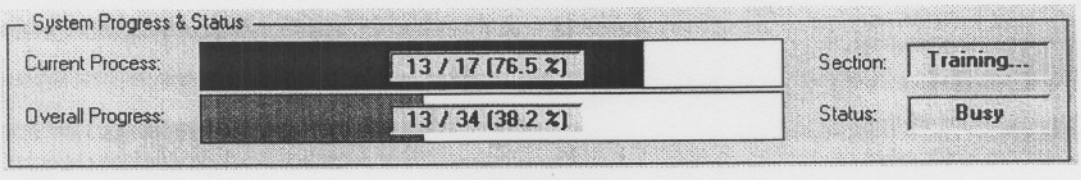


Figure C.8. *System Progress and Status*.

C.5.1.4 Fuzzy System Control

The fuzzy system control section is devoted to the control of the fuzzy logic processes. This section allows the commencing and stopping of a fuzzy process, the mode selection, the clearing of figures and the command window and the saving and loading of fuzzy setting files. Refer to the bottom section of Figure C.6.

The *Stop* button has multiple tasks. The first task allows the user to terminate a running training, verification or validation process. When a process has been terminated the *Stop Flag* check box is checked. The user has no control over this check box and is purely supplied as visual feedback. This check box is set and cleared in software.

The second task allows the user to clear the command window and/or created figures. In order to clear the command window and/or created figures their related check boxes (*CMD Window* and *Figures* respectively) should first be checked.

Training, Reading and Validation commences once the *Train FLS/Read FLS/Validate FLS* button has been clicked. Note that when in training mode, as determined by the *Fuzzy Mode* drop-down box, the caption of the lower left hand button changes to *Train FLS*. When in reading mode, the caption changes to *Read FLS* and in validation mode changes to *Validate FLS*.

Settings can either be saved or loaded, by clicking on the *Save Settings* and *Load Settings* buttons respectively, to the specified file within the *Settings File* field. This file must have an *.esd (Experiment Settings Data) extension.

C.5.2 Read FLS

This application is used to read system settings and data of a previously trained FLS. The aim of this application is to allow the user to monitor a FLS's training, verification and validation settings.

As apposed to the *Train FLS* application, not all fields are available during the read application. These unavailable fields have no direct bearing on the *Read FLS* application.

C.5.2.1 Training File Information

The filename to load, containing the FLS parameters and results, is supplied in the *Load Filename* field. The saved file contains the *Base Path* and therefore is not used in this application.

C.5.2.2 Fuzzy Settings

All fields are greyed out except for the *Display Time Delay Matrix* checkbox. This checkbox is used to select whether or not to display the time delay matrix in the Matlab command window.

C.5.2.3 System Progress and Status

Due to the nature of the *Read FLS* application, the progress bars are not used. The *Section* and *Status* fields however indicate *Reading* and *Busy* respectively if the application reads a FLS's parameters, settings and results.

C.5.2.4 Fuzzy System Control

This section's features and operation does not change drastically when selecting between the three different modes. The *Train FLS* button's caption changes to *Read FLS* and the *Save Settings* button is disabled in this mode.

C.5.3 Validate FLS

This application is used to validate files which have not been encountered by the trained FLS. Most of the fuzzy settings can not be changed because these are read from the saved filename.

C.5.3.1 Training File Information

Both fields are available where the *Base Path*, containing all the data files, are specified as well as the file path (*Save Filename*) containing the trained FLS parameters, settings and results.

C.5.3.2 Fuzzy Settings

The only settings available when validating a number of files are the *Validation File Seq* and the option of displaying the time delay matrix. The remaining settings do not have any bearing on this application because these settings are read from the save file in the *Save Filename* field.

C.5.3.3 System Progress and Status

The progress bars indicate the status of the validation process. The *Section* field indicates that a validation process is currently in progress and the *Status* field shows that the system is currently *Busy* with a process.

C.5.3.4 Fuzzy System Control

When validating a number of files, the *Train FLS* button changes to a *Validate FLS*. In this mode the *Save Settings* button is inactive implying that the selected system settings cannot be saved to the experiment settings file.

C.6 QUICK REFERENCE FUZZY MATLAB GUI

Without having to read through and understand the entire GUI in detail, this section is devoted to a quick reference manual of all three applications namely the *Train FLS*, *Read FLS* and *Validate FLS* applications. This section allows the user to start fuzzy logic training within a very short period of time.

C.6.1 Train FLS

Training the FLS involves various steps within the following sections:

Training File Information

- Supply a *Base Path* (without a terminating "\") which contains the training, verifying and validation files
- Supply a file name in the *Save Filename* field which is used to save the training and verifying data

Fuzzy Settings

- Supply the *Radius*, *Sigma*, *File Transition Time* and *Sampling Period* parameters

- Change the *Training* % parameter if need be. It is strongly advised that this parameter should remain set at 70 %
- Change the *Input Channels* if required. The input channel numbers are in direct relation to the column numbers within the processed file
- Select the *Output Channel* from the dropdown box
- Select if the random file sequence should be generated automatically or manually from the *File Random Seq*. If the random file order should be generated manually, add the file sequence in the available field
- Select if a time delay matrix should be used. If so supply the time delay matrix in the available field. Select if the time delay matrix should be displayed in the Matlab command window by checking the *Display Time Delay Matrix* checkbox
- Select if the input data should be normalised and de-normalised, when graphed, in the *Normalise Data* and *Plot De-Normalised Data* fields respectively
- Select if the training data should be saved (*Save Training Data*) and if the process should verify the trained data (*Verify with Training Data*)
- Select if the fuzzy logic input data should be graphed (*Plot Fuzzy Inputs*)

Fuzzy System Control

- Supply a settings file within the *Settings File* field. This file should have a *.esd extension
  - If all settings have been set up click the *Save Settings* button
- Make sure that the *Fuzzy Mode* is set to *Train FLS*
- Check the appropriate *Clear* check boxes
- Click the *Stop (C)* button to clear all figures and the command window
- Click the *Train FLS*

C.6.2 Read FLS

Reading a previously trained FLS involves the following steps:

Fuzzy System Control

- Change the *Fuzzy Mode* to *Read FLS*
- Supply an *experiment settings data* (*.esd) file in the *Settings File* field and click the *Load Settings* button. These files are only available if the settings were saved during the training process. Make sure that all the settings are correct and if necessary check the *Display Time Delay Matrix* checkbox and then proceed the reading process by clicking the *Read FLS* button.

C.6.3 Validate FLS

The validation process involves the following steps:

Fuzzy System Control

- Supply a settings file within the *Settings File* field
- Click the *Load Settings* button
- Change the *Fuzzy Mode* to *Validate FLS*
- Select the appropriate *Clear* check boxes
- Click the *Stop(C)* button

Training File Information

- Supply or modify the *Base Path* which contains the processed files
- Supply or modify a *Load Filename* containing the previously trained FLS parameters, settings and results

Fuzzy Settings

- Select if the validation process should automatically or manually generate the file sequence. When selecting *Automatic* from the drop down box, the validation process uses the remaining percentage files to validate the trained FLS. On the other hand, when selecting *Manual* from the drop down box, any number of files can be validated, even previously trained files
- Select if the time delay matrix should be displayed in the command window by checking the *Display Time Delay Matrix* check box

APPENDIX D

Data CD Content

This appendix is dedicated to the layout of the data CD. The data CD includes items such as the raw physical data acquired from the control valve, the data management software, Matlab specific source files and the final control valve models. The data CD layout can be seen in Figure D.1.

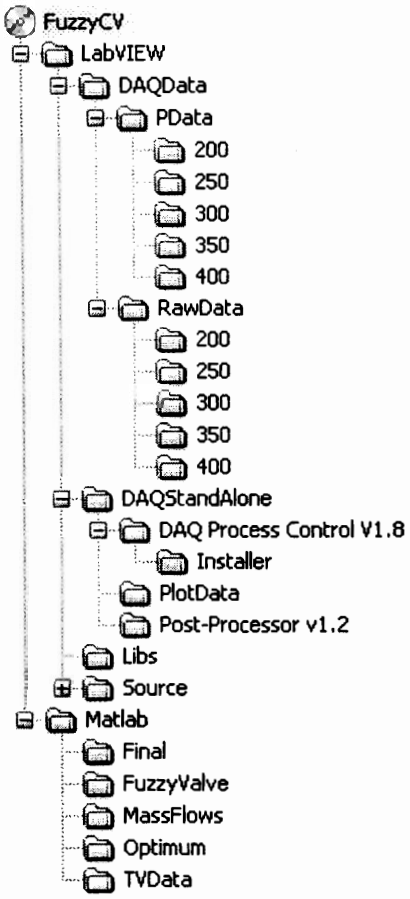


Figure D.1. *Data CD Layout*.

From this figure it is evident that the fuzzy control valve (*FuzzyCV*) data CD is divided into two main sections namely *LabVIEW* and *Matlab*. *LabVIEW* is the main directory for the data management software and the raw acquired data. *Matlab* is the main directory for the fuzzy logic system source files, the training and validation data and the final control valve models to name but a few.

D.1 LABVIEW

This is the main directory for the *LabVIEW* source code and acquired data. The *LabVIEW* directory consists of four subdirectories namely, *DAQData*, *DAQStandAlone*, *Libs* and *Source*.

D.1.1 DAQData

This subdirectory consists of two directories; *PData* and *RawData*. Both directories consist of five subdirectories each representing an initial pressure data set where *PData* contains the processed data and *RawData* the raw physical acquired data.

The processed data directory (*PData*), data files are used for training, verifying and validation. The filename structure for the *DAQData* can be seen in Figure D.2.

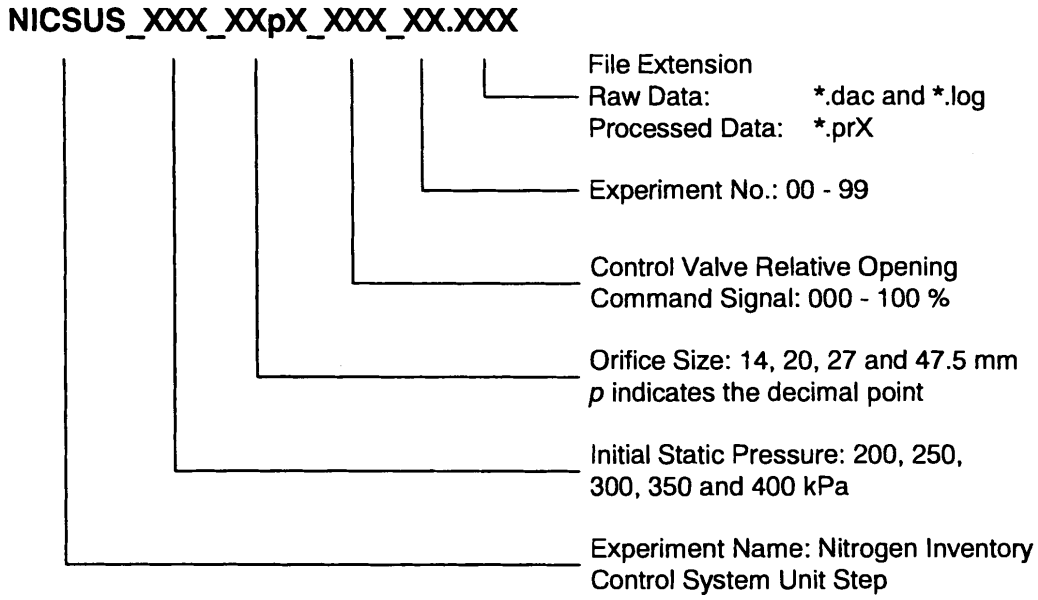


Figure D.2. *DAQData Filename Strucure.*

D.1.2 DAQStandAlone

The *DAQStandAlone* directory includes all the standalone applications created from the source code VIs. This includes the data acquisition process control application (*DAQ Process Control v1.8*), the data plot application (*PlotData*) and the post processing application (*Post-Processor v1.2.1*).

The data acquisition process control application has an installer which contains the LabVIEW Run-time engine; a prerequisite to execute any of the applications.

D.1.3 Libs

The *Libs* (library) directory contains subVIs used in the applications. This includes subVIs such as *AddItem*, *Chokeflow* and *ResampledMean*.

D.1.4 Source

This directory contains all the relevant source code files used to acquire, manage and process the raw physical data.

D.2 MATLAB

The subdirectories within the Matlab directory include: *Final*, *FuzzyValve*, *MassFlows*, *Optimum* and *TVData*.

D.2.1 Final

This directory contains the final fuzzy control valve models derived from all five data sets based on the optimal fuzzy logic parameters.

The filename structure used in this directory can be seen in Figure D.3.

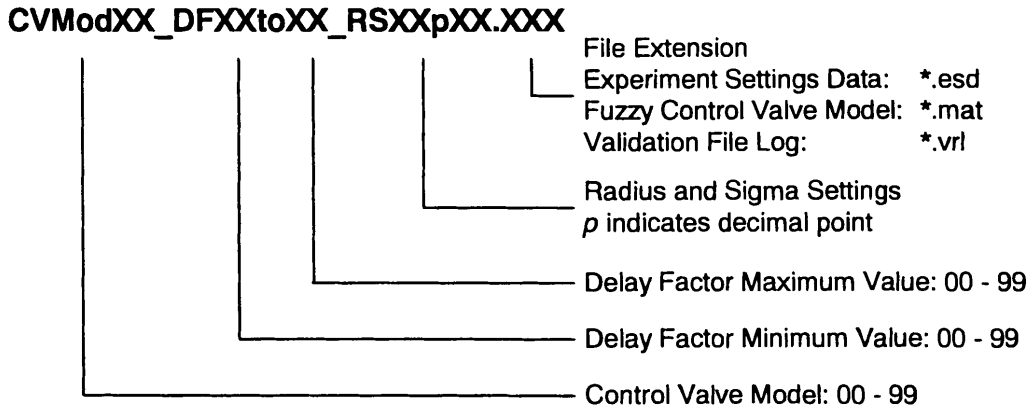


Figure D.3. *Final Control Valve Model Filename Structure.*

D.2.2 FuzzyValve

This directory contains the Matlab source files used to create the fuzzy logic system. Both graphical user interface and function source files are included. The *ShutdownEx* application is used to shutdown the computer once training and verification has stopped.

D.2.3 MassFlows

This directory contains some of the processed data channels in Matlab workspace file format (*.mat). The naming convention used for these files are the same as used in LabVIEW.

D.2.4 Optimum

The *Optimum* directory contains the partial models derived from a single data set which was used to search for the optimum fuzzy logic parameters.

The naming convention used for these files can be seen in Figure D.4.

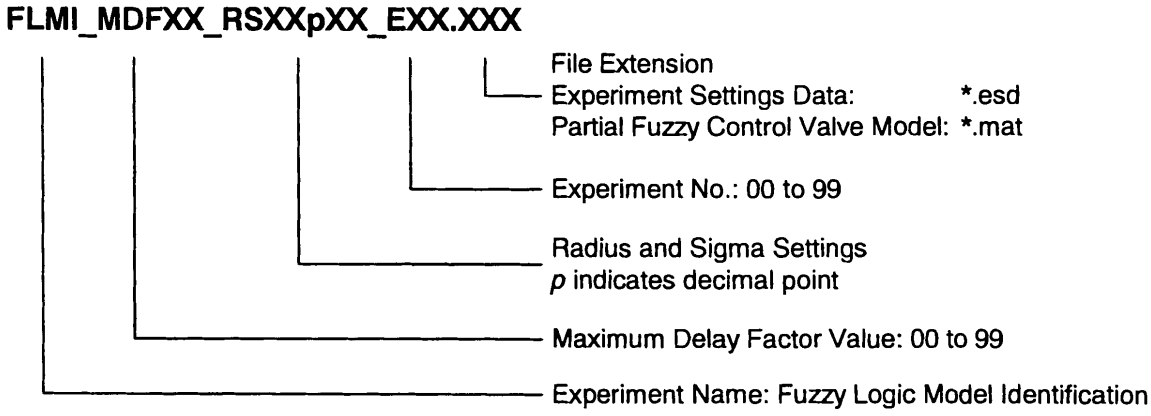


Figure D.4. *Fuzzy Logic Model Identification Filename Structure.*

D.2.5 TVData

This directory contains the training, verification and validation data sets. The data files in the *PData* directory are copied to this single directory.

References

[1] F.R. Giordano, M. D. Weir and W. P. Fox, "A first course in mathematical modelling", 2nd ed., Brook-Cole, California, 1997.

[2] C. P. Bodenstein, "ERIE876: Process modelling and identification", ver 1.081, 2003.

[3] V. Venkatasubramanian, R. Rengaswamy, K. Yin and S. N. Kavuri, "A review of process fault detection and diagnosis – Part 1: Quantitative model-based methods", Computers and Chemical Engineering, vol. 27, pp 293 – 311, April 2002.

[4] R.C. Dorf and R. H. Bishop, "Modern Control Systems", 8th ed., Addison-Wesley, Menlo Park, California, 1998.

[5] L.X. Wang, "Adaptive fuzzy systems and control: Design and stability analysis", Prentice-Hall, Upper Saddle River, Englewood Cliffs, NJ, 1994.

[6] ISO 5167-1:1991(E), "Measurement of fluid flow by means of pressure differential devices – Part 1", 1st ed., 1991.

[7] I. H. Shames, "Mechanics of fluids", 3rd ed., Mc-Craw Hill, New York, 1992.

[8] M. Husu, I. Niemelä, J. Pyötsiä, M. Simula, M. Hauhia, J. Riihilahti, "Flow Control Manual", 3rd ed., Neles Automation, Helsinki Finland, 1997.

[9] I. Niemelä, E. Lumme, P. Kanerva, "Control Valve Sizing Coefficient", 4th ed., Neles Automation, Helsinki Finland, 1999.

[10] Masoneilan, "Masoneilan Handbook for Control Valve Sizing", 6th ed., Masoneilan International Inc., 1977.

[11] R. Babuška and H. B. Verbruggen, "An overview of fuzzy modelling for control", Control Engineering Practice, vol. 4, No 11, pp 1593 – 1606, May 1996.

[12] The MathWorks, Inc., "Matlab 6 R13: Fuzzy Logic Toolbox", ver. 6 R13, June 2002. [Online]. Available: http://www.mathworks.com.

[13] B. Arnett (2004, April 14). Sedna. [Online]. Available: http://www.nineplanets.org. [Date of access: July 2004].

[14] M. Brown (2004, August 10). Sedna (2003 VB12). [Online]. Available: http://www.gps.caltech.edu/~mbrown/sedna. [Date of access: July 2004].

[15] IAU Secretariat (2004, March 30). Naming of planets and Sedna. [Online]. Avaiable: http://www.iau.org. [Date July 2004].

[16] R. M. Stitt and D. Kunst, "Input overload protection for the RCV420 4-20 mA current-loop receiver", Burr-Brown Corporation, Tucson AZ 85734, Application Bulletin AB-014, July 1990.

[17] National Instruments, "Measure and Automation Explorer 3.0: Help system", ver. 3.0, 2003, [Online]. Available: http://www.ni.com.

[18] R. M. Stitt and D. Kunst, "IC Building block form complete isolated 4 – 20 mA current-loop systems", Burr-Brown Corporation, Tucson AZ 85734, Application Bulletin AB-032A, June 1992.

[19] R. H. Bishop, "Student Edition LabVIEW 6i", 2nd ed., Prentice Hall, 2001.

[20] D.H. Friedal jr A. Potts, "Java programming language handbook: Internet programming series", Coriolis group books, 1996.

[21] Valtek, "Sizing and Selction: Control Valve Sizing", vol. 3, Rev. 6/94.

[22] A. Luchetta and S. Manetti, "A real time hydrological forecasting system using a fuzzy clustering approach", Computers and Geosciences, vol. 29, pp 1111 – 1117, May 2003.

[23] Burr-Brown Corporation, "Precision 4 mA to 20 mA current loop receiver", Burr-Brown Corporation, Tucson AZ 85734, PDS-837E, October 1997.

[24] National Instruments, "Measure and Automation: Catalog 2003", [Online]. Available: http://www.ni.com, e-mail: info@ni.com.

[25] National Nuclear Regulator, "Requirements for licensing submissions involving computer codes and evaluation models for safety calculations". Licensing Guide: LG-1038, rev 0.

Additional References – Not directly referenced in text

[26] Reader's Digest Association, Inc, "Reader's Digest, Atlas of the world", Toucan Books Limited, London, 1989.

[27] J. C. Kotz and P. Treichel, Jr., "Chemistry and chemical reactivity", 3rd ed., Saunders College Publishing, 1996.

[28] S. Haykin, "Neural Networks, A comprehensive foundation", 2nd ed., Prentice Hall, 2001.