Dynamic control of a hybrid control valve

A dissertation presented to

The School of Electrical, Electronic and Computer Engineering

North-West University

In partial fulfilment of the requirements for the degree

Magister Ingeneriae

in Computer and Electronic Engineering

by

Douw Gerbrand Breed

Supervisor: Prof. G. van Schoor
Assistant supervisor: Mr. C. van Niekerk

March 2006

Potchefstroom Campus

SUMMARY

In industries around the world control valves are commonly used to control integrated systems. The operational range of a control valve may therefore extend over large excursions of pressure and fluid or gas density. These valves should be able to exert strict control over pressurised gasses or heated fluids in order to keep systems safe and manageable. However, the desired operational range and accuracy guaranteeing peak performance is often not available in a single valve, and therefore a combination of valves may be considered to achieve this.

The purpose of this project is to develop an optimised algorithm that will control such a hybrid control valve system. The algorithm should optimise the coordination of two separate valves, of which the maximum flow coefficients differ by a large degree. This should be done in such a way that the two valves essentially function as a single valve with new characteristics. The new valve should subsequently be able to accurately function over both small and large ranges of mass flow.

Two different hybrid valve controllers are discussed. The first is a linear, PID based controller that is designed for simple inputs. The controller's main task is to assist in developing a better comprehension of the main challenges that will be faced in coordinating the operation of two separate valves. Although the controller provides stable control for step inputs with a low frequency of change, it is seen that it fails to deliver satisfactory results when faced with complex request signals. It is consequently concluded that more complex control will be required.

The second hybrid valve controller is therefore designed with the purpose of controlling more sophisticated input request signals. The controller's design is based on Fuzzy Logic which provides an effective platform for complex control. The complete control system consists of four main elements: The low pass filter, which filters out unachievable high frequencies from the input request, the Neural Network based signal predictors, which increases the efficiency of the controller, the Fuzzy Inference System, which is responsible for all the control decisions, and the crisp controller, which aids the Fuzzy Inference System with control executions that cannot be fuzzified.

The final optimisation to the Fuzzy Logic based hybrid valve controller is done by Genetic Algorithms. The membership functions that lend itself to optimisation are identified, and their parameters are optimised in order to further minimise the controller's mean control error.

OPSOMMING

In industrieë regoor die wêreld word beheerkleppe gebruik in die beheer van geïntegreerde stelsels. Die operasionele werksgebied van 'n beheerklep kan daarom oor enorme ekskursies van druk en vloeistof- of gasdigtheid strek. Hierdie kleppe moet daartoe instaat wees om streng beheer uit te oefen oor gasse onder druk, of verhitte vloeistof sodat stelsels veilig en beheerbaar gehou kan word. Die verlangde operasionele werksgebied en akkuraatheid wat optimale werkverrigting sal verseker is egter nie altyd beskikbaar in 'n enkele beheerklep nie, en daarom kan a kombinasie van kleppe oorweeg word om dit te bereik.

Die doel van hierdie projek is om 'n ge-optimeerde algoritme te ontwikkel wat sò 'n hibriede beheerklepstelsel sal beheer. Die algoritme moet die koördinering optimeer van twee afsonderlike kleppe waarvan die vloeikoeffisiënte grootliks verskil. Dit moet op so 'n manier gedoen word dat die twee kleppe uiteindelik as 'n enkele klep met nuwe karakteristieke funksioneer. Hierdie klep moet gevolglik daartoe instaat wees om met hoë akkuraatheid oor beide groot en klein massavloeiwerksgebiede te kan funksioneer.

Twee afsonderlike hibriede klepbeheerders word bespreek. Die eerste is 'n lineêre, PID gebasseerde beheerder wat ontwerp is vir eenvoudige insette. Die beheerder se hoofdoel is om 'n beter begrip te ontwikkel van die belangrikste uitdagings wat die koördinasie van twee kleppe inhou. Alhoewel die beheerder stabiele beheer bied vir trapinsette wat teen 'n lae frekwensie verander, word daar gesien dat dit nie daarin slaag om bevredigende resultate te lewer wanneer dit met meer komplekse insetseine gekonfronteer word nie. Om hierdie rede word die afleiding gemaak dat meer komplekse beheer benodig sal word.

Die tweede hibriede klepbeheerder is daarom ontwerp met die spesifieke doel om meer gesofistikeerde insetseine te kan beheer. Die beheerder se ontwerp is gebasseer op Wasige Logika, wat 'n baie effektiewe platform vir komplekse beheer bied. Die volledige beheersisteem bestaan uit vier hoofelemente: Die laaglaatfilter, wat onbereikbaar hoë frekwensies uit die insetsein filter, die Neurale Netwerk gebasseerde seinvoorspellers, wat die effektiewiteit van die beheerder verbeter, Die

Wasige Inferensie Sisteem, wat verantwoordelik is vir al die beheerbesluite, en die nie-wasige beheerder wat die Wasige Inferensie Sisteem bystaan met beheeruitvoerings wat nie verwasig kan word nie.

Die finale optimering van die Wasige Logika gebasseerde hibriede klepbeheerder word gedoen met behulp van Genetiese Algoritmes. Die lidmaatskapfunksies wat hulself leen tot optimering word geïdentifiseer en hulle parameters word ge-optimeer om uiteindelik die klepbeheerder se gemiddelde fout te minimeer.

ACKNOWLEDGEMENTS

Firstly, I would like to thank M-Tech Industrial and THRIP for funding this research and granting me the opportunity to further my studies.

I would also like to acknowledge the following people, in no particular order, for their contributions during the course of this project.

- Professor George van Schoor, my supervisor, for his guidance, advice and support without which this project would not have been a success.
- My assistant supervisor, Carl van Niekerk, for his valuable inputs and support during this project
- Chris Niewoudt and Louwrence Erasmus of PBMR. Without their inputs it would not have been possible to grasp the numerous different aspects of this project.
- My family and loved ones for their support

TABLE OF CONTENTS

NOMENCLATURE

LIST OF FIGURES

LIST OF ABBREVIATIONS

AFS	Adaptive Fuzzy System
FL	Fuzzy Logic
FLC	Fuzzy Logic Controller
FRBS	Fuzzy Rule Base System
FIS	Fuzzy Inference System
FS	Fuzzy System
GA	Genetic Algorithm
GUI	Graphical User Interface
KB	Knowledge Base (Mamdani FRBS)
MF	Membership function of Fuzzy Logic system
NN	Neural Network
PBMR	Pebble Bed Modular Reactor
PID	Proportional-Integral-Differential control
RCGA	Real Coded Genetic Algorithms
RBF	Radial Basis Function
SISO	Single-input-single-output

LIST OF SYMBOLS

C_v	Valve inherent flow coefficient
Δp	Pressure differential
$E_{l\,\mathrm{arg}\,e}(s)$	Mass flow error adjusted for large valve
$E_{small}(s)$	Mass flow error adjusted for small valve
K_l	Large valve incremental command gain
K_s	Small valve incremental command gain
$\dot{m}$	Normalised mass flow rate
$\dot{m}(\mathrm{large})_{\min}$	Minimum possible mass flow rate of larger valve

R_p	Low pass filter maximum pass band attenuation
R_s	Low pass filter minimum stop band attenuation
s	Complex frequency
t	Time / simulation time
U	Input request signal to hybrid valve system
ω_p	Low pass filter pass band frequency
ω_s	Low pass filter stop band frequency

1
Introduction

1.1. Problem overview

1.1.1. Control valves

In Hydraulic systems, it is often necessary to control the speed and amount of gas or liquid that flows from one phase in the system to the next [1]. This can be done by using a control valve.

A control valve is a device capable of modulating flow at varying degrees between minimal flow and full capacity in response to a signal from an external control device. The control valve - often referred to as "the final control element" - is a critical part of any control loop, as it performs the physical work and is the element that directly affects the process [2].

Control valves should therefore be able to exert strict control over pressurized gasses or heated fluids in order to keep systems safe and manageable [1]. However, the desired operational range and accuracy guaranteeing peak performance is often not available in a single valve. This is caused by non-linearities in the flow characteristics of some valves.

1.1.2. Control valve non-linearities

Essentially, as mentioned, the aim of a control valve is to modulate the flow of gas or liquid through a system. This is done by opening or closing the valve according to the required effect. The flow rate through a control valve, at constant pressure, can be modelled as *directly proportional* to the valve's flow coefficient (C_v) which is, in essence, an indication of the effective flow crossection of the valve [2]. Perceptibly, the C_v value increases as the valve is being opened – therefore increasing the flow rate through the valve and vice versa. This will be discussed in more detail in Chapter 2

The distance or amount which a control valve is opened is known as the control valve's *travel*. A graph that shows how the flow coefficient of a valve changes as a function of travel gives an idea of

the valve's *inherent flow characteristics* [2]. Figure 2 shows an idealised graph of the inherent flow characteristics of the control valves that will be considered in this dissertation. For the sake of simplicity, the flow coefficients and travel distances were normalised in the graph – and are therefore shown relative to their maximum values.

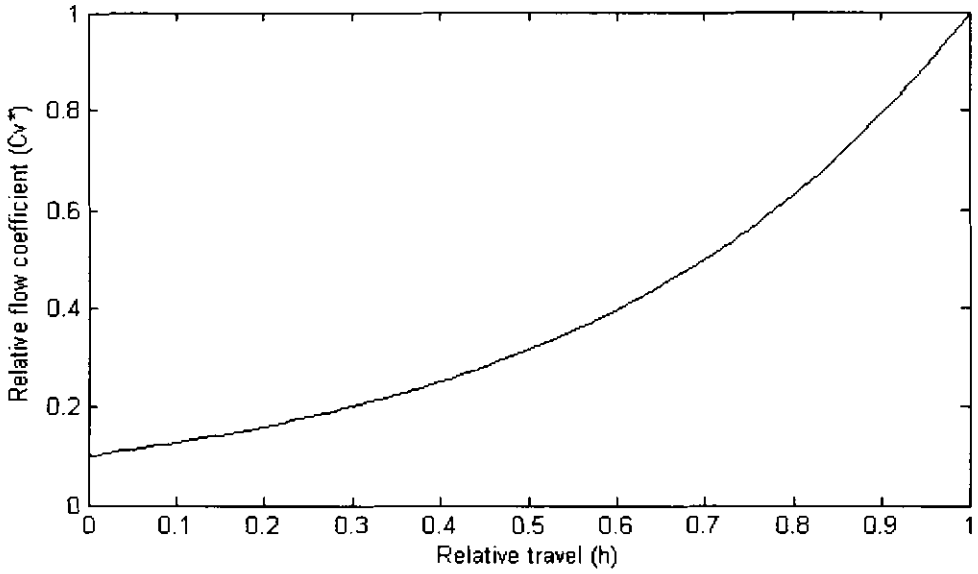

Figure 1.1 Idealised graph of inherent flow characteristics

From the graph in Figure 1.1, it is possible to identify the mentioned non-linearities that exist in the flow characteristics of the control valves under investigation. As can be seen, the graph is essentially continuous, except at very small valve travel, where an abrupt transition takes place. This phenomenon causes valves with large maximum flow coefficients to immediately allow high mass flow upon opening. This means that low mass flow rates through the valve cannot be controlled – thus limiting the controllability of the system as a whole.

1.1.3. Hybrid control valve system

To solve the problems caused by the non-linearities discussed, a hybrid control valve system, consisting of two or more valves, each with differing flow characteristics may be considered. An example of such a system is shown in Figure 1.2.

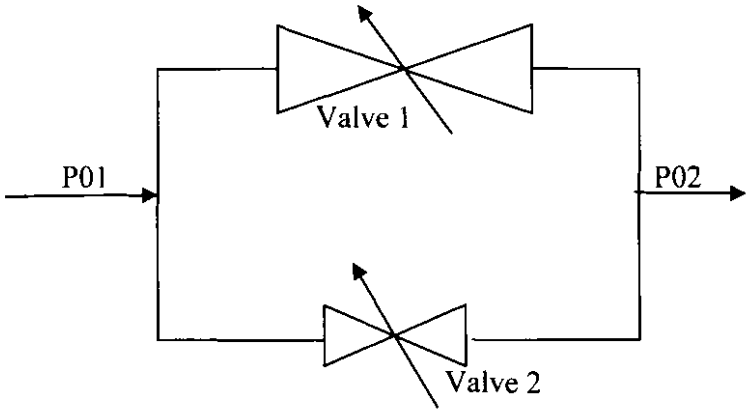

Figure 1.2 Hybrid control valve system consisting of two valves

In Figure 1.2 it can be seen that the hybrid valve system consists of a larger and a smaller valve. If the flow characteristics of these valves are arranged in such a way that the smaller valve's *maximum* mass flow is close to the larger valve's *minimum* mass flow, it will be possible to reach mass flows throughout the whole required range. Figure 1.3 illustrates the process in mind.

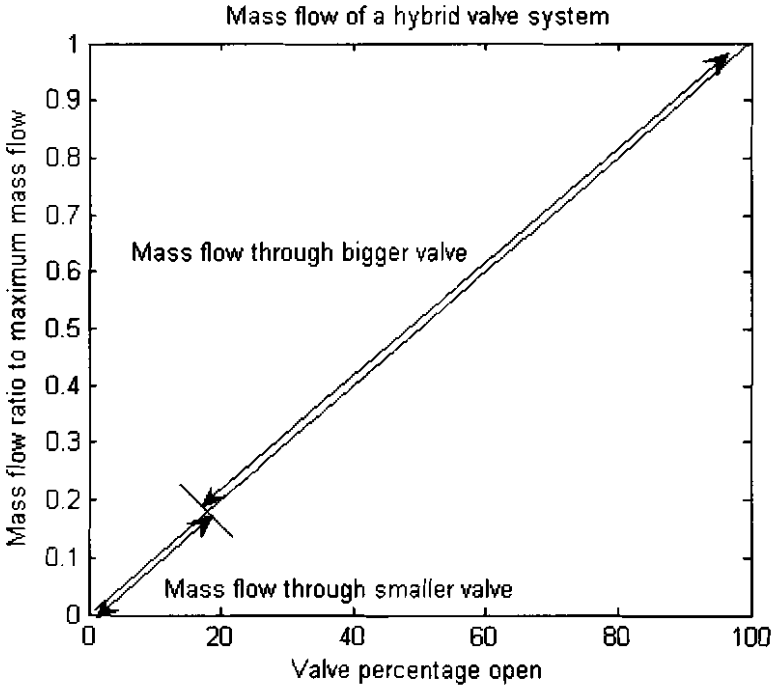

Figure 1.3 Hybrid valve cooperation

From the figure it is apparent that required mass flows below the minimum mass flow of the larger valve is obtained by opening the smaller valve, while mass flows above the maximum mass flow of the smaller valve can still be obtained by opening the larger valve.

1.1.4. Implication of non-linearity on valve cooperation

As was explained in section 1.1.2, the valves that will be considered in this dissertation causes an immediate jump in mass flow the moment it is opened. With that in mind, consider the step request to the hybrid valve system as illustrated in Figure 1.4.

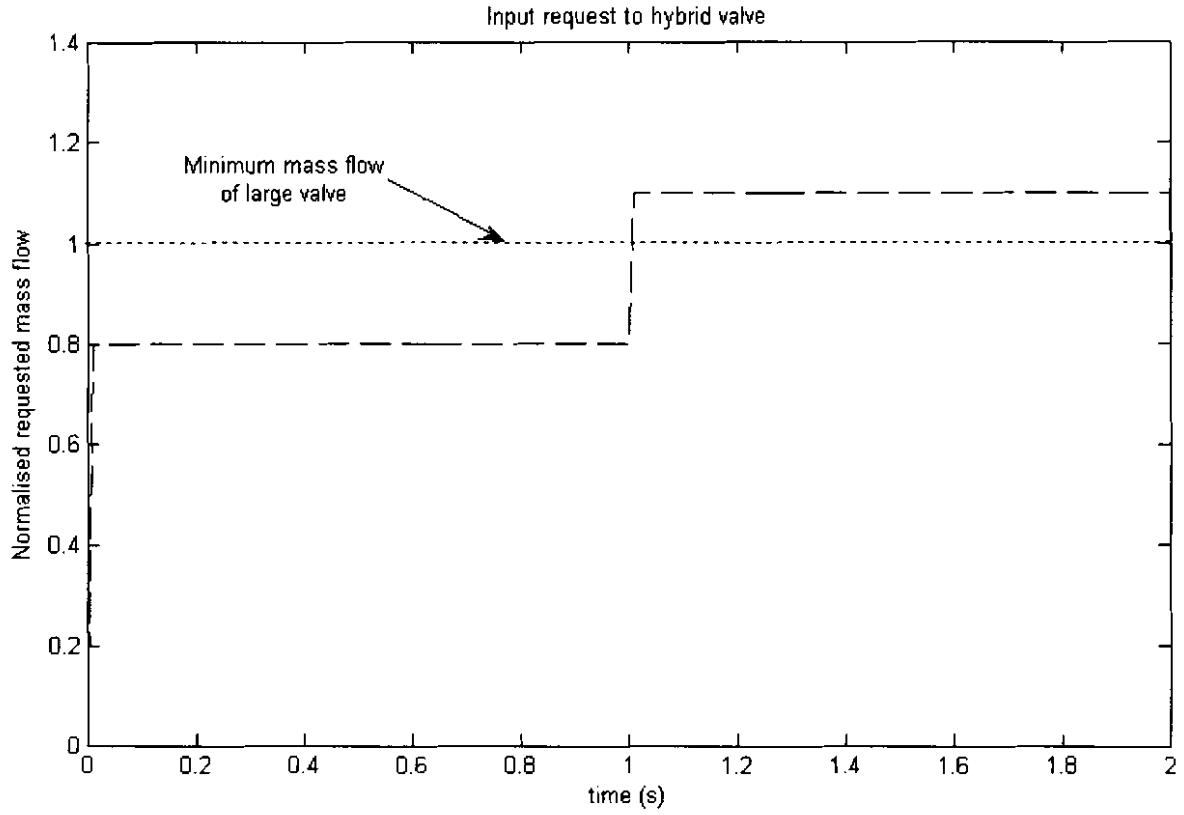

Figure 1.4 Example of step request to the hybrid valve system

In order to avoid irregularity, the mass flows in this dissertation will be normalised to the minimum mass flow of the larger valve. As can be seen, the request signal in Figure 1.4 steps from a value below the minimum mass flow of the large valve, to a value just within its range. Let us assume that the smaller valve is designed in such a way that its maximum mass flow is approximately equal to the minimum mass flow of the larger valve. This means that, in order to reach the requested mass flow after the step has occurred, the large valve *has* to be opened. Figure 1.5 shows the practical implication of this.

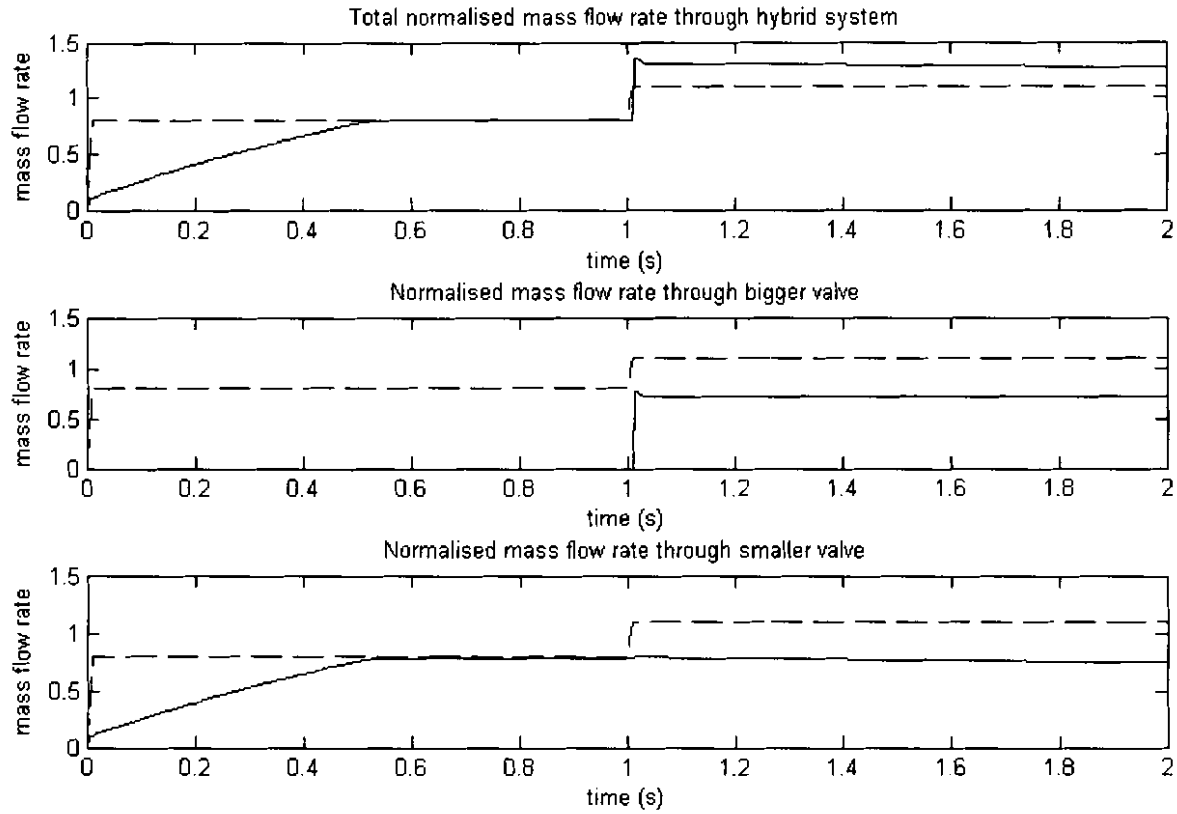

Figure 1.5 Result of opening the larger valve

The dashed line in Figure 1.5 represents the desired mass flow signal, while the solid line represents the actual mass flow rate through the valves. Three graphs are used to illustrate the process. The top graph shows the total mass flow rate through the whole hybrid valve system, the second graph the bigger valve's mass flow rate and the third graph shows the mass flow rate through the smaller valve. At t = 0 s, both valves are completely closed. Graphs such as these will be used throughout this thesis to illustrate the hybrid valve's response.

As can be observed from Figure 1.5, the jump in mass flow resulting from opening the bigger valve causes the mass flow through the hybrid valve to be far more than requested. Because the larger valve is at its minimum mass flow, the smaller valve would consequently have to be closed in order to reach the mass flow request.

This example illustrates that, in order for the hybrid valve system to have practical significance, the two valves' operation will have to be stringently coordinated. The goal of this study, therefore, is to create a complex controller that will coordinate the operation of the two valves.

1.1.5. Possible application of hybrid control valve in PBMR

The Pebble Bed Modular Reactor (PBMR) is a South-African initiated project with international partners involving a closed cycle (Brayton-cycle) based nuclear power generation plant. The inherent safety and modularity of the design renders it an ideal alternative to meet the future energy needs of not only South-Africa, but the world in general.

The system will make use of helium in the closed loop gas cycle to transfer the heat from the nuclear fusion elements to the power turbine. Since helium is both chemically and radiologically inert, nuclear contamination to the plant and environment is prevented [3].

Because of the nature of the PBMR's operation, valves are extremely important. Valves regulate the flow of helium from one phase to the next, and are therefore largely responsible for the amount of power generated. The plant controller may for instance open the bypass valves in order to reduce the production of power, or open the injection valves to increase it. Therefore, if the performance and range of the control valves can be improved, it will have a positive effect on the controllability of the plant as a whole.

1.2. Purpose of research

In this section, a quick overview will be given on what this project aims to accomplish, and some general details will be discussed to explain the problem at hand.

1.2.1. Problem statement

The formal problem statement for this project is to develop an optimised algorithm that will control a hybrid control valve system. The algorithm should optimise the coordination of two separate valves, of which the maximum flow coefficients differ by a large degree. This should be done in such a way that the two valves essentially function as a single valve with new characteristics. The new valve should subsequently be able to function over both small and large ranges of mass flow.

As will be seen in Chapter 5, some additional requirements are added to the control algorithm's operation to ensure that it is of practical significance. This will, however be explained in Chapter 5.

1.2.2. Controller configuration

Figure 1.6 shows the main elements that are expected to be present for the general controller setup. As will be seen in the next sections, the valve controller may need some additional inputs depending on the complexity of the control required, but Figure 1.6 gives a good illustration of the lay-out of the system in general.

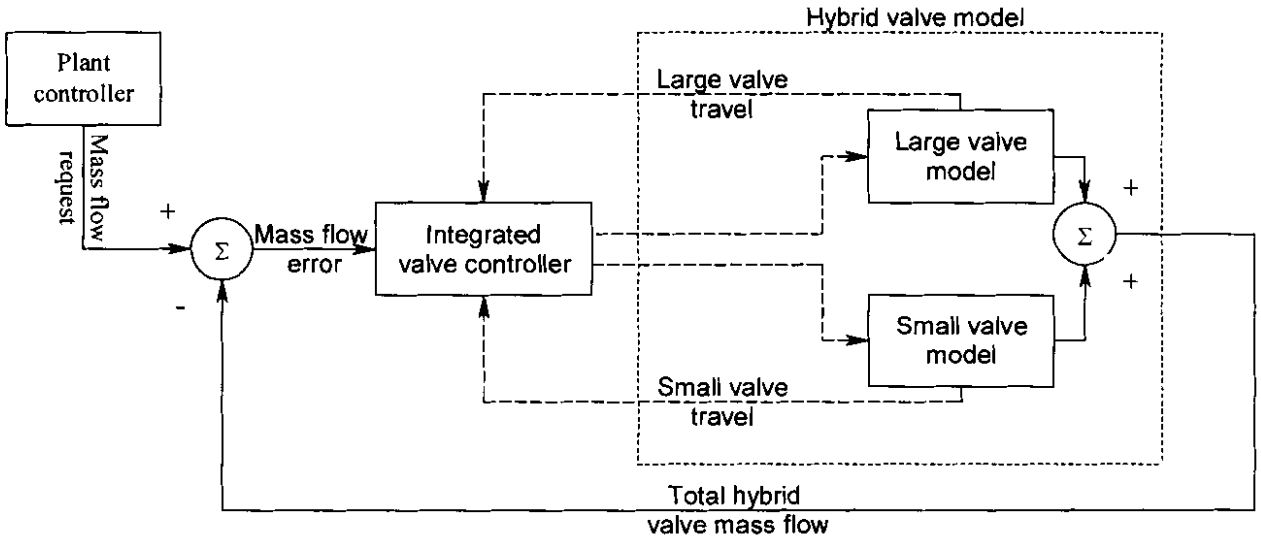

Figure 1.6 Main elements of hybrid valve system

The valve controller in Figure 1.6 is seen to provide each of the two valves with an individual command according to the information available to it. From section 1.1.4 it is apparent that this is necessary because of the non-linearity of the valves. The two outputs to each of the valves will have to be completely independent of each other, and, as was seen from Figure 1.5, may have to perform completely opposite roles at times.

Feedback is an important characteristic of almost any control system and essentially forms the foundation for control system analysis and design [4]. As can be seen from Figure 1.6, several feedback loops were added to the general valve controller.

Subtracting the total mass-flow output from the mass flow request produces the current mass flow error. The error is an important parameter in this control system, and the controller may manipulate it through differentiation, integration, amplification or other logic, according to the type of controller used [4].

Each of the valves also provides inputs to the controller in the form of their current valve travel. These inputs enable the controller to perform much more direct control and provide the controller with knowledge as to how much each valve can still influence the total mass flow. In the chapters describing the controllers that were designed, the various additional inputs, as well as their significance to the valve controller will be further discussed.

1.3. Issues to be addressed and research methodology

1.3.1. Overview

The study can be divided into four main issues that have to be dealt with. These are:

- Creation of an accurate model for the hybrid valve system
- Choosing the method of control and creating a practical control algorithm
- Optimising the control algorithm
- Testing and evaluating the entire system

This section will give a quick overview of each of the issues, and will discuss the work strategy that was followed to reach the outcomes of each issue.

1.3.2. Modelling hybrid system

Before any controller can be created, there has to, at first, exist something that can *be* controlled. This is necessary to verify the correctness and effectiveness of the controller that was created.

In order to obtain such a "controllable entity", there are usually two options available:

- Accurate modelling of the plant or process to be controlled.
- Direct control of an actual plant or process

The disadvantage of creating a model of the plant or process is that modelling usually requires some form of abstraction to certain properties. This means that not all aspects of the plant or process will be taken into account in the creation of the controller, which may lead to poor actual results. Direct

control, however, is an extremely costly process, and is usually only used for final preparation of the controller before implementation.

As this project is the first examination into possible further research of implementing a hybrid control valve, it was decided to follow the first option mentioned, namely the accurate modelling of the hybrid valve system.

The options available for modelling the valve will be discussed in Chapter 2, and the choice of model, as well as the model that was created will be discussed in Chapter 3.

1.3.3. Choosing the method of control and creating the valve controller

As will be discussed in Chapter 2, many different methods exist to create a controller. For each implementation, a different method of control may be optimal.

In the later chapters of this dissertation it will be seen that techniques of both linear and non-linear control were considered as possible control methods for the hybrid control valve. The different advantages and disadvantages of each controller were considered, and eventually an informed choice was made as to the best controller.

Because creating the controller is the main focus of this study, it was important to invest a great deal of time and effort into finding the best suited controller. For this reason, more than one controller was created, and its performance measured, in order to gain enough knowledge and eventually come to a decision.

1.3.4. Optimising valve controller

In control theory, optimisation is the process of improving a system in certain ways to increase the effective execution speed and/or reduce the mean execution error [5]. For this project, optimisation of the control algorithm comprises minimization of the time taken to reach the target mass-flow through the valve system and minimise the error of that process.

In the choice of control technique, the possibility of optimisation of the final controller had to play a large part, since part of the project's objectives is to create an optimised valve controller.

As will be seen, it was eventually decided to use genetic algorithms to optimise the control algorithms because of the many advantages (discussed in Chapter 2) it offers.

1.3.5. Testing and evaluating controller

Without thorough testing, there will be no way of ensuring that the controller that was created is optimal, or even at all stable. Therefore the controller will have to be subjected to a wide range of tests that evaluate its performance and identifies possible weak points.

Because it is impossible to test absolutely every response of the controller to absolutely every input signal possible, the tests had to focus on the areas that are most likely to pose a problem.

1.3.6. Software implementation

This project makes use of the software program MATLAB to develop the needed algorithms. Given the versatility of MATLAB's high-level language, problems can be coded in m-files in a fraction of the time that it would take to create C or FORTRAN programs for the same purpose. Couple this with MATLAB's advanced data analysis, visualisation tools and special purpose application domain toolboxes and the user is presented with a uniform environment with which to explore many different software solutions.

As will be seen in the next chapters, MATLAB facilitates the development of graphical user interfaces (GUIs). GUIs aid the developer tremendously in research, since the influence of changes in design parameters can be investigated with ease, and results can be displayed in a compact and systematic manner.

1.4. Thesis overview

Chapter 2 contains a detailed literature study on some different options available for designing and implementing a controller. It also discusses some adaptive pattern recognition techniques and concludes by discussing the use of evolutionary algorithms for optimisation. Chapter 2 therefore provides the theoretical background on all the techniques applied in this project.

Modelling the hybrid control valve system is the first step of this design. The complete mathematical valve model that will be the subject of this study is described in Chapter 3.

In Chapter 4 the first, PID based, hybrid valve controller that was designed is discussed. The controller serves as an excellent preamble to the more complex controller discussed in Chapter 4, since the fundamental difficulties encountered while controlling a hybrid valve system are explored in depth, while the need for more complex control becomes apparent.

Chapter 5 discusses each of the main elements of the final, Fuzzy Logic based hybrid valve controller that was developed. The chapter commences by explaining some of the more important design decisions made, and then proceeds to explain the role and function of each of these elements.

In Chapter 6 the use of Genetic Algorithms for two purposes are discussed. The first is to find the optimal parameters for the low pass filter that constitutes the first element of the hybrid valve controller. The second purpose is the optimisation of the Fuzzy Inference System (FIS) that is responsible for making the control decisions of the controller discussed in Chapter 5.

Chapter 7 subjects the optimised hybrid valve controller to a few complex request signals in order to evaluate its ability to deal with these signals compared to the PID controller discussed in Chapter 4. Although the stability of the controller cannot be mathematically proved, it is shown that no indication of instability exists.

The final conclusions and recommendations resulting from this project as well as the areas that may require future work are discussed in Chapter 8.

2
Literature study

2.1. Introduction

Chapter two will focus on the theoretical background of this study, and will explain the software techniques that had been applied during the course of this project.

2.2. Linear PID based control

This section will give a quick review of a particular control structure that has become almost universally used in industrial control. It is based on a specific fixed-structure controller family, called the PID family.

The letters P, I and D stand for *Proportional, Integral and Derivative Control,* and simply means that PID control is control with an up-to-second-order controller. The controllers have proven to be robust in the control of many important applications, and their surprising versatility ensures continued relevance and popularity [6].

2.2.1. PID structure

Consider the simple SISO (single input, single output) control loop shown in Figure 2.1

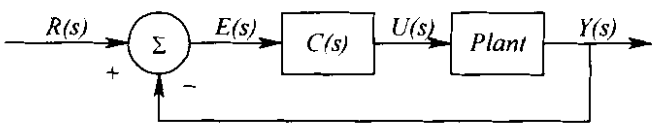

Figure 2.1 Basic feedback control loop [6]

The traditional PI, PD and PID controllers can be described by their transfer functions, relating error $E(s) = R(s) - Y(s)$, and controller output $U(s)$ as given by (2.1) to (2.4) [6]:

$$C_P(s) = K_p \tag{2.1}$$

$$C_{PI}(s) = K_p\left(1 + \frac{1}{T_r s}\right) \tag{2.2}$$

$$C_{PD}(s) = K_p\left(1 + \frac{T_d s}{\tau_D s + 1}\right) \tag{2.3}$$

$$C_{PID}(s) = K_p\left(1 + \frac{1}{T_r s} + \frac{T_d s}{\tau_D s + 1}\right) \tag{2.4}$$

where T_r and T_d are known as the *reset time* and *derivative time*, respectively.

As seen from (2.1) to (2.4), the members of this family include, in different combinations, three control modes or actions. These are *proportional (P), integral (I) and derivative (D)*. The impact of each of these actions will be discussed in the next section.

Equation (2.4) is known as the *standard form* of PID representation. Two alternative forms exist, the *series form* in (2.5) [7]:

$$C_{series}(s) = K_s\left(1 + \frac{I_s}{s}\right)\left(1 + \frac{D_s s}{\gamma_s D_s s + 1}\right) \tag{2.5}$$

or the *parallel form* (which will be used in this thesis) in (2.6) [7]:

$$C_{parallel} = K_p + \frac{I_p}{s} + \frac{D_p s}{\gamma_p D_p s + 1} \tag{2.6}$$

2.2.2. Effect of control modes on controller

Although their impact on the closed loop is far from independent of each other, the P, I and D parameters' effect on the controller's performance can be summarized as follows [6]:

Proportional action contributes to the controller according to the instantaneous value of the control error. Although a proportional controller can control any stable plant, it may provide limited performance and nonzero steady-state errors.

The contribution of the **Integral action**, on the other hand, is proportional to the accumulated error. This implies that it is a *slow reaction* control mode. A big advantage of the integral action is that it forces the steady-state error to zero in the presence of a step reference or a disturbance.

Derivative action can be seen as a *fast reaction* control mode. The action acts on the rate of change of the control error, and consequently disappears in the presence of constant errors. The derivative action is sometimes referred to as a *predictive mode*, because of its dependence on the error trend.

PID control will be applied in Chapter 4, and PD control in Chapter 6 of this thesis.

2.3. Non-linear Fuzzy logic control

This section will introduce the basic aspects of Fuzzy Logic (FL) and fuzzy rule-based systems (FRBSs) and will describe the composition and functioning of FRBSs.

2.3.1. Fuzzy logic and Fuzzy systems

Fuzzy logic is based upon the concept of variables, called *linguistic* variables, whose values are *words* rather than *numbers*. Effectively, FL may therefore simply be viewed as a formal methodology for computing with words, and not with numbers, as had become the custom. Although words convey much less precise information than numbers, their advantage is that their use is closer to human intuition, and therefore easier to implement. Furthermore, computing with words exploits the tolerance for imprecision and thereby lowers the cost of solution [8].

One of the most important areas of application of fuzzy set theory is fuzzy rule-based systems (FRBSs). These kinds of systems represent an extension to the classical rule-based systems, because the IF-THEN rules on which they are based are composed of fuzzy logic statements instead of the classical logic ones [9].

In a broad sense, a FRBS is a rule-based system where FL is used as a tool for representing different forms of knowledge about the problem at hand, as well as for modelling the interactions and relationships that exist between its variables [10]. Fuzzy logic has been applied in a wide range of problems in many different domains where some degree of uncertainty or vagueness emerge and most of the success of such applications can be contributed to the properties introduced by FRBSs [10].

At present, two types of FRBSs exist that can be applied for engineering problems: The *Mamdani* and *Takagi-Sugeno* FRBSs. Because the *Takagi-Sugeno* rule base system does not offer a fuzzified output as well as a fuzzified input [8], and since the *Mamdani* FRBS is considered more suitable for control applications [6], this dissertation will focus on the *Mamdani*, and not the *Takagi-Sugeno* FRBS.

2.3.2. The Mamdani rule base system

The Mamdani rule base system was first introduced by E. H Mamdani in 1974 [11]. He was able to augment the initial fuzzy formulation put forward by Zadeh in a way that allows it to apply a fuzzy system (FS) to a control problem. These kinds of FSs are also referred to as *fuzzy logic controllers* (FLCs), since control system design constitutes the main application of Mamdani FRBSs [6].

Figure 2.2 shows the generic structure of a Mamdani FRBS.

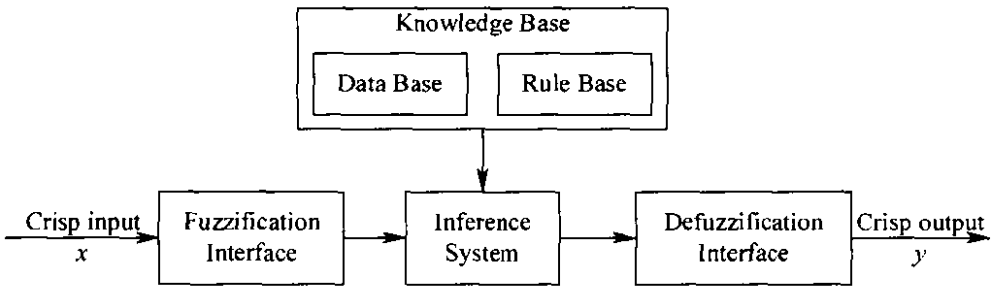

Figure 2.2 Basic structure of a Mamdani Fuzzy Rule Based System [6]

From the figure it can be seen that the structure can be divided into four main parts, the knowledge base (KB), the fuzzification interface, the inference system and the defuzzification interface. The *knowledge base* seen in the figure stores the available knowledge about the problem in the form of fuzzy "IF-THEN" rules. The other three components compose the fuzzy inference engine, which by means of the "IF-THEN" rules puts into effect the inference process on the system inputs.

Each of these parts will now be discussed.

2.3.3. The knowledge base

The KB establishes the fundamental part of the Mamdani FRBS. Its job is to model the relationship between input and output of the underlying system. The inference process therefore uses the knowledge base to generate an associated output from every observed input. As can be seen from Figure 2.2, the KB can be divided into two separate entities, the data base and the rule base.

The data base contains the linguistic term sets that are considered in the linguistic rules of the FS, as well as the membership functions that define the semantics of the linguistic labels. Each input and output variable associates with a certain partition that consists of one or more membership functions. Subsequently the membership functions of that partition each receives a linguistic association that defines its purpose. For instance, the speed of a car may be described by the partition shown in Figure 2.3:

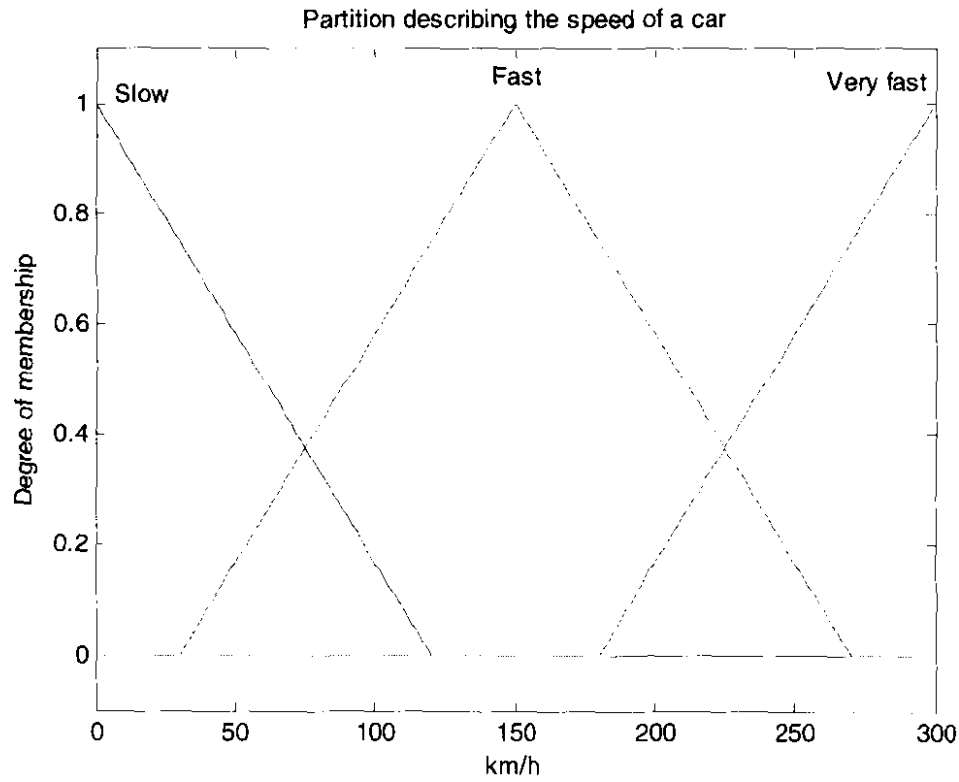

Figure 2.3 Partition describing the speed of a car

The rule base comprises a collection of linguistic rules that tell the fuzzy system how to react to certain conditions. For instance, consider an FRBS where two input variables, X_1 and X_2, and a single output variable, Y, are described by the partitions: {slow, medium, fast}, {short, medium, long} and {small, medium, large}, respectively. A typical rule base for this FRBS may consist of the following rules:

R1 : IF X_1 is *slow* and X_2 is *short* THEN Y is *small*

R2 : IF X_1 is *slow* and X_2 is *medium* THEN Y is *small*

R3 : IF X_1 is *medium* and X_2 is *short* THEN Y is *medium*

R4 : IF X_1 is *fast* and X_2 is *medium* THEN Y is *medium*

R5 : IF X_1 is *fast* and X_2 is *long* then Y is *large*

2.3.4. Fuzzification

The fuzzification interface enables Mamdani-type FRBSs to deal with crisp input values. Fuzzification establishes a mapping from crisp input values to fuzzy sets defined in the universe of discourse of that input. The membership function of the fuzzy set A' defined over the universe of discourse U associated with a crisp input value x_0 is computed as given by (2.7)

$$A' = F(x_0) \tag{2.7}$$

in which F is a fuzzification operator.

2.3.5. The inference system

The inference system is the component that derives the fuzzy outputs from the input fuzzy sets according to the relation defined through the fuzzy rules. This is done with the aid of an *implication operator*. Different implication operators exist, the most popular being the minimum t-norm and the product t-norm.

2.3.6. Defuzzification

The inference process in Mamdani-type FRBSs operates on the level of individual rules. Thus, the application of the compositional rule of inference to the current input using *m* rules in the KB generates *m* output fuzzy sets [6]. The defuzzification interface has to aggregate the information provided by the *m* output fuzzy sets and obtain a crisp output value from them. This consists of two steps, namely aggregation and defuzzification.

Aggregation is usually done by using one of three methods, the max, sum, or probalistic-or method, and defuzzification is most commonly done by using the "centre average" method. The detail behind these methods will not be discussed here, and can be found in any text book on fuzzy logic.

2.3.7. PI type fuzzy process control

The fuzzy controller that is discussed in Chapter 5 uses a fuzzy control architecture named "PI type fuzzy process control" in some of its control decisions. This architecture will be shortly discussed in this section.

The closed loop control-setup for the PI type fuzzy process control is illustrated in Figure 2.4.

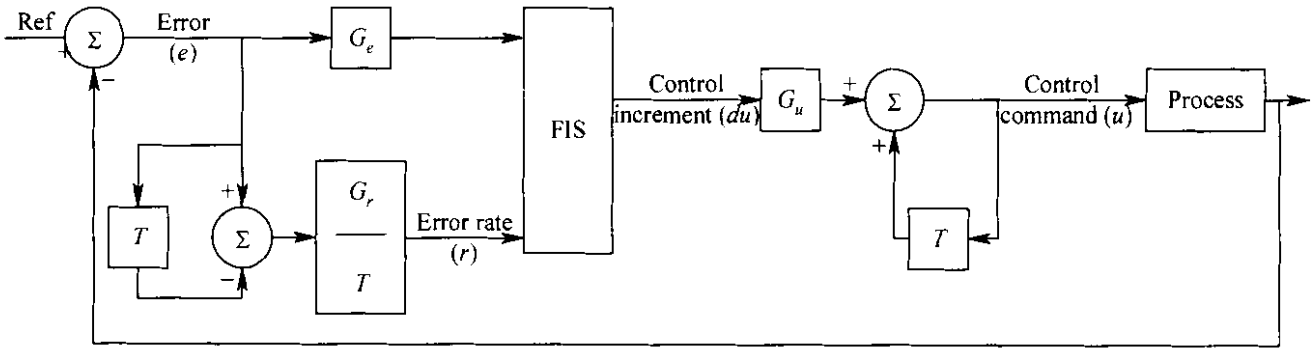

Figure 2.4 PI type fuzzy logic closed loop control

From the figure it can be seen that the fuzzy inference system accepts two scaled values as inputs: the scaled value of the current error, and the scaled value of the rate with which the error is currently changing (first derivative). The output of the FIS is a control increment (*du*) that is scaled, and subsequently added to the previous control command to obtain the final command to the process (*u*). In the figure, the letter *T* is used for a single time delay in a discrete-time system.

Partitions describing both inputs (the scaled error and scaled error rate) must now be selected. Trapezoidal membership functions are used as shown in Figure 2.5

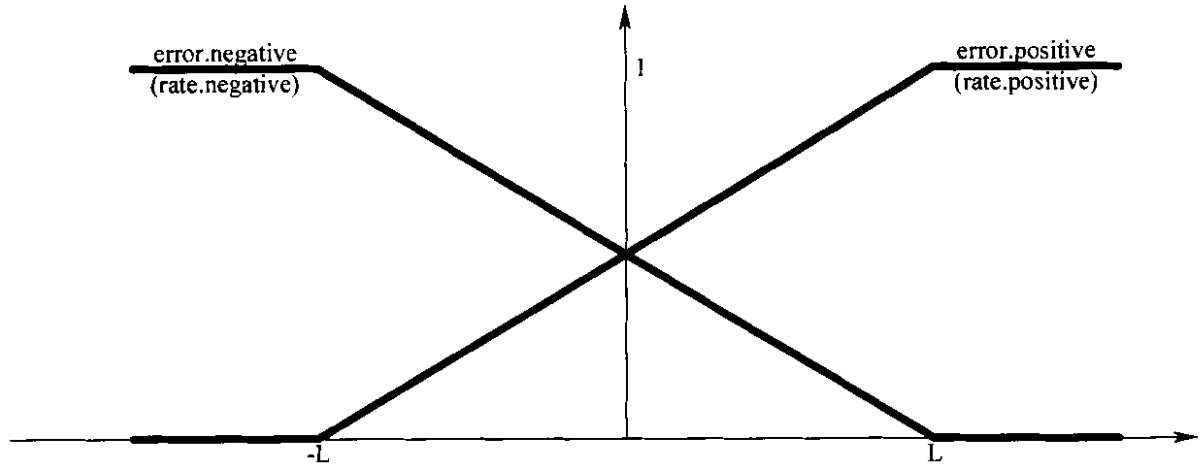

Figure 2.5 Fuzzy sets for the error and error rate [13]

The membership functions selected for the incremental output of the fuzzy controller are shown in Figure 2.6

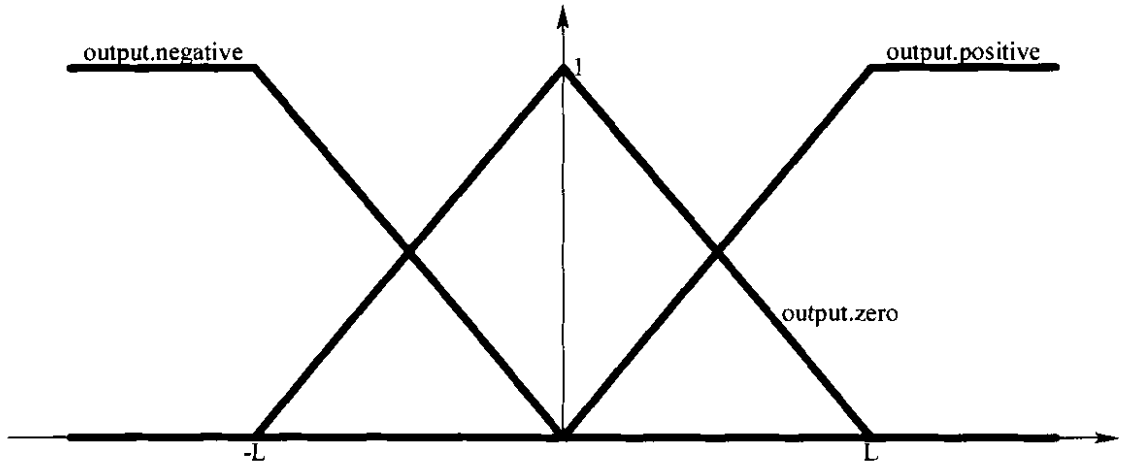

Figure 2.6 Fuzzy sets for the incremental output of the controller [13]

For basic PI-type control, 4 IF-THEN rules are required for the membership function setup as shown in the two figures. These rules are [13]:

R1: IF error is positive AND rate is positive THEN output is negative

R2: IF error is positive AND rate is negative THEN output is zero

R3: IF error is negative AND rate is positive THEN output is zero

R4: IF error is negative AND rate is negative THEN output is positive

2.3.8. Advantages of fuzzy logic control

The success of FLCs can be understood from a theoretical and practical perspective [12][11]. Theoretical reasons for fuzzy control are:

- As a general rule, a good engineering approach should be able to make effective use of all the available information. There are two sources for information in control: sensors, providing numerical measures for the system variables, and human experts, providing linguistic descriptions about the system and control instructions. FLCs, as FRBSs, constitute one of the few tools able to include both kinds of information, so they may be applied when there is a lack of one of them.
- In general, fuzzy control is a model-free approach, i.e., it does not depend on a model of the system being controlled (as does several classical control schemes such as PID control). Model free approaches make the controller design easier, since obtaining a mathematical model of the system is sometimes a very complex task. Therefore the need and importance of model-free approaches continues to increase in the realm of control engineering
- FLCs are universal approximators and are therefore suitable for non-linear control system design

Although the previous reasons illustrate the generality and rigour of FLCs, the practical significance of this control method is only proved in the possibilities of potential *applications* for it. Practical reasons for increasing utilisation of FLCs are the following:

- Fuzzy control is very easy to understand. Since FLCs emulate human reasoning, they are easy to understand for people that are non specialists in control. This has caused its application to increase in comparison to classical control techniques based on a crisp mathematical framework.

- FLCs' hardware implementation is easy and quick, and it allows a large degree of parallelisation to be used. For instance, Fuzzy logic can be easily implemented into actual systems with the use of Real Time Development Tools, which allows the controller to be tested, evaluated on-line, and adjusted if necessary.
- Developing FLCs is cheap. From a practical point of view, development costs are a key aspect for obtaining business success. Fuzzy control is easy to understand and may be mastered in a relatively short amount of time, so "software costs" are low. Once the initial software and equipment are obtained, hardware costs are low as well, since fuzzy control is easy to implement. All these reasons make fuzzy control a technique with an attractive balance between performance and cost.

2.4. Adaptive pattern recognition techniques

In recent years neural computing has emerged as a practical technology, with successful applications in many fields. The majority of these applications are concerned with problems in pattern recognition, and make use of feed-forward network architectures such as the multi-layer perceptron and the Adaptive Fuzzy Network [14]. This section will provide a quick overview of modern techniques that can be applied for pattern recognition.

2.4.1. Statistical pattern recognition

The most general and most natural framework in which to formulate solutions to pattern recognition problems is a statistical one. Such a framework recognises the probabilistic nature of the information someone seek to process as well as the form in which he should express the results. Statistical pattern recognition is a well established field with a long history. It includes techniques such as Polynomial Fitting, Error Minimisation, Correlation, Probability and Density Estimation and Bayes' theorem.

The main advantage offered by statistical pattern recognition is simplicity. However, as the complexity of the problem increases, solving it may become progressively more difficult with the use of statistical methods. For that reason many designers turn to more complex techniques like Neural Networks (NN), or Adaptive Fuzzy Systems (AFS) when advanced recognition is required by the design.

2.4.2. Pattern recognition using Neural Networks

This section will discuss an alternative approach for pattern recognition which circumvents the determination of probability densities (required by statistical pattern recognition) and is based on the idea of a discriminant function. Two techniques will be discussed, the Multi layer Perceptron and Radial Basis Functions (RBFs).

2.4.2.1. The multi-layer perceptron neural network with back-propagation

An Artificial Neural Network is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation [15]. It involves a network of simple processing elements (neurons) which can exhibit complex global behaviour, determined by connections between the processing elements and element parameters.

The Multi-layer perceptron network consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has direct connections to the neurons of the subsequent layer. The connections are each assigned a value that defines the impact the previous neuron's output plays on the input to the neuron in the next layer, the effect of this is that the output is computed as the *weighted sum* of the different networks. This is illustrated in Figure 2.7.

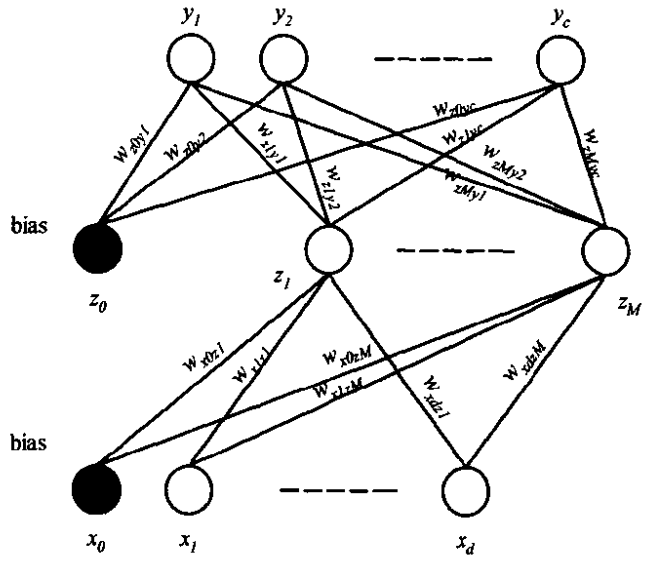

Figure 2.7 A feed-forward network having two layers of adaptive weights

For the network shown in Figure 2.7, the output can therefore be defined as (2.8) [14].

$$y_k = \tilde{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right)\right) \tag{2.8}$$

In (2.8) the functions $\tilde{g}$ and g are called *activation* functions and are generally chosen to be monotonic. A very popular choice for the activation function is the sigmoid function which is defined as:

$$g(a) = \frac{1}{1+\exp(-a)} \tag{2.9}$$

The function is plotted in Figure 2.8

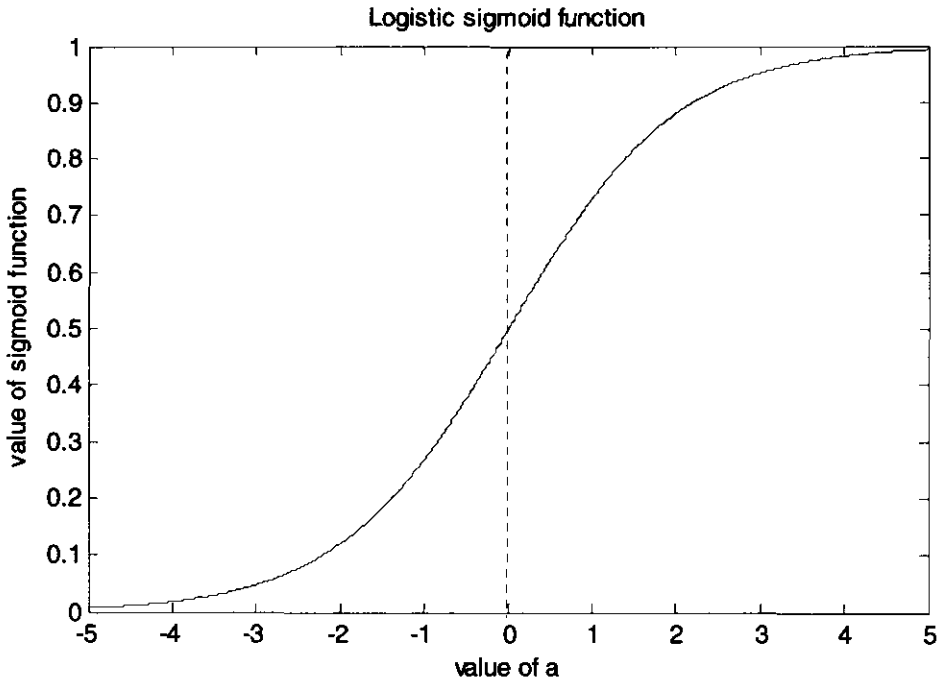

Figure 2.8 Plot of the logistic sigmoid function given by (2.9)

The logistic sigmoid function is the activation function that will be applied in this project.

However interesting the above mentioned functions may be in themselves, what has attracted the most interest in neural networks is the possibility of *learning*, which in practice means the following [16]:

Given a specific *task* to solve, and a class of function *F*, learning means using a set of *observations*, in order to find $f^* \in F$ which solves the task in an *optimal sense*.

Multi-layer networks use a variety of learning techniques, the most popular (and the one used in this project) being *back propagation*. Here the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculation is small. In this case one says that the network has *learned* a certain target function.

To adjust weights properly one applies a general method for non-linear optimisation that is called *gradient descent*. For this, the derivation of the error function with respect to the network weights is calculated and the weights are then changed such that the error decreases – thus going downhill on the surface of the error function.

2.4.2.2. Radial Basis Functions

Radial Basis Functions (RBFs) are powerful techniques for interpolation in multidimensional space. A RBF is a function which has built into it a distance criterion with respect to a centre. Radial basis functions have been applied in the area of neural networks where they may be used as a replacement for the sigmoidal hidden layer transfer function in multi-layer perceptrons.

RBF networks have the advantage of not suffering from local minima in the same way as the multi-layer perceptrons. This is because the only parameters that are adjusted in the learning process are the linear mapping from hidden layer to output layer. Linearity ensures that the error surface is quadratic and therefore has a single easily found minimum.

However, the networks have the disadvantage of requiring good coverage of the input space by radial basis functions. RBF centres are determined with reference to the distribution of the input data, but without reference to the prediction task. As a result, representational resources may be wasted on

areas of the input space that are irrelevant to the learning task [17]. For this reason, RBF networks may require immense computational power if the required results have to be very accurate, or when large amounts of training data are available.

2.4.3. Pattern recognition using Adaptive Fuzzy Systems

The basic concepts underlying FL have already been discussed in section 2.3, and will therefore not be repeated in this section. The section will rather focus on a subset of FL called Adaptive Fuzzy Systems (AFS). An Adaptive Fuzzy System is simply a Fuzzy System that can be trained with the use of some training algorithm, like back-propagation. The system makes use of all the elements that Fuzzy Systems usually consist of (like fuzzification, inference and defuzzification techniques), but provides the designer with the option of optimising the design with training data. Intricate detail behind the mathematical foundations of AFSs can be obtained from numerous sources, like Wang [12], and will not be discussed in this thesis, since the aim is set on a more practical approach.

The two most common techniques for training AFSs, namely back-propagation and nearest neighbourhood clustering will be discussed next.

2.4.3.1. Training an Adaptive Fuzzy System using back-propagation

The back propagation algorithm discussed in section 2.4.2 can also be used for updating the parameters of an AFS [12]. In this case, however, the values that are updated are not meaningless weights, but are parameters that represent linguistic based variables. In this project the values of a FLS with centre-average defuzzifier, product inference rule and Gaussian membership functions are updated. Such a FLS can be represented by the form given in (2.10) [12]:

$$f(\underline{x}) = \frac{\sum_{l=1}^{M} \bar{y}^l \left[\prod_{i=1}^{n} a_i^l \exp\left(-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} a_i^l \exp\left(-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]} \tag{2.10}$$

The update process consequently updates *three* sets of values for each fuzzy rule with each training-data pair: The input Gaussian mean ($\bar{x}_i^l$), the input and output Gaussian width (σ_i^l) and the output

Gaussian mean ($\bar{y}^l$). The mathematical details and derivations of the update-process will not be discussed in this thesis, and can be found in Wang [12].

Two clear advantages of the use of back-propagation in conjunction with AFSs become evident [12]. The first is that the parameters of the AFS have clear physical meanings, based on which the designer is able to develop a very good initial parameter-choosing method which greatly speeds up the convergence of the training procedure. On the other hand, the parameters of the back-propagation neural network have no clear physical meanings. Therefore, their initial values have to be chosen randomly, which results in slow convergence. The second advantage is that the AFS can incorporate the linguistic information in a systematic manner, whereas the back propagation neural network cannot make use of any linguistic information.

2.4.3.2. Training an Adaptive Fuzzy System using nearest neighbourhood clustering

For some practical problems, sample data may be expensive to obtain. For example, a test flight with a new helicopter is very expensive. For these small-sample problems, the designer may want a FLS that is capable of matching all the input-output pairs to any given accuracy. Training an AFS with the use of nearest neighbourhood clustering provides such a solution.

The technique of using nearest neighbourhood clustering in conjunction with AFS compares very closely to Radial Basis functions discussed in section 2.4.2.2 when the membership functions are Gaussian. Therefore its operation will not be discussed in detail, and only the dissimilarities between these two techniques will be highlighted [12]:

- Fuzzy Logic has the freedom of choosing between a wide variety of fuzzyfication, inference, and defuzzification methods. This means that the AFS that is trained with nearest neighbourhood clustering offers the designer much more design options, while RBFs does not, and is therefore essentially a special case of fuzzy logic systems.
- The membership functions of the fuzzy logic systems can take many different forms (Gaussian, triangular, trapezoid, logistic, and so on) and can be inhomogeneous, whereas the RBFs usually take few functional forms and are usually homogeneous.

2.5. Optimization using Evolutionary Algorithms

This section will give a quick overview of the method that was used for optimising the hybrid valve controller that was designed in this project.

2.5.1. Overview of Genetic Algorithms

2.5.1.1. What are Genetic Algorithms?

The GA is a stochastic global search method that mimics the metaphor of natural biological evolution. GAs operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation [18].

2.5.2. Population Representation

GAs operate on a number of potential solutions, called a *population*, which is typically composed of between 30 and 100 *individuals*. Individuals are usually encoded as *chromosomes*, which are essentially strings of values in a certain numeric representation, like binary or real-valued. Each value in the chromosome is known as a *genotype*, and represents a value in a real system that can be tuned to enhance the system's performance. The terms *population, individual, chromosome* and *genotype* will be used in the subsequent paragraphs. The reader may benefit from ensuring full understanding of their definitions.

The most commonly used representation in GAs is the binary alphabet {0, 1} and a problem with two variables, x_1 and x_2, may be mapped onto the chromosome structure as illustrated in Figure 2.9

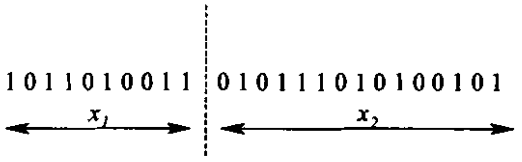

Figure 2.9 Mapping of chromosone structure in binary alphabet

In the figure, x_1 is encoded with 10 bits and x_2 with 15 bits, possibly reflecting the level of accuracy or range of the individual decision variables. Examining the chromosome string in Figure 2.9 in isolation yields no information about the problem we are trying to solve. It is only with the decoding of the chromosome into its actual values that any meaning can be applied to the representation.

2.5.3. Objective and Fitness functions

Having decoded the chromosome representation into the decision domain, it is possible to assess the performance, or *fitness* of individual members of a population. This is done through an objective function that characterises an individual's performance in the problem domain. Another function, the *fitness function*, is normally used to transform the objective function value into a measure of relative fitness [19], thus it can be written as in (2.11)

$$F(x) = g(f(x)) \tag{2.11}$$

where f is the objective function, g transforms the value of the objective function to a non-negative number and F is the resulting relative fitness. This mapping is always necessary when the objective function is to be minimized as the lower objective function values correspond to fitter individuals. In many cases, the fitness function value corresponds to the number of offspring that an individual can expect to produce in the next generation.

2.5.4. Selection

Once the individuals have all been assigned a fitness value, they can be chosen from the population, with a probability according to their relative fitness, and recombined to produce the next generation. Genetic operators manipulate the genes of the chromosomes directly, using the assumption that certain individual's gene codes, on average, produce fitter individuals.

2.5.5. Crossover

The basic operator for producing new chromosomes in the GA is that of crossover. Like its counterpart in nature, crossover produces new individuals that have some parts of both the parents' genetic material. The simplest form of crossover is that of single-point crossover.

Consider, for instance, the two binary strings:

P1 = 1 0 1 1 1 0 0 1

P2 = 1 0 0 1 0 1 0 0

The principle is to select a crossover position *(i)* at random and subsequently producing new individuals by exchanging the genetic information between P1 and P2 at this point. For instance, if the crossover point $i = 3$ is selected, the two offspring that will be produced are:

O1 = 1 0 1 1 0 1 0 0

O2 = 1 0 0 1 1 0 0 1

Other crossover methods, like multi-point crossover [20], uniform crossover [21], shuffle crossover [22] and reduced surrogate based crossover [23] also exist, but the fundamental idea remains the same, and therefore these methods will not be discussed in this thesis.

2.5.6. Mutation

In natural selection, mutation is a random process where one allele of a gene is replaced by another to produce a new genetic structure. In GAs, mutation is randomly applied with low probability and modifies the elements in the chromosomes. Usually considered as a background operator, the role of mutation is often seen as providing a guarantee that the probability of searching any given string will never be zero and acting as a safety net to recover good genetic material that may be lost through the action of selection and crossover [24].

The effect of mutation on a binary string is illustrated in Figure 2.10. Here, binary mutation flips the value of the bit at the loci selected to be the mutation point. As can be seen, the mutation generates a significant difference in the value

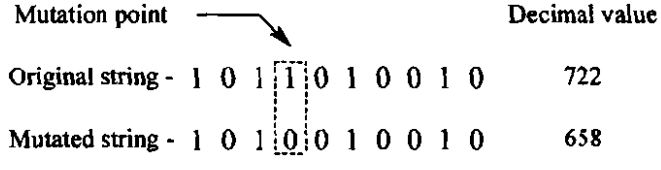

Figure 2.10 Binary mutation

2.5.7. Reinsertion

Once a new population has been produced by selection and recombination of individuals from the old population, the fitness of the individuals from the new population may be determined.

To maintain the size of the original population, some new individuals have to be reinserted into the old population, and some old individuals have to be left out. Different reinsertion schemes are available, but the general idea is to replace the least fit individuals with offspring that has a better fitness value. For an individual to survive successive generations, it must be sufficiently fit to ensure propagation into future generations. Such a strategy is known as an *elitist strategy* of reinsertion.

The new population that is selected is subsequently assigned recombination probabilities, and the whole process continues through subsequent generations. In this way the average performance of individuals in a population is expected to increase, as good individuals are preserved and bred with one another while the less fit individuals die out.

The GA is terminated when some criteria are satisfied, e.g. a certain number of generations, a mean deviation in the population, or when a particular point in the search space is encountered.

2.5.8. Real-coded Genetic Algorithms

Real coded genetic algorithms (RCGAs) operate on precisely the same principles as binary coded GAs, the only difference being that the chromosomes do not consist of binary values, but real values. For that reason the methods of recombination and mutation must be somewhat different – and since real coded GAs will be used in this project, those differences will now be discussed shortly.

2.5.8.1. Recombination in real-coded GAs

As in binary recombination, a number of different techniques exist for recombining different individuals in RCGAs. Techniques differ from flat crossover [9], BLX-alpha crossover [25] and discrete crossover [25], but the technique most commonly applied is simple crossover, which is the real-coded equivalent to single-point crossover. Therefore the offspring of two individuals, x^1 and x^2 each with n genotypes and random crossover point k, will be as shown in Figure 2.11

$$O1 = (x_1^1, x_2^1, \ldots, x_k^1, x_{k+1}^2, \ldots, x_n^2)$$

$$O2 = (x_1^2, x_2^2, \ldots, x_k^2, x_{k+1}^1, \ldots, x_n^1)$$

Figure 2.11 Simple crossover

2.5.8.2. Mutation in real-coded GAs

Mutation in real-coded genetic algorithms can take on two forms, random mutation, where the mutated gene is drawn randomly, from the interval over which the genotype ranges, and non-uniform mutation, in which the mutation step-size decreases with increasing number of generations [26]. The latter technique involves some mathematical computations, and the interested reader may refer to Herrera, Lozano and Verdegay [25].

2.6. Conclusion

Chapter 2 examined and discussed some of the software and computational techniques that had been applied during the course of this project. In the next chapters it will become clear that most of the background discussed in this chapter was used to make informed design choices during this project. Therefore Chapter 2 serves as design basis for the rest of the thesis.

3
Modelling the hybrid control valve

3.1. The hybrid valve model

As mentioned before, many control problems require the accurate modelling of some plant or process in order to cut costs, but still provide the controller with an entity that can offer a calculated response to the control inputs generated. For this particular project, as discussed in section 1.3.2, a hybrid valve system has to be modelled.

In Chapter 2, Multi Layer Neural Networks (NN), and Adaptive Fuzzy Systems (AFS) are discussed for the application of recognising patterns or behaviour properties. These techniques may offer an excellent way of modelling a control valve. However, adaptive techniques such as these require a very large amount of training data before accurate recognition is possible. Obtaining such data is not an easy feat, and may be very time consuming as well as extremely costly.

Moreover, this project's aim is not the modelling of a valve, but rather the development of a hybrid valve controller. The controller is meant to primarily focus on the phenomenon caused by the non-linearity of certain control valves, as discussed in section 1.1.2.

For this reason, a simple, idealised model for the valve system is proposed. The model assumes relatively low mach[1] values of mass flow rates, and do not take into account effects caused by pressure changes across the valve, like valve-choking. Figure 3.1 shows an illustration of the model that was created.

[1] The mass flow rate of gasses through a valve is usually measured relative to the speed of sound, which is 1 *mach*

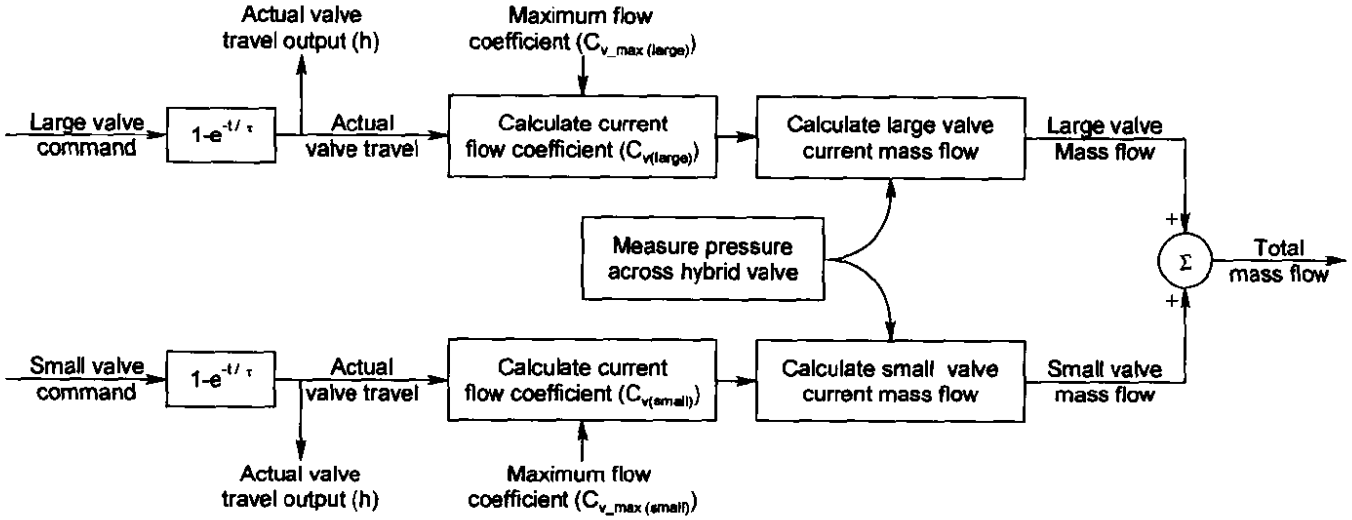

Figure 3.1 Hybrid valve model created

From the diagram in Figure 3.1 it is seen that the time constants associated with opening the valves are taken into consideration. From this calculation, the actual valve travel at any point in time is obtained. As can be seen from Figure 1.6 on page 7, actual valve travel is an important part of valve-control, and therefore the model provides this component as an additional output (this is also the case with actual valves in practice).

The model has been designed in such a way that the maximum flow coefficient of each valve can be adjusted; the reason for this will become clear in later chapters. The model makes use of the inherent flow characteristics of the valves (as described in section 1.1.2) to calculate the current flow coefficient of the valve from its current travel.

The mass flow rate through each valve is calculated by taking into account the current flow coefficient as well as the pressure across the valves. The pressure drop across each valve is assumed to be equal.

The simplified equation used for calculating the mass flow rate is given by (3.1)

$$\dot{m} = C_v \cdot \sqrt{\Delta p} \tag{3.1}$$

where Δp is the pressure differential across the valve.

Of course, the total mass flow through the whole hybrid system is calculated by adding the mass flows through each individual valve.

3.2. Overpowered valve command

As will be seen in the following chapters, the valve controller is allowed (at some occasions) to issue an "overpowered" command to either one of the two valves. Overpowering the command simply means that the command issued to the valve exceeds its maximum capability. Although the command is, of course, not expected to be met, issuing such an overpowered command usually increases the reaction time of the valves – and is therefore used to improve the hybrid valve system's performance.

3.3. Conclusion

This chapter discussed the techniques used for modelling the hybrid valve model that is the subject of this project. The model that has been created was discussed, and the concept of overpowering the valve command was explained.

4
Valve controller for simple input requests

4.1. Overview and motivation

As was mentioned in section 1.3.3, it is important to investigate a wide range of possibilities in order to find the best suited controller for the hybrid control valve system. In Chapter 2, the many advantages and popularity of PID control were discussed. Because of these features, this study will not be complete without examining the application of PID control on the hybrid valve controller.

It was mentioned that simplicity is probably the PID controller's main advantage. However, the simplicity of these controllers is also their weakness [3]. As will be seen in this chapter, the PID based valve controller that was developed is quite complex, but is only capable of handling rather simple input signals.

A very important purpose that this controller served, however, was to assist in developing a better comprehension of the main challenges that would be faced in coordinating the operation of two separate valves. This chapter will discuss the operation of the crisp, PID based hybrid valve controller that was developed and will also point out the difficulties and challenges that were discovered for developing a hybrid valve controller.

4.2. The crisp, PID based valve controller

4.2.1. The Logic controller and PID structure

Figure 4.1 gives an illustration of the PID based controller that was developed for the hybrid control valve system.

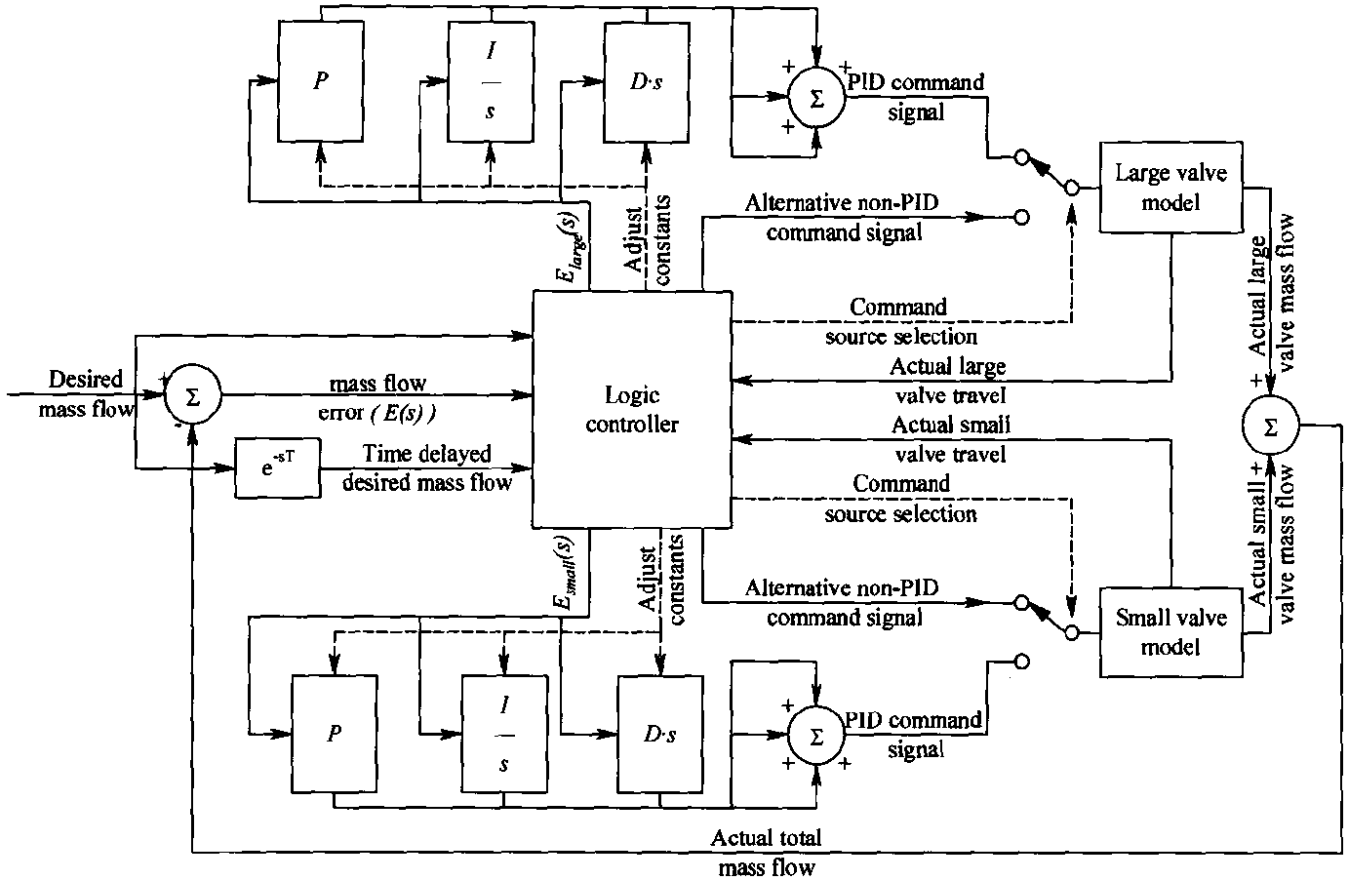

Figure 4.1 Illustration of the PID based hybrid valve controller

It is quite apparent from Figure 4.1 that the complete PID-based control setup of the hybrid valve system is by no means an uncomplicated configuration. The complexity of this problem necessitates complex control, and therefore the burden is simply shifted to the logic controller shown in the figure.

The PID control structure does not leave room for non-linear control [3]. Therefore, one of the main tasks of the logic controller is to subdivide the input-output space into separate, *linear* regions and to identify the regions capable of being controlled by a PID controller. Each of these regions may have different optimal P, I and D constants, and for that reason the logic controller must be able to adjust the constants accordingly.

Some regions, however, require unique control and simply supplying the PID controller with a mass flow error will not produce the required results. For instance, if the desired mass flow through the hybrid system is below the minimum mass flow of the larger valve, the only way to reach it would be

to close the larger valve completely. For these "special" situations, the controller has the ability to bypass the PID control structure, and to provide alternative valve commands.

Another challenge the logic controller may encounter is providing the PID-controllers of each valve with realistic mass flow errors. Because the separate valves work together, the actual total mass flow error of the hybrid system might often exceed the maximum mass flow of a single valve. To overcome this problem, the logic controller should be aware of the flow characteristics of each valve, and should be able to adjust the mass flow error provided to the valve's PID controller according to its capacity.

As will be seen next, the valve controller shown in Figure 4.1 provides control only for simple and very predictable input request signals. For more intricate control, additional information about the expected behaviour of the input request signal will be necessary. In order to obtain this, more inputs to the logic controller is required. This might sound like a simple solution, but for each input added, the number of input-output regions increases at an astounding rate, which, in turn complicates the control.

4.2.2. Input request constraints

For this study, the linear regions of the valve controller in Figure 4.1 were optimised for a reference signal that consists of step requests that occur at a low frequency compared to the reaction time of the valve. This means that the stable hybrid valve system will have matched the last input request long before a new step input occurs. It should be noted that this does not imply that the frequency *content* of the step requests are low, but merely that a sufficient delay in time is required before the next step may occur.

For instance, if we assume that it normally takes the valve just over 1.2 seconds to reach the required mass flow, a typical request signal may look like the one in Figure 4.2.

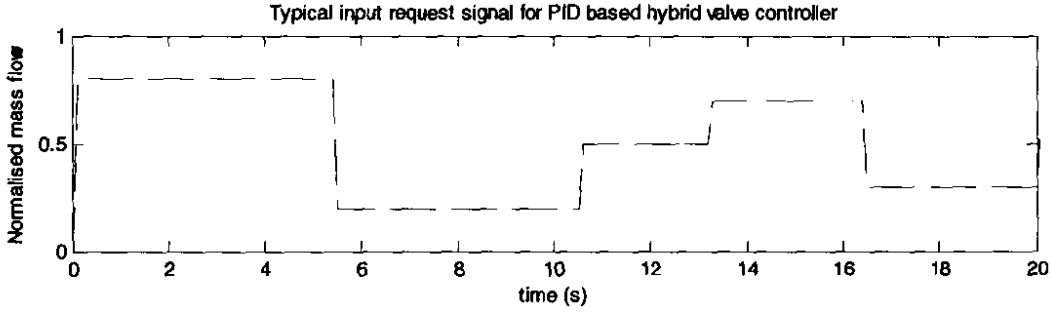

Figure 4.2 Typical input request signal for PID based hybrid valve controller

4.3. Input Regions – operation of the PID based controller

4.3.1. Step request classification sections

The logic controller for this application uses the five inputs shown in Figure 4.1, and divides the input-output space into 15 different regions. These regions are set up in such a way as to keep the control as simple as possible while still obtaining satisfactory settling times on the desired mass flow. Furthermore, they are valid only for the constrained input request signals described above.

In order to explain the operation of the PID controller that was developed, the *input* space of the PID controller will be broadly divided into 5 different sections, each requiring diverse reactions from the PID controller. Although the 5 sections have some relationship to the 15 regions mentioned above, they should not be confused; the 5 sections are purely defined for purposes of explanation. The 5 sections are:

1. A step request that stays below the minimum capability of the large valve.
2. A step request from below the minimum capability of the large valve to above the maximum capability of the small valve
3. A step request that stays within the range of the large valve
4. A step request above the maximum ability of the large valve alone.
5. A step request from above the maximum capability of the small valve to below the minimum capability of the large valve.

It has already been mentioned that the mass flows in this dissertation will be normalised to the minimum mass flow of the larger valve. In Chapter 3 it was mentioned that the valve model allows adjustment of the maximum flow coefficient of each of the valves. For the PID controller, the valves were designed in such a way that the minimum mass flow of the larger valve is almost equal to the maximum mass flow of the smaller valve. A mass flow of $1 \cdot \dot{m}(\text{large})_{\min}$ is therefore also equal to the maximum mass flow of the smaller valve. The maximum mass flow of the larger valve is $10 \cdot \dot{m}(\text{large})_{\min}$, so the maximum mass flow of the whole hybrid system is $11 \cdot \dot{m}(\text{large})_{\min}$. For this controller, no overpowering of the valve commands was allowed. The operation of the different sections will now be described.

4.3.2. Input section 1

The first section that will be discussed for the PID controller is a step request that stays below the minimum capability of the large valve. For instance, if we assume that both valves are closed at t = 0 s and a step request occurs to a mass flow of $0.6\,\dot{m}(\text{large})_{\min}$, the response will be as shown in Figure 4.3.

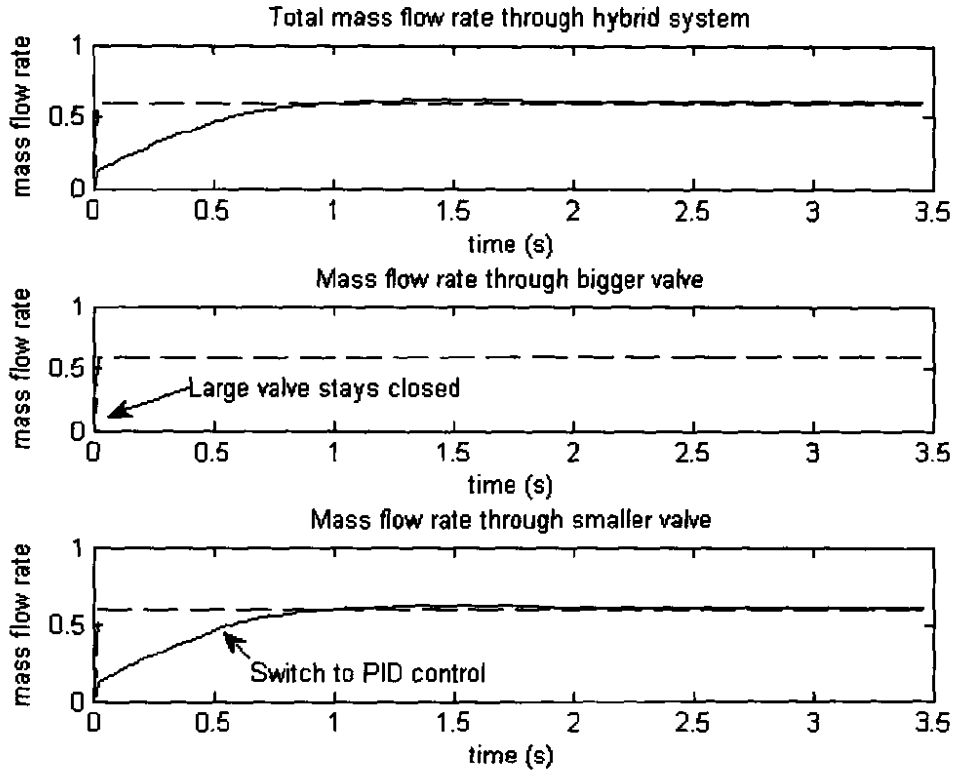

Figure 4.3 Response to a step input from $0\,\dot{m}(\text{large})_{\min}$ to $0.6\,\dot{m}(\text{large})_{\min}$

It can be seen from Figure 4.3 that the large valve stays closed at all times, while the logic controller uses the smaller valve to reach the requested mass flow rate. At roughly t = 0.6 s, a small change in the reaction of the small valve can be discerned. This change can be better explained by referring to the graphs in Figure 4.4. The top graph demonstrates how the normalised valve *travel* changes in reaction to the commands given to the valves, while the lower graph shows how the mass flow rate changes.

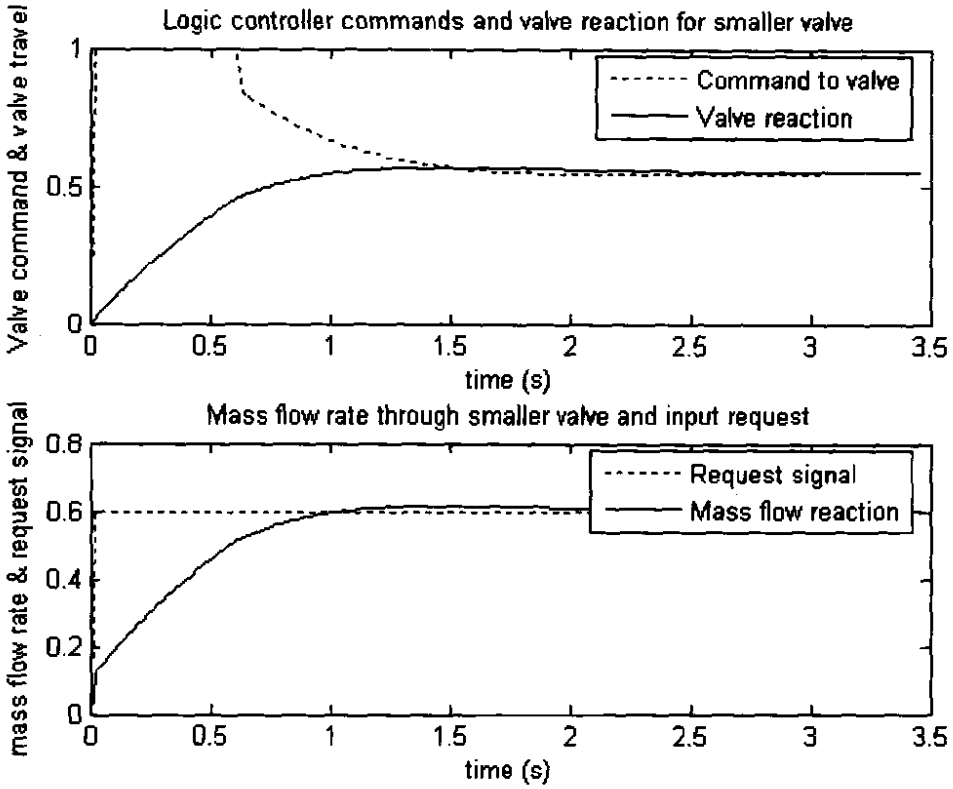

Figure 4.4 The commands given to the small valve, and its reaction to it

From the top graph of Figure 4.4 it is possible to derive the actions taken by the logic controller to reach the mass flow request as quickly as possible. For roughly the first 0.6 seconds, it can be seen that the command given to the small valve is to open *completely*. This can be called the "bang-bang control stage". Referring to Figure 4.1, this stage is made possible by the "bypass" ability of the logic controller which enables it to take over control from the PID controller for a short while. For more accurate control, the command is handed back to the PID controller after 0.6 seconds, and the requested mass flow is reached soon after.

If, for instance, a step request is next received to a mass flow rate of $0.3 \cdot \dot{m}(\text{large})_{\min}$, it can be seen from Figure 4.5 that the controller will respond in a very similar manner, the only difference being that the "bang-bang control" will now command total closure, rather than commanding the valve to open completely.

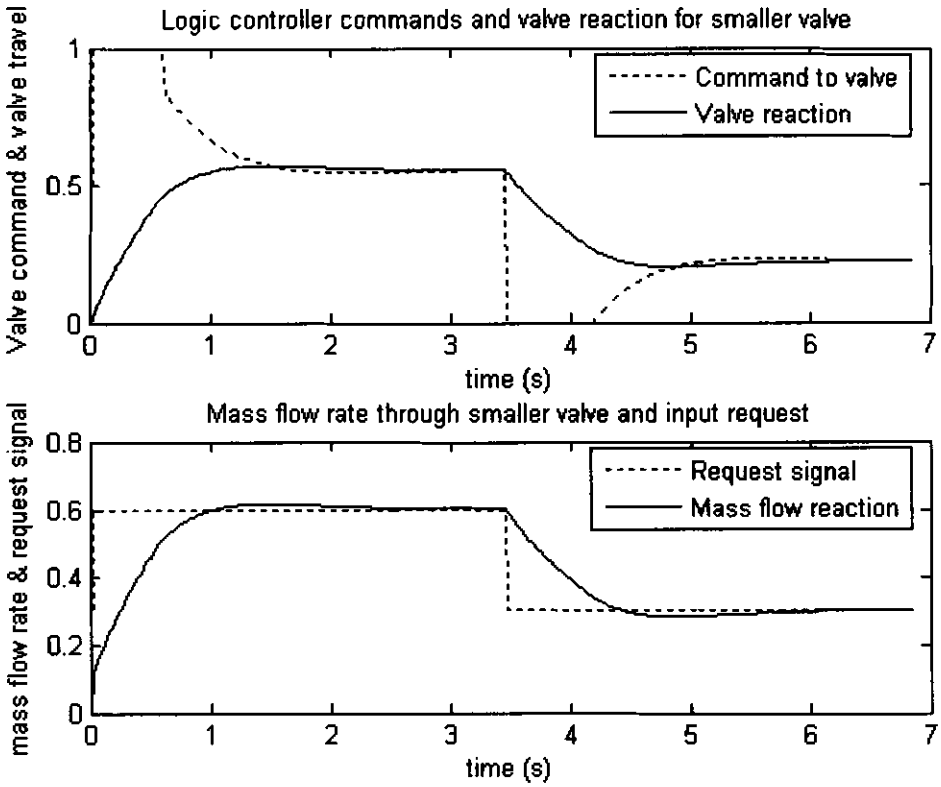

Figure 4.5 Response of the controller to a step input from $0.6 \cdot \dot{m}(\text{large})_{\min}$ to $0.3 \cdot \dot{m}(\text{large})_{\min}$

4.3.3. Input section 2

Section 2 covers a step request from below the minimum capability of the large valve to above the maximum capability of the small valve. Two different situations can occur in this section. The first is illustrated in Figure 4.6.

In the figure, a step request occurs from a steady state mass flow rate of $0.8 \cdot \dot{m}(\text{large})_{\min}$ to a mass flow rate of $1.3 \cdot \dot{m}(\text{large})_{\min}$. As can be expected, the mass flow rate of $0.8 \cdot \dot{m}(\text{large})_{\min}$ is generated by the smaller valve alone. However, as was mentioned before, the maximum mass flow of the smaller valve is close to $1 \cdot \dot{m}(\text{large})_{\min}$. Therefore, in order to reach the required mass flow rate, the larger valve will have to be opened.

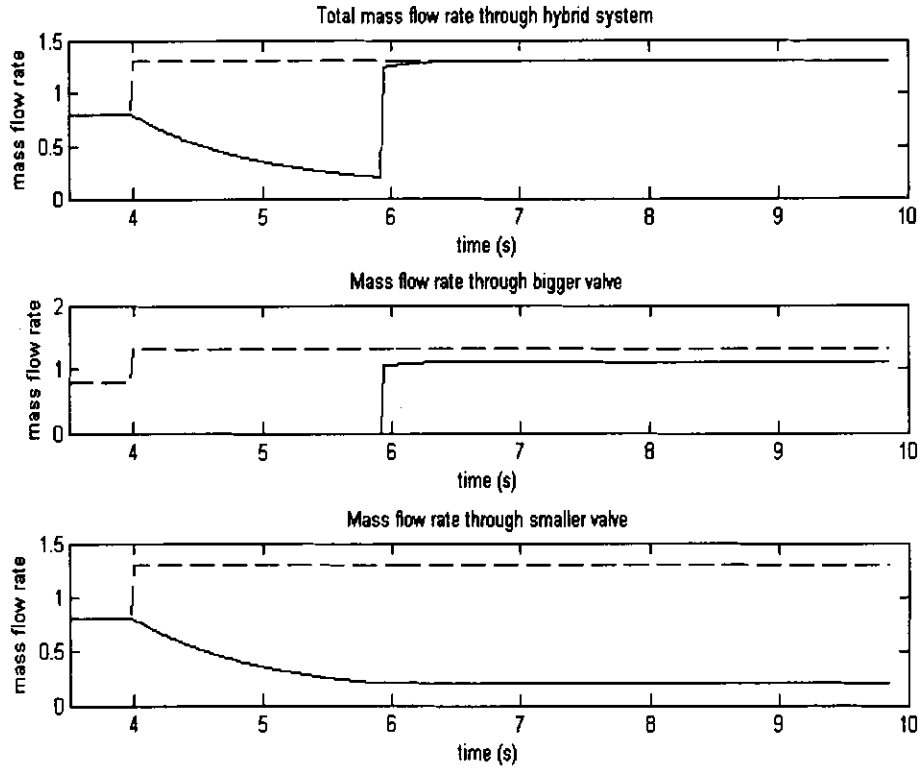

Figure 4.6 Response of both valves to a step input from $0.8 \cdot \dot{m}(\text{large})_{\min}$ to $1.3 \cdot \dot{m}(\text{large})_{\min}$

Because of the non-linearity of the larger valve, the moment it is opened, it immediately allows a mass flow rate of $1 \cdot \dot{m}(\text{large})_{\min}$. Furthermore, at the moment the step request is received, the smaller valve is allowing a mass flow rate of $0.8 \cdot \dot{m}(\text{large})_{\min}$. This means that, if the larger valve is opened immediately, the hybrid valve system will allow a total mass flow rate of $1.8 \cdot \dot{m}(\text{large})_{\min}$!

Figure 4.6 illustrates the process that is followed in order to minimise the overshoot caused by the valve's non-linearity. However, the process can be better described by inspecting the commands given to each valve separately.

Figure 4.7 illustrates the response of the smaller valve to the step request.

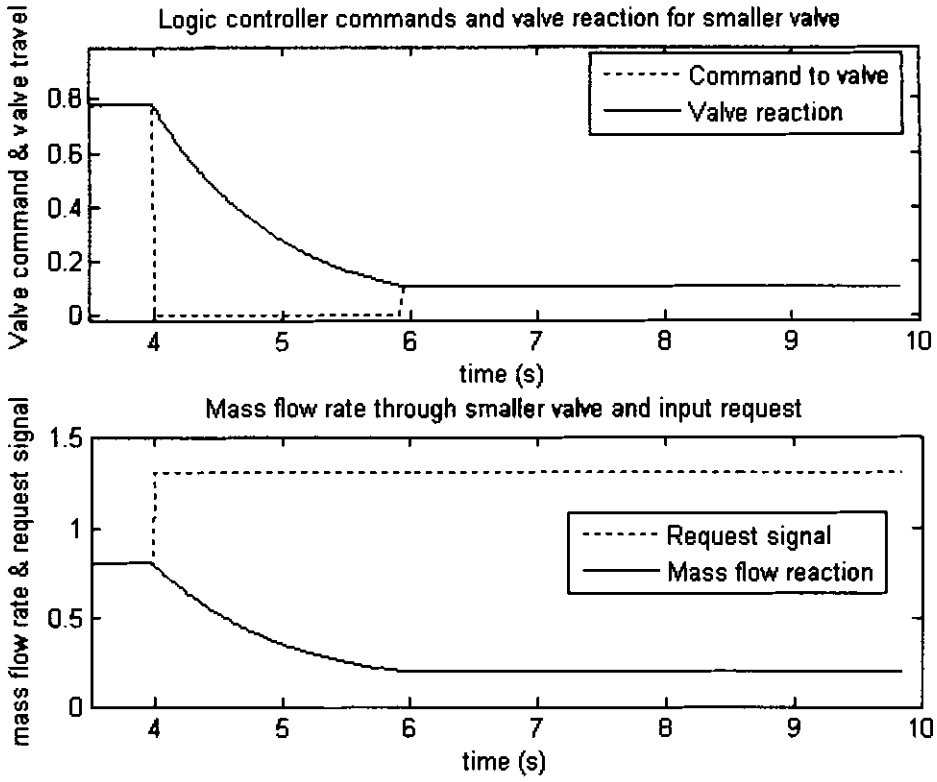

Figure 4.7 Response of the smaller valve to a step input from $0.8 \cdot \dot{m}(\text{large})_{\min}$ to $1.3 \cdot \dot{m}(\text{large})_{\min}$

In Figure 4.7 it can be seen that the logic controller's first reaction upon receiving the step request is to give the smaller valve the command to start closing. The smaller valve is closed until the mass flow generated by it is close to 0.3. This means that the larger valve can now be opened without causing much overshoot.

Because the large valve has a much bigger flow crossection than the small valve, it can correct small mass flow errors much quicker. For this reason, the control is handed to the PID controller of the large valve immediately after it has been opened.

Figure 4.8 shows the commands given to the larger valve.

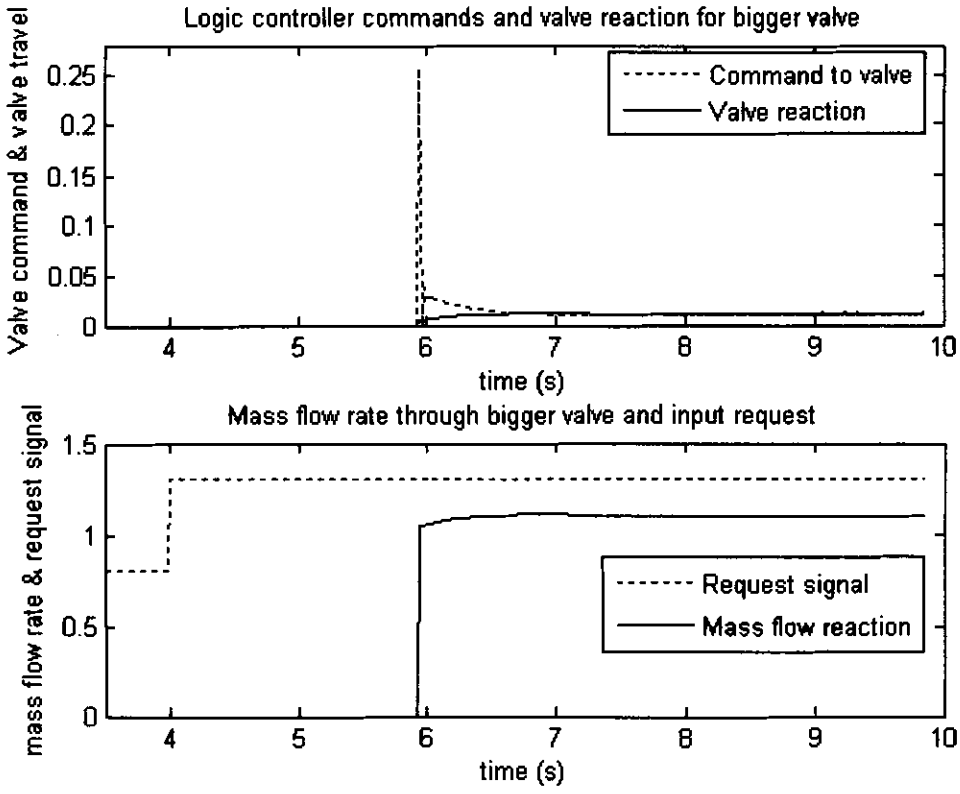

Figure 4.8 Response of the large valve to a step input from $0.8 \cdot \dot{m}(\text{large})_{\min}$ to $1.3 \cdot \dot{m}(\text{large})_{\min}$

In the top graph of Figure 4.8 it can be seen that the first PID command to the larger valve is quite high. This is because the mass flow error at that moment is more than $1 \cdot \dot{m}(\text{large})_{\min}$. However, the moment the large valve opens, the error decreases drastically and, as can be seen, the PID controller soon guides the large valve in such a way that the required mass flow is accurately reached.

The second situation that can occur for section 2 is much simpler. This is when the step request is of such a nature that the large valve can be opened immediately without generating any overshoot.

An example of this is shown in Figure 4.9 where the step request occurs at t = 4 s from a mass flow rate of $0.8 \cdot \dot{m}(\text{large})_{\min}$ to $2 \cdot \dot{m}(\text{large})_{\min}$.

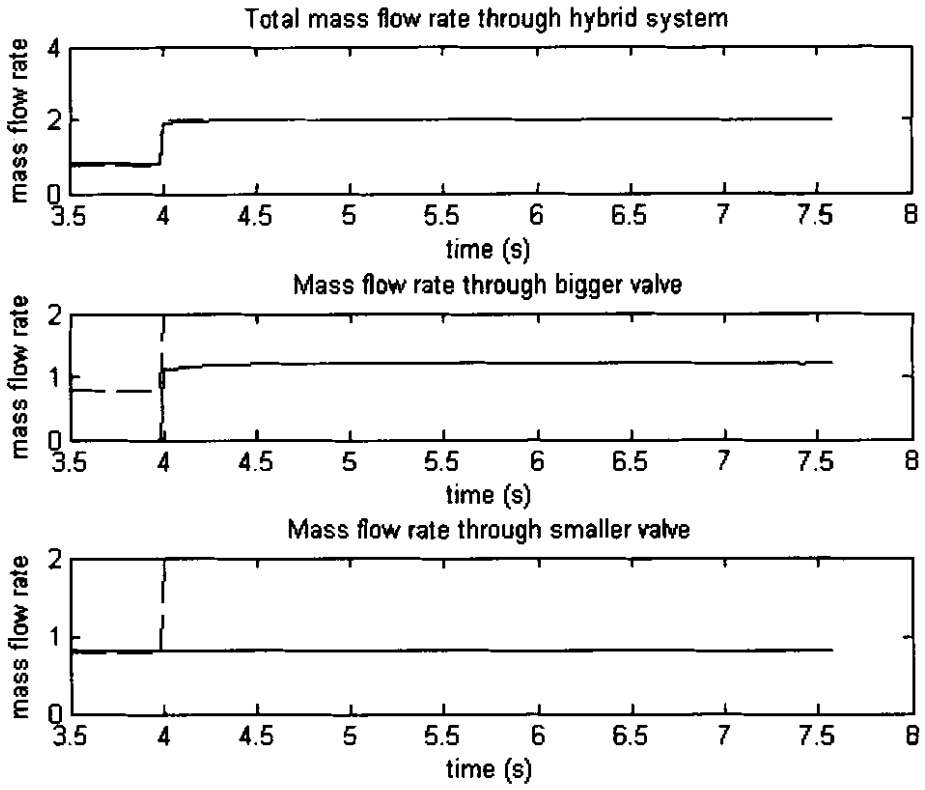

Figure 4.9 Response of both valves to a step input from $0.8 \cdot \dot{m}(\text{large})_{\min}$ to $2 \cdot \dot{m}(\text{large})_{\min}$

4.3.4. Input section 3

Section 3 is a step request that stays within the operational range of the large valve. Technically, it might be possible in this region to close the small valve completely, and only use the larger valve to reach the requested mass flow rates. This is, however, not done for the following reasons:

- The smaller valve may aid the hybrid system to reach the requested mass flows quicker.
- If a mass flow request is received in the future that requires only the smaller valve, time will be saved because the controller will not have to waste time opening the valve.
- Closing and opening the smaller valve also produces a non-linear jump in the mass flow (although much smaller than the larger valve)

Keeping the smaller valve open does, however, complicate the control considerably. This will now be explained by referring to some examples.

The first example that will be explained is the simplest one. This is simply a mass flow step request change within the range of the large valve which is aided by the small valve.

Figure 4.10 shows the response of the PID based controller to a step request from $5 \cdot \dot{m}(\text{large})_{\min}$ to $8 \cdot \dot{m}(\text{large})_{\min}$. It can be seen that the small valve is used to aid the hybrid system to reach the required mass flow as soon as possible, but the accurate PID control is done with the large valve.

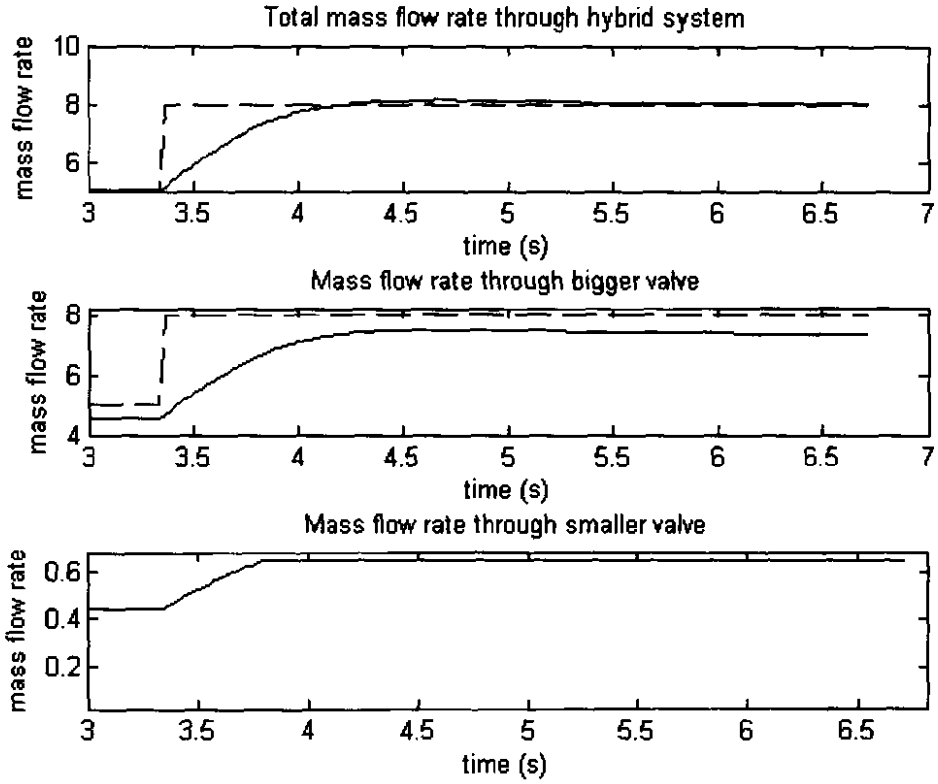

Figure 4.10 Response of both valves to a step input from $5 \cdot \dot{m}(\text{large})_{\min}$ to $8 \cdot \dot{m}(\text{large})_{\min}$

The commands given to the large valve is illustrated in Figure 4.11. As can be observed, the same "bang-bang control" method is used for the large valve in order to reduce settling time.

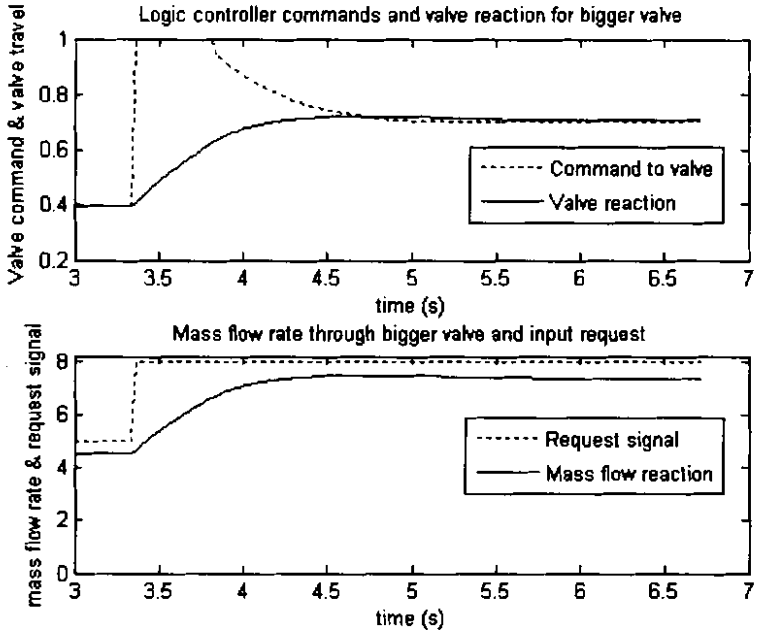

Figure 4.11 Response of the large valve to a step input from $5 \cdot \dot{m}(\text{large})_{\min}$ to $8 \cdot \dot{m}(\text{large})_{\min}$

The aid provided by the smaller valve, also to accelerate the hybrid valve's mass flow increase, is shown in Figure 4.12. The aid is only provided for a short while, so that the small valve does not impede on the PID controller's accuracy.

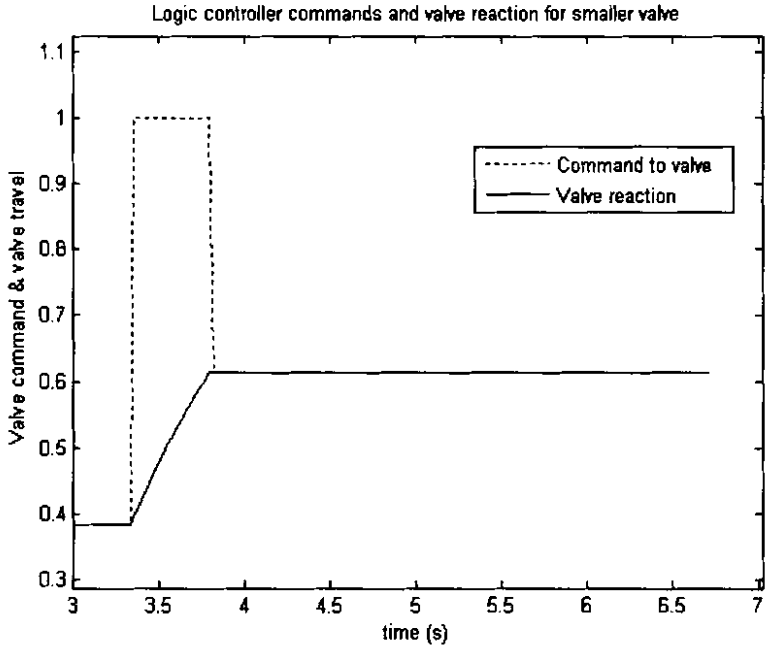

Figure 4.12 Response of the small valve to a step input from $5 \cdot \dot{m}(\text{large})_{\min}$ to $8 \cdot \dot{m}(\text{large})_{\min}$

Now, consider the series of step requests shown in Figure 4.13:

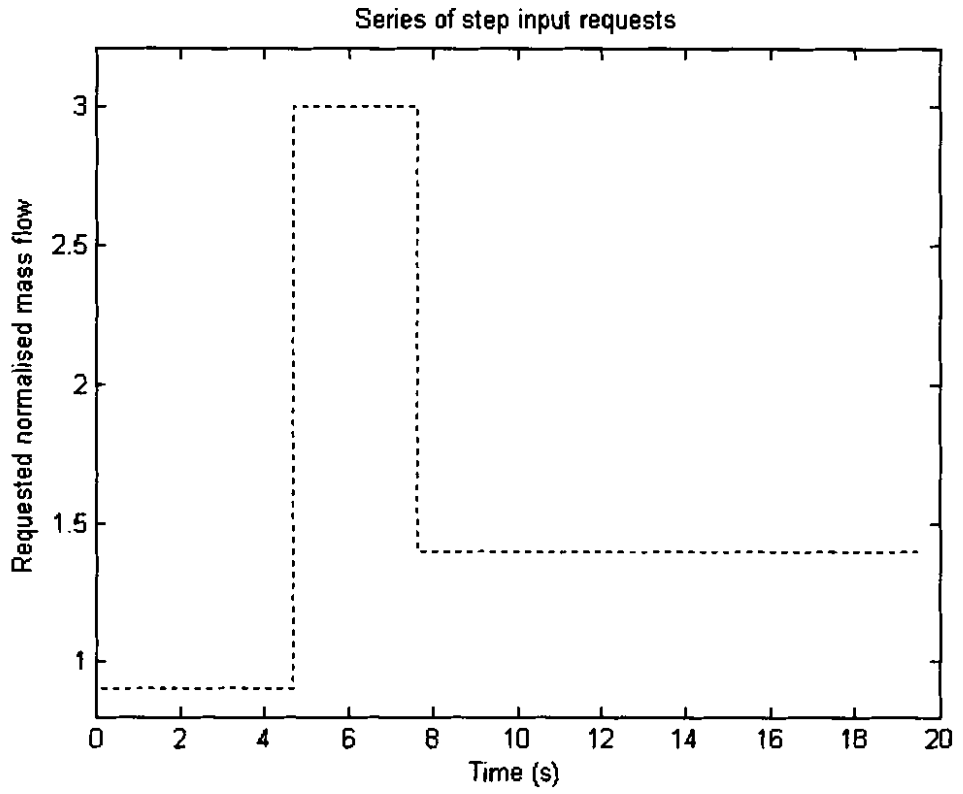

Figure 4.13 Example of a series of step input requests

As was explained, the first step request (to a mass flow rate of $0.7 \cdot \dot{m}(\text{large})_{\min}$) will cause the small valve to be opened almost completely. The second step request will cause the larger valve to open, and the request will be matched by the larger valve with the aid of its PID controller. The third step, to a mass flow of $1.4 \cdot \dot{m}(\text{large})_{\min}$ can, however, not be acquired by only controlling the large valve. This is because the small valve still provides a mass flow rate of $0.7 \cdot \dot{m}(\text{large})_{\min}$. The minimum possible mass flow for this setup, without closing the small valve, is therefore $1.7 \cdot \dot{m}(\text{large})_{\min}$.

This means that the smaller valve will have to be closed a bit at first. Figure 4.14 shows how the controller deals with the series of step inputs.

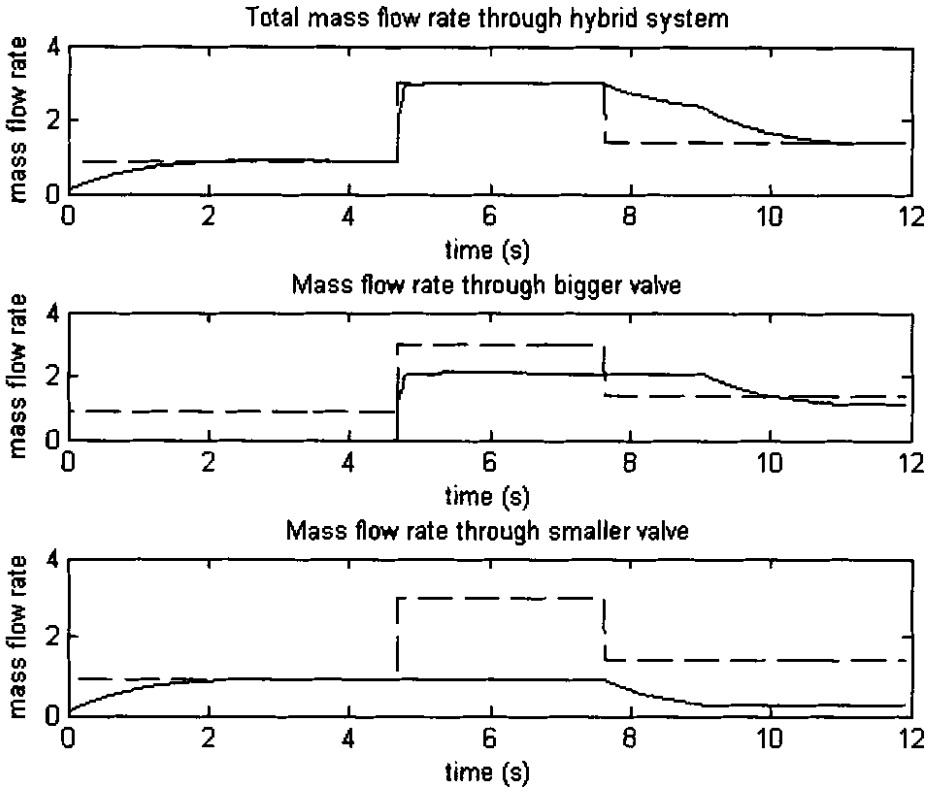

Figure 4.14 Response of both valves to the series of step inputs shown in Figure 4.13

In the bottom graph in Figure 4.14 it can be seen how the smaller valve closes until the mass flow through it is less than $0.4 \cdot \dot{m}(\text{large})_{\min}$. The large valve is then able to close as well so that the required mass flow of $1.4 \cdot \dot{m}(\text{large})_{\min}$ can be reached.

In the example explained above, the large valve waited until the small valve has reached an appropriate mass flow before it started closing. The controller does this to ensure that the large valve does not reach its minimum mass flow, and closes before the mass flow could be reached. This, however, is not optimal, especially if the large valve has a long way to go, for instance from $9 \cdot \dot{m}(\text{large})_{\min}$ down to $1.5 \cdot \dot{m}(\text{large})_{\min}$.

In order to save time in such an instance, the controller allows both valves to simultaneously close if the downward step is quite large. This is shown in Figure 4.15

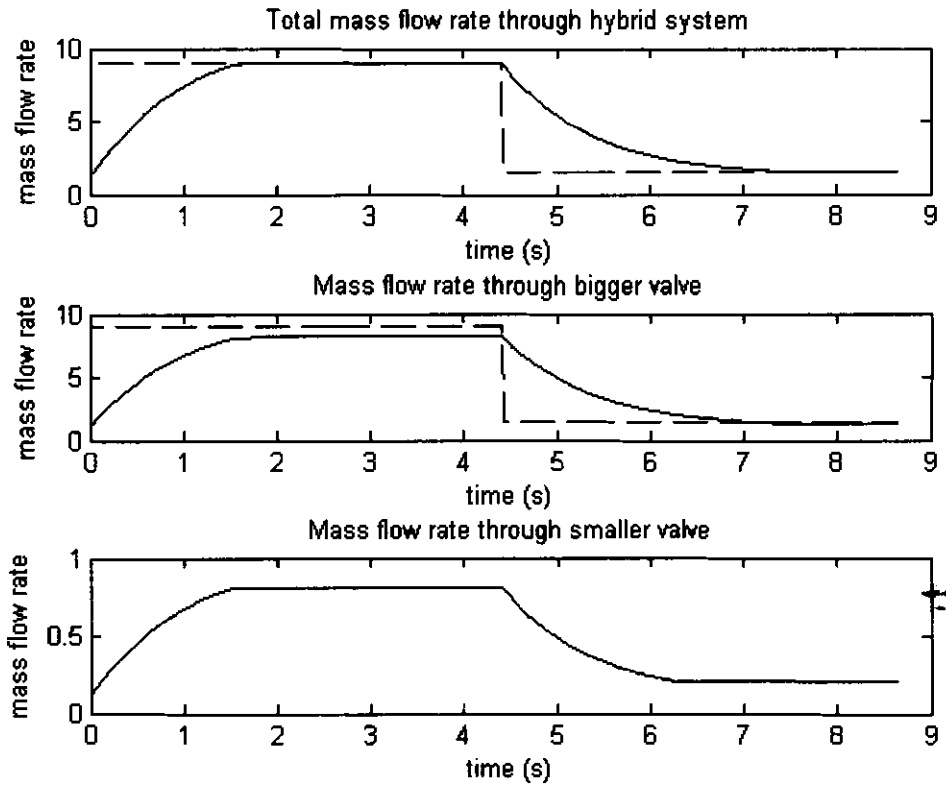

Figure 4.15 Simultaneous closing of both valves to save time

4.3.5. Input section 4

The addition of an extra valve does not only allow mass flow control below the minimum capability of the large valve, but also extends its range a bit, to mass flow rates above its usual capability. In this case, the range of the hybrid valve is roughly from $0.1 \cdot \dot{m}(\text{large})_{\min}$ to $11 \cdot \dot{m}(\text{large})_{\min}$. This means that its usual range from $1 \cdot \dot{m}(\text{large})_{\min}$ to $10 \cdot \dot{m}(\text{large})_{\min}$ was expanded.

Mass flow requests above $10 \cdot \dot{m}(\text{large})_{\min}$ is dealt with almost identically to mass flow requests below $1 \cdot \dot{m}(\text{large})_{\min}$, the only difference being that the large valve is opened completely. In other words, all command is handed to the PID controller of the *small* valve. Figure 4.16 shows what happens if a step request is received from $0 \cdot \dot{m}(\text{large})_{\min}$ to $10.5 \cdot \dot{m}(\text{large})_{\min}$.

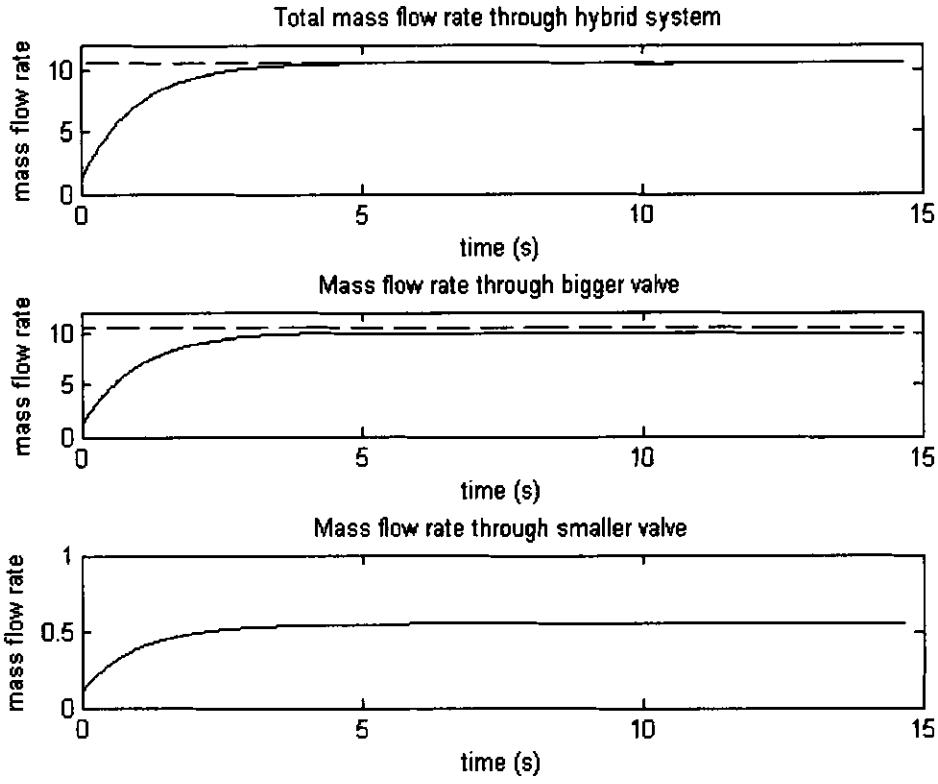

Figure 4.16 Response of the small valve to a step input above $10 \cdot \dot{m}(\text{large})_{\min}$

4.3.6. Input section 5

A step request from above the maximum capability of the small valve to below the minimum capability of the large valve simply requires the large valve to be closed completely. The required mass flow rate can then be achieved by controlling the small valve. However, the large valve takes some time to close completely, and if the small valve remains inactive while the large valve closes, a lot of time is wasted.

Therefore, the controller makes a rough calculation as to the amount that the small valve will have to close or open beforehand. It then simultaneously closes the large valve, and controls the small valve to the calculated valve travel. After the large valve closes completely, the small valve's PID controller takes over, and guides the small valve until the correct mass flow is reached.

Figure 4.17 shows the process of a step request from $3 \cdot \dot{m}(\text{large})_{\min}$ down to $0.5 \cdot \dot{m}(\text{large})_{\min}$

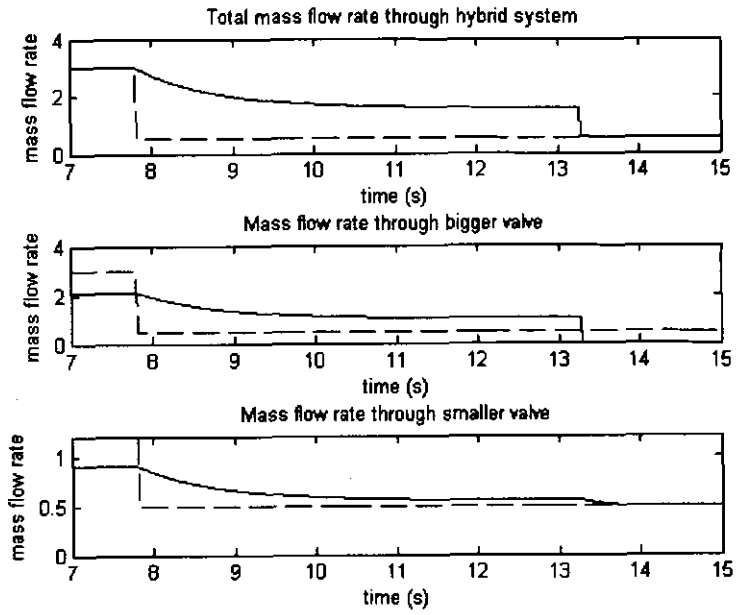

Figure 4.17 Response of both valves to a step input from $3 \cdot \dot{m}(\text{large})_{\min}$ down to $0.5 \cdot \dot{m}(\text{large})_{\min}$

Figure 4.18 shows how the small valve settles on the roughly calculated valve travel and waits for the large valve to close (at t = 13.3 s) before handing over to PID control.

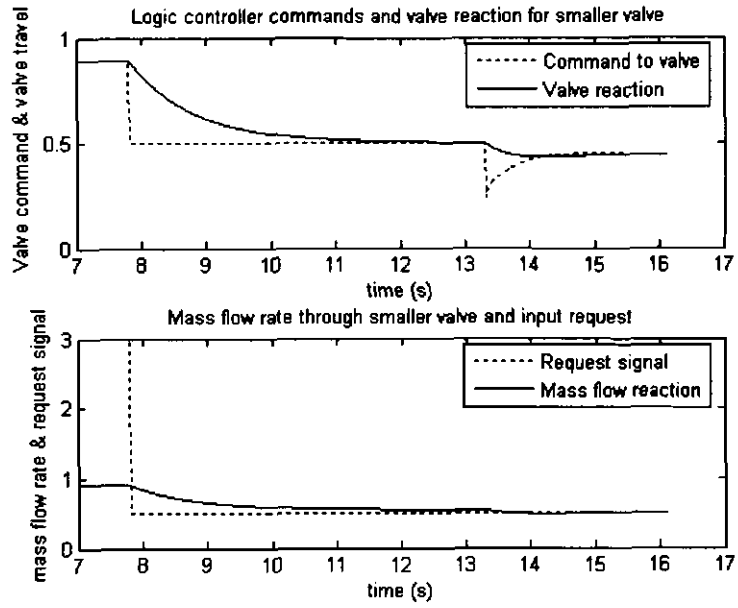

Figure 4.18 Reaction of the small valve for a jump from $3 \cdot \dot{m}(\text{large})_{\min}$ down to $0.5 \cdot \dot{m}(\text{large})_{\min}$

4.4. Complex request signals

4.4.1. Motivation

It was mentioned that the PID based controller that was discussed in this chapter served a very important purpose in this project. The reason for this is that the controller provided an excellent comprehension of some of the challenges that the cooperative control of the two valves would pose.

The limitations that were put on the controller's input signals, however, are not at all practical. Control valves usually operate as part of a much larger control structure, and are subsequently subjected to control inputs from a global controller that may vary continuously. Some of the possible difficulties that the PID controller might encounter because of this will now be discussed.

4.4.2. Gaussian noise

From Figure 4.1, it can be seen that the hybrid valve controller compares the required mass flow rate input to the current measurement of mass flow rate through the system to calculate the mass flow rate error at each interval of time. However, in practical implementations, the accuracy of mass flow rate measurements are currently not as accurate as might be optimal. Therefore, the addition of Gaussian white noise to the error signal seems to be a necessity if the controller is to be effective. Figure 4.19 shows the effect this has on the output of the PID controller.

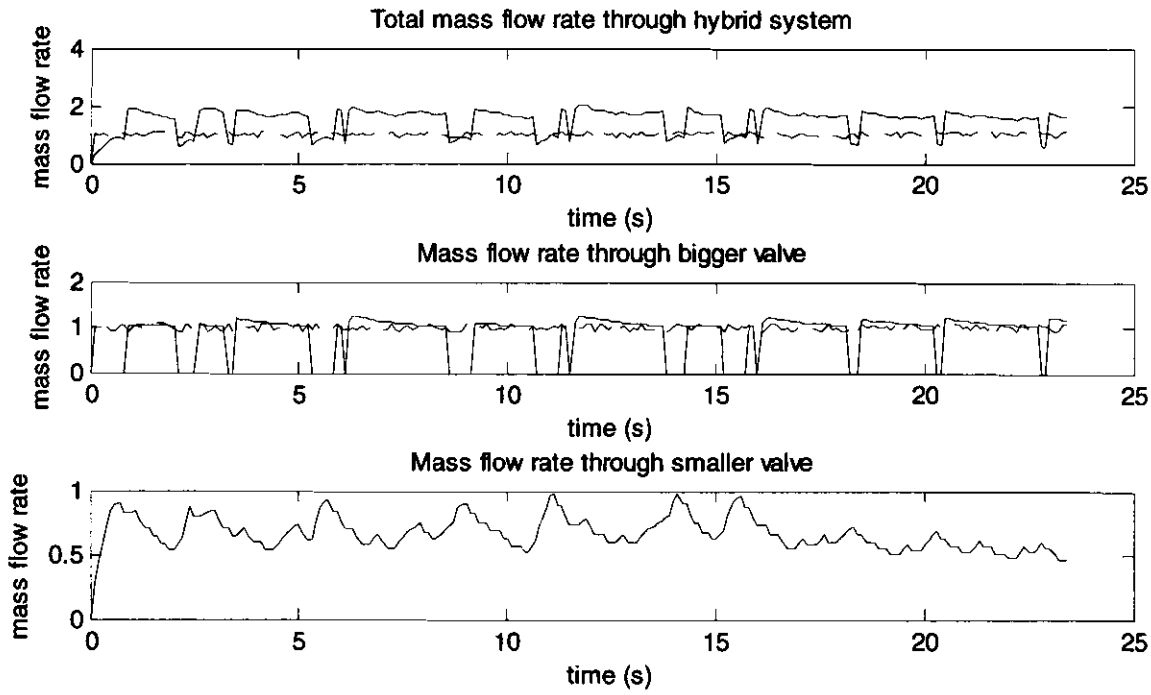

Figure 4.19 Effect of Gaussian white noise to the input on the PID controller

The request signal of Figure 4.19 is a noisy step input at $1 \cdot \dot{m}(\text{large})_{\min}$, which is, of course, the maximum mass flow of the smaller valve, as well as the minimum mass flow of the larger valve. As can be seen, the PID controller is not at all equipped to deal with such a noisy input. The larger valve frequently opens and closes, which produces large jumps in the mass flow that is not optimal at all.

4.4.3. Oscillation between the valves' operational ranges.

Another possible difficulty for the PID controller is continuous, but gradual changes in the request signal that varies across the boundaries of the two valves. This can be illustrated well by a sinusoidal wave with a dc-value close to $1 \cdot \dot{m}(\text{large})_{\min}$, like the one shown in Figure 4.20.

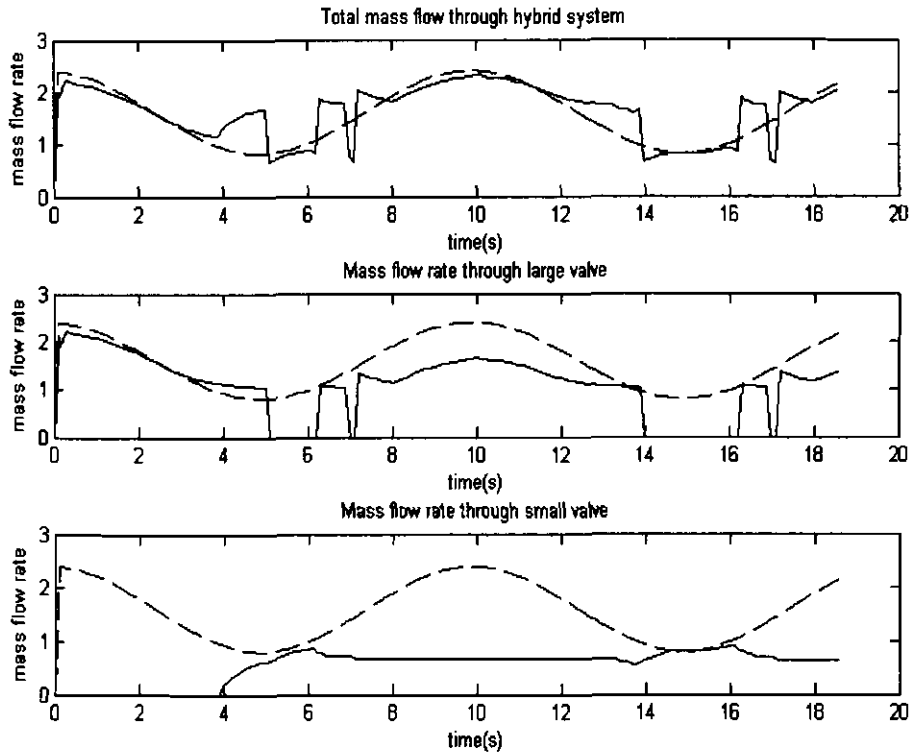

Figure 4.20 A sinusoidal input request that varies over the boundaries of the two valves

From the figure it can be seen that the PID controller does not deal with such an input well. The large valve continuously opens and closes, and the total mass flow rate almost never reaches the requirement.

4.5. GUI implementation

It has already been mentioned that graphical user interfaces (GUIs) may help to simplify the design of the hybrid valve controller. For the PID controller discussed in this chapter a graphical user interface was therefore developed. The GUI enables the user to obtain a better understanding of the way the valve controller operates, and presents the results in a logical and aesthetically pleasing form. The GUI that was created for the PID controller is shown in Figure 4.21.

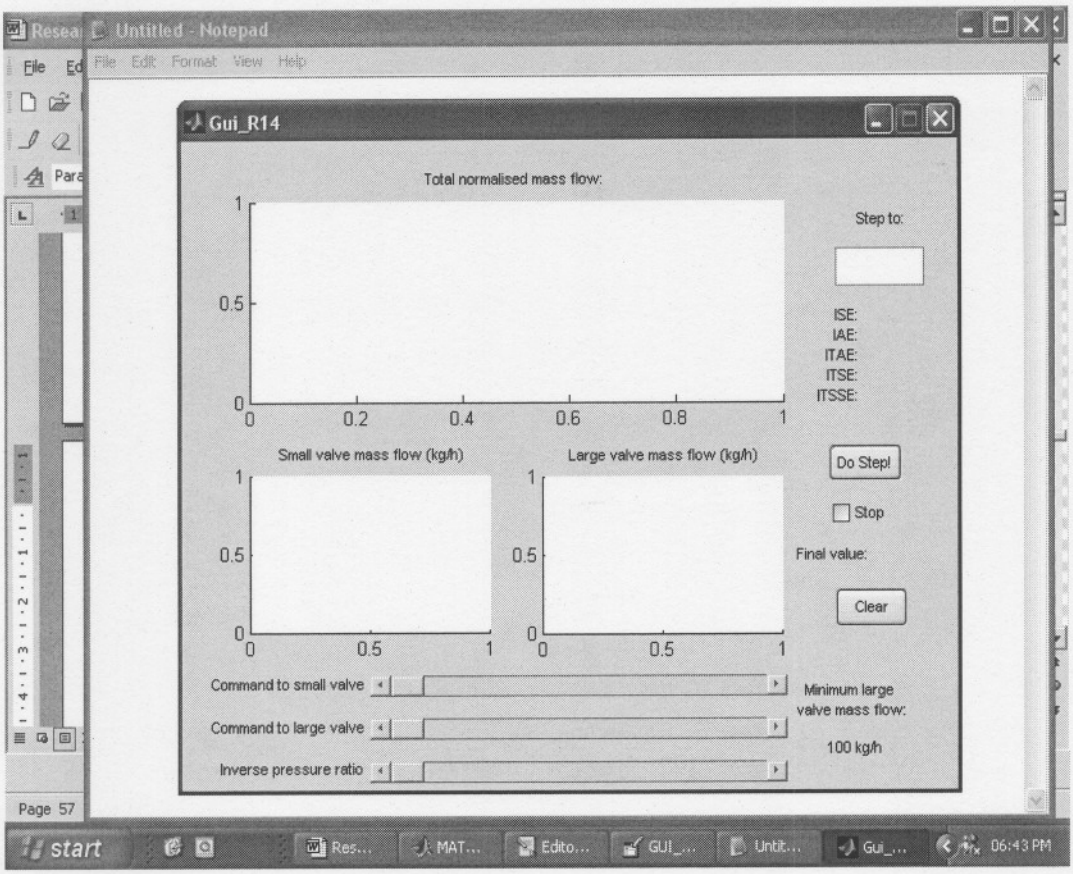

Figure 4.21 Graphical user interface created for the PID controller

The three graphs shown in Figure 4.21 are used to illustrate the reaction of the valve. The bottom left graph shows the mass flow through the smaller valve, the bottom right graph the mass flow through the larger valve, and the graph on the top shows the total, normalised mass flow through the hybrid valve system.

As can be seen, the GUI provides the user with the option of entering a normalised value, and subsequently generates a step request to that specific value. When the button named "Do Step!" is pushed, the GUI provides a "real time" simulation of the controller's reaction to the step request. The

simulation stops when a total mass flow rate sufficiently close to the target value has been reached. For instance, a step request to a normalised value of $5\,\dot{m}(\text{large})_{\min}$ will be illustrated as in Figure 4.22

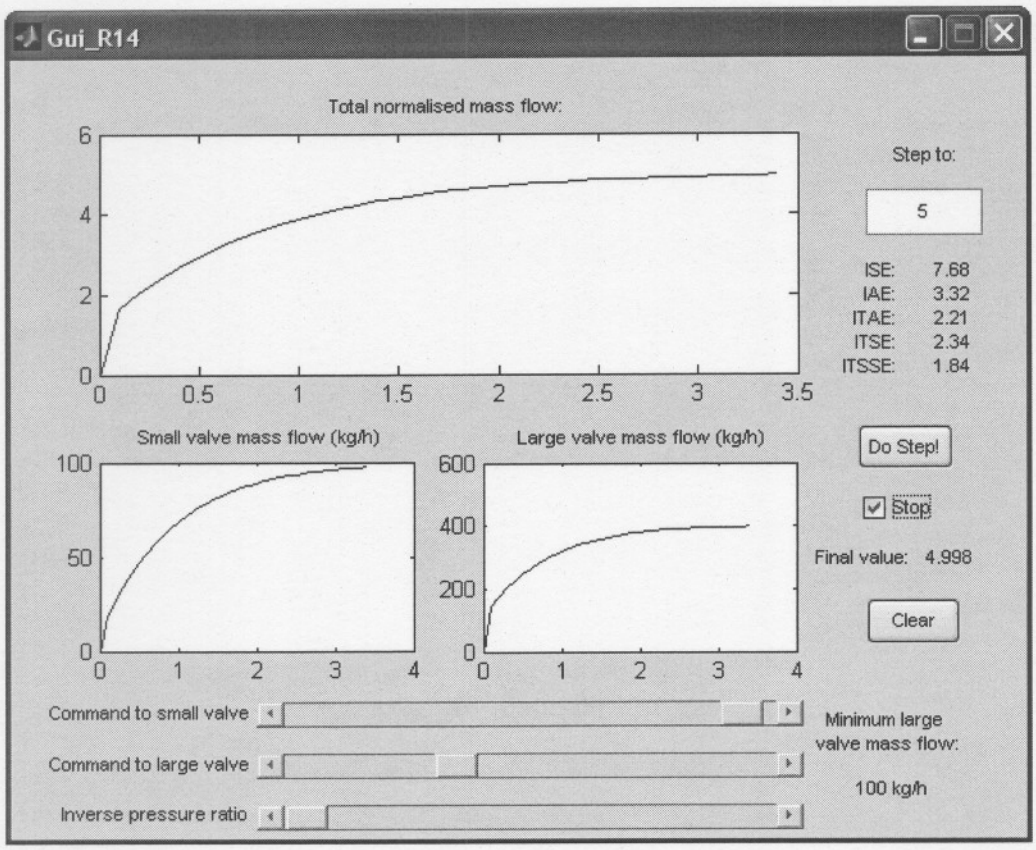

Figure 4.22 The GUI's illustration of the hybrid valve controller's reaction to a step input to $5\,\dot{m}(\text{large})_{\min}$

From the figure it can be seen that the values of some popular performance indices are calculated after each completed step. The indices are, from the top ISE (integral of the square of the error), IAE (Integral of the absolute value of the error), ITAE (Integral of the absolute value of the error multiplied with time), ITSE (integral of the square of the error multiplied with time), ITSSE (integral of the square of the error multiplied by the squared value of the time). Performance indices are very important design tools available to engineers. It enables the designer to quantitatively evaluate the performance of different system parameters – and subsequently choose the optimal combination.

The first two graphic sliders at the bottom of the GUI are used to illustrate the commands issued to each valve at each moment in time during the simulation. The user can therefore simultaneously obtain a good understanding of the operation of the controller, as well as the response of the valves.

The user may make use of the third graphic slider to adjust the inverse pressure ratio across the valves, and therefore the actual mass flow rates through each of them.

The m-file containing the GUI code, as well as an executable GUI is available on the CD provided with this dissertation.

4.6. Conclusion

It is seen that the PID controller, although in itself being quite complex, does not meet the requirements for this project since it is not capable of dealing with advanced request signals. For that reason it is concluded that a more complex controller may have to be implemented. Such a controller is discussed in the next chapter.

5
Complex, non-linear valve controller

5.1. Overview and motivation

In the previous chapter, the implementation of a crisp, PID based hybrid valve controller for relatively simple inputs was discussed. The controller revealed many of the challenges posed by this project, but it was seen that the implementation of *practical* control signals will not be dealt with well by such a controller. It was subsequently decided to implement more complex control.

In Chapter 2, non-linear, Fuzzy Logic (FL) was discussed as control technique. Because the linguistic "IF-THEN" rules (on which the fuzzy logic controller is based) simplifies the design, the complexity of the controller can be increased significantly without complicating the design. It was therefore decided to implement this technique for the control of the hybrid control valve.

Chapter 5 will discuss the operation and design of the non-linear, Fuzzy Logic hybrid valve controller that was developed for practical implementation. However, some of the design-decisions and additional accessories of the controller will first be discussed.

5.1.1. Input signal limitations

In Chapter 4 the input signals to the PID controller was limited to step inputs with a relatively low frequency of change. However, it was eventually seen that the limitations that were put on the input signals ultimately limited the controller to purely theoretical applications.

The goal of the Fuzzy Logic controller that will be discussed in this chapter is to be of *practical* significance. Therefore, the controller will be designed in such a way that the allowed input space will not be limited. As will be explained later, the controller will rather make use of techniques like filtering to ensure a controllable input signal.

5.1.2. Overshoot and "undershoot" in practical implementations

In section 1.1.5 it is mentioned that this project may be applied to the Pebble Bed Modular Reactor (PBMR). Frequent contact with the PBMR was therefore of great significance, and much of the requirements for the controller were obtained from such meetings.

One of the requirements that were clear is that mass flow dips to below 2 % of the required mass flow are not acceptable. This means that the controller must ensure that the opening and closing of the valves rather cause overshoot than causing the actual mass flow rate to dip below the requirement.

The controller discussed here is therefore designed to ensure that no dips in mass flow ever occur to below 2 % of the requirement.

5.1.3. The need for prediction

In Chapter 3 it was seen that the valve model that had been used in this study takes into account the time constant associated with opening the valve. This implies that a certain amount of time is needed for the valve to reach a required mass flow, depending on the amount the valve has to be opened in order to reach it.

Because of this property of the valve, it would be extremely handy to have some way of predicting whether the mass flow request will necessitate a valve to be opened or closed in the near future. This will give the controller time to react in such a way as to prevent the valve's non-linearity to cause, for instance, a mass flow dip of more than 2 % below the request. It would also aid tremendously in optimising settling times and minimising total error.

It was therefore decided to include some form of prediction into the valve controller. The kind of prediction and the detail behind its operation will be discussed in the sections to follow

5.1.4. Valve overlap

Figure 3.1 illustrates the valve model that was created. From the block diagram, it is observable that the maximum flow coefficients of each individual valve can be adjusted. For the PID controller discussed in Chapter 4, the maximum flow coefficients were chosen in such a way that the maximum mass flow of the smaller valve is approximately equal to the minimum mass flow of the larger valve.

In order to make the best use of the added complexity of the Fuzzy Logic controller, it was decided to allow the two valves' operational ranges to overlap. This provides the controller with more alternatives to avoid frequently having to open or close the valves to reach the mass flow requirement. This will subsequently minimise the amount of jumps and dips caused by the valves' non-linearity.

Therefore, for this controller, Figure 1.3 can be redrawn as in Figure 5.1

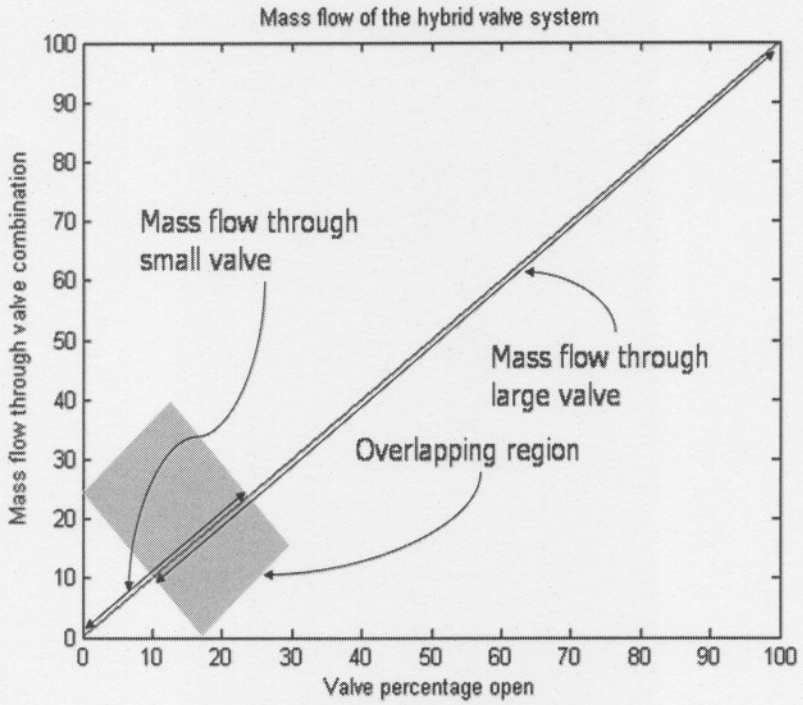

Figure 5.1 Overlap of the hybrid valve

5.2. Predicting request signal behaviour

5.2.1. Filtering of input request signal

5.2.1.1. Effect of filtering on valve response

As mentioned, the aim of this controller is to provide stable control for any possible input signal. This means that even pure white noise as input must not cause the hybrid valve to react in any undesired way. There is, however, a limitation on the reaction time of the valves. Because of the time needed to open the valve, it is impossible to meet the demands of request signals above a certain frequency.

In Figure 5.2 a high frequency, random input is given to a single control valve. From the valve's response, it can be seen that it acts almost as a low pass filter, filtering out the high frequencies, and reacting only to the lower frequencies.

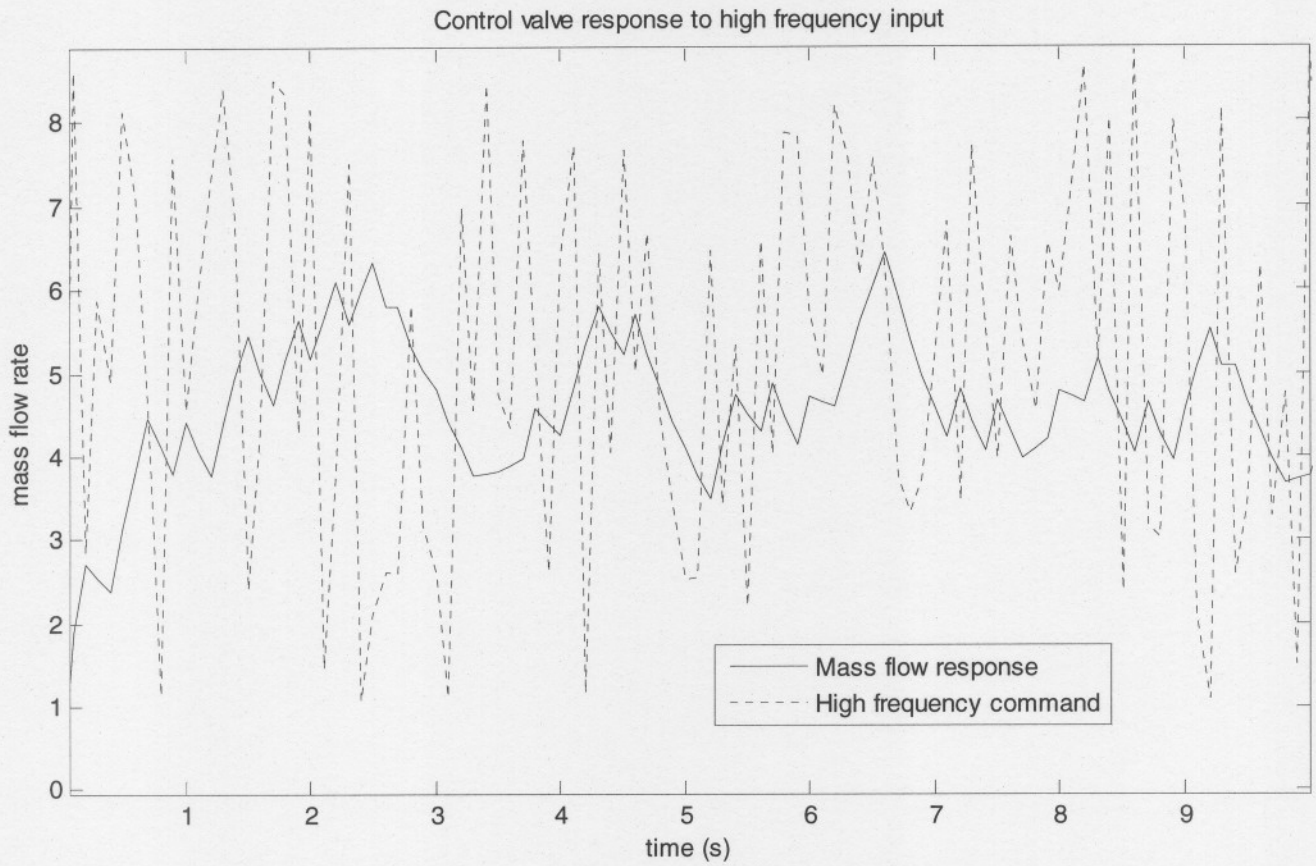

Figure 5.2 Control valve response to high frequency inputs

It can therefore be derived that there exist parameters for a low pass filter that can be applied to the input signal, without affecting the reaction of the valve whatsoever.

These parameters will change for different valve sizes and for different levels of overpowering. Therefore, a new filter will have to be designed each time one of these parameters changes.

5.2.1.2. Advantages of filtering the request signal

Now that it has been established that it is feasible to filter the input to the hybrid valve, the main reason for applying filtering will be discussed.

In section 5.1.3 it was pointed out that prediction can enhance the performance of the hybrid valve, and help it to avoid unnecessary jumps or dips in the total mass flow. Prediction, however, is not possible if the frequency content of the input signal is not limited.

For example, the request signal shown in Figure 5.2 has high frequency content. As can be seen, there is no way to predict what the next value of the request might be. However, if the input was filtered with a low pass filter, only slight deviations about the mean (5 in this case) will need to be predicted, which is much easier to do. Furthermore, since the control valve cannot react nearly as fast as is expected from the request, no significant difference will be visible on the output.

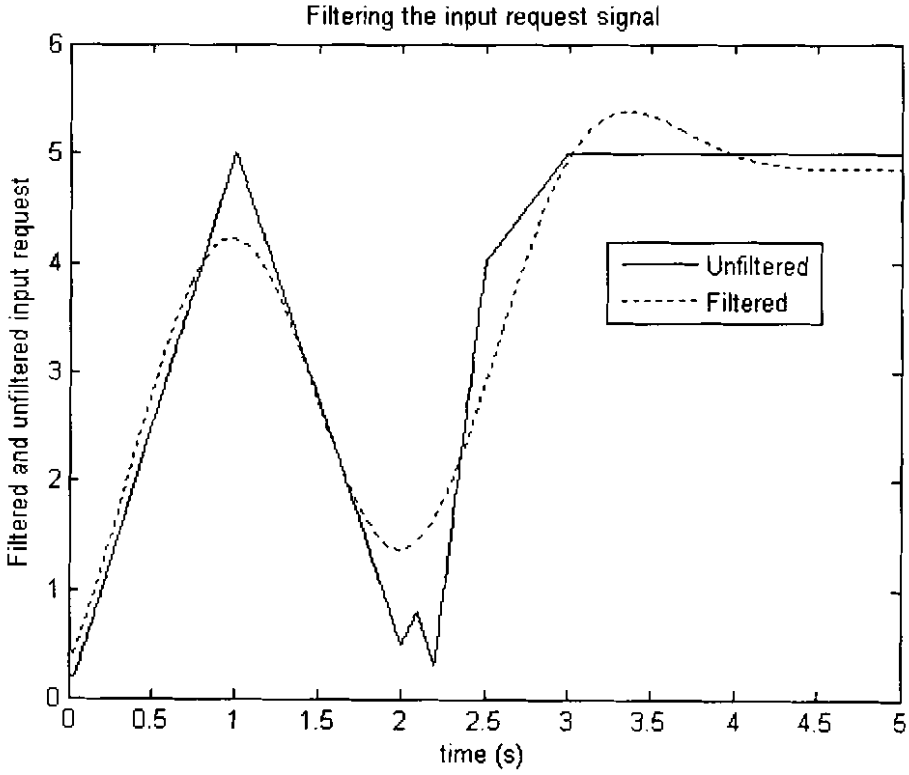

Figure 5.3 Effect of filtering the input signal

Figure 5.3 illustrates the advantages of filtering the input excellently. It is assumed that the filter is an ideal, zero phase shift filter, and that its parameters were chosen in such a way that the output of the valve will not be affected. It is further assumed that the minimum mass flow of the larger valve is equal to 1 in the graph.

As can be seen, the *unfiltered* input request signal crosses the minimum mass flow of the larger valve at approximately t = 1.9 s and re-crosses it again at about t = 2.3 s. If the request signal was not filtered, the controller would have given the command to close the larger valve at about t = 2 s. However, it would soon have been forced to open the larger valve again in order to reach the higher mass flow requests that followed. This process would have caused two unnecessary jumps in the mass flow that could easily have been avoided. In contrast, the *filtered* input never even crosses the minimum mass flow boundary of the larger valve, hence avoiding both jumps without any effort.

5.2.1.3. Parameters for prediction

It was found that the chances for a low frequency signal to cross a certain value can be predicted quite successfully by monitoring the following parameters:

- The distance from the value
- The first derivative of the signal
- The second derivative of the signal

In order to explain why these parameters are needed and how it is used, let the part of Figure 5.3 where the filtered input almost crosses the minimum mass flow of the smaller valve be enlarged. This is shown in Figure 5.4

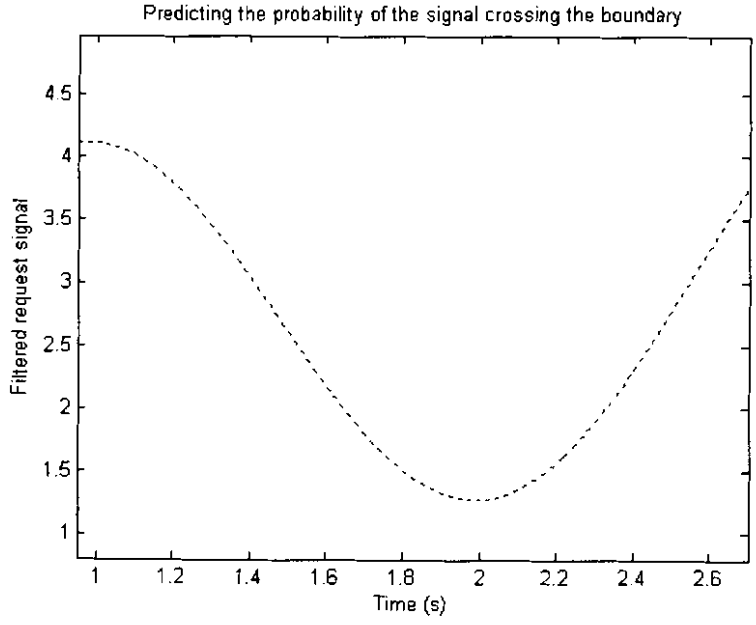

Figure 5.4 Signal approaching the boundary of 1

From close inspection of the signal shown in the figure, it is seen that it starts to experience an upward acceleration (second derivative) at about t = 1.6 s. So let it be assumed that the prediction has to be whether the signal will cross the boundary of $1 \cdot \dot{m}(\text{large})_{\min}$. What will be taken into account?

It can be seen that the signal is moving *downward* at that particular point in time (first derivative) which makes crossing the boundary a possibility. It can further be seen that the signal is getting quite close to the boundary (distance from the boundary) which implies that the possibility of crossing is increasing.

The question therefore arises: Is the acceleration large enough to ensure that the signal starts moving upward before crossing $1 \cdot \dot{m}(\text{large})_{\min}$?

If it is known that the signal's frequency content is limited, it can be safely assumed that the acceleration of that signal will not be able to increase indefinitely, or change the signal's direction very suddenly - which enables the predictor to make a reasonably accurate prediction.

Such a prediction is, however, not an uncomplicated task. If a signal differs only slightly, for instance, in its rate of acceleration, its reaction may be much different. For that reason it was decided to make use of the adaptive recognition techniques discussed in Chapter 2 to aid in predicting the signal behaviour. This will be discussed in the next few sections.

5.2.1.4. Filter requirements and limitations

Finding the correct filter for the hybrid valve controller is a task that is based on many different considerations.

The filter has to be chosen in such a way that the input signal is not affected too much - which will, of course, have an undesirable effect on the output. However, the filter must remove as much of the higher frequencies as can be afforded so that prediction can be done as accurately as possible. The accuracy of prediction obviously increases as the frequency content of the signal decreases.

Moreover, overpowering of the control valve increases the frequency content it can deal with in the input request signal. Therefore the limits of overpowering must first be established before making any decisions about the filter.

5.2.1.5. Intuitive choice of filter and overpowering parameters

In order to obtain the fastest possible response from the valves, it was decided to allow the valves' commands to be overpowered by a maximum of 90 %. This means that, if the valve requires a normalised command of 1 in order to open completely, commands up to 1.9 may be allowed to decrease its reaction time. The constant may be adjusted for different applications, but, as mentioned, that would necessitate subsequent adjustments to the filters and prediction parameters as well.

The actual choices for the filter and filter parameters in this project were made by the use of Genetic Algorithms. This will be discussed in section 6.2. However, initial investigation showed that frequencies above 2 Hz become difficult to deal with for the control valves that are overpowered by 90 %. Therefore it was decided to design a filter a pass band of with at least 2Hz. The initial filter parameters were chosen purely with the intent of testing the optimisation results. The results should be tested since the choice of objective function for the Genetic Algorithm may be flawed, and intuitive choices are usually not far from optimal.

The intuitive choice of filter was a second order Butterworth Filter with a pass band frequency of 2 Hz and a stop band frequency of 10 Hz. The maximum pass band attenuation was designed to be 2 dB and the minimum stop band attenuation was designed to be 10 dB.

5.2.2. Generating training- and testing-data

In many cases where Artificial Intelligence is used for system identification, one of the main problems is the availability of training data [15]. Training data should cover the whole range of possible inputs and subsequent reactions of the system that is to be identified [14].

Fortunately, however, in this case the training data doesn't have to be sampled from some expensive plant in operation, but can be generated mathematically. As was explained, the only limitation put on the request signals is that they have to be limited to some maximum frequency.

Sinusoidal waves therefore provide an excellent way of generating training data. Since many signals can be built up from a collection of sinusoidal waves (or parts thereof), it is a reasonable assumption that the networks can be trained using sinusoidal waves of which the amplitude, dc-offset and frequency are chosen at random.

The amplitudes and dc offsets of the sinusoidal signals will, of course, be limited by the range of the control valve, while the frequency will be limited by the filter that is implemented. To increase accuracy, the networks will only be trained for highly specialised functions, and the required predictions will rather be divided into several different networks, than training one network to predict the outcome for a number of different situations.

More detail on the training data used for each network will be given in section 5.3 where the created networks will be discussed thoroughly.

5.2.3. Deciding on applied prediction technique

Chapter 2 discussed only a small selection of the multitude of prediction and recognition techniques that are available. Each of these techniques has its own advantages and disadvantages, and it is subsequently necessary to find the technique that will provide the best results for each application.

Because of the complexity of the prediction required for the valve controller, it was decided to consider two techniques for this project. These are:

- The Multilayer Perceptron Neural Network
- The Adaptive Fuzzy Logic Network

These networks were considered because of their widely recognised ability to identify highly complex patterns. Both of these techniques are discussed in Chapter 2.

After deciding on the two techniques mentioned above, both techniques had to be tested for this project's particular application to find which one is most suited. The techniques were trained by a

training set which has the three inputs mentioned in section 5.2.1.3, and one output. The output of the training data can be either a 0 or a 1, depending on whether the value is crossed or not.

Figure 5.5 shows the training progress that was made by using the Multi Layer Perceptron Neural Network.

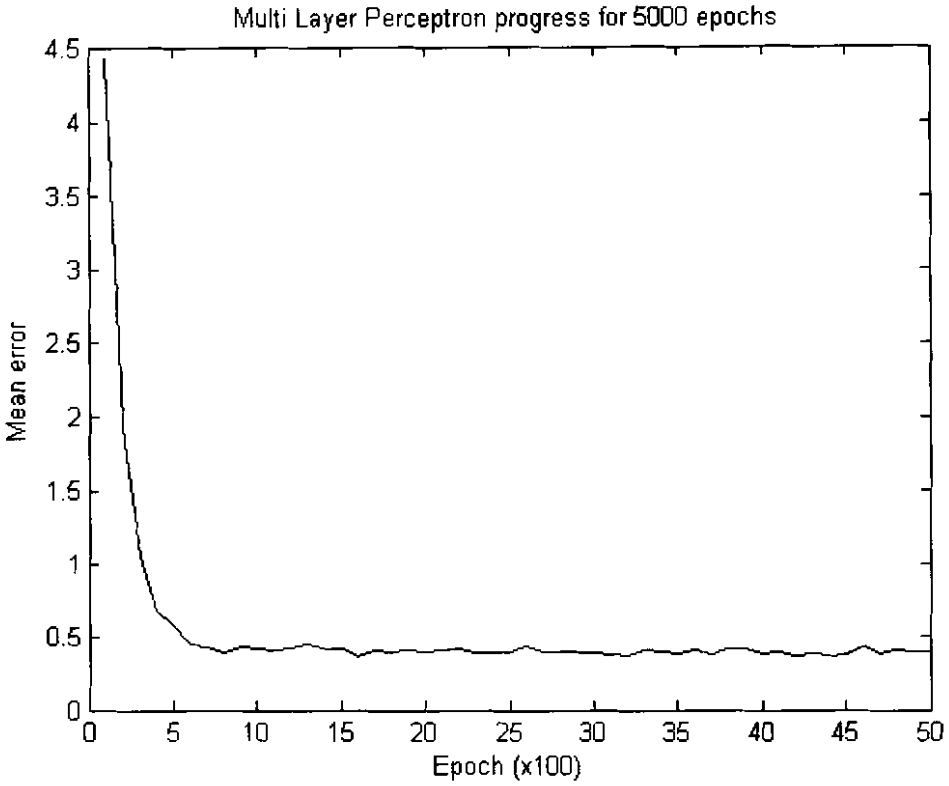

Figure 5.5 Training progress of the Multi Layer Perceptron Neural Network

As can be seen, the mean error is quite large at first, but gradually descends to a value below 0.5. This means that on average the signals were classified correctly. Although the margin for misclassification seems to be very small, it has to be taken into account that the mean error is evaluated over the whole input signal, and accurate prediction is not always simple for all the data points.

To explain this, refer to Figure 5.6. As can be seen, the network is at first not sure whether the value of 1 will be crossed, but as the signal draws near the value, the network becomes more confident. In the graph of Figure 5.5, the mean error is calculated for all the data points, which explains why the mean error is so close to 0.5.

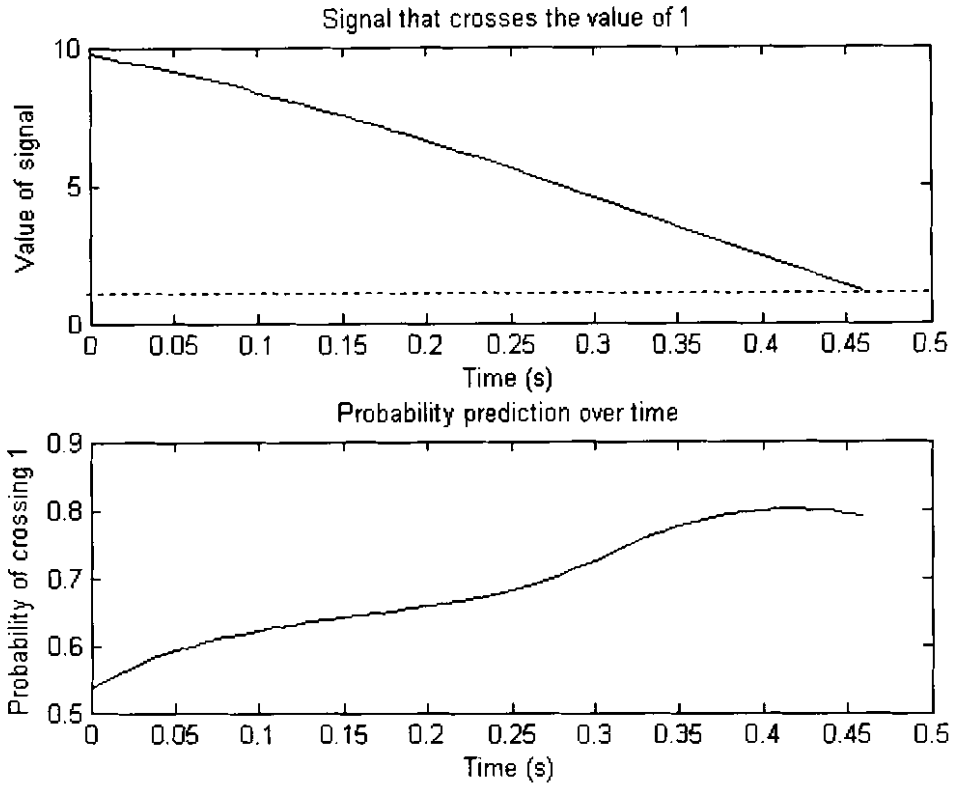

Figure 5.6 Prediction for the probability of a signal crossing the value of 1

In evaluating which method to use, the following was considered:

- Time required for training
- Ability to converge to best solution

Figure 5.7 shows the progress made with training the Adaptive Fuzzy Logic network. As can be seen, the linguistic association of the network makes it possible to initialise the parameters in such a way that the outputs are immediately in the required range. This means that the network can converge faster to a better solution. The time required to train 5000 different signals is also much less. The Multi Layer Perceptron Network required almost three times the amount of time to complete 5000 signals, and did not converge to the same accuracy of prediction.

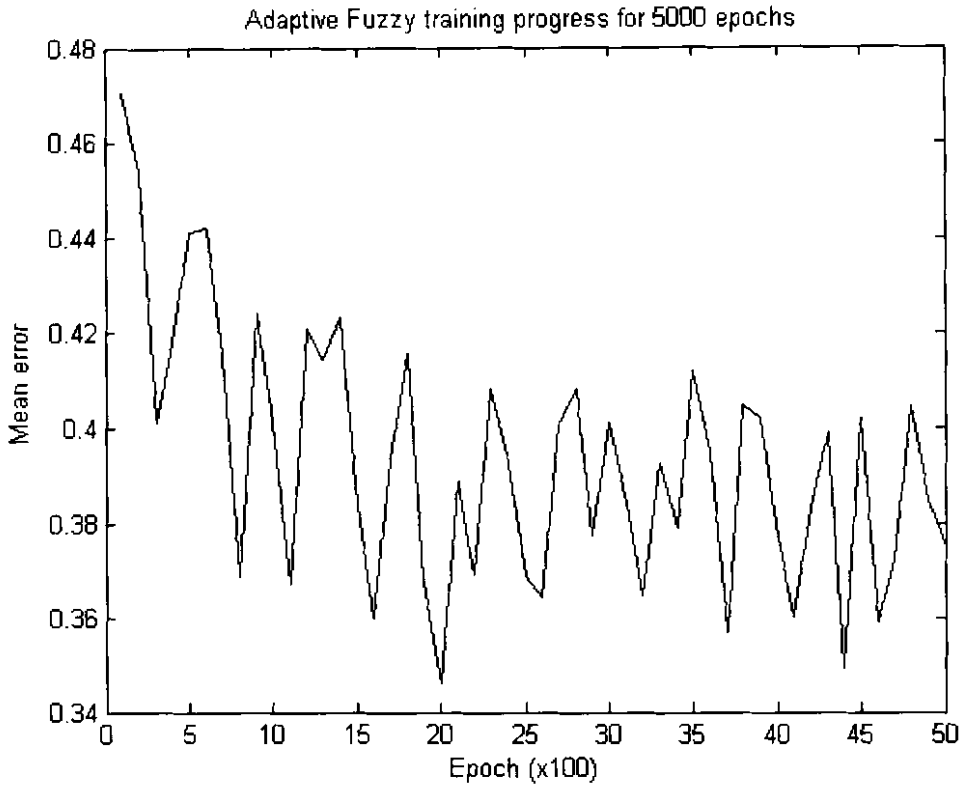

Figure 5.7 Training progress of the Adaptive Fuzzy Logic Network

It can be seen that the mean error signal of the Adaptive Fuzzy network fluctuates much more than that of the Multi Layer Perceptron. This is because the Adaptive Fuzzy network adjusts three constants for each rule, while the Multi Layer Perceptron only adjusts one weight for each layer. Although the Multi Layer Perceptron network is much smoother, the added complexity of the Adaptive Fuzzy Network is preferred.

It was consequently decided to use the Adaptive Fuzzy Logic Network to train the predictors for the Hybrid valve controller.

5.2.4. Method of avoiding over-training

The main advantage of the fact that no limit exists to the amount of training data, is that the danger of over-training the network can be avoided.

Over-training of a network occurs when the network is trained repeatedly on the same data, and the network subsequently starts losing its generalising abilities. This means that the network will be able to predict very well with the data it was trained with, but will not be able to predict well with other data.

The chances of over-training are avoided by generating a *completely new* signal each time the network is trained. This means that the network will never be trained on the same data twice. The result of this is that the network will in time converge to the *most optimal* solution that generalises as good as possible for the chosen complexity of the network.

5.3. Prediction applied in valve controller

The creation of the specific prediction networks will now be discussed

5.3.1. A declining request signal near the minimum operational range of the large valve

5.3.1.1. Motivation

When the request signal declines from above the maximum range of the small valve to below the minimum range of the large valve, it usually implies that the large valve will have to be closed. This, of course, causes a dip in the mass flow.

In order to avoid the dip in mass flow falling below the margin of 2 % below the request, it is imperative that the small valve be used to absorb that dip. Such a process is shown in Figure 5.8.

As can be seen, the small valve opens, so that the total mass flow through the system is far enough above the required mass flow that the dip caused by the large valve's non-linearity does not create any "undershoot".

It can, however, also be seen that the process of closing the larger valve is very costly in terms of overshoot. Therefore, it is sometimes better not to close the large valve at all. For instance, if the request signal declines to $0.85 \cdot \dot{m}(\text{large})_{\min}$, and then starts going up again, it would be much better to close only the *smaller* valve – and not going through the whole process shown in Figure 5.8. Predicting whether the signal would change direction, would therefore be very helpful.

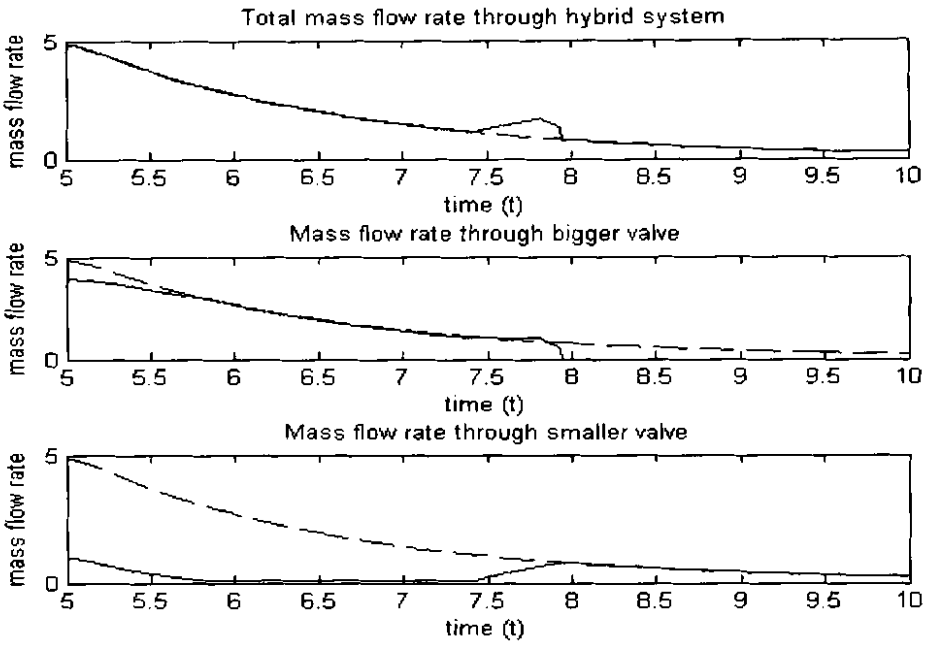

Figure 5.8 Small valve absorbing dip in mass flow to avoid undershoot

Figure 5.9 illustrates how overshoot is minimised by closing the smaller valve, but keeping the larger valve open.

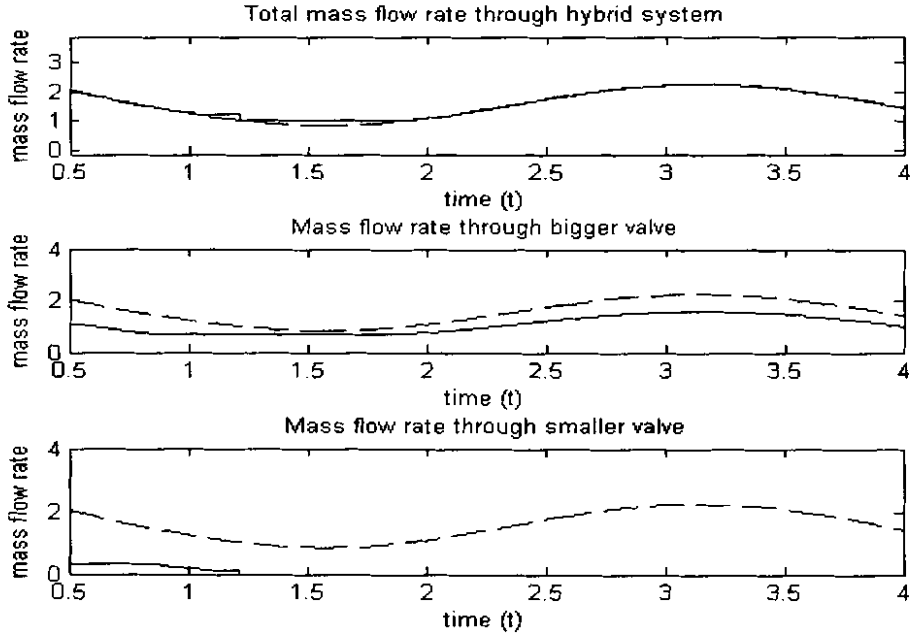

Figure 5.9 Small valve closing to minimise overshoot

5.3.1.2. Training data generation

It has already been mentioned that sinusoidal waves with random frequencies, dc offsets and amplitudes were used for training the predictors. This section will give more details on the data generated specifically for training this particular predictor.

A very important aspect of generating the training data is making the data as useful as possible. For instance, it will serve no purpose to train this network with signals that is *inclining*, since the goal of the predictor is to predict whether *declining* signals will cross a certain boundary. Furthermore, data points that has *already* crossed the boundary, and data points that are above the operational range of the valves are also useless, since they do not fit into the network's range. Let us assume, for illustration purposes, that the network was trained to predict whether the signals will cross the boundary of 1. This can, of course be easily adjusted.

Figure 5.10 shows a few examples of training signals that were used. Solid lines represent signals that will cross the boundary, and the dotted lines represent signals that will turn before reaching it.

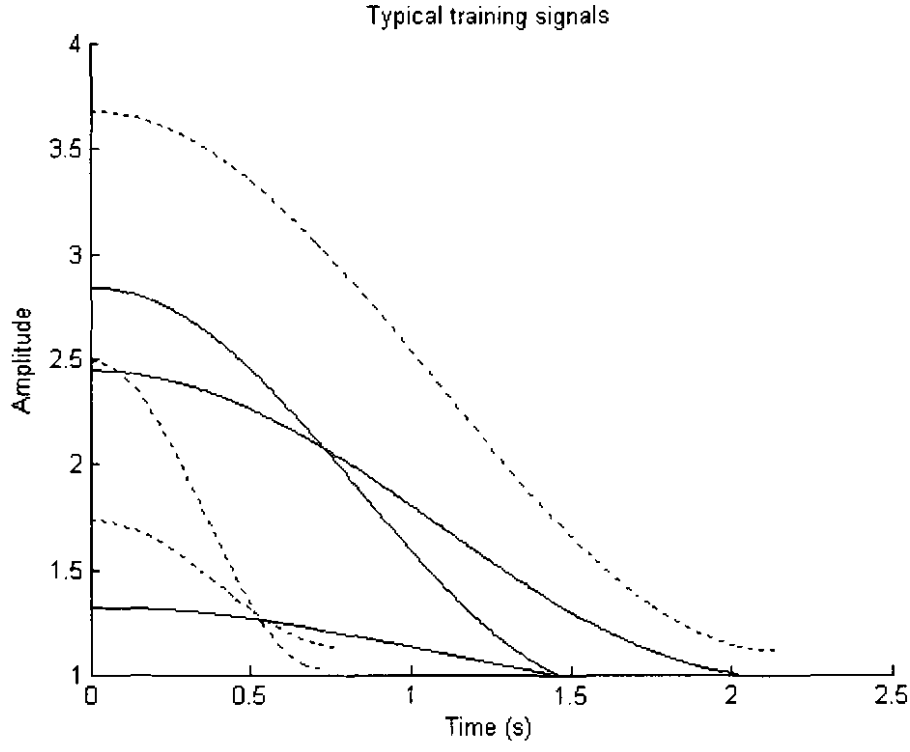

Figure 5.10 Typical training signals for predicting the probability of crossing for declining signals

5.3.1.3. Prediction accuracy – Asymmetric training

After training the network, its accuracy of prediction was tested. The accuracy was, however, not as good as was hoped for. On average, the signal failed to predict the correct outcome about 35 % of the time.

Therefore, a compromise had to be made – it was decided to train the network *asymmetrically*. This is done by focussing the data used for training on a certain behaviour pattern (for instance, signals not crossing the boundary). This causes the network to recognise virtually all signs of such behaviour, and subsequently predicting it whenever there exist an outside chance that it may occur. If the network therefore predicts that it will not occur, no sign exists that contradicts such a prediction, and the prediction can be almost completely trusted.

In this case, it was decided to train the network in such a way that its prediction that the signal *will* cross the boundary can be trusted. This, however, also means that the network may sometimes predict that a signal will not cross the boundary – and then it does.

The impact of such a prediction scheme on the controller is that the controller will rather not start the process of closing the large valve (as illustrated in Figure 5.8) if it is not *definitely* necessary. It may be that the prediction is wrong, and the request signal crosses the boundary, but it was seen that the signal then usually starts going upwards not long after crossing the boundary – and not much harm was consequently done.

5.3.2. Inclining request signal near the maximum operational range of the small valve

5.3.2.1. Motivation

Predicting whether an inclining request signal will cross a value above the maximum operational range of the small valve holds the most value if both overshoot and "undershoot" are allowed for the controller to minimise the total mean error.

For instance, the controller may decide not to open the large valve if the request will turn *back* shortly after crossing the maximum range of the small valve. However, for this project "undershoot" is not allowed – which means that this cannot be done.

What *can* be done, however, is to open the larger valve a bit earlier than is strictly needed, and closing the smaller valve to allow the larger valve more room to manoeuvre. Because the larger valve has a bigger range, it tends to react much quicker to high frequency changes than the smaller valve - so opening it earlier may improve the controller's performance.

Figure 5.11 shows such a case.

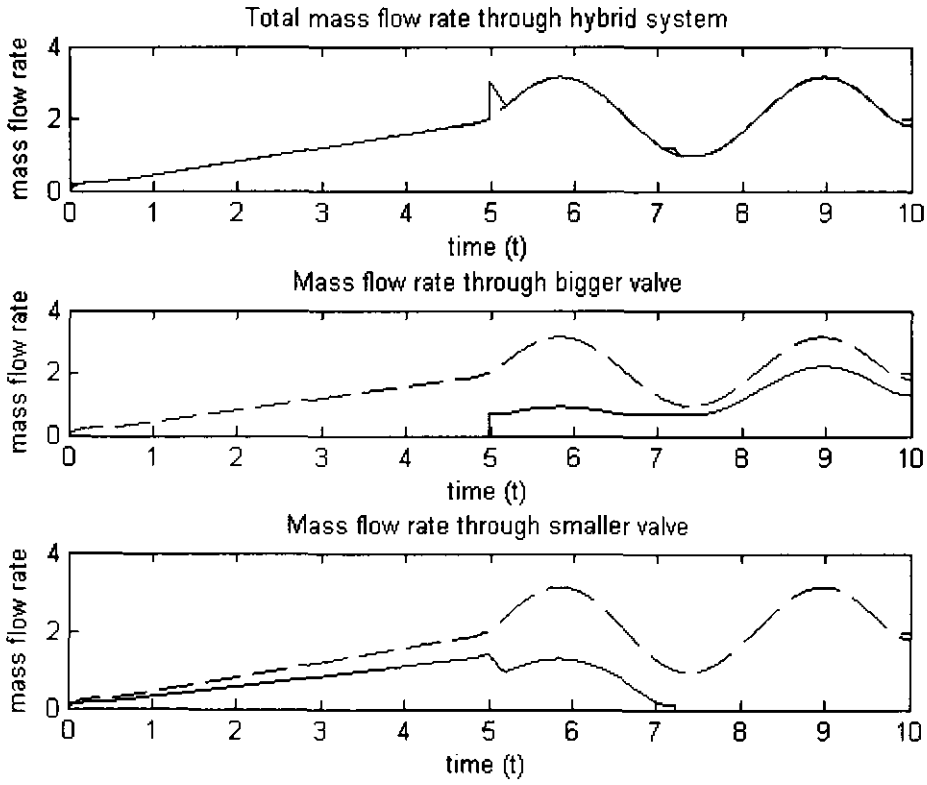

Figure 5.11 Opening the larger valve earlier to improve manoeuvrability

The prediction was therefore included into the controller for improving its performance, but also to make it possible to adjust the controller for optimal control in other applications where "undershoot" is allowed.

5.3.2.2. Training data generation

The training data for this predictor has much resemblance to the predictor discussed in section 0. The main difference being that the signals are all *inclining*. Figure 5.12 shows some examples – when it is

assumed that the network is predicting whether the value of 1 will be crossed. Once again, the solid lines represent signals that cross the boundary and dotted lines represent signals that do not.

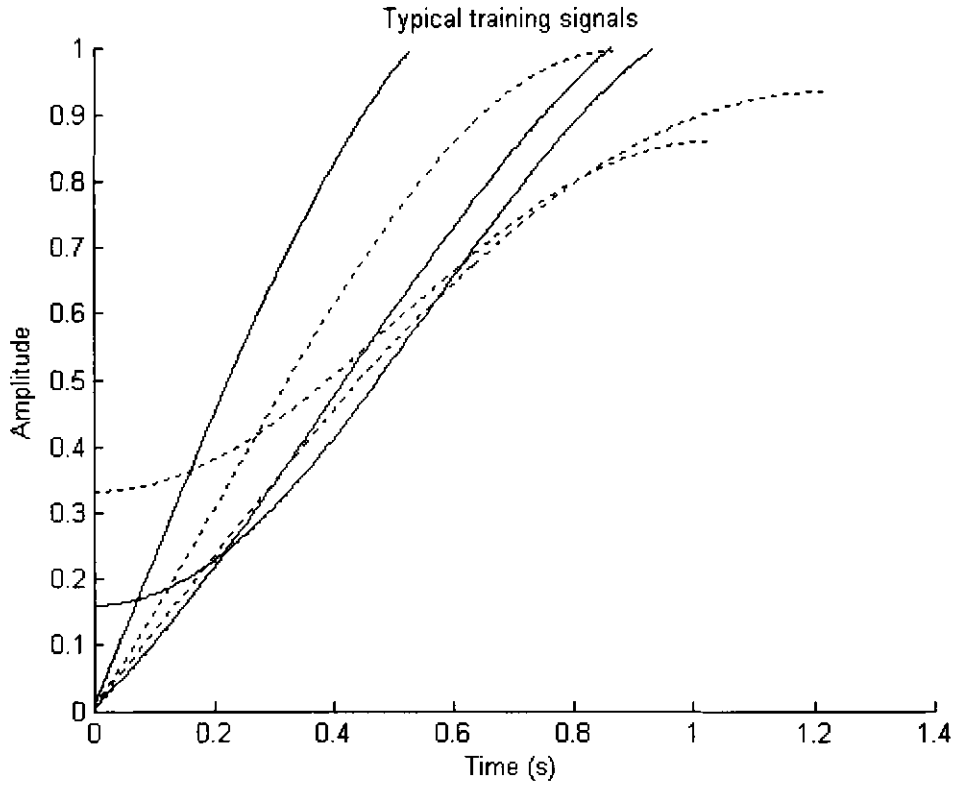

Figure 5.12 Typical training signals for predicting the probability of crossing for inclining signals

5.3.2.3. Prediction accuracy – Asymmetric training

For this predictor it was also decided to make use of asymmetric training in order to improve the credibility of one of the predictions. The predictor can be trusted for a prediction that the signal will cross the upper boundary – which means that the larger valve will not be opened unnecessarily.

5.4. Non-linear, Fuzzy-logic based valve controller

5.4.1. Motivation for Fuzzy control

It has already been mentioned that Fuzzy Logic will be used as design foundation for the more complex hybrid valve controller discussed in this chapter.

The structure of the Fuzzy Logic design makes it possible to combine human expert knowledge with other important inputs - such as the result of a Neural Network Predictor. Furthermore, as was seen

in Chapter 2, it also allows the imitation of other essential design functions, such as PID control and Adaptive Networks, and lends itself excellently to optimisation in the form of Genetic Fuzzy Systems. It comes, therefore, as no surprise that Fuzzy Logic is considered to be extremely suited for the purposes of this project.

5.4.2. Fuzzy controller implementation

In Figure 5.13, the Fuzzy Logic based hybrid valve controller that was developed is illustrated. It is immediately evident that the controller is much more complex than the PID controller discussed in Chapter 4. This is apparent from the fact that the fuzzy inference system uses 14 different inputs to generate the most optimal response. From the figure, the four main elements of the controller are discernible:

- The low pass filter
- The two predictors
- The fuzzy inference system
- The crisp controller

Both the low pass filter and the two predictors have already been discussed in the previous sections of this chapter. This section will focus on the last two elements - the fuzzy inference system that was developed to manage the largest part of the control, and the crisp controller that manages the parts of the system that cannot be fuzzified.

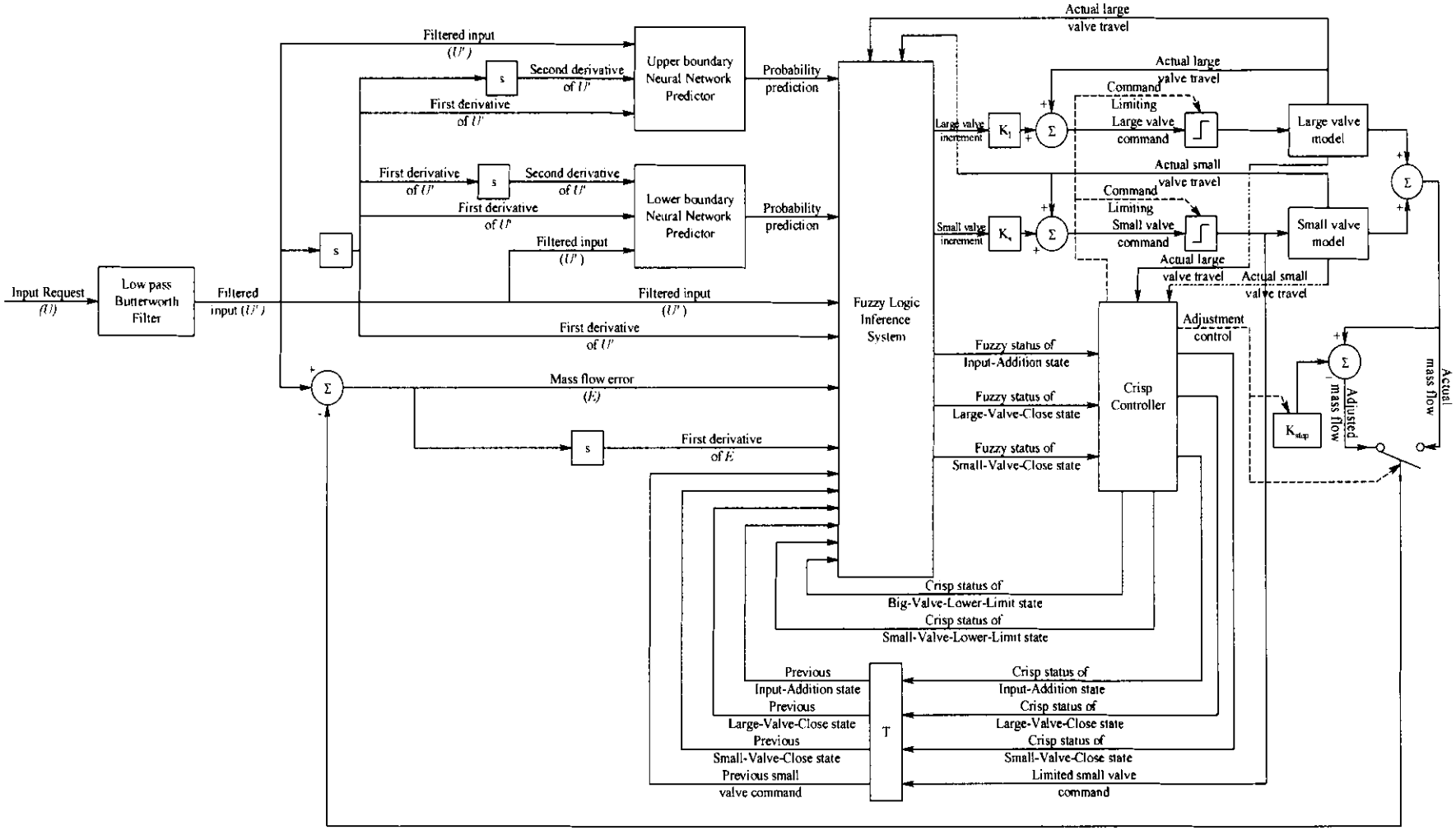

Figure 5.13 The Fuzzy Logic based hybrid valve controller

5.4.3. Fuzzy controller inputs and membership functions

As was discussed in Chapter 2, both fuzzyfication and defuzzyfication relies on the identification of membership functions for each input and output. Each of the 14 inputs to the fuzzy inference system provides it with valuable information, on which it bases its decisions. The membership function arrangement for each input will now be discussed shortly, along with the role the input plays in the operation of the controller.

5.4.3.1. Input 1: The current mass flow error

In Chapter 2 PI Type Fuzzy Process Control was discussed as a possible method of imitating PI control by using fuzzy methods. This method is applied by the fuzzy hybrid valve controller, and therefore the membership functions of the mass flow error input to the fuzzy inference system will closely relate to those discussed in Chapter 2. However, there are two extra membership functions added to this input that are not discussed in Chapter 2. The membership functions of the current mass flow error input are shown in Figure 5.14.

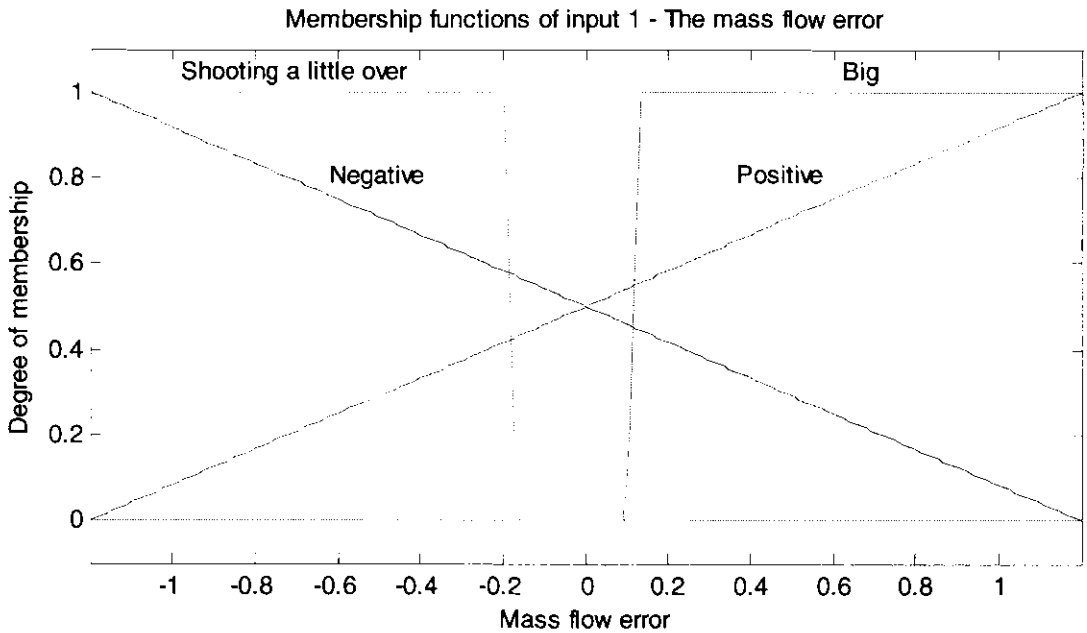

Figure 5.14 Membership functions for input 1 – The current mass flow error

From the figure, it can be seen that the two extra inputs are named "Shooting a little over" and "Big". To explain the role of the membership function named "Shooting a little over", refer to Figure 5.15.

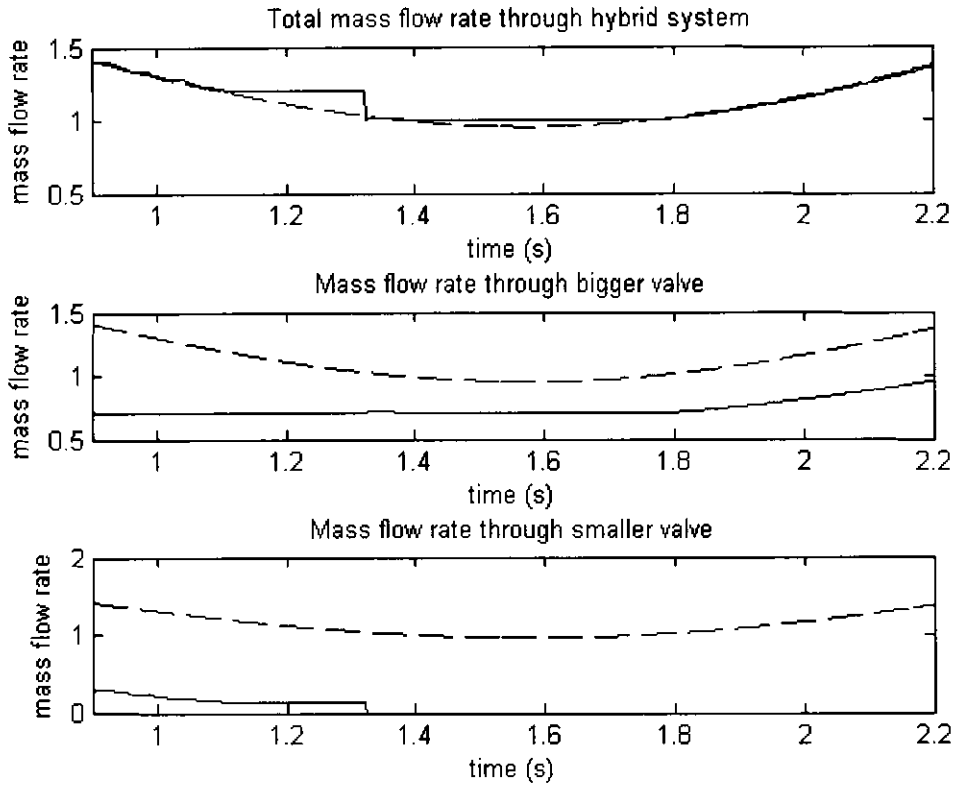

Figure 5.15 The role of the membership function "Shooting a little over"

In the figure, the request signal declines to a value below the minimum capability of the large valve. However, the controller decides *not* to close the large valve, and decides to rather risk the small amount of overshoot. The controller then waits until the membership function "Shooting a little over" is active before closing the smaller valve. This membership function therefore ensures that the dip in mass flow caused by closing the valve does not result in "undershoot" of more than 2 %.

It has already been mentioned that the smaller valve, because of its smaller size, does not have the ability to deal with the same frequencies as the larger valve. However, the low pass filter that is implemented is adjusted to allow the *larger* valve to work to its maximum ability. This means that some input signals may be of a frequency too high to deal with for the smaller valve. The membership function named "Big" ensures that the larger valve is opened if the "undershoot" becomes too big because of this. Figure 5.16 shows such a case.

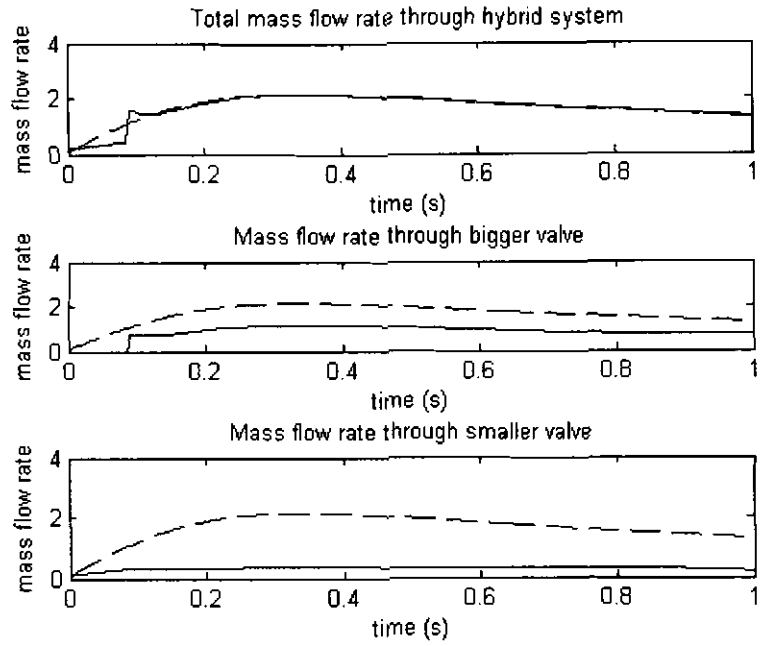

Figure 5.16 The role of the membership function "Big"

5.4.3.2. Input 2: Derivative of the current mass flow error

The derivative of the mass flow error input is used only for the purpose of PI control, as is discussed in Chapter 2. The range was, however, adjusted a bit. The two membership functions of this input are shown in Figure 5.17.

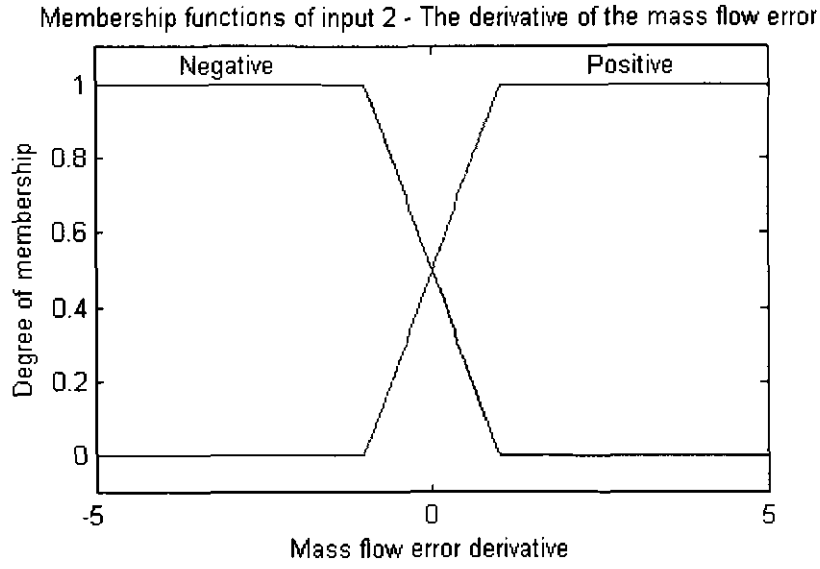

Figure 5.17 Membership functions for input 2 – The current mass flow error derivative

5.4.3.3. Input 3: Current request signal value

Input 3, the value of the request signal, is the most significant input. It is therefore also the input with, by far, the most membership functions. Input 3 has 7 membership functions that provide the controller with the current position of the request signal. The 7 membership functions are so closely arranged however, that it is very difficult to portray it well by drawing it. Figure 5.18 is an attempt to do so.

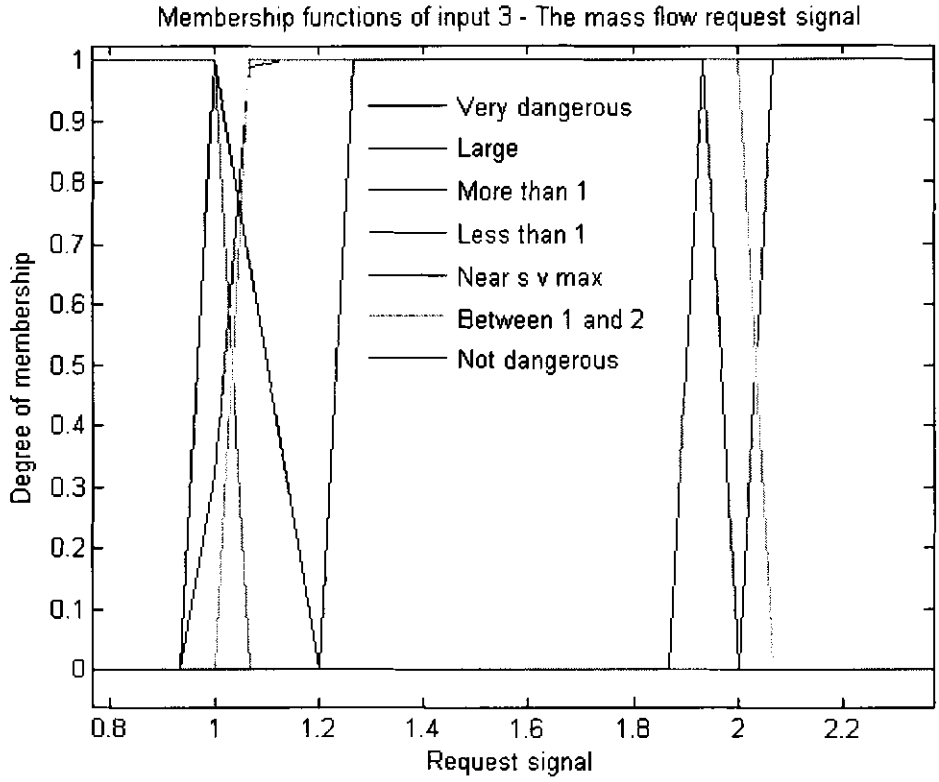

Figure 5.18 Membership functions for input 3 – The request signal

The membership functions named "Less than 1", "More than 1", "Between 1 and 2" and "Large" all fulfil similar purposes. They tell the controller in what prominent region the input request is. For instance, in the region "Large", which is when the mass flow request is more than $2 \cdot \dot{m}(\text{large})_{\min}$, and therefore above the maximum capability of the smaller valve, the controller cannot consider closing the larger valve – because the smaller valve will never be able to reach the required mass flow. However, in the region "Between 1 and 2" both valves are capable of reaching the requested mass flow, and the controller may choose to close either one of them. When the required mass flow is in the region "Less than 1" the controller is usually forced to follow the request by using the smaller valve alone.

The membership functions named "Very dangerous" and "Near s v max" (which stands for "Near small valve max") tells the controller that prediction is starting to become necessary. The controller therefore requests predictions from one of the two predictors when in one of these regions, and uses the subsequent results to decide, for instance, whether to close the larger valve or not.

When the "Not dangerous" membership function is active, it indicates that no immediate decision has to be made about opening or closing the valve.

5.4.3.4. Input 4: Request signal derivative

From Figure 5.19 it can be seen that the derivative of the request signal is only defined in terms of whether it is negative or positive. This tells the controller if the signal is declining or inclining, which plays a very important role in some control decisions.

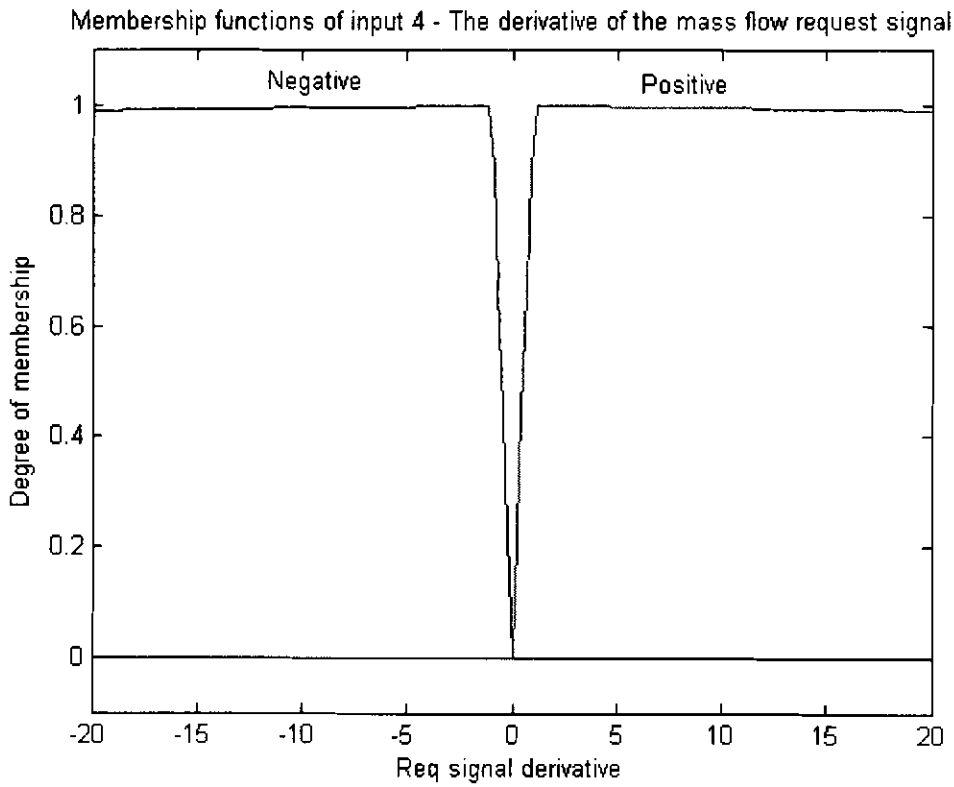

Figure 5.19 Membership functions for input 3 – The request signal

5.4.3.5. Input 5: Large valve closing state

Input 5 is a logical state that, when active, means that the controller is in the process of closing the larger valve. Because the valve does not close immediately, it is important to complete this specific state before continuing with normal control. This state can only become active if the "Input addition" state has been active long enough for the process to settle. As can be seen from Figure 5.20, the input is only defined for a 1 or a 0.

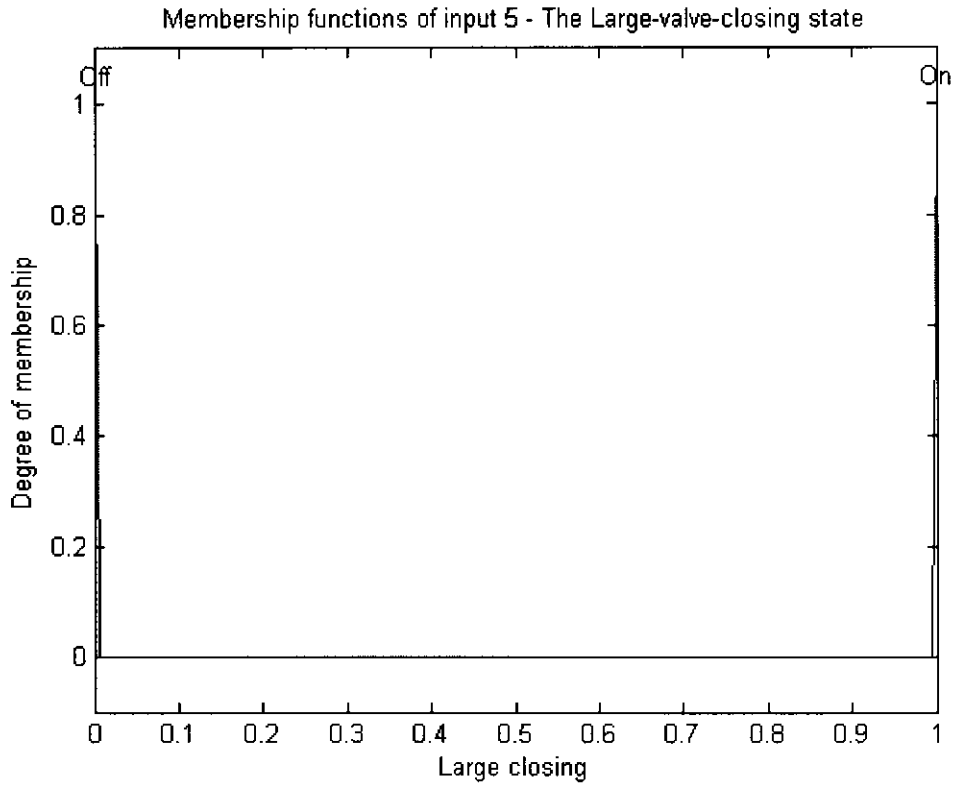

Figure 5.20 Membership functions for input 3 – The request signal

5.4.3.6. Input 6 and 7: Current small and large valve travel

The valve travel of both the small valve and the large valve is very valuable to the controller since it informs the controller how much each valve can still be used to generate more (or less) mass flow. For instance, if both valves are almost closed, they can both be used to increase the mass flow quicker, but if one of them is already fully open, the mass flow will not be reached as fast since only one valve is available to increase it.

The valve travel input also lets the controller know whether a valve is completely closed, this is valuable since it allows the controller to avoid unnecessary jumps in the mass flow. For instance, the controller would not make use of the large valve to go from a mass flow of $1.2 \cdot \dot{m}(\text{large})_{\min}$ to $1.8 \cdot \dot{m}(\text{large})_{\min}$ if it knows that the large valve is closed at that moment. It would rather use the smaller valve alone, and in doing so avoid having to open the large valve and causing a jump in mass flow.

5.4.3.7. Input 8 and 9: Probability prediction from networks

The value of these inputs has already been discussed in section 5.3, and will not be repeated here. It can, however be mentioned that the decision boundary for the prediction was shifted a bit to increase the asymmetric reliability of the trained networks. This can be seen in Figure 5.21 – The probability is deemed "high" from about 0.42, and not from 0.5 as will be expected – this value was optimised with the use of Genetic Algorithms (see Chapter 6).

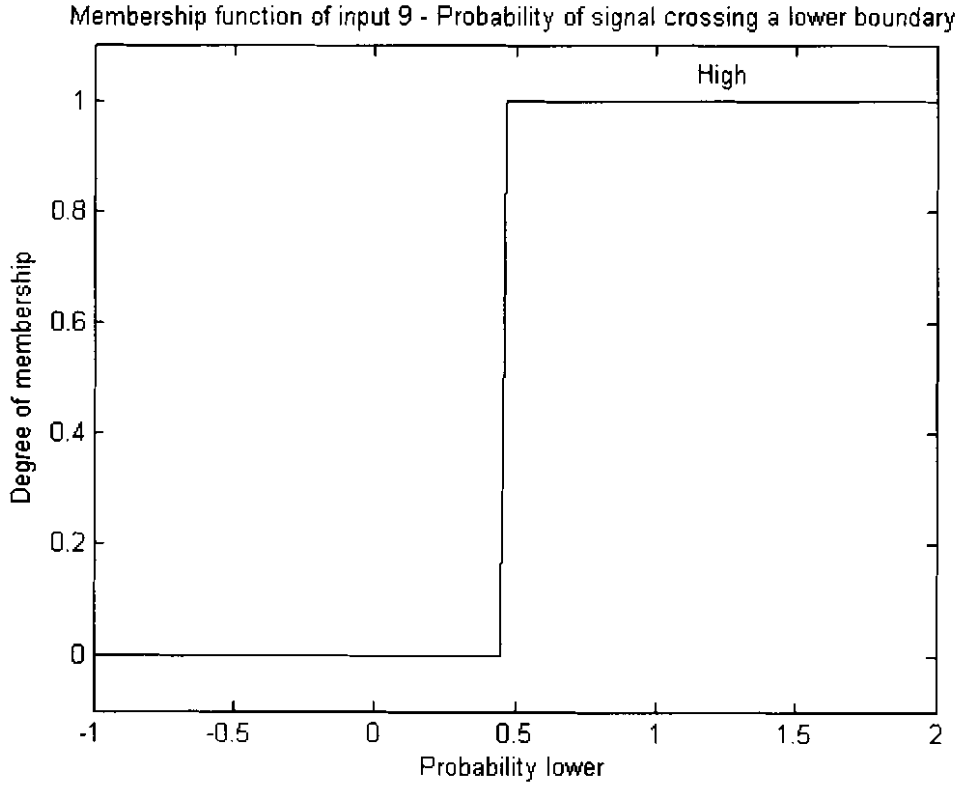

Figure 5.21 Shifted decision boundary

5.4.3.8. Input 10: Last command given to small valve

Input 10 is used in conjunction with input 11 when the controller decides to close the larger valve. Refer to Figure 5.22 for an example.

As can be seen, the large valve is not closed until the smaller valve is generating enough mass flow to absorb the dip caused by the large valve's non linearity. As the small valve's mass flow rate draws near the goal, the commands to it will obviously start to get smaller. The commands will eventually be very small when the mass flow has settled on the required rate (in this case at about t = 5.6 s) – when this happens, the controller knows the system is ready, and it subsequently closes the larger valve.

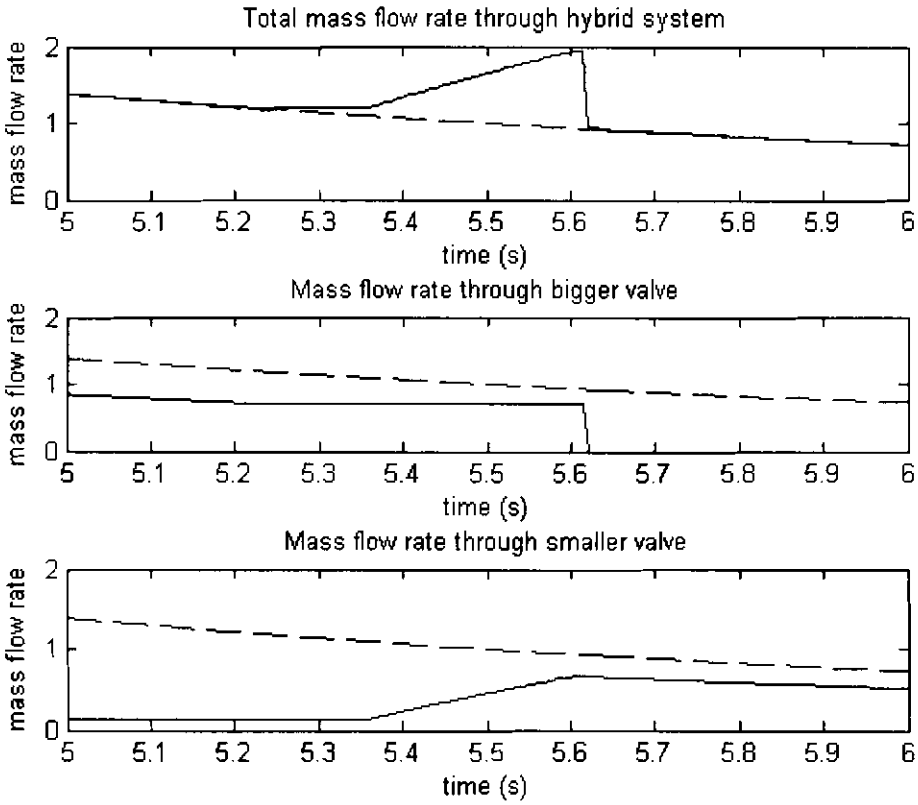

Figure 5.22 The process of closing the large valve

5.4.3.9. Input 11: Input addition state

The input addition state is also only defined as either a 0 or 1 (true or false). When this state is true, the crisp controller adds a constant to the mass flow error of the valve that would absorb the dip in mass flow caused by closing the larger valve. In Figure 5.22 the Input addition state is true from about t = 5.35s to about t = 5.6 s.

5.4.3.10. Input 12: Small valve closing state

Input 12 fulfils exactly the same function for the smaller valve as input 5 does for the larger valve. The main difference between closing the smaller valve and closing the larger valve is that no "Input addition" state exists for the smaller valve. The dip in mass flow created by it is so small that it is rarely necessary to close it. Furthermore, the controller was created in such a way that the small valve never closes if its dip in mass flow will cause the total mass flow to dip below 2 % of the requirement.

5.4.3.11. Input 13 and 14: Large and small valve minimum valve-travel state

The minimum valve-travel state of each valve is a logical state that is activated by the external crisp controller when the valve is not meant to be closed, but has reached such a small travel distance that only a tiny negative command will cause it to close completely. When this state is activated, the valve in question is no longer given any negative commands. This state can also be used to avoid non-linear phenomena like valve-choke.

Figure 5.23 shows an example where the larger valve activates its minimum valve travel state and is kept inactive while allowing the smaller valve to follow the request signal in its descent. This happens from about t = 3.6 s.

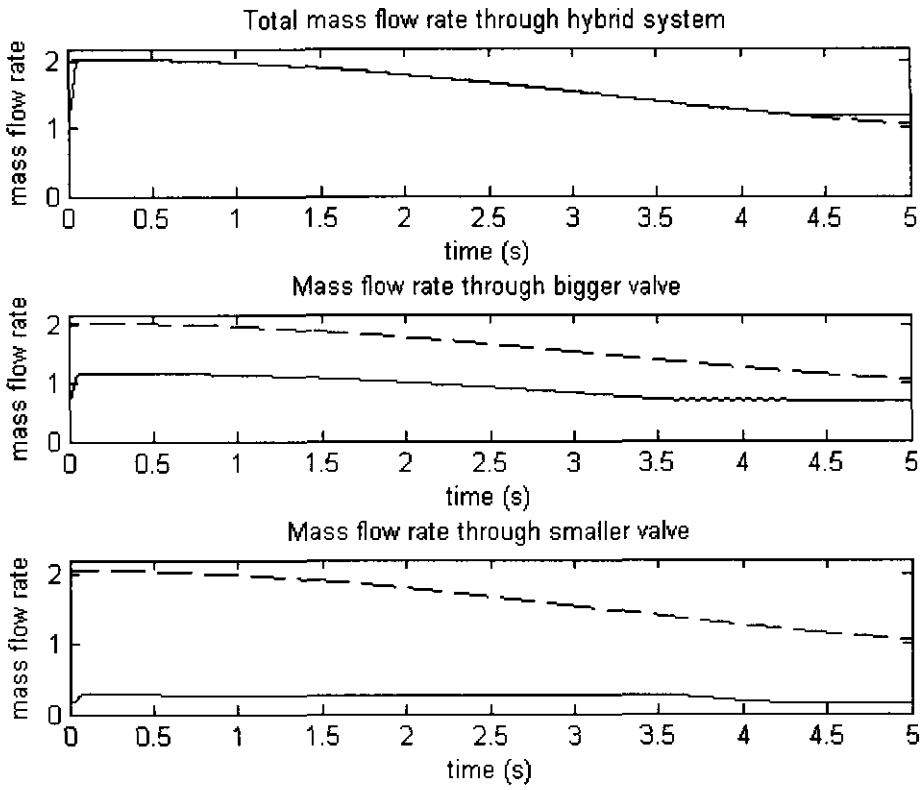

Figure 5.23 Example where the minimum valve-travel state is used

5.4.4. Fuzzy controller outputs and membership functions

As can be seen from Figure 5.13, the Fuzzy Inference System used for this project has 5 outputs. The two main outputs are the separate commands to the two valves, and the other three are logical states that can be activated by the fuzzy system, but is done so in cooperation with the crisp controller. These outputs and their membership functions will now be briefly discussed.

5.4.4.1. Output 1 and 2: Large and small valve incremental command

The membership function arrangement for both the small and large valve command outputs for this fuzzy system are exactly the same. For that reason they will be discussed together.

As can be seen from Figure 5.24, the three membership functions of the Fuzzy Process Control inference system, as discussed in Chapter 2, is applied for both outputs as explained. However, two membership functions, named "Open" and "Close" were added.

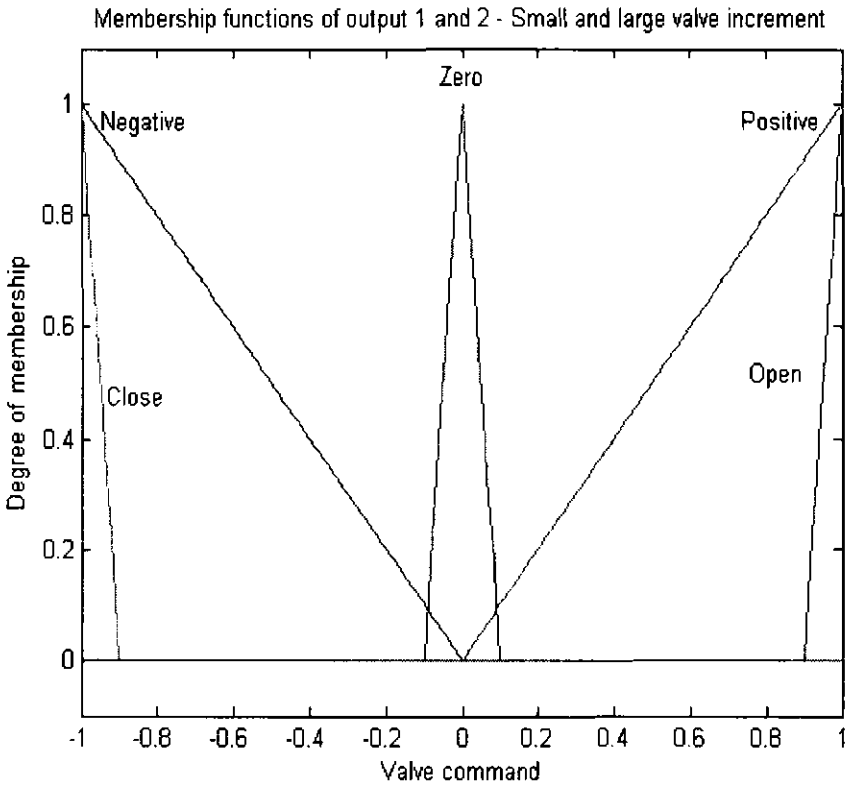

Figure 5.24 Small and large valve command outputs

The two added membership functions enable the controller to give large commands to the two valves when needed. For instance, if the controller decides to close the large valve, it issues the "Close" command which results in the valve receiving maximum negative commands – and causes it to close as quickly as possible. The membership functions are therefore added to increase reaction time.

5.4.4.2. Output 3, 4 and 5: Large valve closing state, input addition state and small valve closing state

These three outputs are, as mentioned, pure logical states – which means that they should only be defined as a 0 or a 1. Because this is not possible in fuzzy logic (as will be expanded on in the next section) it was decided to pair the control of these outputs with the crisp controller. The fuzzy logic controller is, however the one that decides when these states should be activated.

5.4.5. Fuzzy rule base

In Figure 2.2 it is seen that the rule base, which is part of the knowledge base (KB) of the Mamdani FRBS, plays an important role in deciding the values of the fuzzy outputs. The final fuzzy rule base for the FLC (fuzzy logic controller) designed in this project has 46 rules. These rules can be grouped according to the role they play in the control, since more than one rule is usually required to obtain a certain result. This section will discuss the fuzzy rules according to these groups, the exact rules that were used can be found in Appendix A

5.4.5.1. Control above the maximum mass flow rate of the smaller valve

Rules 1 to 3 and 35 to 37 are concerned with controlling the hybrid valve system when the required mass flow rate is above the maximum mass flow of the smaller valve. Both the smaller and larger valves are used for control in this case, and the method used is the PI type fuzzy process control technique discussed in section 2.3.7. As will be seen, this technique is used for all regions of control.

5.4.5.2. Closing the smaller valve

In certain situations, as discussed in section 5.4.3.1 and shown in Figure 5.15, the fuzzy controller decides to close the smaller valve in order to minimise overshoot. Rules 4-6, 19 and 20 are concerned with this situation.

5.4.5.3. Closing the larger valve

Figure 5.22 illustrates the process that is followed when the large valve is closed without causing a dip below 2 % of the request. Fifteen rules are used in order to successfully complete the process in any situation. The rules are 7-12 and 21-29.

5.4.5.4. Control below the minimum mass flow of the small valve

Control below the minimum mass flow of the small valve must, of course be dealt with by the small valve alone. Rules 13-15 are responsible for giving the small valve the commands in such a case. Again, the commands are based on the PI-type fuzzy process control technique.

5.4.5.5. Control in the overlapping region

The region between the minimum mass flow of the large valve and the maximum mass flow of the small valve may be very eventful. This is because either one, or both of the valves may be open in this region, and the controller must subsequently decide what to do. However, while the controller is analysing the signal, the request signal must also be controlled, and unnecessary jumps must be avoided. The control of the open valves in the overlapping region is done by rules 16-18, 30-34, 39 and 45.

5.4.5.6. Providing values to uninvolved outputs

The five outputs of the fuzzy inference system shown in Figure 5.13 were added to each fulfil a certain role in specific circumstances. However, there exist certain circumstances when the role of some of these outputs is not relevant at all. For instance, the logical output "Input addition", that is used when the large valve is being closed, will never be used when the input request is above the maximum mass flow of the smaller valve, since the larger valve *has* to be open. However, the Mamdani FRBS is composed in such a way that the rules must provide an output value for each fuzzy output, for every possible combination of inputs.

For that reason, rules 38, 40-44 and 46 are used to provide values for the outputs when they are uninvolved.

5.4.6. Crisp external controller

In Figure 5.13 it can be seen that a crisp controller was added to the hybrid valve control system. The reason for this inclusion as well as the crisp controller's tasks will be discussed in this section.

5.4.6.1. Motivation

The reasons for using Fuzzy Logic as design platform for this project have already been discussed. However, the control valves modelled for this project possesses some crisp features and requires some crisp control inputs that are not possible to deal with by using a fuzzy system.

In Fuzzy logic, for instance, the Boolean "Input addition state" that was discussed in section 5.4.4 will never be *completely* true. It may be *very nearly* true – but it is highly unlikely that the fuzzy inference

system will define the output as exactly 1. As another example, when the large valve opens, it will always produce a certain jump in mass flow – it is not possible to connect a membership function to that jump – the most optimal way to deal with it would be to measure it, and subsequently make provision for it.

For that reason it was decided to pair up the Fuzzy Inference System with a crisp controller that can deal with the parts of the control that cannot be fuzzified. The crisp controller does not deal with any of the valve control itself, but is only used as a "co-operator" of the fuzzy controller.

In Figure 5.25 the crisp controller is shown separated from the hybrid valve control system. From the figure, the different tasks of the crisp controller are visible. These tasks will now be discussed briefly.

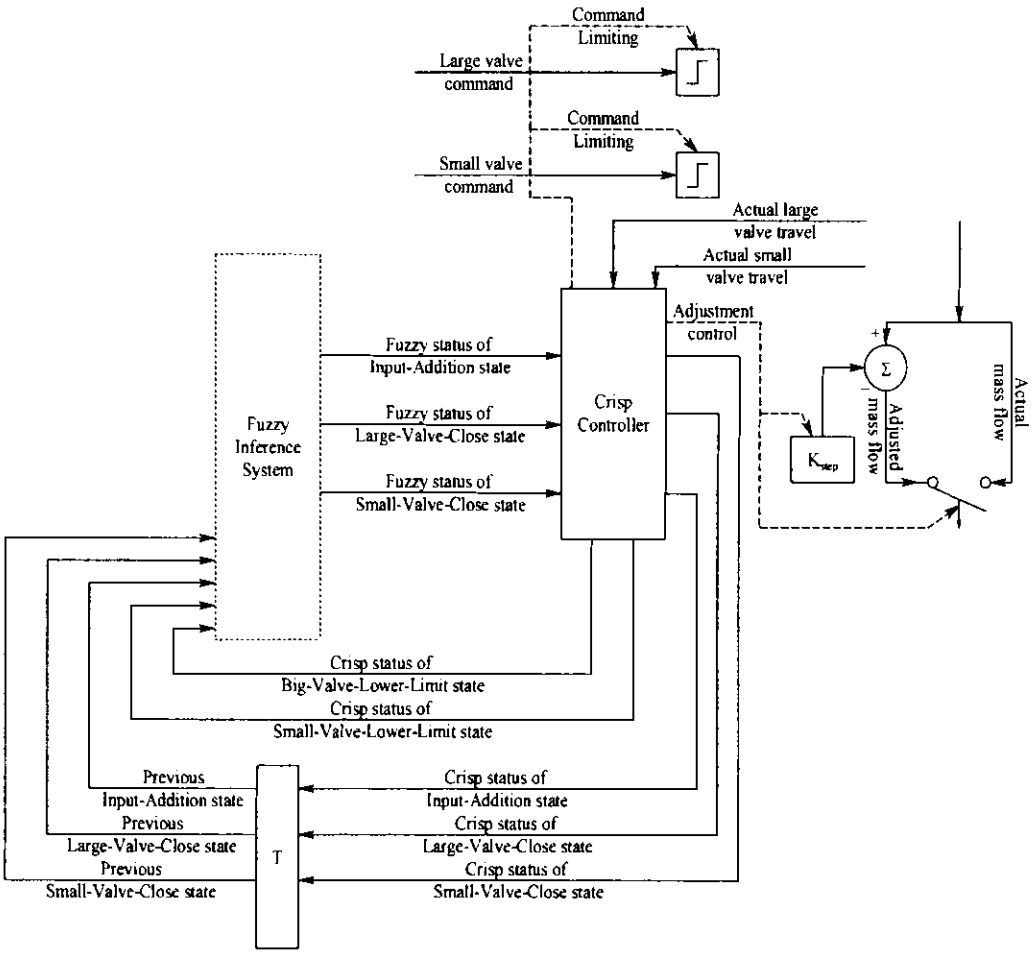

Figure 5.25 The crisp controller

5.4.6.2. First task: Interpreting Boolean outputs from Fuzzy Inference System

From Figure 5.25 it can be seen that the three Boolean outputs of the fuzzy logic controller are given as inputs to the crisp controller. The task of the crisp controller is to ensure that these outputs are always defined as either a 0 or a 1 – this is done because the outputs of the Fuzzy Inference System is not capable of defining something as simply true or false.

5.4.6.3. Second task: Activating the valves' minimum valve-travel states

It is clear from Figure 5.25 that the crisp controller receives the travel distances of each of the two valves as inputs. The controller uses this to determine whether the valves may receive any further negative commands from the fuzzy system. When the crisp controller is informed that one of the valves has reached its minimum, and will close soon, it activates that valve's minimum valve-travel state. This ensures that the valve will no longer receive any commands that might cause it to close – consequently bringing about an undesired dip in mass flow.

5.4.6.4. Third task: Limiting the valve commands

The third task of the crisp controller is to limit the commands given to the two valves. For each valve there is a maximum voltage that can be applied to it, and it is the task of the controller to ensure that this voltage is not exceeded.

5.4.6.5. Fourth task: Adding a constant to the mass flow error

The process of closing the larger valve when the controller is sure that the request will fall below its minimum has already been discussed. In short, the process requires the smaller valve to compensate for the dip in mass flow by allowing much more mass flow than necessary through it.

The amount of extra mass flow required depends on the size of the dip that is caused in the total mass flow rate, when the larger valve closes. It is the task of the crisp controller to add the correct amount of required mass flow to the request when that is needed. This is done (as illustrated in Figure 5.25) by *subtracting* the amount from the current actual mass flow – which subsequently increases the demand as is required.

5.5. GUI implementation

The graphical user interface that was developed for the Fuzzy Logic hybrid valve controller is shown in Figure 5.26

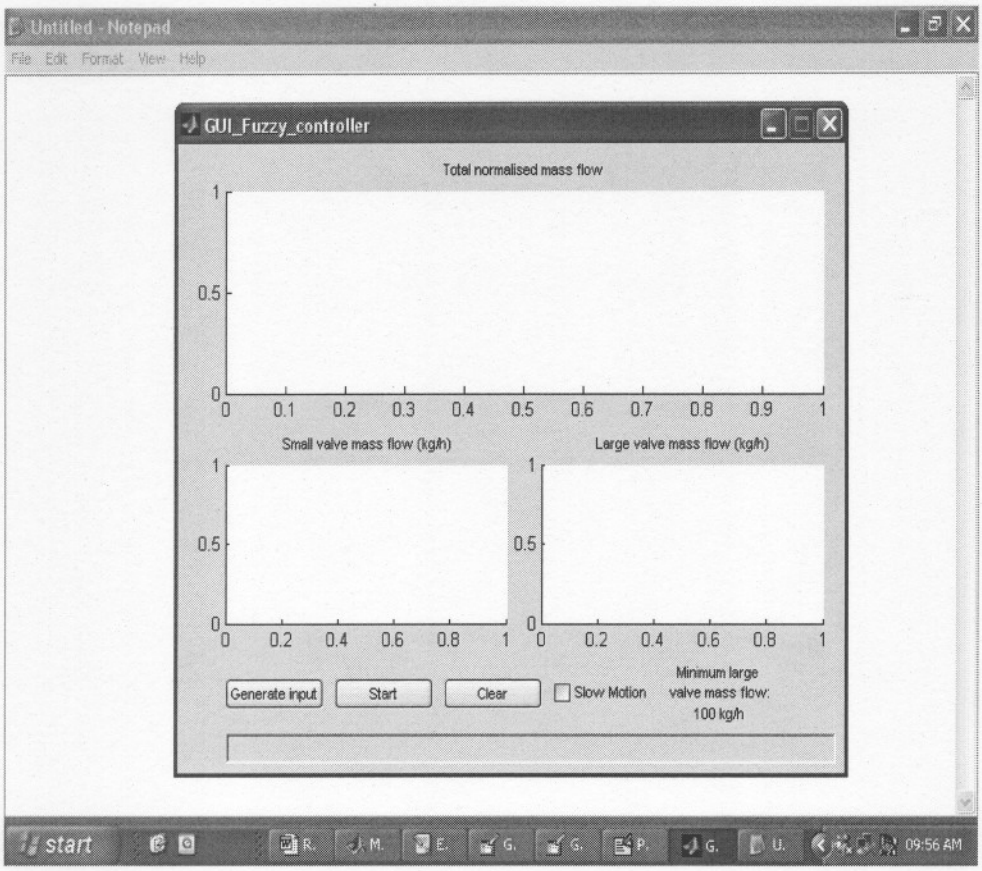

Figure 5.26 Graphical user interface for the fuzzy logic controller

As can be seen, the structuring of the three graphs is identical to the GUI created for the PID controller (see section 4.5). The user operates the GUI with three pushbuttons, and a checkbox. The checkbox allows the user to choose between a "real time", slow motion simulation of the controller's reaction to the request signal, or a quicker mathematical calculation that provides only the result. The progress bar at the bottom informs the user what progress is made, since calculating the response can be very time-consuming.

When the user pushes the button called "Generate input", the window shown in Figure 5.27 appears.

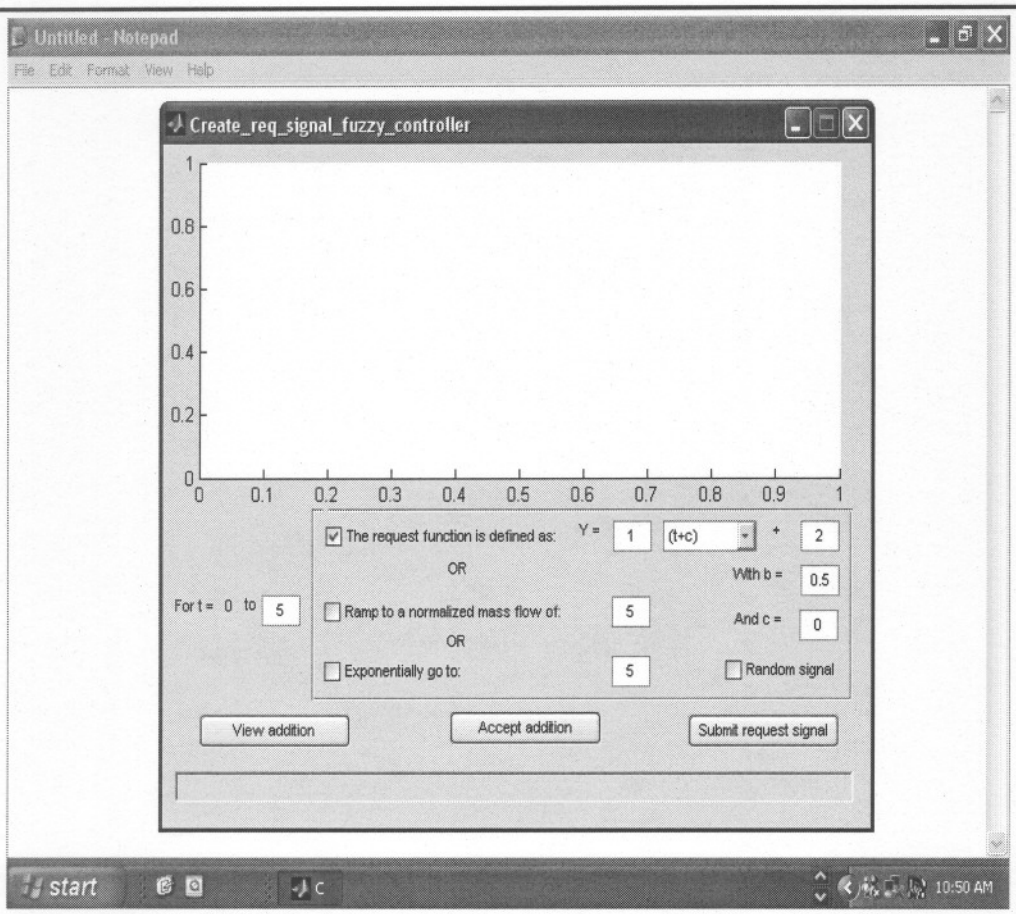

Figure 5.27 Window used for generating an input request signal

The window provides the user with many choices to "build up" almost any input request signal. The first step is to choose the time span of the part of the request signal that will be created next. For instance 5 seconds, as is illustrated in Figure 5.27.

Next the user must decide which form the request signal should take. The first option in the function panel allows the user to choose between a number of different functions to define the request with. This is done by choosing the desired function from a drop-down menu, as illustrated in Figure 5.28. In this case the sinus function is selected.

The next step is to change the gain parameter, and the values for b and c, by entering the required values into the editable text boxes provided.

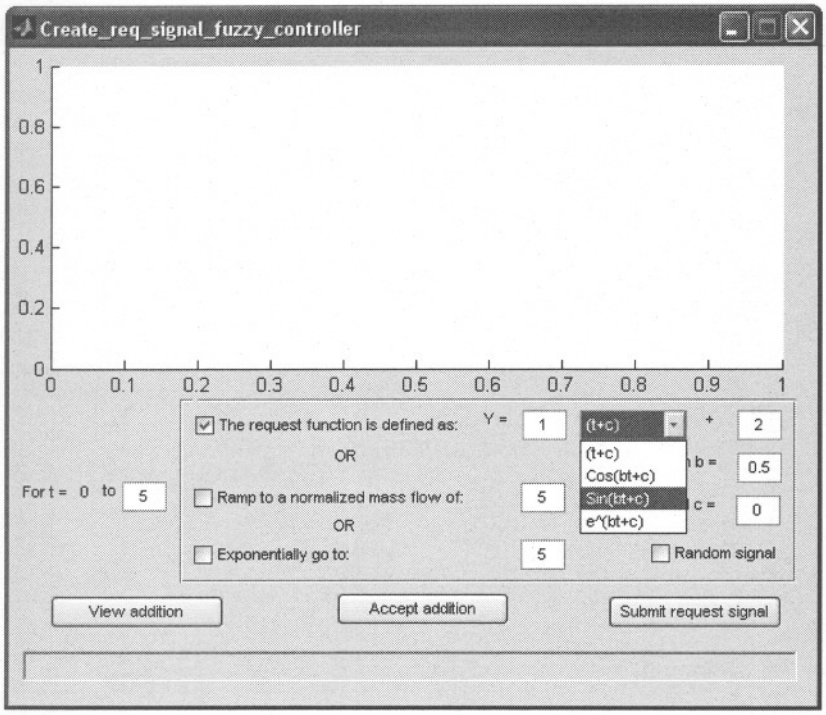

Figure 5.28 Choosing a function from the drop-down menu

For instance, if the function $3\sin(2t+5)+4.5$ is required, the user will enter the values as illustrated in Figure 5.29

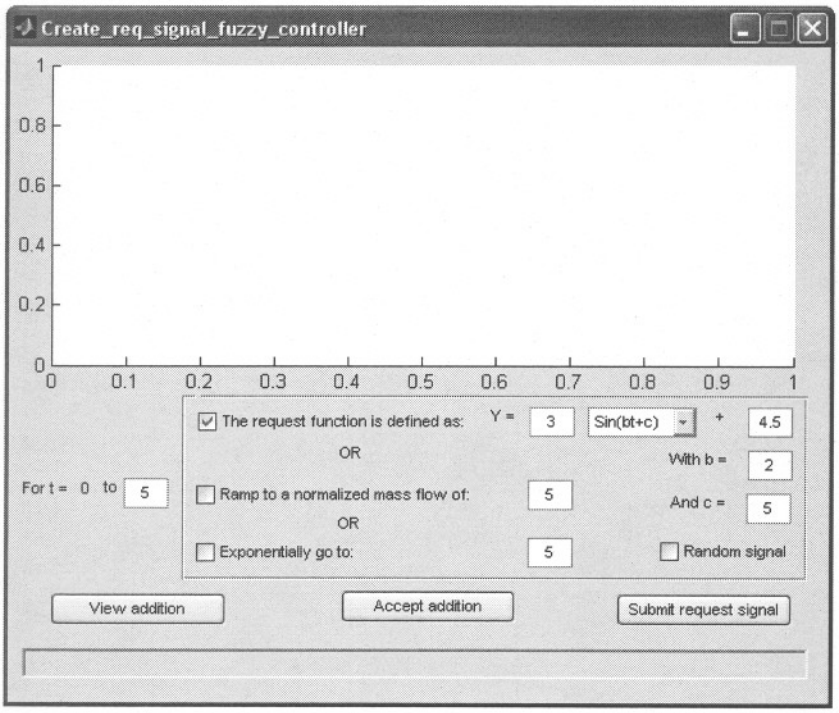

Figure 5.29 Generating the function $3\sin(2t+5)+4.5$

The user can now view the addition he has made by pressing the "View addition" button. For the case shown in Figure 5.29 the signal will be displayed as shown in Figure 5.30.

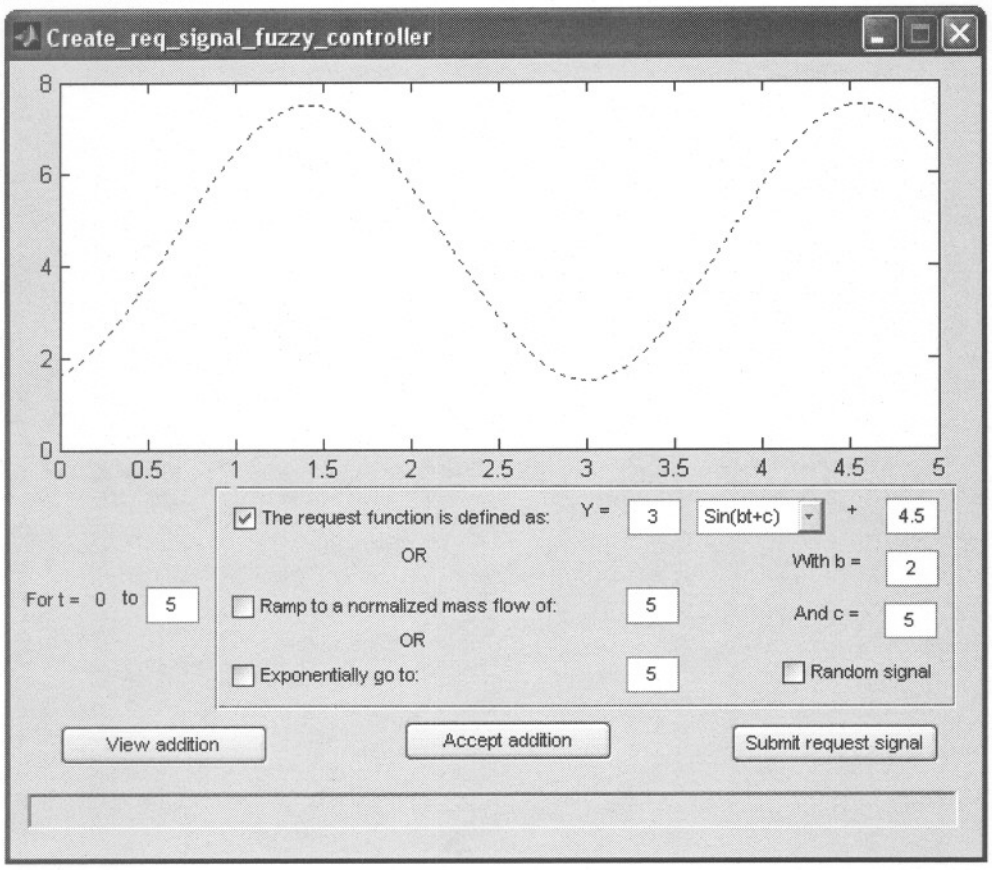

Figure 5.30 Viewing the function $3\sin(2t+5)+4.5$

If the user decides to accept the addition, he can do so by pressing the "Accept addition" button. The program automatically adds white noise to the signal, and filters it with the low pass filter. The noisy signal together with the filtered signal is then displayed to the user.

This is shown in Figure 5.31. The solid line represents the noisy signal created, and the dotted line represents the signal after it has been filtered.

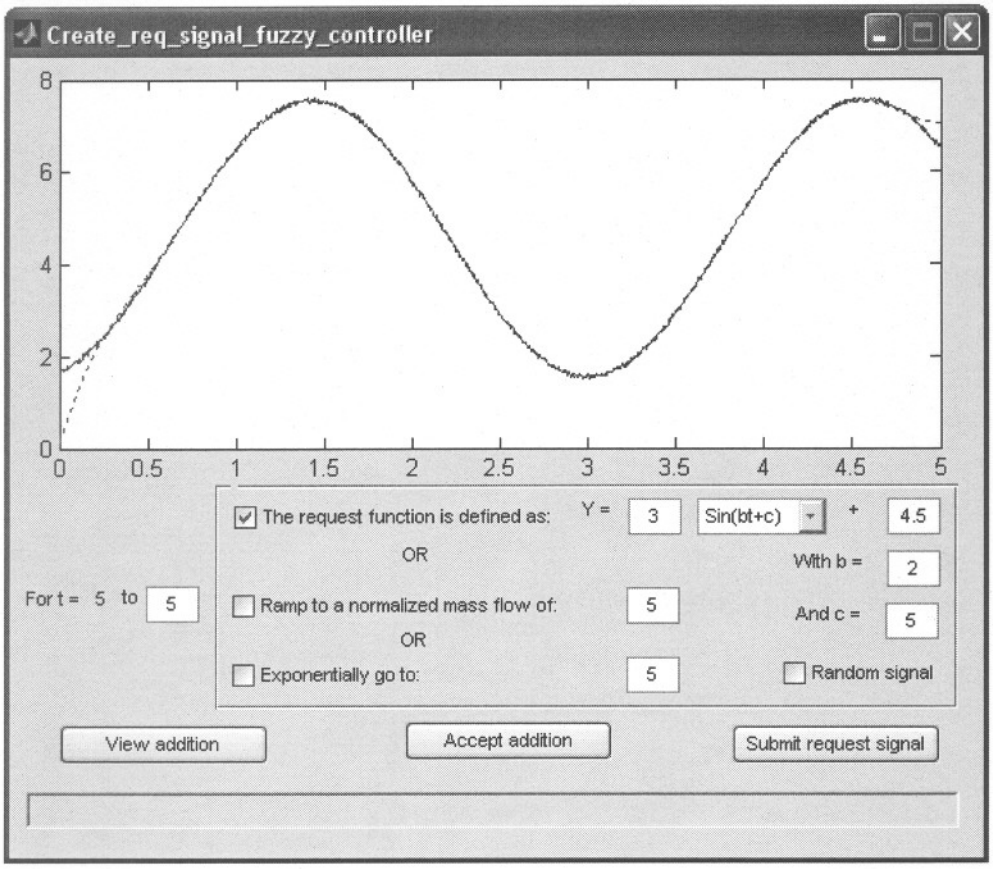

Figure 5.31 Noisy and filtered input request signal

The user may next, for instance, decide to ramp the request slowly down to a value of $0.5\,\dot{m}(\text{large})_{\min}$. This can be done by selecting the second option named "Ramp to a normalized mass flow of:" and entering the value 0.5 into the editable text box.

The time span of the ramp must also be selected. If the user prefers the request to reach $0.5\,\dot{m}(\text{large})_{\min}$ after 12 seconds of simulation, the accepted addition will be as shown in Figure 5.32.

From the screen shots of the GUI, it is apparent that the user may choose to create a random input signal by selecting the "Random signal" checkbox. As will be seen in Chapter 6, this function was used to test for flaws in the controller's stability.

Using the functions available, the user can therefore submit the valve controller to a multitude of tests, and create input signals that inspect every aspect of the control.

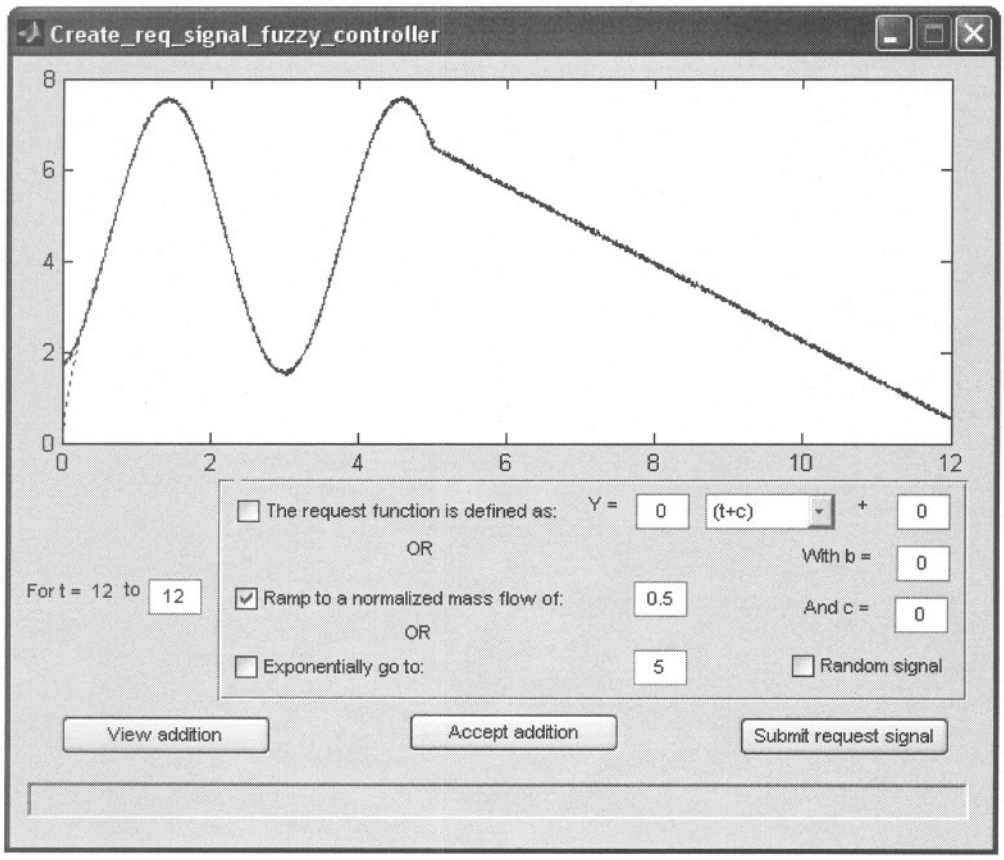

Figure 5.32 Adding a ramp of 7 seconds to $0.5\,\dot{m}(\text{large})_{\min}$

When the user is satisfied with the input signal created, the "Submit request signal" button can be pressed. This causes the program to download the filtered request signal to the fuzzy control GUI introduced in Figure 5.26. The GUI will display the created signal in the graph at the top. For the request signal created in Figure 5.32 the GUI will therefore look as shown in Figure 5.33.

The response of the controller can now be examined by pressing the "Start" button.

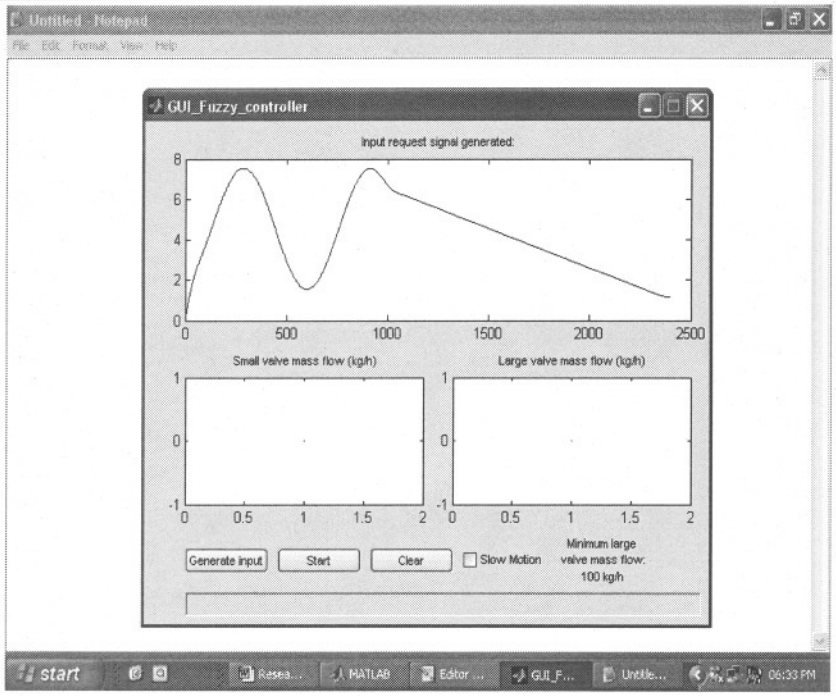

Figure 5.33 The Fuzzy logic controller displays the filtered request signal created

Figure 5.34 shows the GUI's illustration of the valve controller's response to the input signal that was downloaded.

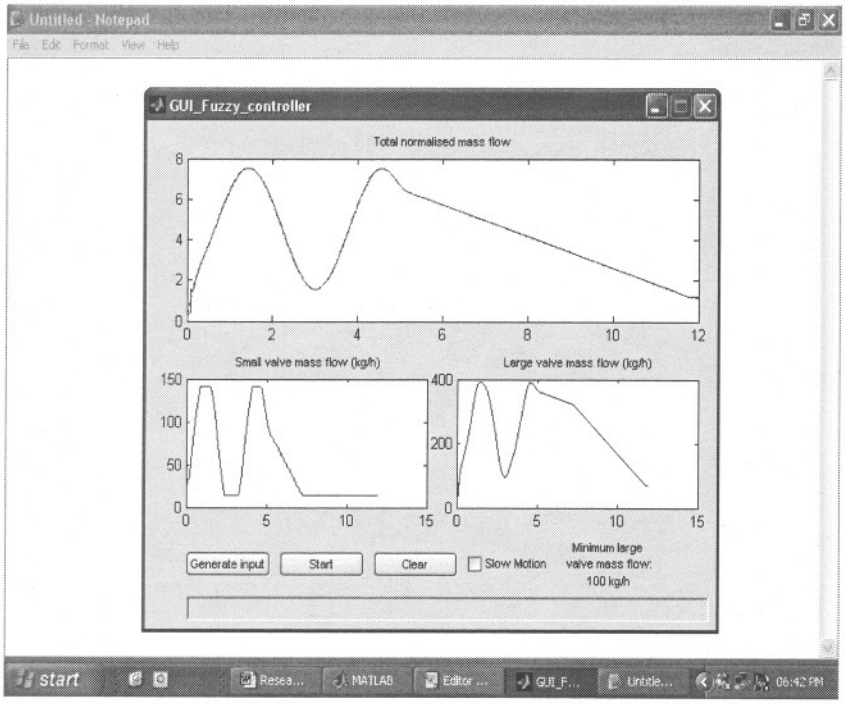

Figure 5.34 the GUI's illustration of the valve controller's response to the input signal

As with the GUI discussed in section 4.5, the m-file containing the code of the fuzzy logic GUI, as well as an executable GUI is available on the CD provided with this dissertation.

5.6. Conclusion

Chapter 5 discussed the operation and design of the non-linear, Fuzzy Logic hybrid valve controller that was developed in this project.

The chapter began with pointing out some of the design decisions that was made for the controller, and then proceeded to describe the four main elements that make up the hybrid valve control structure.

It was seen that it is feasible to filter the input to the controller without much negative impact on the response. Filtering the input also ensures that prediction can be done more effectively. The exact filter parameters were chosen with the use of Genetic Algorithms, and will be discussed in the next chapter.

Predicting the behaviour of the input request signal helps to reduce the amount of jumps and dips in the mass flow through the hybrid valve system, and decreases the total mean error. The two predictors that were used for this controller were discussed.

The Fuzzy Inference System that is the heart of this controller was examined next. The membership functions and tasks of the fourteen inputs and five outputs that are used by the fuzzy system to optimise the control were discussed to explain the operation of the system.

Lastly the role of the crisp controller in aiding the fuzzy system was discussed. It was seen that the hybrid valve system cannot operate exclusively on a fuzzy design foundation.

The next chapters will expand on the optimisation and testing of the non-linear Fuzzy Logic based hybrid valve controller that was examined in this chapter.

6
Control optimisation

6.1. Motivation and overview

The final step of creating the controller is to optimise the design. The hybrid valve controller system discussed in the previous chapter has many parameters that lend themselves very well to optimisation. For this reason it was decided to use Genetic Algorithms (GAs) for this purpose.

The process of using GAs to optimise a range of parameters was discussed in Chapter 2. From a design point of view, the main challenge of optimisation using GAs is the development of an appropriate objective function. As mentioned in Chapter 2, objective functions should give an accurate measure of the performance of each individual. If the objective function is flawed, the results of the GA optimisation will most certainly be flawed as well.

For this project two different elements had to be optimised. The first was choosing the best low pass filter for filtering the request input to the controller and optimising its filter parameters. The parameters had to be optimised in such a way that the frequency content of the resulting signal is high enough not to affect the response of the filters, but still low enough to ensure accurate prediction.

The second element of the control structure that had to be optimised was the Fuzzy Inference System. Optimising the parameters of the membership functions of many of the inputs and outputs helped to minimise rise time as well as the total mean error.

The parameters that were optimised, the design of the objective function as well as the results of the optimisation will now be discussed.

6.2. Filter optimisation

In section 5.2.1 it was determined that choosing the correct filter and optimising its parameters is the first step towards training networks for prediction. It is therefore imperative that the choice of filter be

done well, since changing it later will require much extra time and effort in retraining the prediction networks.

For this reason it was decided to ensure optimal filter selection by employing Genetic Algorithms.

6.2.1. Parameters optimised

For the purposes of this project four filter types were considered as possibilities. These are:

- The Butterworth filter
- The Chebyshev type I filter
- The Chebyshev type II filter
- The Elliptic filter

The parameters of the filters that were optimised were:

- The pass band frequency (ω_p)
- The stop band frequency (ω_s)
- The maximum pass band attenuation (R_p)
- The minimum stop band attenuation (R_s)

These parameters were optimised for the large valve alone. This unfortunately implies that the frequency content of the optimised filter might be too high for the small valve, but it was decided that more performance will be lost by facilitating the smaller valve than would be the case if its slower reaction were ignored.

The consequence of this was that only one valve needed to be controlled for this exercise. This, of course, simplified the controller tremendously, and it was possible to use a simple, linear PD controller for the valve.

However, if the PD controller had not been absolutely optimal, it would have had a negative effect on the reaction of the valve. Therefore, the two constants (P and D) of the PD controller were also added as parameters to be optimised.

Each chromosome therefore constituted 7 genes:

Chrom = {P_const; D_const; ω_p; ω_s; R_p; R_s; Filter_choice}

The genes were all real valued, except the last, which was either a 1, 2, 3 or 4 – depending on the filter selected.

The order of the filter was determined by the choice of parameters, and was therefore not specified specifically.

6.2.2. Objective Function

As mentioned, the goal of the objective function is to ensure that the optimised filter facilitate accurate predictions, but also optimal valve reaction.

The objective function for finding the best filter was eventually chosen as the sum of the normalised correlation of two pairs of signals. These are:

- The normalised correlation of the *filtered* request signal with the controlled output of the system – with the *filtered* request signal as input.
- The normalised correlation of the controlled output of the system – with the *filtered* request signal as input with the controlled output of the system – with the *unfiltered* request signal as input.

This might sound very confusing, and will be explained much better by referring to Figure 6.1.

The first pair of signals that is correlated (the *filtered* request signal with the controlled output of the system – with the *filtered* request signal as input) ensures that the parameters of the filter are not too relaxed.

As can be derived from Figure 6.1, the unfiltered request signal is one with frequency content somewhat higher than can be dealt with by the valve. This means that, if the parameters of the filter are too relaxed, the system will not be able to follow the request, and the controlled output of the

system – with the filtered request signal as input will not correlate well with the filtered request signal itself.

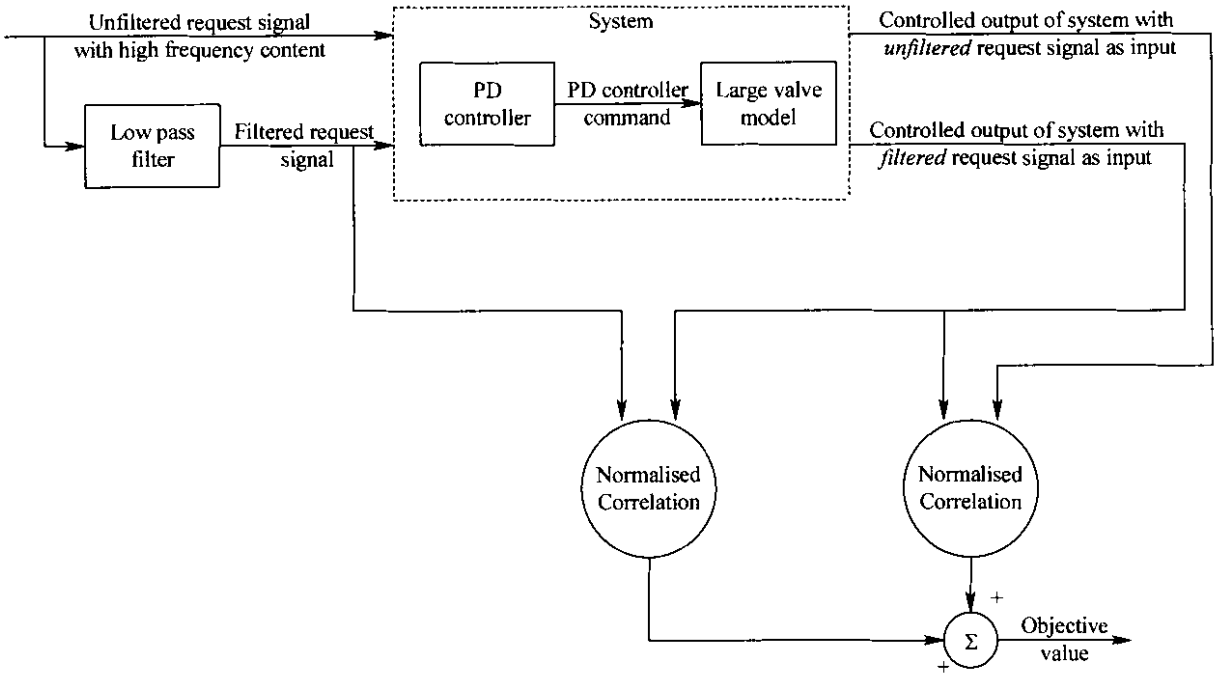

Figure 6.1 Determining the objective value for optimising the low pass filter

The second pair of signals that is correlated ensures, on the other hand, that the filter parameters are not too strict. This will cause the reaction of the system to the *filtered* request to differ greatly from its reaction to the *unfiltered* input. The correlation between the two signals will therefore be low.

6.2.3. Optimised filter parameter values

The results of the optimisation of the filter parameters are shown in Table 6.1:

Table 6.1 Results of filter optimisation

Parameter	Symbol	Value
Pass band frequency	ω_p	12.32 rad/s (1.96 Hz)
Stop band frequency	ω_s	182.96 rad/s (29.13 Hz)
Maximum pass band ripple	R_p	13.64 dB
Minimum stop band attenuation	R_s	41.39 dB

The best suited filter type was seen to be a Butterworth filter.

6.2.4. Performance difference compared to intuitive parameter choices

In Figure 6.2 and Figure 6.3 the performance of both the intuitive filter parameters (see section 5.2.1.5) and the optimised filter parameters are compared. The improvement is not easy to see, but close inspection will reveal that the signals on the right have higher correlation than those on the left. This can also be proven mathematically.

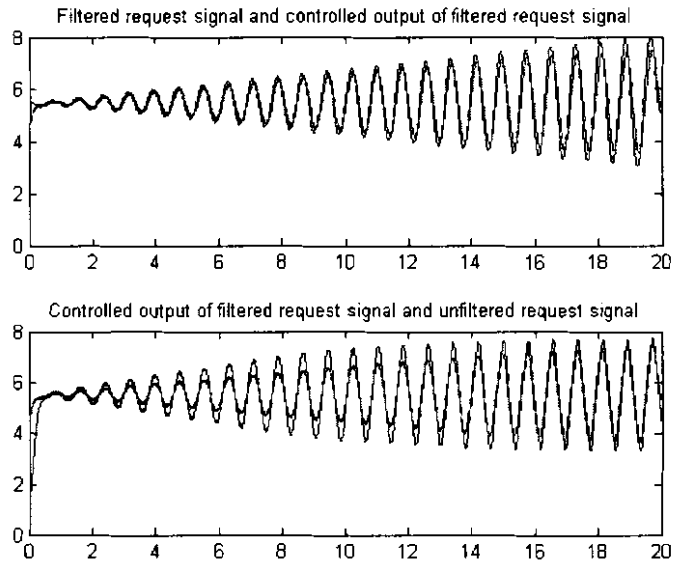

Figure 6.2 Results with intuitive filter parameters

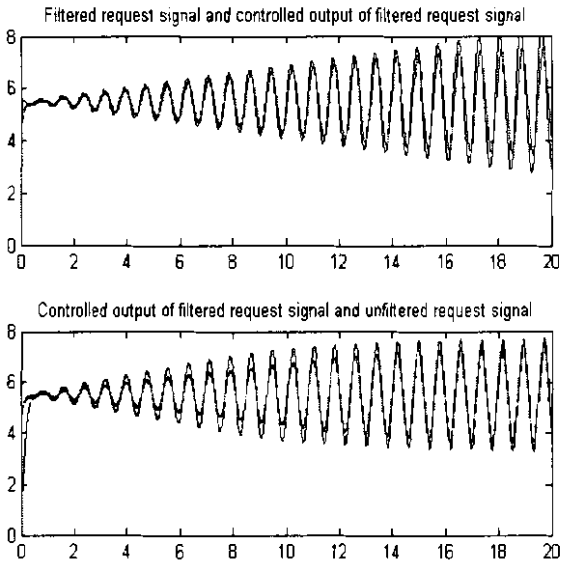

Figure 6.3 Results with optimised filter parameters

The optimised filter parameters were consequently used in the design of the controller.

6.3. *Fuzzy controller optimisation*

In section 1.2 the formal problem statement for this project was said to be the development of an *optimised* algorithm that will control a hybrid control valve system. This implies that some form of optimisation will have to be done to ensure that the hybrid valve controller reaches the target value as soon as possible, and with as little total mean error as possible. This section will discuss how the Fuzzy Logic based hybrid valve controller discussed in Chapter 5 was optimised.

6.3.1. Parameters optimised

In section 5.4 the functions of the different inputs and outputs for the Fuzzy Inference System used for the control of the hybrid valve were discussed together with their membership functions. Each membership function consists of three or more adjustable variables, depending on its form. For instance, a triangular membership function has three adjustable variables: its starting value, centre value and ending value. A triangular membership function with the values [1, 2, 3] will consequently look like the one shown in Figure 6.4.

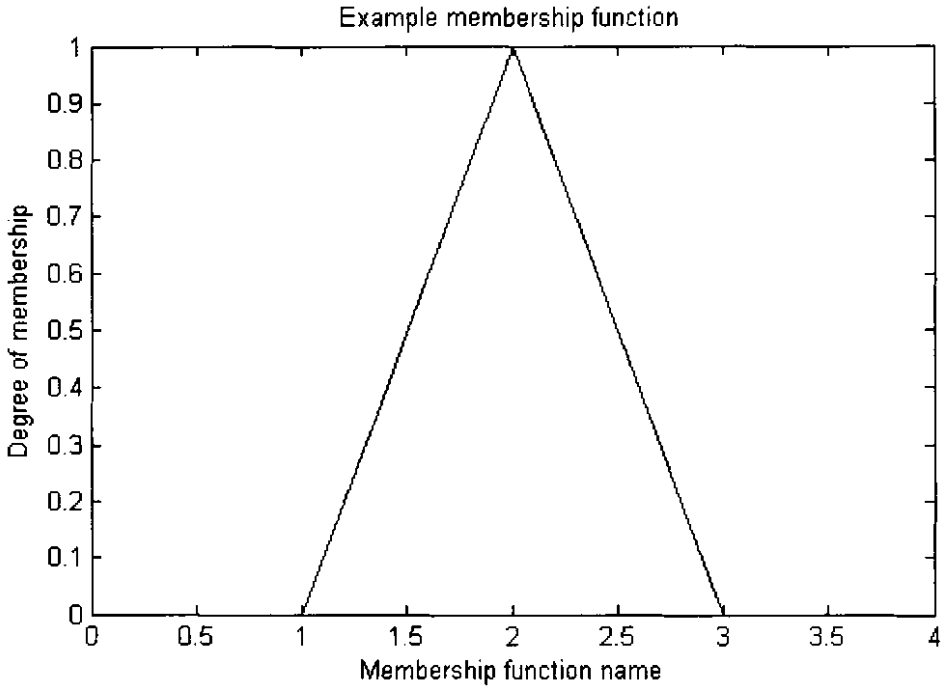

Figure 6.4 Triangular membership function with values [1, 2, 3]

It is these adjustable variables whose values can be optimised by the use of GAs. If the values are chosen correctly, it may improve the performance of the Fuzzy system.

The optimisation was done for many of the membership functions of the Fuzzy System used in this project. However, some of the membership functions used in the hybrid valve controller's Fuzzy Inference System will not derive benefit from such optimisation. For instance, the 5 logical inputs to the system can only be defined as either a 0 or a 1 – which does not leave much room for optimisation! Therefore the membership functions of the inputs that will derive benefit from optimisation were identified and their values were used as genes in the optimisation chromosome.

The membership functions that were chosen to be optimised were:

- Input: Mass flow error
  - Membership function: "Big" (2 values)
  - Membership function: "Shooting a little over" (2 values)
- Input: Request signal
  - Membership function: "Very Dangerous" (3 values)
  - Membership function: "Almost above small valve max" (4 values)
  - Membership function: "Not Dangerous" (4 values)
- Input: Probability of crossing lower boundary
  - Membership function: "High" (2 values)
- Input: Small valve last command
  - Membership function: "Small" (4 values)

Two more values were included in the chromosome of each individual. These are the small valve command gain and the large valve command gain. They are illustrated as K_s and K_l in Figure 5.13, and determine the weight of each valve's fuzzy command. As can be seen, some of the membership functions chosen to be optimised do not submit all their values for optimisation. This is because some of them have values that can be defined as "infinite" and will not gain any advantage from such a process.

The final chromosome was therefore equipped with 23 genes for which the most optimal values had to be found.

6.3.2. Objective Function

The objective value for evaluating the performance of each individual were simply based on finding the individual that caused the smallest total mean error. However, some additional conditions had to be met.

The first condition was to ensure that no dip in mass flow causes the total hybrid valve mass flow to fall below 2 % of the request. The objective function was therefore designed to test any dips in mass flow, and disqualify individuals that caused such a dip.

The second condition originates from the definition of the values for the membership functions. For most membership functions to be valid, the first value has to be smaller than the second, the second smaller than the third, and so on.

The first attempt to ensure this, was to allow the whole chromosome to be chosen randomly, and then disqualify all chromosomes that did not fulfil the condition. However, it was quickly realised that the chances of generating a random chromosome with 23 genes all of which conform exactly to those rules are very slight indeed. This made an already lengthy process even more time consuming, since thousands of worthless chromosomes were generated. Therefore it was decided to attempt a different strategy.

The strategy was to choose the *first* value for the membership function randomly, then generate a random value to be *added* to the first to obtain the second value, then a random value to be added to the second to get the third value and so on. The triangular membership function shown in Figure 6.4 will therefore be defined with the values [1, 1, 1] and not [1, 2, 3] as was originally the case. This strategy ensured that almost all chromosomes that were generated could be evaluated.

6.3.3. Optimisation input signal

The input signal to the optimisation system is almost as important in this project as the objective function.

The signal must challenge the controller with each tricky part of controlling the hybrid valves, and must not cause the optimisation to focus on one aspect too much.

However, the input signal must also not contain too many data points. The controller that was created is extremely computationally intensive. The consequence of this is that the process of optimisation takes several days to complete - for each added second of input signal, an extra day of optimisation is usually needed. Therefore, choosing the correct signal was by no means an easy feat.

The input signal that was eventually chosen is shown in Figure 6.5. It covers all the aspects of control that can be optimised. As can be seen, the request increases at first to almost $10 \cdot \dot{m}(\text{large})_{\min}$. This

ensures that both the larger and smaller valve is open, and exposes the optimiser to control requirements in the region far above $1 \cdot \dot{m}(\text{large})_{\min}$. From about t = 1.2 s, the request starts decreasing, and eventually decreases to well below the $1 \cdot \dot{m}(\text{large})_{\min}$ boundary. In this situation it is optimal to close the larger valve, and hand control to the smaller valve. From about t = 3 s, the request oscillates closely around $1 \cdot \dot{m}(\text{large})_{\min}$. This fine-tunes the decision parameters of the control system, since the control decisions are not the least bit obvious, and can only be acquired by the optimisation algorithm.

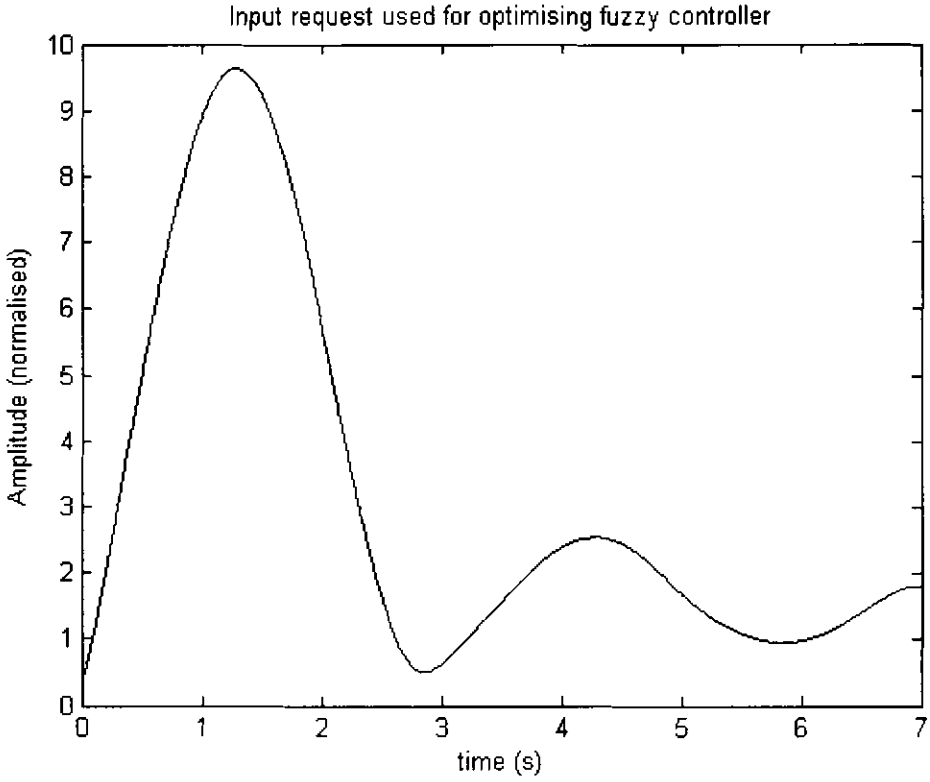

Figure 6.5 Input request signal used for optimisation

6.3.4. Optimisation progress

Figure 6.6 is a plot that shows the decrease in objective function value against the generations. As can be seen, after about 70 generations, the value does not improve any further – therefore it was assumed that this chromosome is the most optimal one.

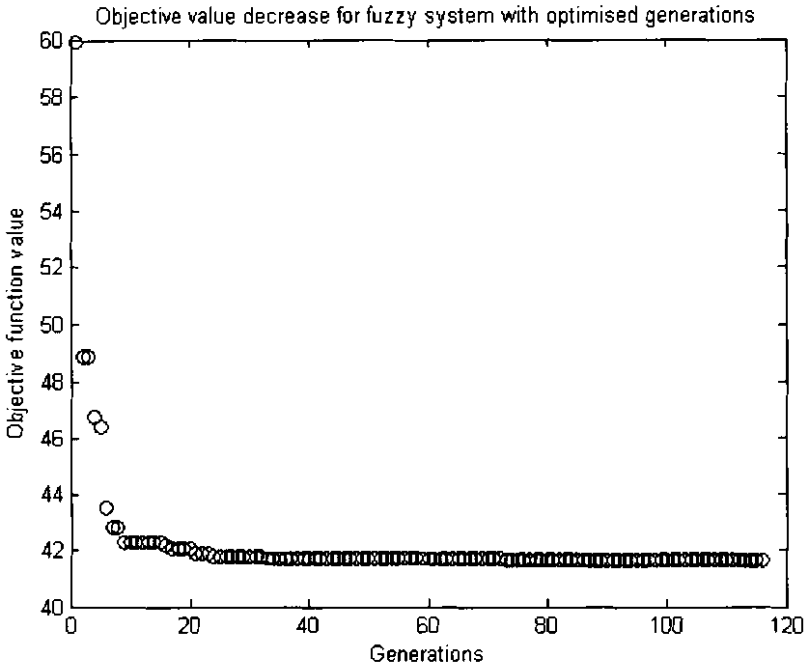

Figure 6.6 Optimisation progress

6.3.5. Performance difference compared to intuitive parameter choices

The most visible improvement of the optimised Fuzzy Inference System is the improvement of its decision when to close the large valve, and when not to. The difference between the optimised and un-optimised controller on this aspect is shown in Figure 6.7 and Figure 6.8.

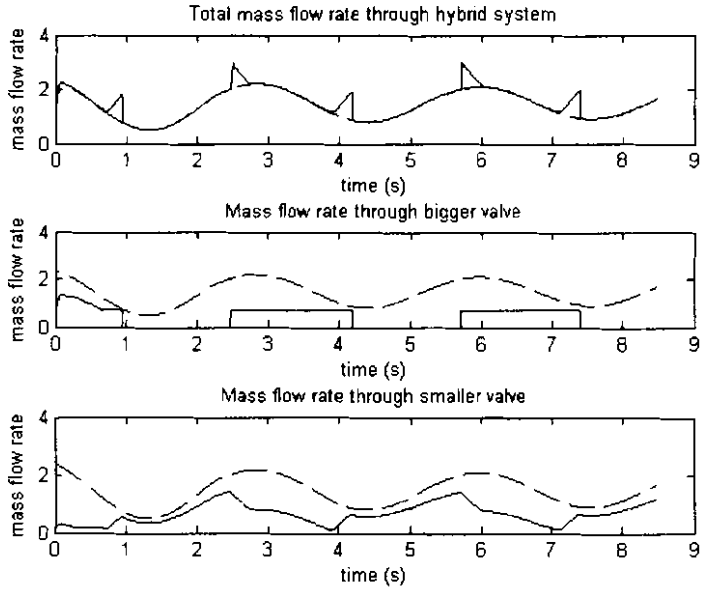

Figure 6.7 Un-optimised controller response

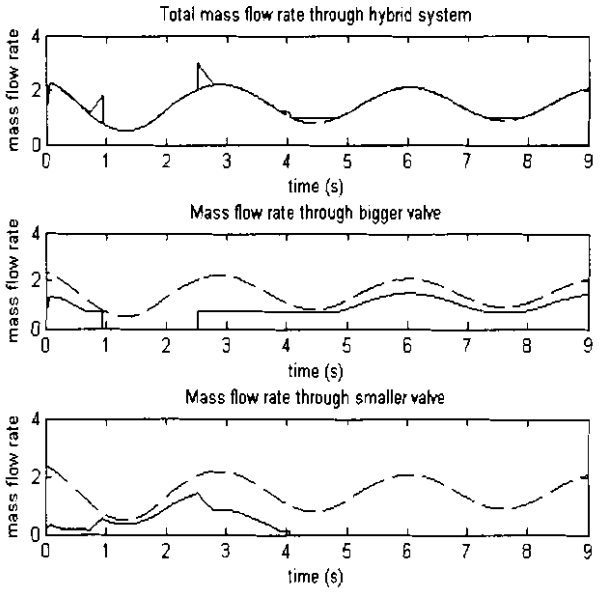

Figure 6.8 Optimised controller response

6.4. Conclusion

Chapter 6 discussed the optimisation of both the low pass filter that is used to filter out the high frequencies of the input request signal, and the membership function values of the Fuzzy Inference System that is discussed in Chapter 5. It was seen that the optimisation improved the results obtained for both the filter and the Fuzzy system. This, of course improves the results that will be obtained from the whole hybrid valve control unit.

7
Control evaluation

7.1. Overview

Extensive testing should accompany any high-quality design. Without evaluation of a project, the value of that project will be lost, and much time and effort will have been wasted.

This chapter will subject the hybrid valve controller to a series of different input signals, and the controller's response will be evaluated to distinguish whether the response is as optimal as can be expected. Lastly the controller will be tested to find any signs of instability.

7.2. Complex control signals

In section 4.4 the PID based hybrid valve controller that was developed was subjected to a few complex input signals. It was seen that the PID based controller is not capable of handling such requests, and that many unnecessary jumps and dips in the mass flow occurred.

This section will evaluate the capability of the more complex, Fuzzy Logic-based valve controller to deal with such inputs.

7.2.1. Gaussian noise

It has already been mentioned that the addition of white noise to the request signal for this project may be a very viable option to consider, since mass flow measurements are not always entirely accurate. However, the low pass filter that was added to the hybrid valve controller resolves a very large part of that problem, since the high frequencies of the white noise are filtered out. In Figure 5.13 the low pass filter is situated so that only the request signal is filtered. Filtering can, however also be applied to the mass flow feed-back loop, as well as other relevant loops without affecting the controller's performance.

Nevertheless, it was decided to test the ability of the controller to deal with a signal of which the noise has not been filtered.

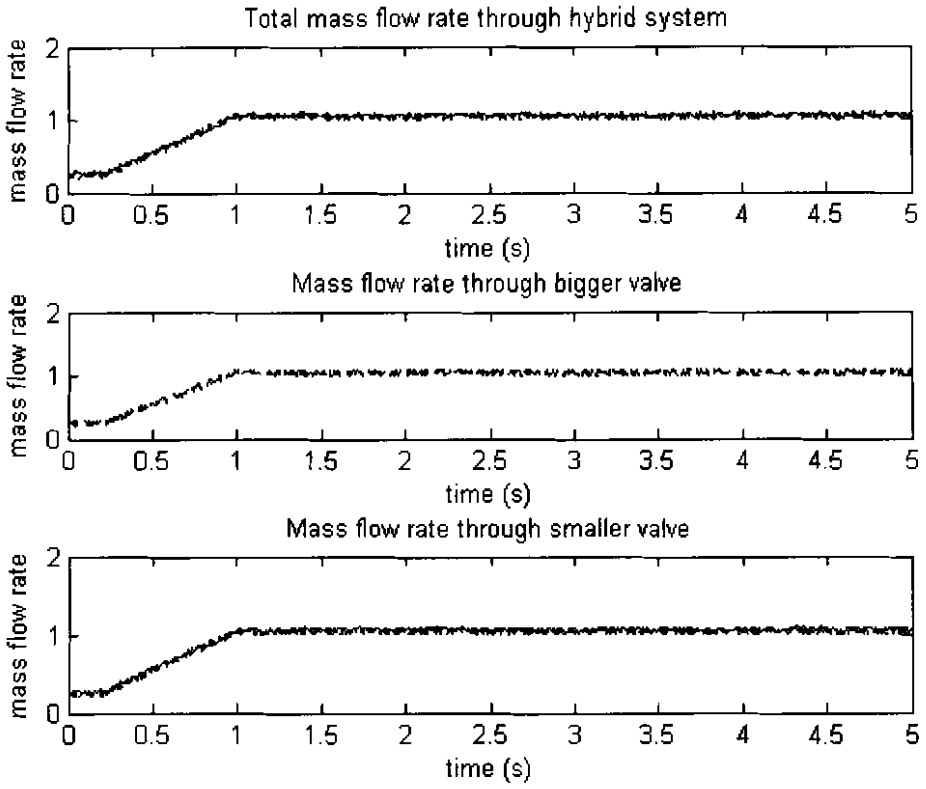

Figure 7.1 Noisy input signal on the minimum-capability boundary of the large valve

Figure 7.1 shows the reaction of the controller to a noisy input signal on the minimum-capability boundary of the large valve. As can be seen, because of the overlap available to the two valves, the small valve is capable of dealing easily with the signal, and no jumps in mass flow occur at all.

Figure 7.2 shows the reaction to a noisy input on the maximum-capability boundary of the small valve. Once again the overlap of the two valves ensure that neither of the two valves is forced to open or close in order to reach the mass flow requirement, and subsequently no undesired jumps in mass flow occur.

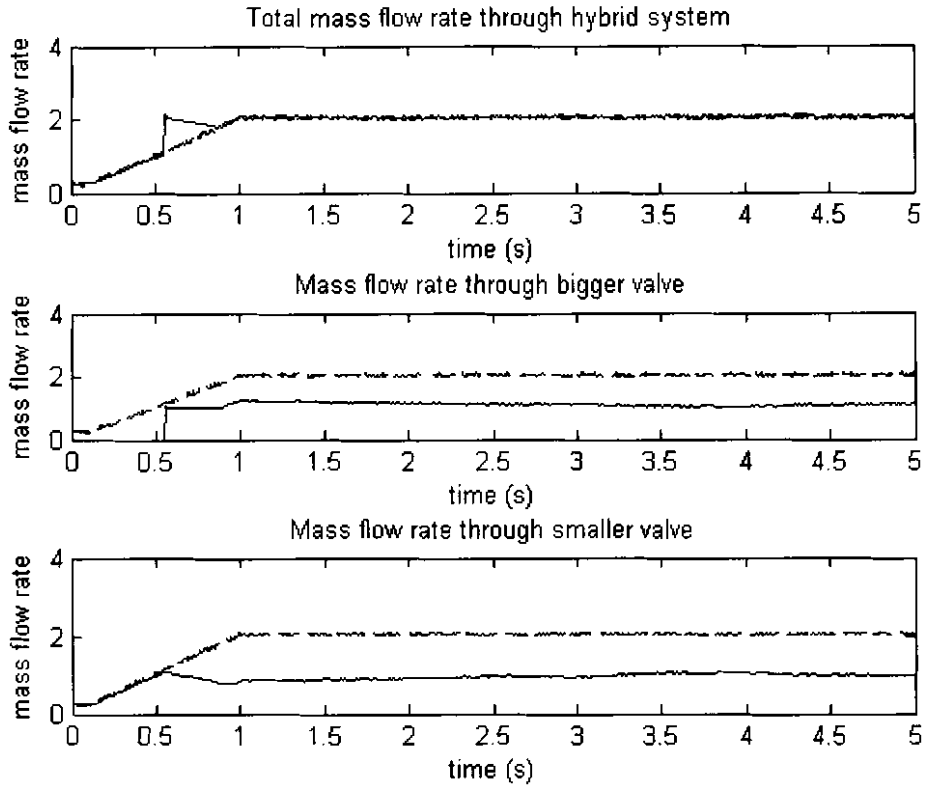

Figure 7.2 Noisy input signal on the maximum-capability boundary of the small valve

7.2.2. Oscillation near the valves' operational boundaries

Signals that oscillate near the operational boundaries of the two valves also presented a control problem for the PID based valve controller. The Fuzzy Logic based controller's response to such signals will now be investigated.

7.2.2.1. Oscillation around the minimum-capability boundary of the large valve

Figure 7.3 shows the reaction of the controller to a signal that oscillates around the minimum-capability boundary of the large valve. As can be seen, since the small valve is capable of handling mass flows up to $2 \cdot \dot{m}(\text{large})_{\min}$ no valve needs to be opened or closed, and consequently no dips or jumps in mass flow occur.

However, for the signal shown in Figure 7.3, the controller was never forced to open the large valve. If the large valve had, for instance already been open at t = 2 s, the request might have been much more challenging.

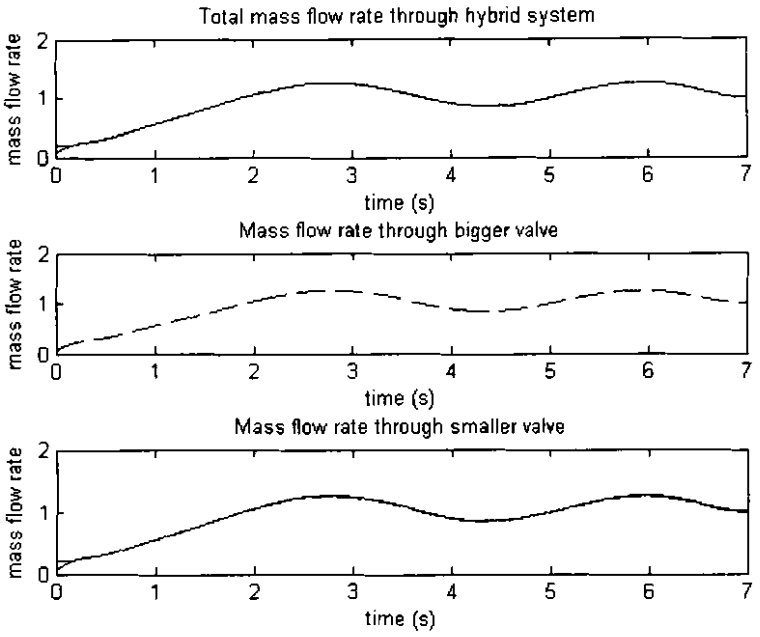

Figure 7.3 Oscillating signal around the minimum-capability boundary of the large valve (1)

Figure 7.4 shows a similar case, but where the controller is at first forced to open the large valve.

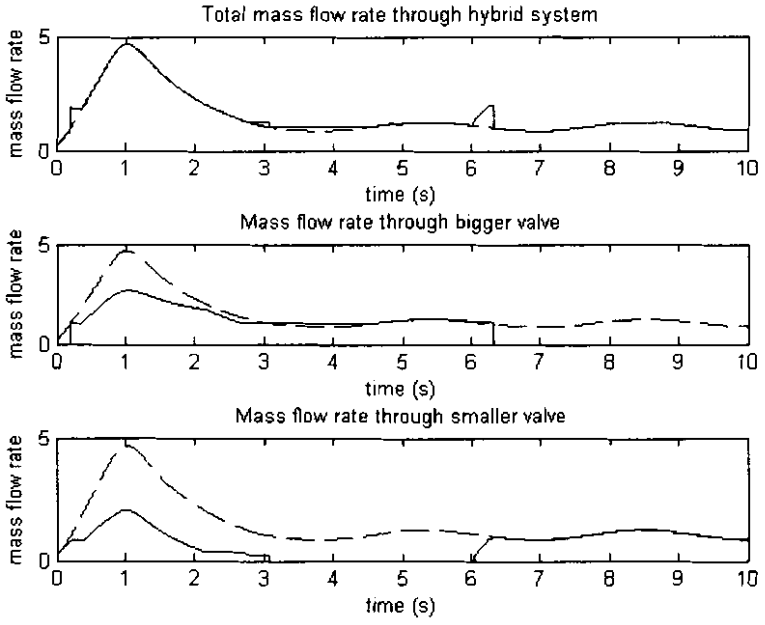

Figure 7.4 Oscillating signal around the minimum-capability boundary of the large valve (2)

From Figure 7.4 it can be seen that the controller closes the small valve at first to minimise overshoot, but when the signal oscillates back towards $1 \cdot \dot{m}(\text{large})_{\min}$ (at about t = 6 s) it closes the large valve,

and uses the small valve alone to follow the oscillating request signal. This is a more permanent solution.

It can therefore be seen that the hybrid valve controller deals well with oscillation around the minimum capability boundary of the large valve.

7.2.2.2. Oscillation around the maximum-capability boundary of the small valve

Figure 7.5 shows the reaction of the controller to a signal that oscillates around the maximum-capability boundary of the small valve. As can be seen, the valve-overlap ensures that only one jump in mass flow is necessary throughout the whole process.

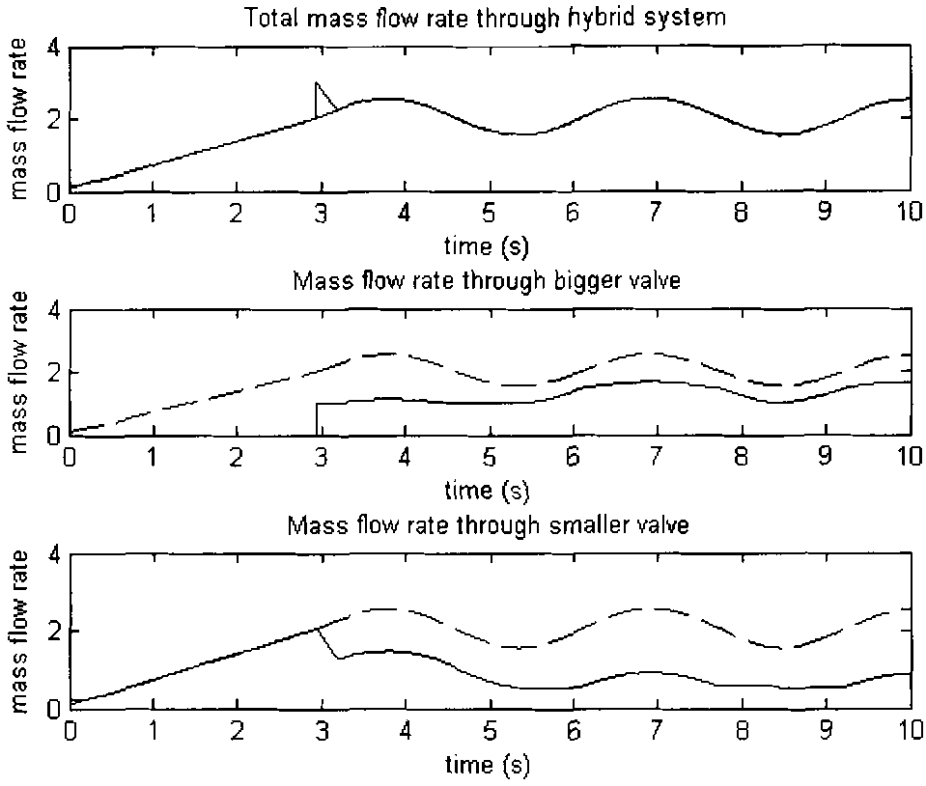

Figure 7.5 Oscillating signal around the maximum-capability boundary of the small valve

7.2.2.3. *Oscillation between the two boundaries*

Signals that oscillate between the maximum capability of the small valve and the minimum capability of the large valve present the most problems, since they may require the assistance of both valves to reach the required mass flow.

Figure 7.6 shows a signal that oscillates between $0.9 \cdot \dot{m}(\text{large})_{\min}$ and $2.1 \cdot \dot{m}(\text{large})_{\min}$. As can be seen, the controller decides at about t = 1.2 s to close the smaller valve, since this would minimise the amount of overshoot. This proves to be a good decision, and the request is met with a very small amount of mean error.

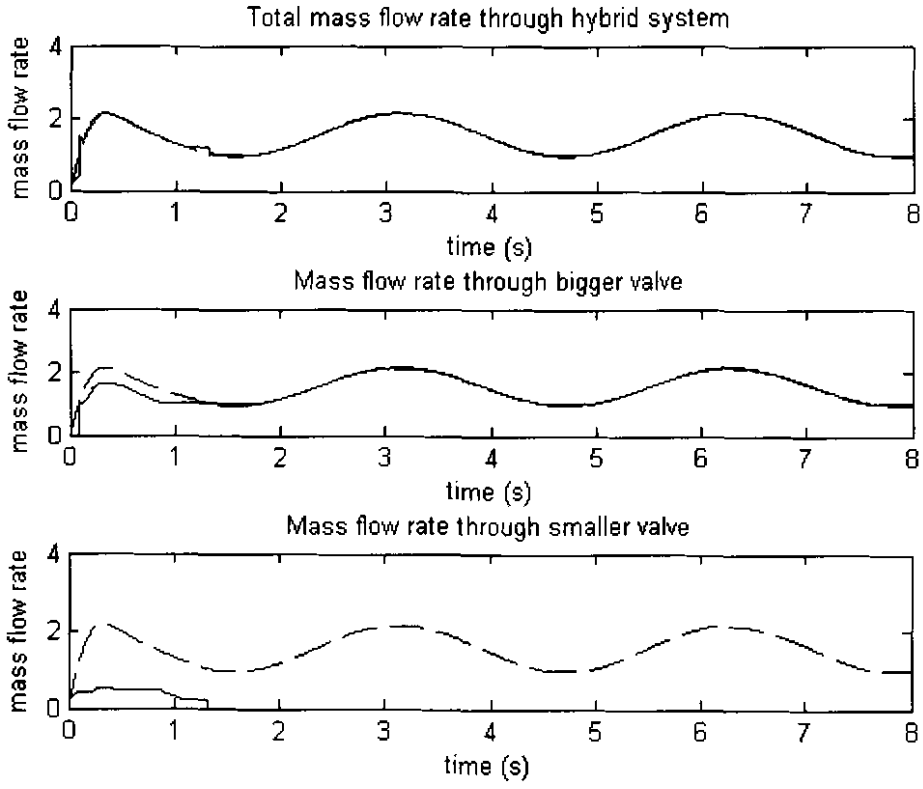

Figure 7.6 Oscillating signal between the two boundaries of the valves

Figure 7.7, on the other hand, shows a request signal that oscillates between $0.6 \cdot \dot{m}(\text{large})_{\min}$ and $2.4 \cdot \dot{m}(\text{large})_{\min}$. In this case, the controller decides to rather close the large valve at t = 1.2 s. This consequently forces the controller to open the valve again at about t = 2.5 s to avoid "undershoot", when the signal crosses the maximum-capability boundary of the smaller valve.

As can be seen, the controller persists with its decision to close the larger valve each time the signal re-crosses $1 \cdot \dot{m}(\text{large})_{\min}$. This, of course, causes a lot of jumps and dips in the mass flow.

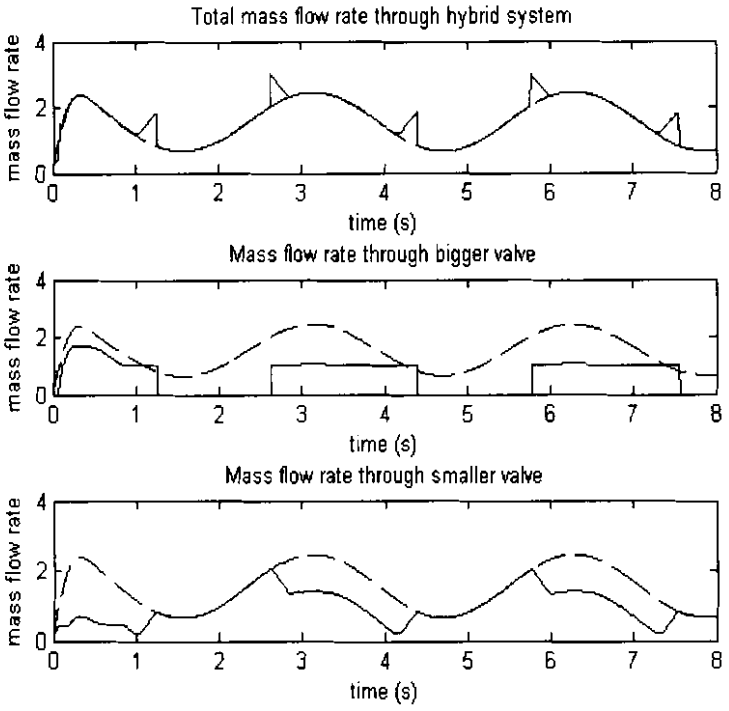

Figure 7.7 Larger oscillating signal between the two boundaries of the valves

However, it can be mathematically proven that (for the request signal in Figure 7.7) the total mean error would have been larger if the controller had decided to close the small valve, and rather risk the overshoot (as in Figure 7.6).

It is therefore seen that a trade-off exists between closing the smaller valve – and risking the small amount of overshoot, and closing the larger valve, which consequently generates some overshoot. The controller makes use of the predictors discussed in section 5.2 to decide which option to follow.

7.3. Stability test

7.3.1. Overview

Probably the most important weak point of fuzzy logic is that it has not yet produced any definite guarantees in the sense of stability or robustness [8]. It is therefore impossible to prove mathematically that the Fuzzy Logic based hybrid valve controller that was developed for this project

will remain stable for any input. However, an attempt may be made to prove the opposite, and if many such attempts fail, confidence in the stability of the controller may increase.

In order to improve the confidence in the Fuzzy hybrid valve controller created, it was decided to subject the controller to an entirely random input signal for a long time – and examine it for signs of instability.

7.3.2. Random input signal properties

The random signal that is generated must, however, have some special properties. For instance, if the signal was created simply by selecting a random value between 0 and 12 for each new sample, the low pass filter will simply remove all the high frequencies, and the resulting request will be a signal that oscillates closely around the mean, 6 in this case. Figure 7.8 illustrates this.

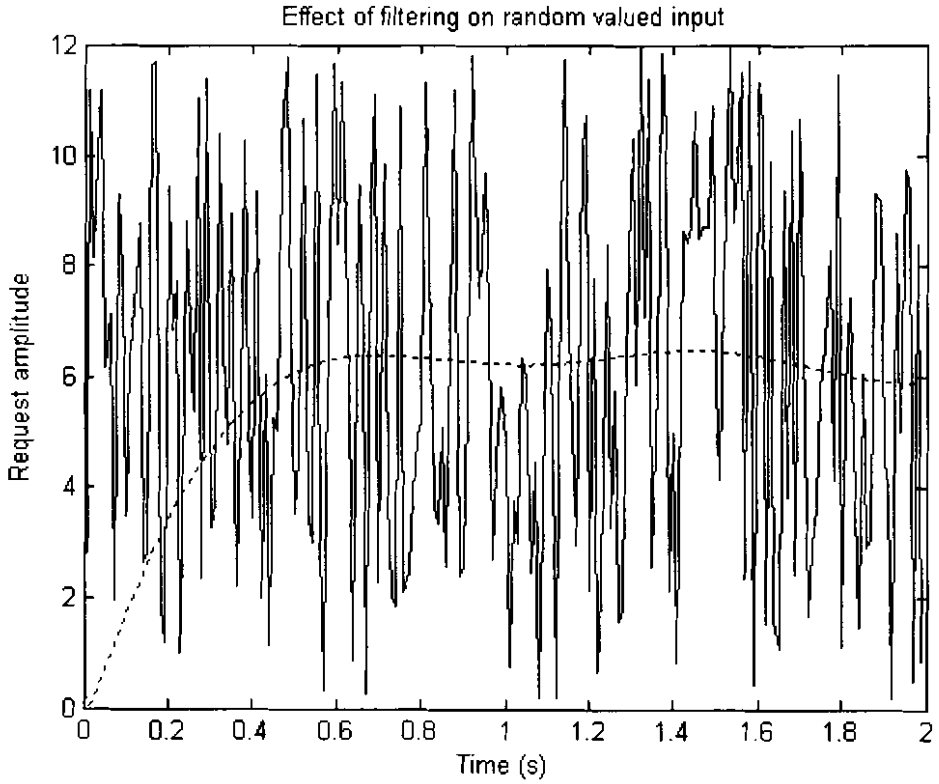

Figure 7.8 Effect of low pass filtering on a completely random input signal

It was therefore decided to apply a different strategy. The first value of the signal is chosen randomly between 0 and 12, after which each consecutive sample is generated by adding a small random

(positive or negative) value to the current one. This ensures that the resulting random signal contains a much lower frequency content, and the result of filtering does not result in an inadequate input request signal. Figure 7.9 shows a signal created by using this strategy.

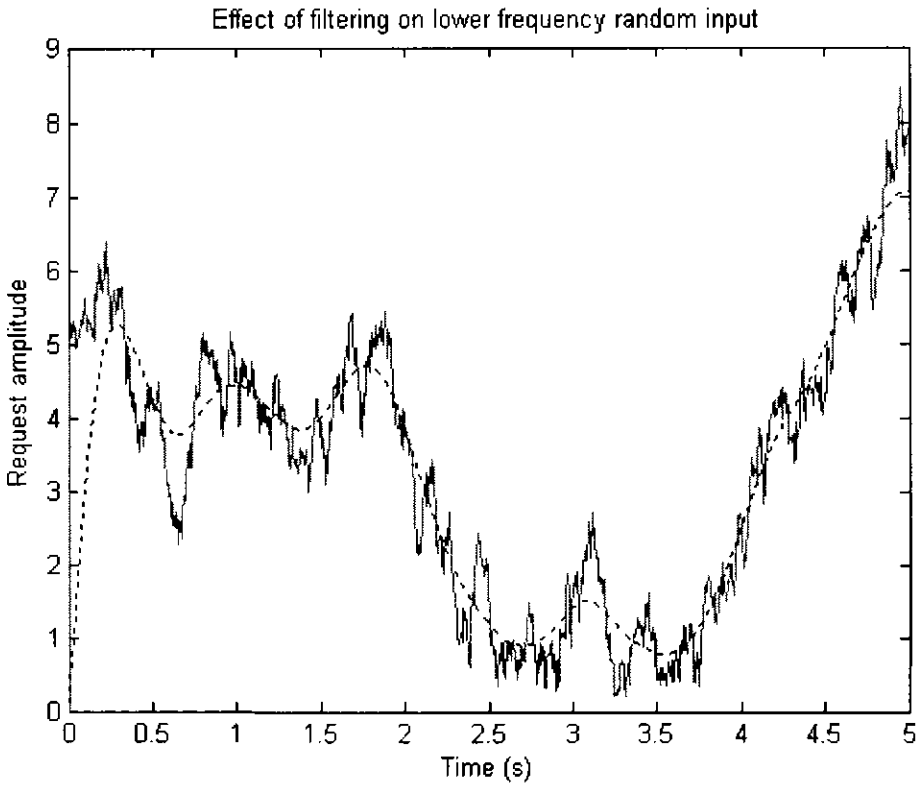

Figure 7.9 Effect of filtering on input signal created by using the different strategy

7.3.3. Results of stability test

Figure 7.10 shows the response of the controller to the filtered random input request generated. As can be seen, 1 hour of data was generated. Generating the random data took three hours of computer time, while calculating the response to the request held the computer busy for another 13 and a half hours.

The graph in Figure 7.10 is very compact, so it is consequently not possible to examine the finer detail of the response. What is possible to see, however, is that the controller never became unstable throughout the whole process. Closer inspection also revealed that no dip in mass flow caused the mass flow through the system to dip below 2 % of the request, and that the predictions were usually very accurate.

It can therefore be said that there is no reason to suspect that the controller developed for this project is unstable.

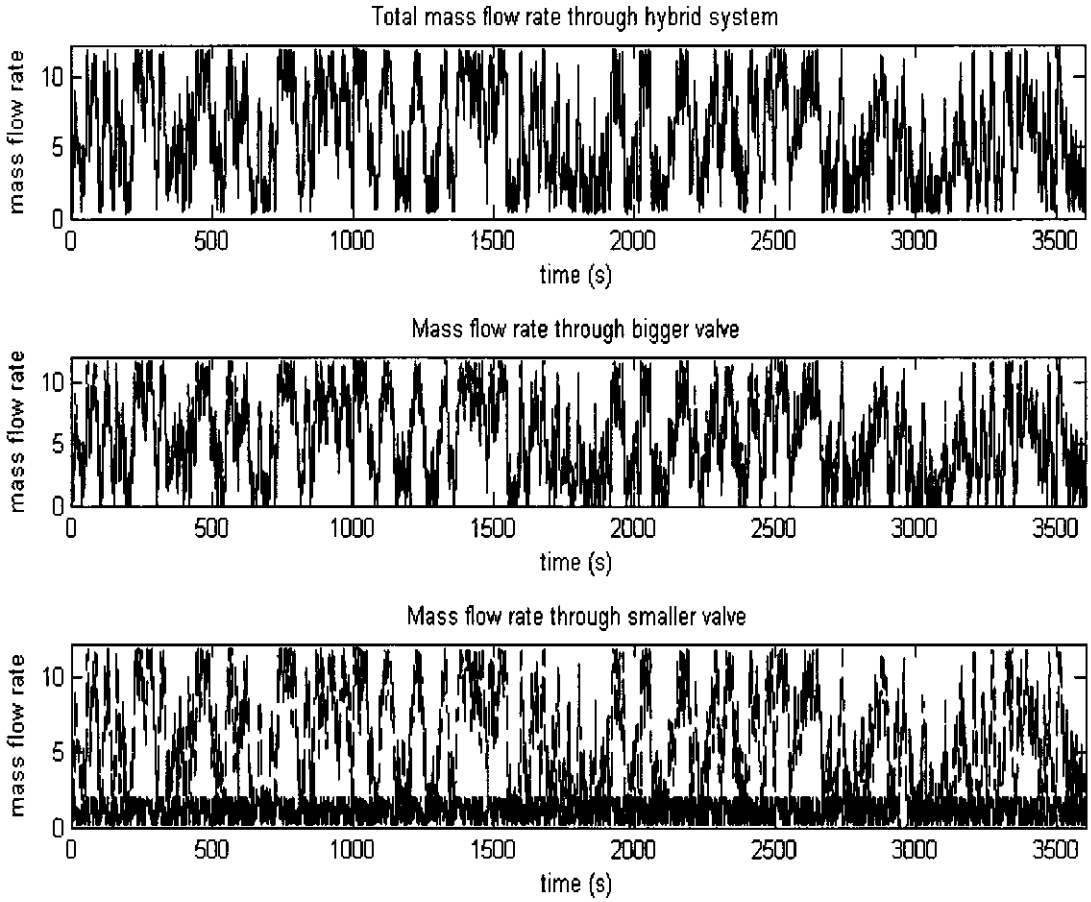

Figure 7.10 Response of controller to very long, random input signal

7.4. Conclusion

This chapter subjected the hybrid valve controller that was developed to a few thorough tests to examine if it is capable of minimising unwanted dips and jumps in mass flow while remaining stable. It was seen that complex input signals are dealt with quite well by the controller and that signals near the transition boundaries do not generate unnecessary overshoot (and no "undershoot" below 2 % of the request).

While not being able to prove without any doubt that the controller is completely stable, no signs of instability was found on the controller despite extensive testing.

8
Conclusions and recommendations

8.1. Overview

This chapter will conclude the dissertation and suggest some areas where future work may be considered.

8.2. Conclusions

This thesis discussed the process followed in designing a controller for a hybrid control valve system.

The preliminary PID based hybrid valve controller was seen to provide stable control for relatively simple input signals, but failed to impress when subjected to more complex signals. The controller nevertheless provided great insight into some of the complications associated with coordinating the two valves and was therefore very helpful in defining the actual problem statement for this project in practical terms. From its response to simple inputs it was plain to see that the final controller will have to possess the ability to give completely contrasting commands to the two valves while having some insight into each of the two valves' flow characteristics. Furthermore, from the poor reaction of the controller to more complex request signals, it could be derived that the final controller should be provided with more information about the request signal, and possess enhanced control intelligence if practical implementation is to be considered.

Fuzzy Logic is the design method selected to form the foundation of the more complex controller. The controller is equipped with neural network based prediction capabilities enabling it to minimise unwanted overshoot and avoid dips to mass flow rates below the request. The fuzzy inference system provided the necessary control intelligence and design simplification to utilise the increased number of inputs and offer the complexity of control required for this application. Furthermore, the control parameters of the Mamdani fuzzy engine could be optimised very effectively by using the random search techniques offered by genetic algorithms.

Tests on the controller have shown that it is capable of dealing with all complex request signals the PID based controller had trouble with, and remains stable without any signs of possible instability.

It is therefore concluded that the problem statement for this project has been met, since the fuzzy controller succeeds in addressing all critical outcomes identified.

8.3. Future work

8.3.1. Addressing valve stick-slip and poor large valve resolution

Valve stick-slip and poor large valve resolution have always been a problem with applications where hybrid valve models are used [28]. For economic reasons, less expensive valves are frequently implemented as the "large" valve in these configurations. As a result, the large valve often does not have the required accuracy to deal with small changes in the mass flow request (poor resolution) and tends to get stuck in one position when it is not moved continuously (stick-slip). The smaller valve is often of much better quality (since buying a small valve of good quality is more affordable) and consequently offers the required resolution for accurate control.

This problem has created the need for a hybrid valve controller that not only extends the control range of the large valve, but also maximises the amount of control performed by the quicker, more accurate smaller valve. This might be achieved by using the smaller valve to meet the high frequency changes in the request while the larger valve simply reacts to changes in the smaller valve, and moves to ensure that it stays within its operational range as far as possible.

An example of the solution in mind is shown in Figure 8.1. As can be seen from the graph, the small valve moves very quickly and oscillates through its entire range, while the larger valve simply moves when the smaller valve gets close to its maximum or minimum.

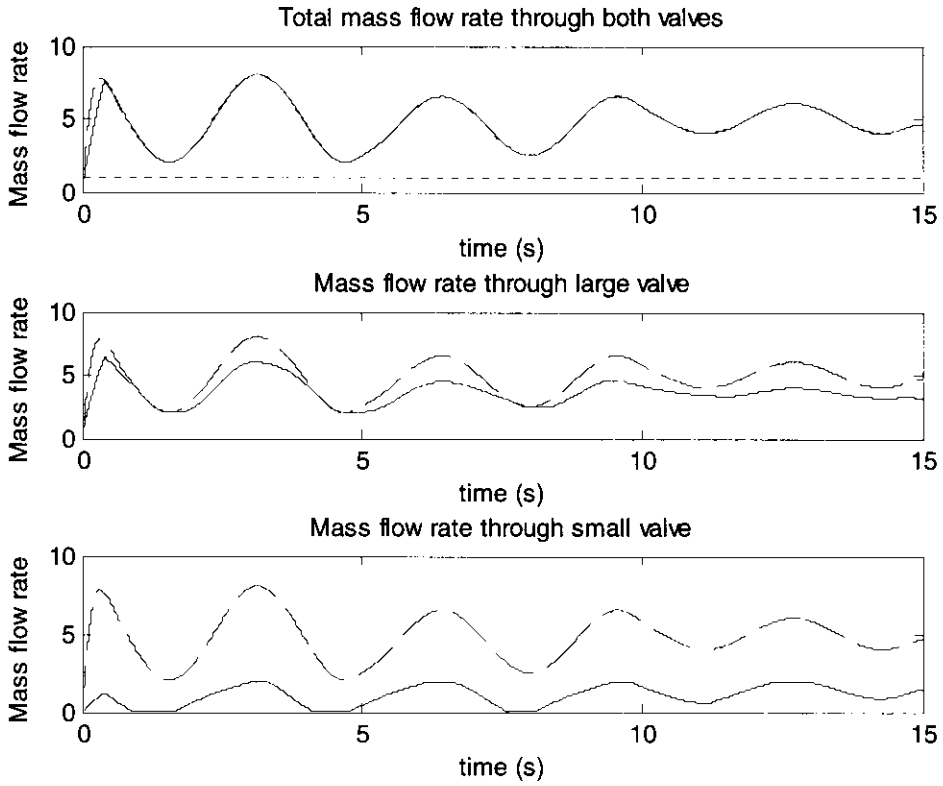

Figure 8.1 Maximising the control performed by the smaller valve

Fuzzy Logic provides an excellent platform to address these problems and develop a controller that maximises small valve control. Membership functions can be created in such a way that the large valve is moved continuously (to avoid stick-slip) but not relied upon for accuracy. These principles can be implemented with fuzzy logic in a very gradual and smooth way not possible when crisp implementations are used. For that reason this is definitely an area for which further research is required.

8.3.2. Complex linear controller

It has already been mentioned that the stability and robustness of Fuzzy Logic controllers cannot be guaranteed at present [8]. For that reason, implementation in the industry of the Fuzzy Logic based controller developed in this project seems unlikely. Industries, and especially high risk industries like the PBMR, require strict guarantees ensuring the stability of every controller implemented.

Creating a valve controller capable of comparable complex control on a linear design foundation must therefore be seriously considered. Such a controller might present numerous challenges, since

Neural Network based predictions will also be prohibited, but the design thereof is by no means impossible.

Other intelligent implementations for controlling a hybrid control valve setup already exist in the industry where model predictive controllers are implemented to simultaneously manipulate the small and big valve [27]. These controllers have been applied with great success in many applications which shows that PID control, although being very popular, is not the only acceptable linear control solution.

8.3.3. Integration into PBMR plant model

Numerous comprehensive thermo-dynamic models of the PBMR have already been created. These models enable the designer to thoroughly investigate the global effect of any new design on the plant before it is integrated into the actual system. For that reason the integration of the hybrid valve, and its controller into a thermo dynamic modelling program, like Flownex, will be of tremendous value.

Of course, implementing the controller into a more complex valve or system model will most definitely require re-optimisation of the control parameters and possibly redesigning of some of the controller's components.

8.3.4. Evaluation of controller in actual system

A further step may be taken in evaluating the hybrid valve controller namely implementing it into an actual system or physical system model. This step is usually taken after comprehensive modelling and successful implementation of the controller is considered very likely. The process is usually very expensive, but it is the only way to investigate the true effect of the hybrid valve on a system and make an informed decision on the advantages of its implementation.

8.3.5. Adjusting valve control to optimise process

The valve controller that was developed in this project was optimised in such a way as to minimise the total mean error for any input request signal. However, when a hybrid valve is installed into an actual system, the system may require different behaviour from the valve in order to perform

optimally. Genetic algorithms may offer a good solution to this problem, but it may happen that more drastic changes will be required to the controller's configuration.

9
References

[1] R.S. Gupta, *Hydrology and Hydraulic Systems*, Waveland Press, 2003

[2] J Radle, *Equipment Notebook Hpac Archive*, Hpac Engineering, Illinois, December 2000

[3] M.C. Nieuwoudt and W Kriel, *PBMR operability summary*, PBMR, Pretoria, Tech. Rep. Revision: 2D, 2001

[4] R.C. Dorf , Bishop, R.H., *Modern Control Systems.*, New Jersey, Prentice-Hall, Inc., 2001

[5] Wikipipedia, (2005 June 10), *Computer simulation* [online] Available: *http://en.wikipedia.org/wiki/Computer_simulation*

[6] G.C. Goodwin, S.F.Graebe, M.E.Salgado, *Control System Design*, New Jersey, Prentice-Hall, 2001

[7] K. Åström, T Hägglund, *PID controllers: theory, design and tuning*, Instrument Society of America, 1995

[8] L. Zadeh, "*Fuzzy Logic*", University of Berkeley, 1995

[9] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, *Genetic Fuzzy Systems*, World Scientific Publishing Co, 2001

[10] A. Bardossy, L. Duckstein. *Fuzzy Rule-based Modeling with Application to Geophysical, Biological and Engineering Systems*, CRC Press, 1995

[11] E. H Mamdani, "Applications of fuzzy algorithm for control of a simple dynamic plant". *Proceedings of the IEE 121*, pp.12-14, June 1974

[12] L. X. Wang, *Adaptive Fuzzy Systems and Control*, New Jersey, Prentice-Hall, 1994

[13] H. Ying, W. Siler, J.J. Buckley, "Fuzzy Control Theory: A Nonlinear Case", *Automatica, Vol. 26, No. 3*, pp 513-520, 2000

[14] C. M. Bishop *Neural Networks for Pattern Recognition*, Oxford University, Oxford University Press, 1995

[15] L. Lawrence, M. Jeanette, *Introduction to Neural Networks*, California Scientific software press, 1994

[16] K. Gurney, *An introduction to Neural Networks*, London: Routledge, 1997.

[17] Wikipipedia, (2006 January 11), *Artificial neural network* [online] Available: *http://en.wikipedia.org/wiki/Computer_simulation*

[18] A. Chipperfield, P. Fleming, H. Pohlheim, C Fonseca, *Genetic Toolbox*, University of Sheffield, 2004

[19] K. A. De Jong, "Analysis of the Behaviour of a Class of Genetic Adaptive Systems", Phd Thesis, Dept of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975

[20] W. M. Spears, K. A. De Jong, "An Analysis of Multi-Point Crossover", *Foundations of Genetic Algorithms* pp. 301-315, 1991

[21] G. Syswerda, "Uniform crossover in genetic algorithms", *Proc. ICgA 3*, pp 2-9, 1989

[22] R. A. Caruna, L. A. eshelman, J. D. Schaffer, "Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover", *Eleventh International Joint Conference on Artificial Intelligence*, Vol. 1, pp 750-755, Morgan Kaufmann Publishers, 1989

[23] L. Booker, 'Improving search in genetic algorithms", *Genetic Algorithms and Simulated Annealing*, pp. 61-73, Morgan Kaufmann Publishers, 1987

[24] D. E. Goldberg, "*Genetic Algorithms is Search, Optimisation and Machine Learning*", Addison Wesley Publishing Company, 1989

[25] F. Herrera, M. Lozano, J.L Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behaviour analysis",*Artificial Intelligence Review 12*, 1998

[26] Z. Michalewicz, *Genetic Algorithms + Data Stuctures = Evolution Programs*, Springer-Verlag, 1996

[27] G. McMillan (2005, August 11), "Model-predictive control can solve valve problem", *Controlglobal.com* [Online]. Available: http://www.controlglobal.com/articles/2005/533.html

[28] B. Ruggiero (2000, October 5), "Design considerations for control valves", *HPAC Engineering* [Online]. Available: http://www.hpac.com

Appendix A

The fuzzy rules that make up the rule base of the FLC that was created in this project is provided in Appendix A. The values that are shown in brackets next to each rule are the weight value of that rule. These rules will be better understood if viewed together with the membership functions of the FRBS discussed in sections 5.4.3 and 5.4.4

- 1. If (Mass_flow_error is Negative) and (Mass_flow_error_der is Positive) and (Request_signal is Large) then (Large_valve_cmd is Negative)(Large_closing is Off) (1)
- 2. If (Mass_flow_error_der is Negative) and (Request_signal is Large) then (Large_valve_cmd is Zero)(Large_closing is Off) (1)
- 3. If (Mass_flow_error is Positive) and (Mass_flow_error_der is Positive) and (Request_signal is Large) then (Large_valve_cmd is Positive)(Large_closing is Off) (1)
- 4. If (Mass_flow_error is Negative) and (Mass_flow_error_der is Positive) and (Small_closing is On) then (Large_valve_cmd is Negative) (1)
- 5. If (Mass_flow_error_der is Negative) and (Small_closing is On) then (Large_valve_cmd is Zero) (1)
- 6. If (Mass_flow_error is Positive) and (Mass_flow_error_der is Positive) and (Small_closing is On) then (Large_valve_cmd is Positive) (1)
- 7. If (Mass_flow_error is Negative) and (Mass_flow_error_der is Positive) and (Add_inp is On) then (Small_valve_cmd is Negative) (1)
- 8. If (Mass_flow_error_der is Negative) and (Add_inp is On) then (Small_valve_cmd is Zero) (1)
- 9. If (Mass_flow_error is Positive) and (Mass_flow_error_der is Positive) and (Add_inp is On) then (Small_valve_cmd is Positive) (1)
- 10. If (Mass_flow_error is Negative) and (Mass_flow_error_der is Positive) and (Large_closing is On) then (Small_valve_cmd is Negative) (1)
- 11. If (Mass_flow_error_der is Negative) and (Large_closing is On) then (Small_valve_cmd is Zero) (1)
- 12. If (Mass_flow_error is Positive) and (Mass_flow_error_der is Positive) and (Large_closing is On) then (Small_valve_cmd is Positive) (1)
- 13. If (Mass_flow_error is Negative) and (Mass_flow_error_der is Positive) and (Request_signal is Less_than_one) and (Large_closing is Off) and (Add_inp is Off) and (Small_closing is Off) then (Small_valve_cmd is Negative) (1)
- 14. If (Mass_flow_error_der is Negative) and (Request_signal is Less_than_one) and (Large_closing is Off) and (Add_inp is Off) and (Small_closing is Off) then (Small_valve_cmd is Zero) (1)
- 15. If (Mass_flow_error is Positive) and (Mass_flow_error_der is Positive) and (Request_signal is Less_than_one) and (Large_closing is Off) and (Add_inp is Off) and (Small_closing is Off) then (Small_valve_cmd is Positive) (1)
- 16. If (Mass_flow_error is Negative) and (Mass_flow_error_der is Positive) and (Request_signal is Between1and2) and (Large_closing is Off) and (Large_valve_travel is not Closed) and (Add_inp is not On) and (Small_closing is Off) then (Large_valve_cmd is Negative) (1)

- 17. If (Mass_flow_error_der is Negative) and (Request_signal is Between1and2) and (Large_closing is Off) and (Large_valve_travel is not Closed) and (Add_inp is not On) and (Small_closing is Off) then (Large_valve_cmd is Zero) (1)
- 18. If (Mass_flow_error is Positive) and (Mass_flow_error_der is Positive) and (Request_signal is Between1and2) and (Large_closing is Off) and (Large_valve_travel is not Closed) and (Add_inp is not On) and (Small_closing is Off) then (Large_valve_cmd is Positive) (1)
- 19. If (Mass_flow_error is Shooting_a_little_over) and (Small_valve_travel is not Closed) and (Add_inp is Off) and (Big_limit_down is True) and (Small_limit_down is True) then (Small_closing is On) (1)
- 20. If (Small_closing is On) then (Small_valve_cmd is Close) (1)
- 21. If (Large_closing is On) then (Large_valve_cmd is Close) (1)
- 22. If (Request_signal is Very_dangerous) and (Req_signal_der is Negative) and (Large_valve_travel is not Closed) and (Probability_lower is High) then (Add_inp is On) (1)
- 23. If (Add_inp is On) then (Small_closing is Off) (1)
- 24. If (Request_signal is Less_than_one) and (Small_valve_last_cmnd is Small) and (Add_inp is On) then (Large_closing is On) (1)
- 25. If (Large_valve_travel is Closed) and (Add_inp is On) then (Large_closing is Off)(Add_inp is Off) (1)
- 26. If (Request_signal is More_than_one) and (Req_signal_der is Positive) and (Add_inp is On) then (Add_inp is Off) (1)
- 27. If (Request_signal is Almost_above_small_valve_max) and (Req_signal_der is Positive) and (Probability_upper is High) then (Add_inp is Off) (1)
- 28. If (Request_signal is Almost_above_small_valve_max) and (Req_signal_der is Positive) and (Probability_upper is High) then (Large_closing is Off)(Add_inp is Off)(Small_closing is Off) (1)
- 29. If (Request_signal is Less_than_one) and (Req_signal_der is Negative) and (Large_valve_travel is not Closed) then (Add_inp is On) (1)
- 30. If (Mass_flow_error is Negative) and (Mass_flow_error_der is Positive) and (Request_signal is Between1and2) and (Large_closing is Off) and (Large_valve_travel is Closed) and (Add_inp is Off) and (Small_closing is Off) then (Small_valve_cmd is Negative) (1)
- 31. If (Mass_flow_error_der is Negative) and (Request_signal is Between1and2) and (Large_closing is Off) and (Large_valve_travel is Closed) and (Add_inp is Off) and (Small_closing is Off) then (Small_valve_cmd is Zero) (1)
- 32. If (Mass_flow_error is Positive) and (Mass_flow_error_der is Positive) and (Request_signal is Between1and2) and (Large_closing is Off) and (Large_valve_travel is Closed) and (Add_inp is Off) and (Small_closing is Off) then (Small_valve_cmd is Positive) (1)
- 33. If (Mass_flow_error is Big) and (Mass_flow_error_der is Positive) and (Request_signal is Between1and2) and (Req_signal_der is Positive) and (Large_closing is Off) and (Large_valve_travel is Closed) and (Add_inp is Off) and (Small_closing is Off) then (Large_valve_cmd is Positive) (1)
- 34. If (Request_signal is Between1and2) and (Req_signal_der is Negative) and (Small_valve_travel is not Closed) and (Add_inp is Off) and (Small_closing is Off) and (Big_limit_down is True) then (Small_valve_cmd is Close) (1)
- 35. If (Mass_flow_error is Negative) and (Mass_flow_error_der is Positive) and (Request_signal is Large) and (Small_valve_travel is not Closed) then (Small_valve_cmd is Negative)(Small_closing is Off) (1)

- 36. If (Mass_flow_error_der is Negative) and (Request_signal is Large) and (Small_valve_travel is not Closed) then (Small_valve_cmd is Zero)(Small_closing is Off) (1)
- 37. If (Mass_flow_error is Positive) and (Mass_flow_error_der is Positive) and (Request_signal is Large) and (Small_valve_travel is not Closed) then (Small_valve_cmd is Positive)(Small_closing is Off) (1)
- 38. If (Request_signal is Less_than_one) and (Large_closing is Off) and (Add_inp is Off) and (Small_closing is Off) then (Large_valve_cmd is Close)(Large_closing is Off)(Add_inp is Off)(Small_closing is Off) (1)
- 39. If (Request_signal is Between1and2) and (Large_closing is Off) and (Large_valve_travel is Closed) and (Add_inp is Off) and (Small_closing is Off) then (Large_valve_cmd is Close)(Large_closing is Off)(Add_inp is Off)(Small_closing is Off) (1)
- 40. If (Request_signal is Large) and (Add_inp is Off) then (Large_closing is Off)(Add_inp is Off) (1)
- 41. If (Request_signal is Not_dangerous) and (Large_closing is Off) and (Large_valve_travel is not Closed) and (Add_inp is not On) and (Small_closing is Off) then (Small_valve_cmd is Zero)(Large_closing is Off)(Add_inp is Off)(Small_closing is Off) (1)
- 42. If (Large_closing is On) or (Add_inp is On) then (Small_closing is Off) (1)
- 43. If (Request_signal is Very_dangerous) and (Req_signal_der is Positive) and (Large_closing is Off) and (Large_valve_travel is not Closed) and (Add_inp is not On) and (Small_closing is Off) then (Small_valve_cmd is Zero)(Large_closing is Off)(Add_inp is Off)(Small_closing is Off) (1)
- 44. If (Request_signal is Less_than_one) and (Small_valve_travel is Closed) and (Large_valve_travel is not Closed) then (Small_valve_cmd is Close) (1)
- 45. If (Request_signal is Between1and2) and (Small_valve_travel is Closed) and (Large_valve_travel is not Closed) then (Small_valve_cmd is Close) (1)
- 46. If (Request_signal is Large) and (Small_valve_travel is Closed) then (Small_valve_cmd is Close)(Small_closing is Off) (1)