To the Graduate Council:

I am submitting herewith a dissertation written by Andreas Ikonomopoulos entitled "Virtual measurements using neural networks and fuzzy logic for complex system monitoring." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Robert E. Uhrig, Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

VIRTUAL MEASUREMENTS USING NEURAL NETWORKS AND FUZZY LOGIC FOR COMPLEX SYSTEM MONITORING

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Andreas Ikonomopoulos

December, 1992

DEDICATION

To all teachers, professors and advisors,

who introduced me to the quest for knowledge.

ACKNOWLEDGMENTS

The major driving intellectual and moral force behind this project has been my advisors Professors Robert Uhrig and Lefteri Tsoukalas. Their skill and foresight provided the stimulation and courage required to navigate through the unclear waters of this work. They devoted continuously time and patience in the most gracious and helpful manner I have come to know. They have my gratitude ever for.

I would like to thank my dissertation committee members, Professor Ohannes Karakashian of the Department of Mathematics, and Professors Rafael Perez and Belle Upadhyaya of the Department of Nuclear Engineering. Their contribution to the present work has been most valuable. Many thanks are due to the Head of the Department of Nuclear Engineering Professor Tom Kerlin for his valuable support throughout the period of my studies in the department. Also, I would like to thank Mr. James Mullens from ORNL for providing the real-world context for implementing and testing the ideas in this research.

This project would not have been possible without the larger community of fine friends and colleagues in Knoxville. Many thanks are due to all of them and in particular to the colleagues of the poker club. So many thanks to the faculty and staff of the Department of Nuclear Engineering, particularly, Professor H. L. Dodds, Mrs. Lynnetta Holbrook, and to all the many fine colleagues there, especially, Israel Alguindigue, Benan Basoglu and Tanju Sofu.

Finally, I would like to thank Miss Efi Pilarinu for her constant encouragement and invaluable help in the completion of this dissertation. The long discussions with her concerning the subtle meanings of some of the ideas incorporated in this work, along with her editorial comments on the first drafts of this dissertation are most appreciated.

ABSTRACT

A new approach is presented that demonstrates the potential of pretrained artificial neural networks (ANNs) as generators of membership functions for the purpose of monitoring nuclear reactor systems. ANNs provide a complex-to-simple mapping of reactor parameters in a process analogous to that of measurement. Through such **virtual measurements** the value of parameters with operational significance, e.g., *valve-position*, *transient-type* or *performance* can be determined. In the proposed methodology the output of a virtual measuring device is a membership function which uniquely and unambiguously represents the state of the system. Utilizing a fuzzy logic representation offers the advantage of describing the state of the system in a condensed form, developed through linguistic descriptions and convenient for application in monitoring, diagnostics and generally control algorithms.

The membership functions generated from independently firing networks, belong to a set of prototypes that describe the universe of discourse. A computational technique based on the dissemblance index between two fuzzy numbers, has been devised for discriminating amongst different network responses. Information criteria have been utilized for best model selection in the development of a rule-based diagnostician that incorporates the time dimension in the stationary behavior of the ANNs. The rule-based system is supplemented with fuzzy algebraic techniques forming an optimization algorithm capable to continuously adjust the membership functions describing the system. The optimization algorithm modifies appropriately the shape and position of the resultant membership functions, following changes in the operating environment.

The proposed methodology is applied to the problems of measuring the disc-position of the secondary flow control valve of an experimental reactor and transient identification in a Nuclear Power Plant (NPP). The results obtained clearly demonstrate the enhanced noise tolerance of the methodology as well as its unique robustness in fault signals. The virtual measuring methodology has been proven capable of identifying the exact system state even in the extreme case, where input signals have been substituted for random time series. The model behavior shows that it is possible to develop virtual measuring devices through artificial neural networks mapping dynamic time series to a set of membership functions, and thus enhance the capability of monitoring systems. Furthermore, the application of optimization algorithms on the neural network responses resolves the issue of designing time-varying membership functions suitable for coping with changes in the operation environment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The main purpose of this doctoral dissertation is to develop a methodology for virtual measuring devices, which will measure dynamic variables that can not be monitored directly (*virtual measurements*) in a complex system such as a nuclear power plant, and predict their future values. The variables will represent actual system parameters and linguistic evaluations of a complex system's general behavior.

The ability to understand the surrounding environment is strongly governed by the ability to recognize temporal patterns (Sung and Priebe, 1988). Temporal patterns can be considered as series of spatial patterns in a form analogous to a movie clip, where spatial patterns are mainly analogous to a single frame of a movie. The order in which events occur may be of higher priority than the events themselves, and an intelligent system must be able to detect and understand this ordering. The time dimension permits access to particular information concerning the past, the present, and most importantly, the future (Rosen, 1985b). Henceforth, every system able to incorporate this dimension into its data acquisition and decision processes may be able to perform advanced operations like recognition of sequence of events, discrimination between cause and effect, and prediction based on logical inferences from past and present.

In order to monitor the performance of a complex system, such as a nuclear power plant, it is necessary to subdivide our attention to the performance of individual

components composing it (Gottinger, 1983). A set of procedures is required to recognize whether the values of various parameters are within normal, off-normal and in general desirable or undesirable ranges (Zadeh, 1973). The parameters to be monitored are typically specific to a particular system, often the outputs of sensors and meters (Pattee, 1977). For example, in the case of a motor-operated valve, the voltage, current, and breaker position are monitored, but it is the gate-position that is of interest. While monitoring the performance of a steam turbine driven pump attention is required to the indications of speed, pressure, and stop valve position. Generally the notion of **performance**, and its evaluations such as **normal, desirable, undesirable**, etc., are quantified in terms of set points and ranges (Dubois and Prade, 1980). Yet, in the course of operations these evaluations are imputed with meanings that vary not only with the history of a particular system or equipment but also with different operators and the state of the plant as a whole.

A *virtual measuring* device is a software-based instrument for the measurement of user-specified dynamic variables with operational significance (Tsoukalas and Ikonomopoulos, 1991b). Usually these variables can not be measured directly. Their measurement may contain significant time lags and in general, the failure of a particular sensor requires that a variable be computed from other system parameters (Ikonomopoulos, et. al., 1992a). A major characteristic of virtual instruments is that they are software based, rather than hardware based. Henceforth their function can be easily modified by programming the software, not altering hardware.

The process of measurement inherently contains procedures for mapping complex input patterns to simple output patterns (Pattee, 1986). Neural networks have been known for their ability to map geometrically complicated spatial or spatiotemporal patterns to

simple, algebraically defined, representations (Wieland and Leighton, 1987). In this context, ANNs may be used for mapping complex input spatiotemporal variables to a simplified set of membership functions representing the values of a fuzzy variable (Tsoukalas, Ikonomopoulos and Uhrig, 1991a). They may reproduce uniquely defined membership functions that unambiguously represent the values of fuzzy variables such as *performance*, *risk*, *safety*, and *availability* (Zadeh, 1965).

The two fundamental requirements for useful measurements are: (a) the **precision** and (b) **reproducibility** of the input-output relation and of the functional value of the entire operation of the system performing the measurement (Pattee, 1986). The requirement of reproducibility means that the measuring device must be **isolatable** from the system being measured. Furthermore, it has to be **resettable** so that the measurements can be repeated an arbitrary number of times and give the same output for the same input pattern. Apart from that, the notion of measurement itself involves the requirement of temporal pattern recognition. In any non-trivial environment in which a "virtual measuring" device will function, the entire range of spatiotemporal information becomes available. In most cases it is imperative to process the information with respect to its relative time-order.

Neural networks have been proven capable of meeting all of the above requirements. Neural computing memory, apart from being *distributed*, is also *associative* (McClelland and Rumelhart, 1986). If the trained network is presented with a partial input, the network will choose the closest match in memory to that input, and generate an output which corresponds to a full input.

In the framework of generalized measurements, neural networks may be used not only as classifiers but also as predictors, a function of utmost significance, in the context of the anticipatory paradigm (Rosen, 1985b). Neural networks may calculate responses in a time frame faster than real time utilizing their unique ability to be trained in a user-defined environment. In this regard the ANN response may offer predictions about a future state, required in every anticipatory system. The basis of any anticipatory system is that it changes state according to present as well as future states (Rosen, 1985b). The operator of a power plant for example, can be considered as part of the entire plant system, thus rendering the system as anticipatory (Gottinger, 1983). In the anticipatory paradigm a control action taken by the operator, is decided on the basis of not only the current measurements but also estimates of what the future condition of the system might be. Any anticipatory process contains an internal model of the system being monitored, and this model computes an estimate of the future faster than real-time, in order to influence the actions taken at the present . In the proposed work neural networks are used in order to provide a model performing faster than real time computations.

Fuzzy-logic-based control and expert systems have been shown to be highly successful, reliable and superior in performance to conventional systems (Zimmerman, 1985 and Zadeh, 1988). Utilizing a fuzzy logic representation offers the advantage of describing the state of a system in a condensed form, developed through linguistic descriptions and convenient for application in monitoring, diagnostics and generally control algorithms (Zadeh, 1973). A major drawback encountered in its implementation is the difficulty of generating the fundamental tool of fuzzy computing, the membership functions. Furthermore, membership functions, once posed, remain the same throughout the operational life of the implementation and never follow changes in the reasoning environment. In the proposed approach, ANNs are used to map a set of spatiotemporal

patters representing the state of a nuclear system to a set of membership functions that describe the values of a fuzzy variable representing a particular system parameter. Thus, the calculational abilities of neural networks are coupled with the representational advantages of fuzzy logic. Additionally, a rule-based system has been developed for manipulating the prototype membership functions computed from the neural networks. The rule-based system is supplemented by fuzzy algebraic techniques in order to form an optimization algorithm. The purpose of the unified code is to identify the appropriate number of fuzzy values required to best describe the state of the system at every time step, their shapes as well as positions at the universe of discourse. The resultant membership function uniquely and unambiguously defines the state of the system at every instant in time. The optimization algorithm will respond to possible changes at the operational environment by properly modifying the membership functions. Based upon the anticipatory paradigm, the decision-making process takes into consideration the system behavior during different instants in time and draws a conclusion concerning the present time step. The above procedure renders the universe of discourse dynamic and thus capable to follow changes in the operating environment.

The original contribution in this work is focused on the development of a methodology for measuring user-defined variables with operational significance. These are variables which can not be directly monitored and need to be inferred from other system parameters. In order to accomplish this task, two original ideas are implemented. The first one is the utilization of artificial neural networks as membership function generators. In addition, a novel methodology for optimizing prototype membership functions has been implemented and tested. The methodology is based on an optimization algorithm capable of modifying membership functions. These two innovations resolve the major problems encountered in the application of fuzzy reasoning. The generation as well as "fine-tuning"

of the membership functions required for every fuzzy logic implementation. Finally, the integrated methodology offers the application environment required for the development of anticipatory systems capable of performing computations in a time-frame faster than real time.

In Chapter 2 a literature review of the issues pertaining to the present work is presented. A brief historical review on predictive models is followed by a more analytical discussion of artificial neural networks. The back-propagation algorithm is discussed in more detail since it has been used for training the networks utilized in the present work. The necessity for more relaxed (fuzzy) approaches to issues related to complex systems is presented through a discussion on fuzzy sets.

An effort to visualize a more rigorous, from the mathematical point of view, formalization of measurement theory is attempted in Chapter 3. The basic issues introduced and analyzed in this chapter have initially been developed by Robert Rosen and later expanded by John Casti and Howard Pattee. The main point of this discussion is focused on the necessity for virtual measurements as well as their prerequisites and possible applications in a complex system environment.

In Chapter 4 an analytical description of the model developed for performing virtual measurements is presented. The application environment is that of an experimental nuclear reactor and the monitored variable is the position of the secondary flow control valve. The individual constituents of the measuring device are presented and analyzed in great detail. The model performance is compared to the crisp values of the monitored parameter for the duration of the start-up.

Following the model completion we examine it through a number of tests presented in Chapter 5. The numerous test scenarios include the addition of noise in the input time series as well as the elimination of input signals. The virtual measuring device responses when presented with novel stimuli corresponding to different fuel cycle start-ups are also presented. The model behavior in the above tests is analyzed from the point of view of the statistical properties of the monitored parameter.

In Chapter 6 the possible application environment of such a measuring technique is explored and different implementations in the form of nuclear power plant (NPP) transient identification are also examined. The developed methodology is seen through the prism of anticipatory systems and the possible role such a software-based tool may play in the anticipatory paradigm, is discussed.

Finally in Chapter 7, a number of conclusions from this work is drawn and the required steps, in a form of a general algorithm, for constructing a virtual measuring device are presented. In addition, some of the subtle issues of this work which need further research are discussed, and a framework for future work is given.

CHAPTER 2

LITERATURE REVIEW

Predictive Models

Astronomy and weather forecast are typical examples of the two types of scientific prediction applied today. In the first case, we know the initial configuration of the system and its dynamic laws. Thus we can make projections concerning the state of the system in the future. In the latter, which is the case of stochastic prediction, we know nothing but the past history of the system. From this, we may only extract the statistical properties of the phenomenon in order to make a forecast, which is always of limited time and of limited reliability (Gabor, 1967).

One of the main hypotheses in predicting the future, is that conclusions about a future event - or series of events - can be made on the basis of *study*, *analysis*, and *generalization* of preceding experience. Yanase (1970) subdivides the essence of probability in stochastic prediction, into two different categories. The probability of the outcome of a die is always one sixth, which is a *prior* probability, since its foundations lie in symmetry considerations. In the case of weather forecasting, we calculate relative probability from past statistical data of similar weather. This is a *posterior* probability.

The problem of designing an optimum linear predictor was first conceived by Kolmogoroff in 1942 and shortly afterwards was solved by Wiener (Gabor, Wilby and

Woodcock, 1961). Kolmogoroff proposed the most general formula for predicting a future value of a stationary random signal, f, (Ivakhnenko and Lapa, 1967):

$$g[f(t)] = r_o + \sum_0^n r_{n_1} f_{n_1} + \sum_0^n \sum_0^n r_{n_1 n_2} f_{n_1} f_{n_2} + \sum_0^n \sum_0^n \sum_0^n r_{n_1 n_2 n_3} f_{n_1} f_{n_2} f_{n_3} + \ldots \qquad (2.1)$$

In this approach we use the values of the continuous signal $f(t)$, as a set of discrete times in the past, to predict the value of $f(t)$ at an instant in the future. The predicted value is a linearly weighted sum of the past (delayed) $f(t)$ values. In order to minimize the mean-square error between the predicted and the actual value, we equate all the partial derivatives of the mean-square error - with respect to the weighted coefficients r_i - to zero. The set of equations obtained, are the *normal equations of regression.* Since the number of equations obtained is equal to the number of Kolmogoroff's expansion terms, solving them gives the "optimal" values for the weights r_i.

Though Wiener (Schetzen, 1980) devoted much of his time to the problem of non-linear processes, the issue of non-linear filter proved to be mathematically non-tractable. Wiener (1958), Zadeh (1953), Kalman (1957), along with many other prominent scientists, have carried out much brilliant work to clarify the nature of the problem, but it had been clear that even if a formal solution could be found, it would be of little use. On the other hand, the problem has never ceased to be attractive. While the application of the non-linear filter as a predictor was at once realized by Kolmogoroff and Wiener, it was Zadeh who first saw that a universal filter could also be used as a recognizer. Since noise in a signal is anything that is not wanted, constructing a machine for that specific signal requires only the elimination of noise.

A major boost in the area of predicting machines was given with the introduction of self-adjusting predicting filters, by Denies Gabor (1961). The **Gabor Polynomial Predictive** method is a straightforward extension of Kolmogoroff's formula for the minimum mean-square error. The mean-square prediction error as a function of the weighted coefficients is a multidimensional elliptical paraboloid whose apex is the search goal. Gabor used the extrapolation method for searching for the minimum of his self-adjusting filter. The filter was constructed as an analog device, using piezomagnetic multipliers. One use of this device was in predicting the amplitude of the ocean waves. In this case the prediction accuracy obtained was of the order of several percent (Gabor 1961). The major deficiency of Gabor's methodology lies in the utilization of many delayed values in order to increase the prediction accuracy. A quick glance at Equation (2.1) is sufficient to ensure that a small increase in the number of time lags will be accompanied by a significant growth of the number of cross terms in Kolmogoroff's formula.

At about the same time Frank Rosenblatt (1962) and his group were working on self-organizing brain-like algorithms with both forced and spontaneous learning. They concentrated on networks called **perceptrons**, in which independent processing units were organized into layers with feed-forward connections between layers with adjustable weights. For the simplest class of perceptrons without any intermediate layers, Rosenblatt proved the convergence of a learning algorithm that changed the weights iteratively to perform the desired computation.

However, there was a problem in the learning algorithm, pointed out by Minsky and Papert (1969). The theorem was applicable, only to those problems which the structure was capable of computing. Minsky and Papert showed that Rossenblatt's two-layer perceptron was unable to perform certain elementary computations. Out of 16

Boolean operations only two could not be performed, the **exclusive-or** (**XOR**) and **exclusive-nor** (**XNOR**). Rosenblatt had also studied structures with more layers and believed that they could overcome the limitations of the simple perceptrons (Hertz, Krogh and Palmer, 1991). However, there was no known algorithm which could be implemented to perform the allocation of weights between layers. Minsky and Papert doubted that one could be found and proposed the exploration of other technical areas. The power of Minsky's personality, as well as his scientific reputation, were sufficient to guarantee the retardation of neural computing for the next thirteen years.

Artificial Neural Networks

After thirteen years of near eclipse, interest in artificial neural networks has grown rapidly over the past decade. Professionals from such diverse fields as engineering, physical sciences, philosophy, physiology, and psychology are intrigued by the potential of this new technology and are seeking applications within their disciplines. Part of this rapid growth was the result of technological advances in personal and main-frame computing, enabling neural network investigators to simulate and test ideas in ways not readily possible before 1980.

Another major impulse came from Hopfield's (1982) work on neural networks with symmetric connections. Such networks were previously dismissed as not brain-like and therefore deviating from the scope of neural computing. Hopfield's papers triggered an explosion, particularly in the statistical physics community, leading to a whole series of dramatic advances in the understanding of symmetric networks and their properties. Special attention has been given to their utility as distributed memory stores, and as solvers

of constrained optimization problems, e.g. versions of the famous Traveling Salesman Problem (Hertz, Krogh, and Palmer, 1991).

At about the same time, another more important development in the neural networks area, was taking place. The publication by Rumelhart, Hinton, and Williams (1986), of the well-known back-propagation algorithm for solving the fundamental problem of training neural networks to compute desired functions. This problem was first formulated by Rosenblatt in the early 1960's in his classical work on perceptrons (Rosenblatt, 1962).

Work, both theoretical and practical, by numerous scientists and engineers, has shown that it is possible to apply computation to realms previously restricted to human intelligence. It seems feasible to make machines learn and remember in ways that bear a striking resemblance to human mental processes and thus give a new and significant meaning to the much abused term *artificial intelligence* (Wasserman, 1989).

Artificial neural networks are biologically inspired; that is, they are composed of elements that perform in a manner that is analogous to the most elementary functions of the biological neuron. These elements are then organized in a way that may (or may not) be related to the anatomy of the brain. Despite this superficial resemblance, artificial neural networks exhibit a number of the brain's characteristics. For example, they learn from experience, generalize from previous examples to new ones, and abstract essential characteristics from inputs containing unrelated data (Khanna, 1990).

Artificial neural networks can modify their behavior in response to their environment. This factor, more than any other, is responsible for the interest they have received. Presented with a set of inputs (in some cases with desired outputs also), they

self-adjust to produce consistent responses. This process is known as *learning*. A wide variety of learning algorithms has been developed, each one with its own strengths and weaknesses. It is very important to note that certain training algorithms are eligible to specific knowledge. It is very common to refer to the neural computing memory as *distributed* (McClelland and Rumelhart, 1986). The term is borrowed from the networks characteristic to store the information acquired through the learning process, in the weights of the artificial connections among the independent processing units (neurons).

Once a neural network has been trained, its response can be, to a degree, insensitive to minor variations in its input. This ability to handle noise and distortion in the pattern that lies within, is vital to pattern recognition in a real-world environment. Overcoming the numerical stiffness of conventional computers, it produces a system that can cope with the fuzzy environment we deal with. This property is called *generalization* by many authors. It is important to note that artificial neural networks generalize automatically as a result of their structure, and not by using human intelligence embedded in the form of ad hoc computer programs. It is this very property of generalization that makes the neural memory *associative*. In simple terms, associativity depicts the property of artificial neural networks to calculate a correct, to a degree, response based on non-exact information. This characteristic is a direct analog to human intelligence and cognition. As an example, a first grader who has been taught to add five with seven and equate the summation to twelve, is capable of *associating* the summation of fifty with seventy , to a number very close to one hundred and twenty.

Particular neural network architectures are capable of abstracting the essence of a set of inputs. This property is called *abstraction*. As an example, consider a network trained on a sequence of distorted versions of the letter **A**. After adequate training, application of

such a distorted example will cause the network to produce an answer that is closest (in a least mean square sense) to the example of the training variable (Wasserman, 1989). This property sometimes - falsely - called *extrapolation.*

The first type of networks studied 30 years ago by Rosenblatt (1962) and his colleagues was called **perceptron** and belongs to a category of networks, widely known as *feed-forward* networks. Figure 2.1 shows an example of a feed-forward network architecture. There is a set of input neurons whose only role is to feed input patterns into the rest of the network. Following this layer there may be one or more intermediate computing layers (here only one is shown), followed by a final output layer where the result of the computation is read off. In feed-forward networks there are no connections from one unit to any units in previous layers or to any units in the same layer and, moreover, to layers more than one level ahead. In summary every unit feeds only the units in the next layer. The units in the first layer (fan-out units) are called **input units**, and the units in the last layer are called **output units**. The units in the intermediate layer are sometimes called **hidden units** because they have no direct connection to the outside world, neither input nor output.

Cybenko (1989), recently proved what is considered as the most significant characteristic of feed-forward networks. Cybenko's work shows that sufficiently complex multilayer feed-forward networks are capable of approximating an unknown mapping f: $\Re' \rightarrow \Re$ arbitrarily well. A year later Hornik, Stinchcombe and White (1990) and White (1990) proceeded one step further and proved that multilayer feed-forward networks with as few as a single hidden layer and an approximately smooth hidden layer transfer function are capable of arbitrarily accurate approximation of an arbitrary function and its derivatives. In fact, these type of networks can approximate functions that are not differentiable in the

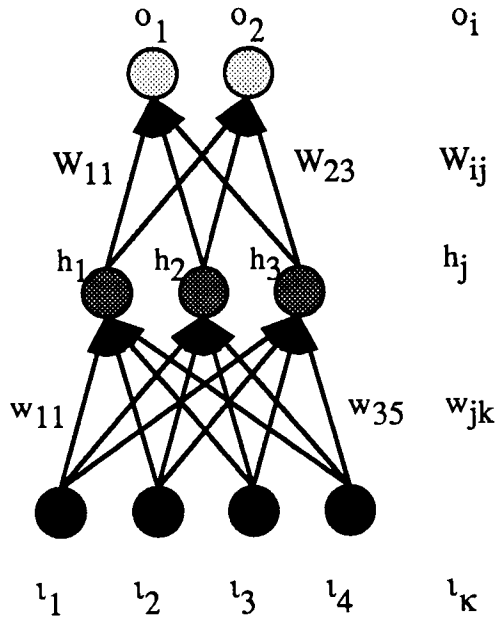

Figure 2.1 A three layer feed-forward network architecture.

classical sense, but possess only a generalized derivative, as in the case of certain piece wise differentiable functions.

The most successful learning algorithm devised for feed-forward architectures is back-propagation. It was invented independently several times, by Bryson and Ho (1969), Werbos (1974), Parker (1985), and Rumelhart, Hinton, and Williams (1986). Back-propagation is a learning algorithm for neural networks that seeks to find weights, w_{pq}, such that given an input pattern from the training set of pairs of input/output patterns, the network will approximate the output of the training set, given the input. Having learned this mapping between input and output from the training set, one can then apply a previously unseen input, and obtain the proper output from the neural network. The network response is based on the network's ability to learn the fundamental relationship between input and output from the training patterns.

The basis of the training algorithm is steepest descent and is closely related to multiple regression (Werbos, 1988). In order to acquire an overall description of the back-propagation algorithm consider the notational convention illustrated in Figure 2.1. The input units are denoted by ι_k, hidden units by h_j, and output units by o_i. There are connections w_{jk} from the input units to the hidden units, and W_{ij} from the hidden units to the output units (Hertz, Krogh and Palmer, 1991), (Lapedes and Farber, 1988). Different input patterns are labeled by the superscript μ, and the input element ι_κ^μ can be binary or continuous-valued.

Given the pattern μ, a hidden unit j receives a net input

$$v_j^\mu = \sum_k w_{jk} \iota_k^\mu + \theta_j \tag{2.2}$$

where θ_j is a user-defined arbitrary number, called *bias*. The hidden unit j produces an output

$$h_j^\mu = g(v_j^\mu) = g\left(\sum_k w_{jk} \iota_k^\mu + \theta_j \right) \tag{2.3}$$

where $g(x)$ is a nonlinear function that is often chosen to be of a sigmoidal form. For example we may choose:

$$g(v_j^\mu) = \frac{1}{1+e^{-(\sum_k w_{jk} \iota_k^\mu + \theta_j)}} \tag{2.4}$$

The output unit i thus receives as an input

$$v_i^{\mu} = \sum_j W_{ij} h_j^{\mu} + \theta_i = \sum_j W_{ij} g\left(\sum_k w_{jk} \iota_k^{\mu} + \theta_j \right) + \theta_i \tag{2.5}$$

and produces final output equal to

$$o_i^{\mu} = g(v_i^{\mu}) = g\left(\sum_j W_{ij} h_j^{\mu} + \theta_i \right) = g\left(\sum_j W_{ij} g\left(\sum_k w_{jk} \iota_k^{\mu} + \theta_j \right) + \theta_i \right) \tag{2.6}$$

If t_i^{μ} is the target output value for the μ input pattern, the error measure, or *cost function*, is calculated as

$$E[\mathbf{W}] = \frac{1}{2} \sum_{\mu} \sum_i \left[t_i^{\mu} - o_i^{\mu} \right]^2 \tag{2.7}$$

which may be rewritten according to (2.6) as:

$$E[\mathbf{W}] = \frac{1}{2} \sum_{\mu} \sum_i \left[t_i^{\mu} - g\left(\sum_j W_{ij} g\left(\sum_k w_{jk} \iota_k^{\mu} + \theta_j \right) + \theta_i \right) \right]^2 \tag{2.8}$$

The function defined by equation (2.8) is clearly a continuous differentiable function in every weight, and thus we can use a steepest descent algorithm to obtain the appropriate weights. In one sense this is all there is to back-propagation, and the so-called *delta rule,* for the minimization of the error is one more numerical way to implement numerical regression, a well-known statistical method (Werbos, 1988). For the hidden-to-output connections the gradient descent rule gives

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} = \eta \sum_{\mu} \left[t_i^{\mu} - o_i^{\mu} \right] g'\left(v_i^{\mu} \right) h_j^{\mu} = \eta \sum_{\mu} \delta_i^{\mu} h_j^{\mu} \tag{2.9a}$$

$$\Delta\theta_i = -\eta \frac{\partial E}{\partial \theta_i} = \eta \sum_{\mu} \delta_i^{\mu} \tag{2.9b}$$

where we have introduced the user-specified constant, η, called *learning rate*, and we have defined

$$\delta_i^{\mu} = g'\left(v_i^{\mu}\right)\left[t_i^{\mu} - o_i^{\mu}\right] \tag{2.10}$$

Equations (2.9a, b) imply that $\Delta E < 0$ and hence E will decrease to a local minimum. For the input-to-hidden connections, we have to differentiate with respect to w_{jk}'s which are more deeply embedded in (2.8). Using the chain rule gives

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \sum_{\mu} \frac{\partial E}{\partial h_j^{\mu}} \frac{\partial h_j^{\mu}}{\partial w_{jk}} = \eta \sum_{\mu} \delta_j^{\mu} \iota_k^{\mu} \tag{2.11}$$

where

$$\delta_j^{\mu} = g'\left(v_j^{\mu}\right) \sum_i W_{ij} \delta_i^{\mu} \tag{2.12}$$

Equation (2.12) allows us to determine δ for a given hidden unit h_j in terms of the δs of the output units. The coefficients are the forward propagated W_{ij}'s, with the only difference that they propagate the errors (δs) backwards. Therefore following the iterative procedure described for minimizing E, we calculate the weight change, for every unit in the hidden layer m as

$$\Delta w_{ij}^{m} = \eta \delta_i^{m} h_j^{m-1} \tag{2.13a}$$

Most back-propagation algorithms include an extra term in equation (2.13a) which represents the importance of previous weight changes to the current weight change, i.e.

$$\Delta w_{ij}^{m}(t+1) = \eta \delta_i^m h_j^{m-1} + \alpha \Delta w_{ij}^{m}(t) \tag{2.13b}$$

where, t is the iteration number, and α is called the *momentum term*. Therefore all connections are updated according to

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij} \tag{2.14}$$

in an iterative effort to minimize the error E as it has been defined in equation (2.7). It should be noted that the use of sum square error and steepest descent in model estimation have been established decades ago. Therefore, the novel feature of back-propagation in this formulation is the dynamic feedback approach (Werbos, 1988). Furthermore, the commonly used steepest descent procedure is very slow in simulation and a better minimization procedure, such as the classic conjugate gradient procedure (Press, 1986), can offer quite significant speed up. Furthermore, Lapedes and Farber (1987) along with many other authors, have found that the dynamic range of the input values is not so critical to the network performance. However, associated numerical problems may be avoided by scaling the input and the output to the interval [0,1], training the network, and then rescaling the w_{ij}'s and θ_i's to encompass the original dynamic range. As we have discovered from personal experience in feed-forward networks (Ikonomopoulos, Uhrig and Tsoukalas, 1991a), scale changes in the input and output values will always be absorbed in w_{ij}'s, θ_i's. This procedure has also been implemented in the present work.

Based on these theoretical developments a lot of attention has been shifted to the utilization of neural networks as floating point number processors operating in a massively parallel fashion. Previous research interest had mainly been focused on the abilities of artificial neural networks to mimic "qualitative reasoning" by manipulating neural encodings of symbols. It is this very property of neural networks to perform arbitrary mappings which is heavily utilized in the present work. The use of neural networks as membership function generators is based on the network ability to map a set of time dependent signals into the universe of discourse of human linguistics (Tsoukalas, Ikonomopoulos and Uhrig 1991a). The prerequisite for accurate representation is satisfied from the complexity of the network architecture.

In closing the remarks on neural network formalism, it is important to refer to a somewhat different aspect of neural network performance. Lapedes and Farber (1987) have proved that feed-forward networks - using sigmoid transfer functions - perform a generalized mode decomposition of the underlying map into sums of Walsh functions. Thus by changing the neurons transfer function from sigmoids to sinusoidals the analysis changes to a generalized Fourier analysis, where the neural network constructs a discrete Fourier series. However, in contrast to a normal Fourier decomposition in which the values of the frequencies are **fixed**, the network has the ability to **adjust** the values of the frequencies to obtain the minimum error. The number of adjustable frequencies is determined by the number of neurons in the hidden layer. Thus, by specifying the number of hidden units, we determine the frequencies available to the network. The network based on this configuration adjusts the numerical value of these frequencies, along with their amplitude and phases, in order to produce the best fit. Presumably, by adding more hidden units, i.e., by adding more frequencies, the network accuracy will be increased. Lapedes and Farber call the mode decomposition performed by the neural networks as "Generalized

Fourier Decomposition", since in the conventional Fourier mode analysis only the amplitudes are adjusted and the frequencies are fixed.

Fuzzy Sets

The continuous human quest for precision and exactness has developed scientific methodologies that reject the fuzziness of concepts in natural use and replace them with non-fuzzy scientific *explicata* by a process of *precisiation*. This philosophy contradicts the human inference procedure. Human reasoning does not rely upon long chains of inferences supported by intermediate data. On the contrary, it accepts minor contradictions and incorporates their effects. In this fashion universal inferences are not driven from such events. Almost 70 years ago, Bertrand Russell (1923) argued that traditional logic requires the use of precise symbols, "... therefore (is) not applicable to this terrestrial life but only to an imagined celestial existence ...". The determination of a satisfying model for a complex process is a matter of approximation. More specifically, when complex systems are considered, there does not exist an optimal criterion for the best model. The problem is "... that of determining those models that are as good as possible in that no simpler or equally simple model is a better approximation to the data ..." (Gaines, 1977).

Set theory forms the foundations of arithmetic and logic. It is a common extension to proceed from the classification of everyday language to the mathematical formalism of sets. A person may be characterized as tall, or a system as stable, based on the individual perception of the set of tall people, or stable systems. However, the step taken from a predicate to a set is a dubious one (Gaines, 1976a). The notion of membership in set theory is a very precise one. An element either belongs to a set, or does not (dyadic logic).

Although there are many people and systems which may be undoubtedly characterized as tall or stable, respectively, there is no formal decision procedure that enables us to classify every element in this binary fashion. The problem arises with borderline cases which have a dubious classification. An alternative approach is to treat these cases separately as distinct classes. Each entity may be regarded as a member, a non-member, or a border-line member of a set. Therefore, a ternary rather than a binary distinction formed. This is similar to the 3-valued logic (L_3), where an event may be true, false, or possible (Lukasiewicz, 1930). Utilizing this approach we may develop arguments which concentrate on definite classes and leave borderline cases outside of the debate. However, it still leaves open the question of where the borderline actually is, giving rise to the secondary phenomenon of entities that are borderline "borderline" cases.

It is between the thesis of definite borderline and the antithesis of no borderline that Zadeh (1965) created the dialectical synthesis of continuously graded degree of membership to a set. This is a natural generalization of the characteristic dyadic function, $\mu_A(x)$,

$$\mu_A(x) : X \rightarrow \{0,1\}$$

where X is a classical set of objects, called the *universe*, whose generic elements are denoted x, A is a classical subset of X, and {0,1} is called a *valuation set*. The membership or non-membership for any $x \in A$ is defined as:

$$\mu_A(x) = 1 \qquad \text{iff} \qquad x \in A, \text{ and,} \tag{2.15a}$$

$$\mu_A(x) = 0 \qquad \text{iff} \qquad x \notin A \tag{2.15b}$$

A graphical representation of the notion of membership is given in Figure 2.2, where the characteristic equation for an ordinary number $\alpha \in \Re$, is plotted.

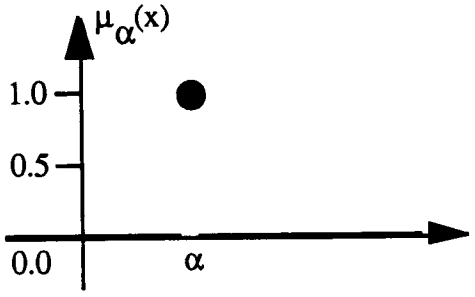

Figure 2.2 Characteristic function of an ordinary number α.

Zadeh (1965) suggested that $\mu_A(x)$ be not restricted to the binary endpoints of the interval [0,1], but instead be allowed to range continuously throughout the interval, i.e.,

$$\mu_A(x) : X \rightarrow [0,1]$$

The semantics of intermediate values of $\mu_A(x)$ are not tightly defined but should be consistent with the natural order relation on the unit interval , e.g. that $\mu_A(x) = 0.7$ denotes a greater degree of membership of x to A than does $\mu_A(x) = 0.3$. Sets with graded characteristic function were called by Zadeh *fuzzy sets*. Based on this formalism Zadeh proposed that such predicates as tall and stable, do not define classical sets with a binary characteristic function, but instead fuzzy sets with graded membership function. Let us define a fuzzy number **A** over an interval of confidence $[\alpha_1, \alpha_2]$ (Kaufmann and Gupta, 1988) as

$$\mu_A(x) = 0, \qquad x < \alpha_1 \tag{2.16a}$$

$$\mu_A(x) = 1, \qquad \alpha_1 \leq x \leq \alpha_2 \tag{2.16b}$$

$$\mu_A(x) = 0, \qquad x > \alpha_2 \tag{2.16c}$$

A schematic representation of the fuzzy number A is given in Figure 2.3. An interval of confidence is an ordinary subset which represents a type of uncertainty. In the above example we know that A cannot be smaller than α_1 and also cannot be greater than α_2.

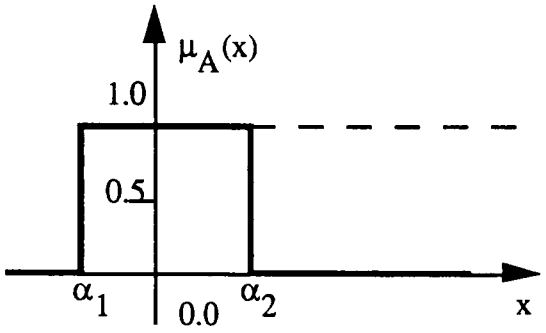

Figure 2.3 Membership function of a number A with an interval of confidence.

The concept of fuzzy set possesses major advantages with respect to the possible approaches to the borderline cases considered so far. No artificial precision is needed to avoid borderline cases, such that, "a tall man has height greater than 1.76543 m", since any increase in height is accompanied by an analogous increase in the degree of membership to the fuzzy set of tall people. Furthermore there is no need for introducing a clear-cut distinction between borderline cases and definite membership, since it would involve the introduction of an arbitrary threshold numerical value. Lukasiewicz (1930), in his 3-valued logic, introduced a third value, 0.5 (possible), between 0 (false) and 1 (true). This could be extended to set theory with 0.5 being the value of the characteristic function for the borderline elements. Through this viewpoint, Zadeh's theory may be seen as a natural generalization of the ternary membership values of a "borderline" set theory, {0, 0.5, 1}, to an infinite range of values in the interval [0,1].

The concept of *fuzzy set* may be seen as providing a new tool, more appropriate than the classical set theory, for a program of precisiation (Gaines and Kohout, 1977). It allows the inherent imprecision of the concepts that we actually use, to be neither excluded nor introduced explicitly in the explicatum, but rather to be subsumed in the universal concept of a degree of membership to a fuzzy set centered on the explicatum. The explosive growth of definitions of apparently simple concepts, such as stability, accompanied with an ever increasing precision, requires new tools and methodologies as system theory attempts to cope with the more subtle features of complex systems. Habets and Peiffer (1973) report 184,320 formally different concepts of stability based on variants of those in the literature. The ultimate precisiation may be achieved only by treating every event in the world as different from every other - as it actually is.

Contrary to Zadeh's theory, Kalman, Falb and Arbib (1969) insist that classical methodologies are adequate for the continued pursuit of universal laws of system theory, similar to Newton's laws in physics. That is, Kalman et. al. are affirming the old doctrine of the theology of science - that the lack of a clear and precise model of a phenomenon should lead to a continuous search for a more accurate model. While continuous search for ever-increasing precision is desirable, there is no formal procedure defining whether we have achieved the maximum possible precision, or even adequate precision. As Zadeh (1992) reports regarding recent advances in the approximate solution of the Traveling Salesman Problem, the time needed to solve the problem to within 1% for 100,000 cities is two days; to within 0.75% for 100,000 cities is seven months; and to within 3.5% for 1 million cities is three and one half hours. What is striking in this example is the dramatic effect in computation time by reducing the accuracy from 0.75% to 1% and 3.5%.

A similar debate has taken place concerning the role of randomness in systems theory. Many eminent scientists, including Darwin and Einstein, have regarded randomness as a sign of ignorance rather than a phenomenon in its own right. Kalman has suggested that it is important to continue to act as if there were precise universal laws. However, Gaines (1976b) has shown that a modeller assuming causality faced with a sequence generated by a system having the slightest non-causal components forms a model which is not just incorrect but totally meaningless. Thus, it is dangerous to assume certain forms of precision when they do not exist in the world. It is possible that we may acquire interesting results which have been generated from the mismatch between presupposition and observation, not from the nature of the observed system itself. Precise models of complex systems are often mathematically intractable. Zadeh (1973) stated what he called the *principle of incompatibility*: "Stated informally, the essence of this principle is that as the complexity of the system increases, our ability to make precise and yet significant statements about its behavior diminishes until a threshold is reached beyond which precision and significance (or relevance) become almost mutually exclusive characteristics".

Bellman and Zadeh (1969) have suggested that most of what tends to be characterized as random noise in systems theory is actually our inability in setting sharply defined criteria of set membership. Fuzzy set theory offers a physical tool for coping with these types of systems allowing a continuous valued membership in a set. Thus, the vagueness in the definition of exact membership boundaries introduces fuzziness, rather than randomness, in the system description. This means that although we deal with deterministic systems, the constraints and the goals set are fuzzy in nature. The decision making process takes place in a fuzzy environment where only the fuzzy goals and the fuzzy constraints can be defined as fuzzy sets (classes) in the space of alternatives.

Closing the discussion on fuzzy sets an important issue is yet to be addressed. The notion of continuous grade of membership inherently implies the existence of arbitrarily designed membership function. Zimmerman (1985) presents a complete collection of algebraically described membership functions suitable for every application in the area of fuzzy sets. Mamdani (1974), and King and Mamdani (1977) in their pioneer work on fuzzy control systems, exhibit the potential benefits of trapezoidal membership functions for industrial applications. Dubois and Prade (1980) address the problem of initial choice of appropriate membership function and describe a set of practical rules useful for selecting membership functions. Apparently there is no formal procedure for selecting the shapes of membership functions or positioning them in the universe of discourse. Zadeh (1992) has realized the problem of membership function calibration and has proposed "... the use of cut-and-trial procedures", until sufficiently powerful adjustment algorithms can be developed.

In order to address the issue of "choosing" appropriate membership functions, the utilization of neural networks as membership function generators is proposed. The response of independent neural networks defines the actual shape of the membership functions, based on user-supplied prototypes. The issues of the area occupied on the universe of discourse, by every membership function, as well as its position, are resolved from an adjustment algorithm devised. The fundamental principles of this algorithm are two independently developed criteria, namely; the *dissemblance index* and *dynamic behavior criteria*, which will be defined in Chapter 4.

CHAPTER 3

VIRTUAL MEASUREMENTS

Overview

Human understanding of physical systems is generated through *measurements* of physical variables, *recognition* of mechanisms, *discrimination* of different states, and finally *classification* of different categories of system behavior. A common ground is shared by all four diverse procedures: namely, the generation of numbers (or invariant labels in general) which serve as identification elements of the process they are associated with. The association is established in a manner where processes bearing the same identification are considered to be alike, and processes bearing different labels are considered to be different.

According to Rosen (1978) three basic notions are associated with the process of measurement; namely, the *system*, the *observable*, and the *state*. Rosen defines a *system* as some part of the real world which comprises our object of study; an *observable* of the system as some characteristic of the system which can, in principle, be measured directly; and a *state* as a specification of what the system is like at a specific time instant. These three concepts are interrelated by two axiomatic propositions, (1) that the only meaningful physical events are those represented by the evaluation of observables on states, and (2) every observable can be regarded as a mapping from states to real numbers (or any kind of labeled sets).

Although with the usual notion of measurement we associate numbers as the output of a measuring device, measurements are also possible without numerical outputs. As Pattee (1986) points out, timing, weighing, navigating, surveying, and numerous interactions with complex systems since earlier times, have been accomplished by symbolic representations: *verbal*, *iconic*, *mimetic* or *analog*. Whether numerical or symbolic, the choice of the type of measurement is made in accordance with the particular functional requirements of the system as a whole and the relationship between the system and the observer. Essentially all measuring devices are associations between input spatiotemporal patterns and output spatial symbols, under the requirements of precision and reproducibility.

Thus, measurement involves the classification of incoming information, a process closely related to the labeling of different states of the system. Implicit in all measurements is a knowledge-based procedure, or model, for a **many-to-one** or **complex-to-simple** mapping. It is possible to employ a neural network as a tool of defining a virtual measurement, i.e., a tool for mapping complex input patterns to simple outputs (Tsoukalas, Ikonomopoulos and Uhrig, 1991a) (Tsoukalas and Ikonomopoulos, 1991b), (Ikonomopoulos et. al. 1992a). The inputs are patterns of parameters from a complex system and the outputs are membership functions representing the meaning of quantifiers and predicates, such as, "high", "adequate", etc., which describe the state of the system.

Theoretical Considerations

The concepts of *state* and *observable* are inter-dependent; it does not seem possible to view them as mutually exclusive notions. In order to evaluate an observable on a state of

an arbitrary system, a particular physical device called a *measuring instrument*, or *meter*, is required. According to the second axiomatic proposition defined in the previous section, the purpose of the *measuring instrument* is to create a mapping, f, from the set of states, $\{s\}$ of the system in question, into the set of real numbers; i.e. f: $\{s\} \rightarrow \Re$. The *measuring instrument* has to be *isolatable* from the system measured. Furthermore, it has to be *resettable* so that it can be calibrated, and the employment of the meter in defining the mapping f depends only on the state s (Pattee, 1986).

Consider an abstract set S of states s of the system of interest, and a single observable which is our only access to the individual states s of S. Through this single meter we are able to define a mapping f: $S \rightarrow \Re$. The mapping f induces an *equivalence* relation R_f on S, defined as

$$sRs' \qquad \text{iff} \qquad f(s) = f(s')$$

A relation R on S is an equivalence relation if it satisfies the following conditions:

a. sRs for every s in S (i.e., R is reflexive)

b. if sRs' then s'Rs (i.e., R is symmetric)

c. if sRs' and s'Rs" then sRs" (i.e., R is transitive)

For example assume the set of natural numbers $\mathbf{N} = \{0, 1, 2, 3, ...\}$. An equivalence relation R on **N** is defined as follows

$$n_1 R n_2 \Leftrightarrow n_1 \equiv n_2 \ (\text{mod}3) \qquad (3.1)$$

Thus, $n_1 \in \mathbf{N}$ is related to $n_2 \in \mathbf{N}$ through the equivalence relation (3.1) iff the two natural numbers n_1 and n_2 have the same residue when divided by three.

For every equivalence relation R_f on **N**, the *quotient set* $\mathbf{N}/R_f$ is defined as the set of all equivalence classes. The latter are the subsets of **N** that contain all equivalent elements of **N** under the assumed relation. The equivalence classes comprise a partition of **N**. For the above example, there exist three equivalence classes. One class is composed of natural numbers that have zero residue when divided by three (denoted as $\overline{0}$). The second class contains all the numbers that have residue equal to one (denoted as $\overline{1}$), and the third one is composed of all natural numbers that have residue two when divided by three (denoted as $\overline{2}$). Thus, the quotient set $\mathbf{N}/R_f$ is

$$\mathbf{N}/R_f = \{\overline{0},\ \overline{1},\ \overline{2}\} \tag{3.2}$$

In other words, if relation (3.2) is our measuring instrument of the natural numbers, the numbers 3 and 60 can not be distinguished as separate events, since both produce the same dynamic effect on the measuring device. They belong to the $\overline{0}$ class because their residue when divided by three is zero. Therefore, only partial knowledge for the system is obtained through one specific observable. If one uses two different observables, then less information is neglected and our knowledge about the system is improved. The latter will actually occur iff the partition induced from the second observable is "finer" than the initial one.

For example, if one chooses as a second observable the mod6 relation, then the quotient set for the new equivalence relation R_g on **N** is

$$N/R_g = \{ \overline{0}', \overline{1}', \overline{2}', \overline{3}', \overline{4}', \overline{5}' \} \tag{3.3}$$

Under this relationship the numbers 3 and 60 (previously indistinguishable) are now distinguishable since $3 \in \overline{3}'$ and $60 \in \overline{0}'$. This holds because the partition mod6 is finer than the partition mod3,

$$\overline{0}' \subset \overline{0}, \overline{1}' \subset \overline{1}, \overline{2}' \subset \overline{2}$$

On the other hand, if the second observable is defined by the mod5 relation, then the quotient set for the equivalence relation R_m on **N** is

$$N/R_m = \{ \overline{0}'', \overline{1}'', \overline{2}'', \overline{3}'', \overline{4}'' \} \tag{3.4}$$

The numbers 3 and 60 are also distinguishable under this relation, i.e., $3 \in \overline{2}''$ and $60 \in \overline{0}''$. However, there are numbers that were previously distinguishable (under the mod3 relationship) like $30 \in \overline{0}$ and $65 \in \overline{2}$, but are indistinguishable under mod5 (i.e., $30 \in \overline{0}''$ and $65 \in \overline{0}''$). In this case it is said that the mod5 relationship induces an observable that is not "linked" to the mod3 observable. According to Rosen (1976), the notion of linkage captures the relationship between two observables that determine partitions (quotient sets) that one is finer than the other.

The measurement process as it has been formulated up to this point involves a process of *abstraction*. For instance, when we measure the values of one or more observables of a system, we thereby obtain a description of the system in terms of a set of reduced states (S/R_f); which are characterized only by the observables we have measured, and the remainder have been abstracted away. Basically, by measuring the observables f,

and g, from the set of states S, we have acquired only partial knowledge for the system S, neglecting information existing in other observables never measured. Thus, performing an act of abstraction involves the "forgetting" or ignoring of some aspect or structure organization of the system under consideration (Rosen, 1987). The performance of such an act of abstraction is the first step in theoretical analysis of system behavior. Two very well-known examples of abstractions have played an important role in the development of physics and biology. In physics, we represent a material body by a mass point, retaining only the observables of position and mass. In biology, a biological neuron is described by a binary switch, retaining only those observables of "firing" or "not firing", and only those dynamics which cause transitions between those gross states.

It is important to recognize that the kinds of subsystems which can be extracted are limited by the types of observational techniques (e.g., meters) which are available to us (Casti, 1988). There are many other subsystems besides the ones which can be extracted with a limited set of observational techniques. Furthermore, these other subsystems can participate in material interactions on precisely the same basis as those which we can characterize empirically with our meters.

The purpose of performing an abstraction stems from the necessity to replace the original system by a subsystem (i.e. by an abstraction) and observe the same behavior, or approximately so. Thus according to Rosen (1978), "... the utility of abstractions, and the subsystem which they represent, is that they allow us to "approximate" the behavior of our original system, at least over some range of conditions, by means of a (simpler) subsystem ...". The performance of an abstraction determines the set of circumstances for which the abstraction can replace the original system. The notion of abstraction contains the essence of modeling. The relationship between the original system and a model of it, may be

envisioned as imposed by an "observable." Furthermore, the recognition that the two systems behave identically reduces to the fact that they impose identical dynamics on the same measuring device.

Let us assume that we can physically measure a variable of the system of interest. Furthermore, we assume that the measured parameter is linked to another system parameter through a mathematical formulation. Since the process of measurement of the first parameter maps the particular state of the variable to a *number* or a *linguistic label*; we need not measure the latter parameter directly; but we can obtain its value by the process of *computation*. Thus, according to Rosen (1978) we may argue, that there must be a close relationship between *measurement*, which is the result of dynamic interaction between systems, and *computation*, which is the result of dynamic manipulation of symbols. The empirical content of a computation stems from the interpretation of the symbols as labels for the states of a physical system. Rosen (1986) has proved that the process of measurement always results in the *establishment of a linkage* between observables in the system being measured and observables in the meter performing the measurement. Likewise, a computation establishes a linkage between observables pertaining to a symbolic system; when these symbols are labels for states of a physical system. This implies the existence of further linkages between observables of that physical system and observables defined on the symbols which represent it. In this sense, it is correct to say that computation and measurement are basically identical processes, which depend crucially on the existence of linkage relations between observables.

Based on this theoretical framework, a number of system variables are physically measured and the obtained numbers are used in order to computationally derive the state of another system parameter which is linked to the previous parameters. The latter system

parameter can not be directly monitored and thus a process of computation is required to estimate its values (*virtual measurement*). The computational process involves the mapping of a set of time series (i.e., numbers) to the space of human linguistics through non-linear classifiers. The resultant linguistic evaluations are further manipulated in order to derive the state of the non-directly measured parameter in a condensed form, suitable for applications in a fuzzy environment.

CHAPTER 4

NUCLEAR REACTOR SYSTEM MONITORING

Introduction

As it has already been discussed in Chapter 3, the notion of measurement is strongly associated with the existence of a measuring device. However, this is not always the case. In a complex system environment, there is a number of parameters with operational significance, which can not be monitored directly. Hence, their values have to be computed from other system parameters. For our purpose we consider a nuclear reactor system as a complex one, and as the immeasurable parameters *reactivity, valve_position, performance, safety,* etc. All these parameters can not be measured directly with the utilization of a gauge, and their values have to be inferred from information acquired through other monitoring devices. Furthermore, there are parameters whose values are indeed monitored, but their actual values are known with a significant time lag, due to measurement complication.

The issue of accurate and on-time knowledge of non-measurable parameters, came dramatically in the forefront with the Chernobyl accident. The accident has been classified as a classical example of excess reactivity insertion, but the operators did not have any idea of the amount of reactivity that existed in the system. A colorful description of the situation at the Chernobyl 4 operating room is given by the former deputy engineer for operations at Chernobyl, Anatoly Dyatlov (1991):

"The reduction in operational reactivity margin was overlooked owing to the lack of suitable instrumentation to indicate its value. The device available to the Chernobyl 4 operators at the time required about five minutes to do a single measurement and was completely useless when reactor parameters were not constant. While it was suitable for controlling the power density distribution when used in combination with the system for physically monitoring power distribution, it was completely inappropriate for the task that had just presented itself to the Chernobyl operators."

The results of "... lack of suitable instrumentation", and computation time required for a single measurement have been widely publicized and there is no need to discuss here. The issue which remains to be addressed, is that of developing suitable computer techniques capable of performing accurate estimations of significant system parameters, which can not be measured with conventional techniques. In addition, these calculations have to be performed in a time frame faster than - or at least comparable to - real time, so that sufficient information will be supplied to human operators in the shortest possible time span.

In order to provide a solution to this most uncomfortable situation, we propose the utilization of artificial neural networks as generators of membership functions for the purpose of monitoring nuclear reactor systems. In this context, ANNs provide a complex-to-simple mapping of reactor parameters in a process analogous to that of measurement. Through such "virtual measurements" the value of parameters with operational significance, e.g., VALVE_POSITION, can be determined. A set of pre-trained neural networks are the receivers of several on-line time series and calculate other system parameters, not in the form of time series, but in the form of membership functions, as it

can be seen in the general schematic shown in Figure 4.1. Each membership function corresponds to a different value of the monitored variable and has a unique shape.

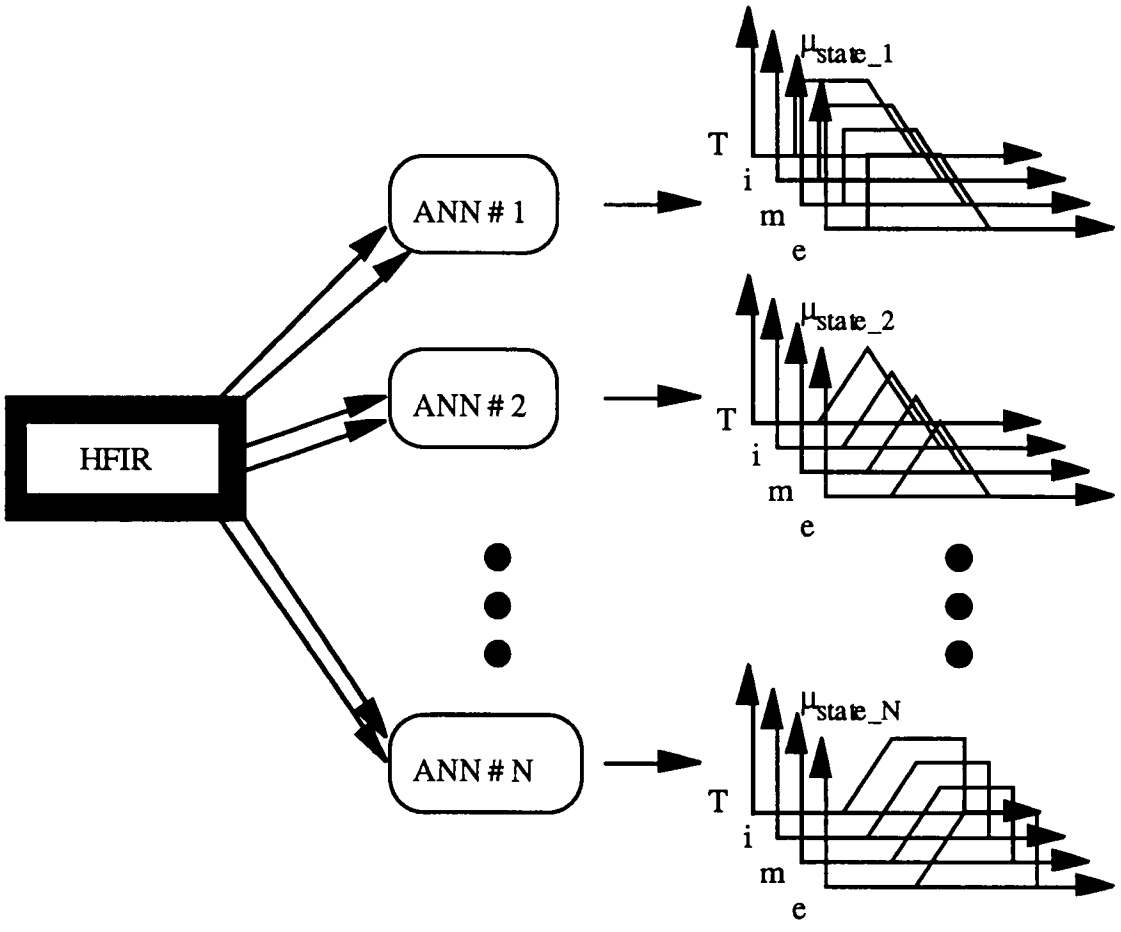

Figure 4.1 Schematic representation of virtual measurement instrumentation.

Model Description

One of the parameters which can not be directly measured in most industrial environments is the VALVE_POSITION. It is the opaque construction material, along with the design, which do not permit the direct knowledge of the valve-disc position. In some industrial applications the position of the disc may be measured with the utilization of

electromagnetic devices, but still the measurement remains inaccurate since it is characterized by significant time lags. In order to demonstrate the proposed methodology we utilized actual data obtained during a start-up period of the High Flux Isotope Reactor (HFIR), three-loop pressurized water reactor, operated at the Oak Ridge National Laboratory. The parameter which is to be simulated is the **position** of the Secondary Flow Control Valve. This particular valve controls the flow of water in the secondary side of the system and is considered as a vital system component. The data used is normalized in the interval 0.1 to 0.9 and sampled every 16 seconds, with a total of 1240 samples.

Five parameters have been chosen for describing the Secondary Flow Control Valve position: **neutron flux, primary flow pressure variation (DP), core inlet temperature, core outlet temperature** and **secondary flow** (Figures 4.2, 4.3, 4.4, 4.5, and 4.6). All but the last one of the above mentioned time series, contain average values of the corresponding parameters of the three-loop system. These parameters are selected in order to provide sufficient description of both the primary and secondary sides of HFIR during start-up. The time series of these five parameters are used to train a set of five neural networks, where each one of them is responsible for recognizing a specific position of the valve-disc. The output of every neural network is a membership function uniquely labeling a particular position of the Secondary Flow Control Valve. The crisp values of the Secondary Flow Control Valve Position are plotted vs. time in Figure 4.7.

The behavior of the Secondary Flow Control Valve is simulated in the space of alternatives (universe of discourse) with the fuzzy variable **VALVE_POSITION**, which takes five fuzzy values, namely; **closed, partially_closed, medium, partially_open**, and **open**. Each one of these fuzzy values is represented with a

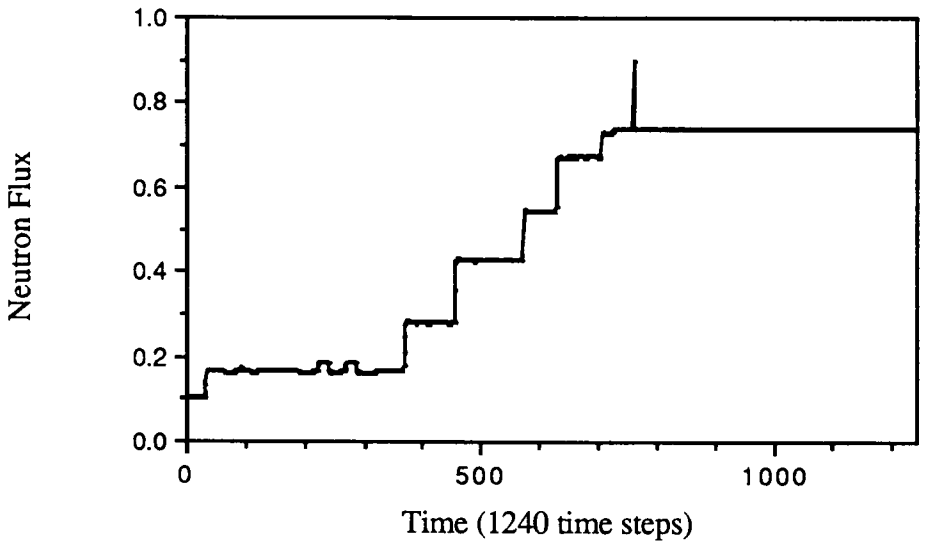

Figure 4.2 Neutron flux vs. time (average three-loop values).

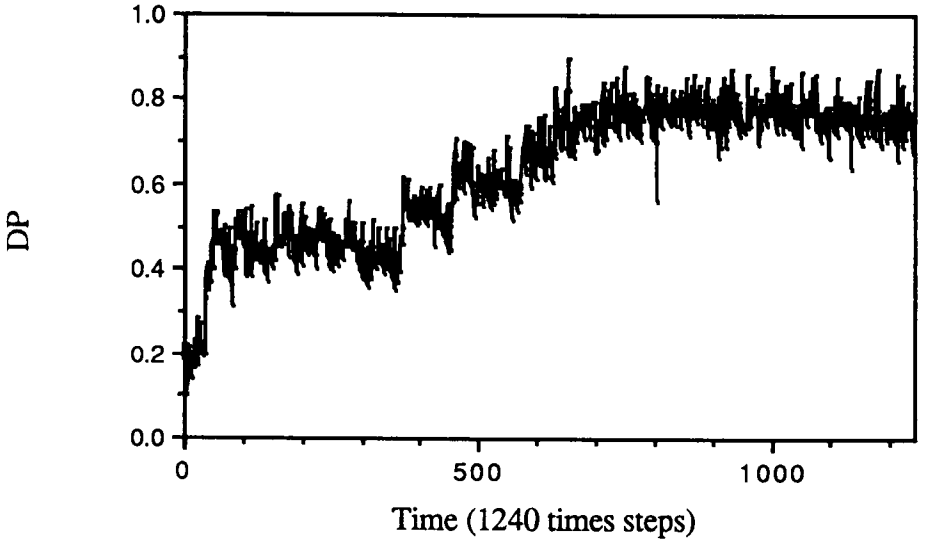

Figure 4.3 Primary flow pressure variation vs. time (average three-loop values).

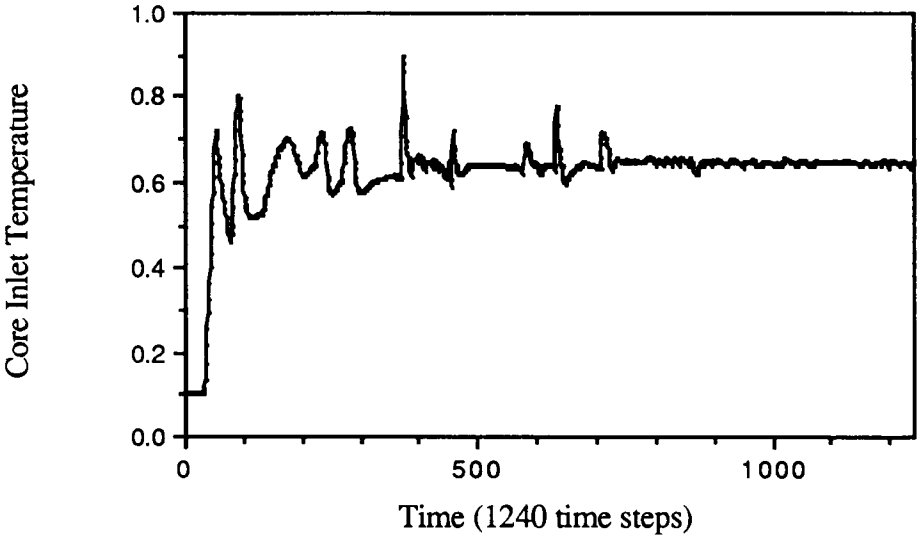

Figure 4.4 Core inlet temperature vs. time (average three-loop values).

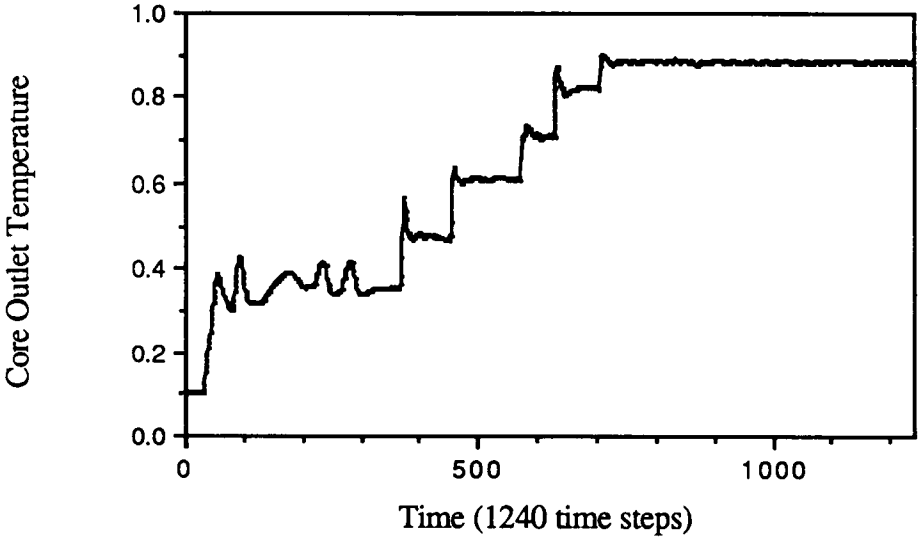

Figure 4.5 Core outlet temperature vs. time (average three-loop values).

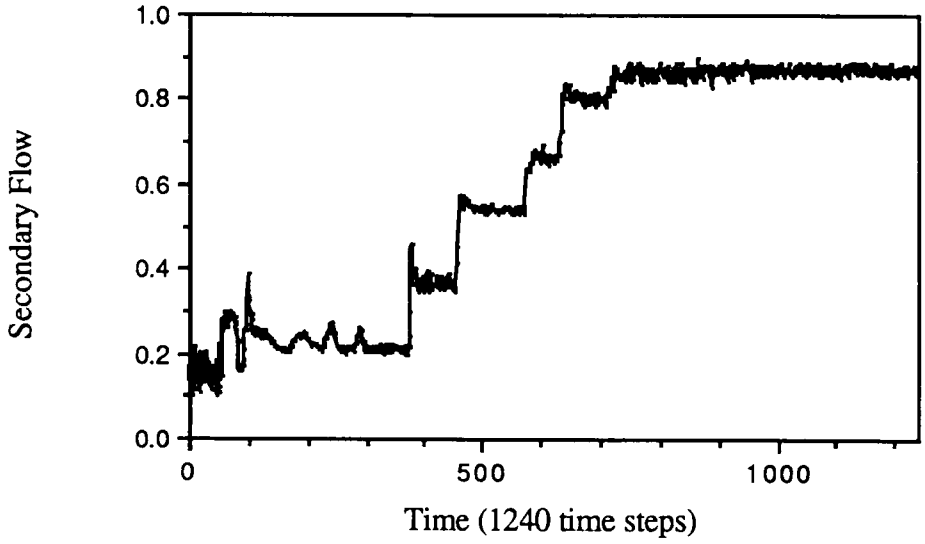

Figure 4.6 Secondary flow vs. time.

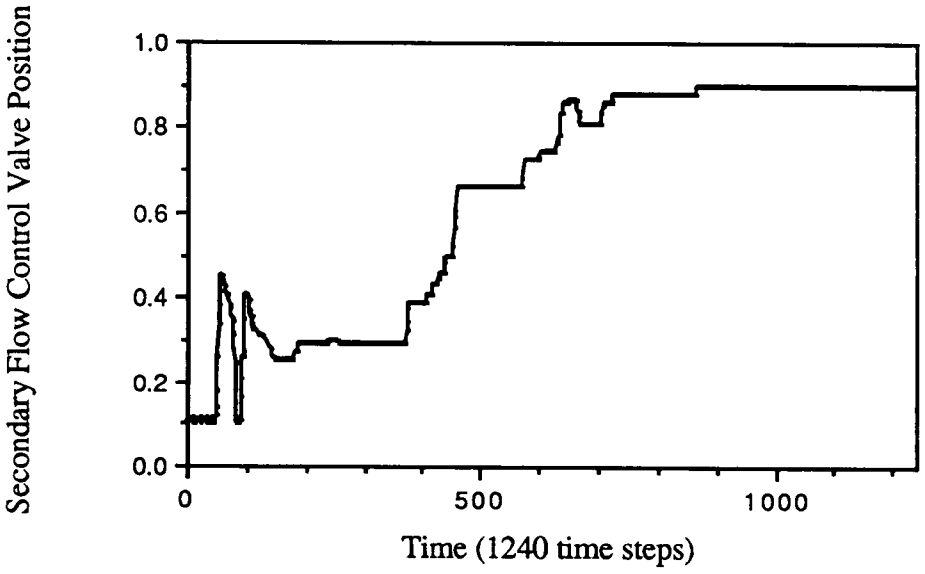

Figure 4.7 Secondary flow control valve position vs. time.

membership function, namely; $\mu_{\mathbf{closed}}$, $\mu_{\mathbf{partially\_closed}}$, $\mu_{\mathbf{medium}}$, $\mu_{\mathbf{partially\_open}}$, and $\mu_{\mathbf{open}}$. These five membership functions are considered sufficient to describe the position of the valve at every instant during the start-up period. A schematic of the membership functions representing the fuzzy values of the fuzzy variable **VALVE_POSITION** in the universe of discourse, is given in Figure 4.8.

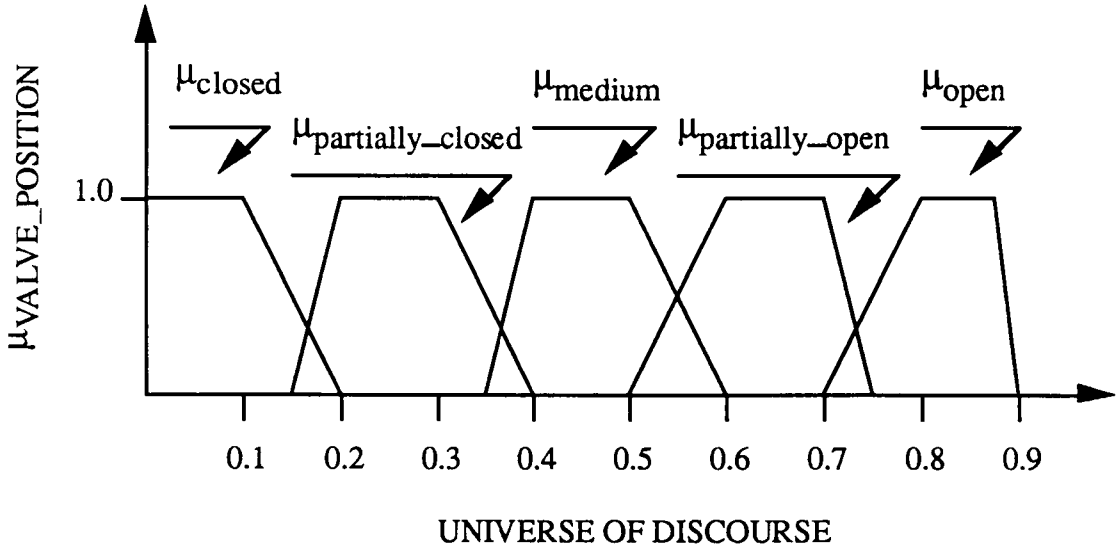

Figure 4.8 Membership function labeling of possible valve positions.

The area occupied by every membership function in the universe of discourse depicts the uncertainty associated with this particular class (Zadeh, 1965). The fuzzy set (or class) **open** in the space of alternatives is characterized by the membership function $\mu_{\mathbf{open}}$ (Figure 4.8), which associates each point in the universe of discourse with the value of $\mu_{\mathbf{open}}$ at this point. In this way the grade of membership (or level of presumption) of this point in the class **open**, is represented. It is apparent that such framework provides a natural way of dealing with problems in which the source of imprecision is the absence of sharply defined boundaries of class membership, rather than the presence of random variables (Zadeh, 1976). In the case under consideration, the vagueness in the definition of the exact position of the valve-disc introduces the fuzziness

of the valve position, and hence is the reason for the absence of sharply defined criteria. This means, that although we deal with a deterministic system, the constraints and the goals set are fuzzy in nature. Therefore the decision-making process takes place in a fuzzy environment where only the fuzzy goals and the fuzzy constraints can be defined as fuzzy sets (classes) in the space of alternatives. The fuzzy decision will be the intersection of the given goals and constraints (Bellman and Zadeh, 1969). In our case both goals and constraints are defined by the same set of classes.

The membership functions developed for this particular study have trapezoidal shape, which is very useful for performing computations in the fuzzy control area (Zadeh, 1988). The five membership functions have been defined with the following sets of equations:

μclosed:

$\mu_{closed}(x) = 33.34*x - 0.6668$ $\quad 0.02 \leq x \leq 0.05$ (4.1a)

$\mu_{closed}(x) = 1.0$ $\quad 0.05 \leq x \leq 0.10$ (4.1b)

$\mu_{closed}(x) = -10.0*x + 2.0$ $\quad 0.10 \leq x \leq 0.20$ (4.1c)

μpartially_closed:

$\mu_{partially\_closed}(x) = 20.0*x - 3.0$ $\quad 0.15 \leq x \leq 0.20$ (4.2a)

$\mu_{partially\_closed}(x) = 1.0$ $\quad 0.20 \leq x \leq 0.30$ (4.2b)

$\mu_{partially\_closed}(x) = -10.0*x + 4.0$ $\quad 0.30 \leq x \leq 0.40$ (4.2c)

μmedium:

$\mu_{medium}(x) = 20.0*x - 7.0$ $\quad 0.35 \leq x \leq 0.40$ (4.3a)

$\mu_{medium}(x) = 1.0$ $\quad 0.40 \leq x \leq 0.50$ (4.3b)

$\mu_{medium}(x) = -10.0*x + 6.0$ $0.50 \leq x \leq 0.60$ (4.3c)

$\mu_{partially\_open}$:

$\mu_{partially\_open}(x) = 10.0*x - 5.0$ $0.50 \leq x \leq 0.60$ (4.4a)

$\mu_{partially\_open}(x) = 1.0$ $0.60 \leq x \leq 0.70$ (4.4b)

$\mu_{partially\_open}(x) = -20.0*x + 15.0$ $0.70 \leq x \leq 0.75$ (4.4c)

μ_{open}:

$\mu_{open}(x) = 10.0*x - 7.0$ $0.70 \leq x \leq 0.80$ (4.5a)

$\mu_{open}(x) = 1.0$ $0.80 \leq x \leq 0.85$ (4.5b)

$\mu_{open}(x) = -20.0*x + 18.0$ $0.85 \leq x \leq 0.90$ (4.5c)

It becomes apparent from the geometrical schemes depicted through Equations (4.1 - 4.5), (Figure 4.8) that there is an overlap between adjacent membership functions. The reason for the overlap is the fuzziness in the definition of the different states of the valve position. It is characteristic of fuzzy sets to assume, that at a particular moment the valve may be described as **closed** and/or **partially_closed** (Zadeh, 1979). Therefore, the uncertainty associated with instrument measurements is reflected on the position of the membership functions in the universe of discourse. Henceforth, whenever the valve position is between 0.5 and 0.6 it could be characterized as **medium**, as well as, **partially_open** (Figure 4.8). If the crisp value of the position is between 0.6 and 0.7, then it is definitely **partially_open**. This representation offers some unique advantages. It maps a set of complicated time series to the universe of discourse of human linguistics, through a set of neural networks which act as interpreters of vital information supplied from the nuclear system. The information encoded in a time series is in the form of rate of increase/decrease, and maximum/minimum values attained in a period of time. Artificial

neural networks are trained to represent this kind of "hidden" information in the form of membership functions which can be used by a rule-based diagnostician.

The shape of the membership function which has been assigned to each state of the valve is unique, and therefore there is a sharp distinction between different states. The membership function provides sufficient information to describe the position of the valve at a particular instant in time (Zadeh, 1965). Furthermore, an ANN trained to recognize a specific complicated time pattern will lose much of its ability to sustain noisy input signals. This stems out of the network tendency when presented with distorted inputs to produce averaged forms of the desired output, missing therefore vital pieces of information (Tsoukalas, Ikonomopoulos and Uhrig, 1991a). This handicap can be overcome by the proposed technique which has an output that is a simple membership function. The actual shapes of the prototype membership functions, as well as their position in the universe of discourse, have been concluded through a trial-and-error procedure. The beginning of the universe of discourse has been intentionally set at point 0.02 (Equation (5.1a)) in order to minimize the computational errors associated with the sigmoid functions. This does not affect the methodology developed, since the data is normalized in the region 0.1 to 0.9.

The number of membership functions chosen to describe the state of the valve has been concluded upon information supplied from the literature (King and Mamdani, 1977). It has been shown that in most industrial applications, no more than seven membership functions are required to describe the state of every system under most circumstances (Mizumoto, 1988). In our earlier approaches to the problem (Ikonomopoulos, et. al., 1992a), we utilized three membership functions and later we increased this number to seven. The latter (more detailed) partition of the universe of discourse was found to be unnecessary for the conditions of the system under investigation. Thus, the number of

membership functions was reduced to five, which is considered sufficient for accurately identifying the possible states of the Secondary Flow Control Valve. Furthermore, characterizations as **almost_open**, **very_close**, etc., introduced by Zadeh (1972) as **linguistic hedges**, do not seem appropriate in this application. Membership function modifiers like "almost" and "very" are mostly used in classification problems dealing with more complex categorizations than the one pursued in this application (Dubois and Prade, 1980).

One of the major advantages of artificial neural networks is their unique ability to be trained in a **user-defined environment**. This characteristic has been proven vital in the area of neural computing. Lapedes and Farber (1987) have shown experimentally, that a neural network is able to make predictions of arbitrarily complex mappings with very good accuracy. This property has been utilized in the present work, also. The set of neural networks developed to calculate the states of the **VALVE_POSITION**, has been trained to predict the position of the valve at the near future. This means that the neural networks receive information concerning the state of the system at instant t in time and predict what the valve position may be at instant $t+\Delta t$. Since the present work is based on sampled data provided from HFIR, the time interval Δt has been set equal to one sampling interval (i.e., 16 seconds), which is the minimum permissible. A higher value for Δt would significantly affect the accuracy of the prediction (Ivakhnenko and Lapa, 1967). Furthermore, the time interval of 16 seconds can be considered sufficient for decision-making procedures, and offers adequate time margin for performing computational tasks.

The architecture of the neural networks used in the present work is shown in Figure 4.9, and is the same for all five networks. It is a three-layer network with one input, one hidden, and one output layers, similar to the one described in Chapter 2. The input layer of

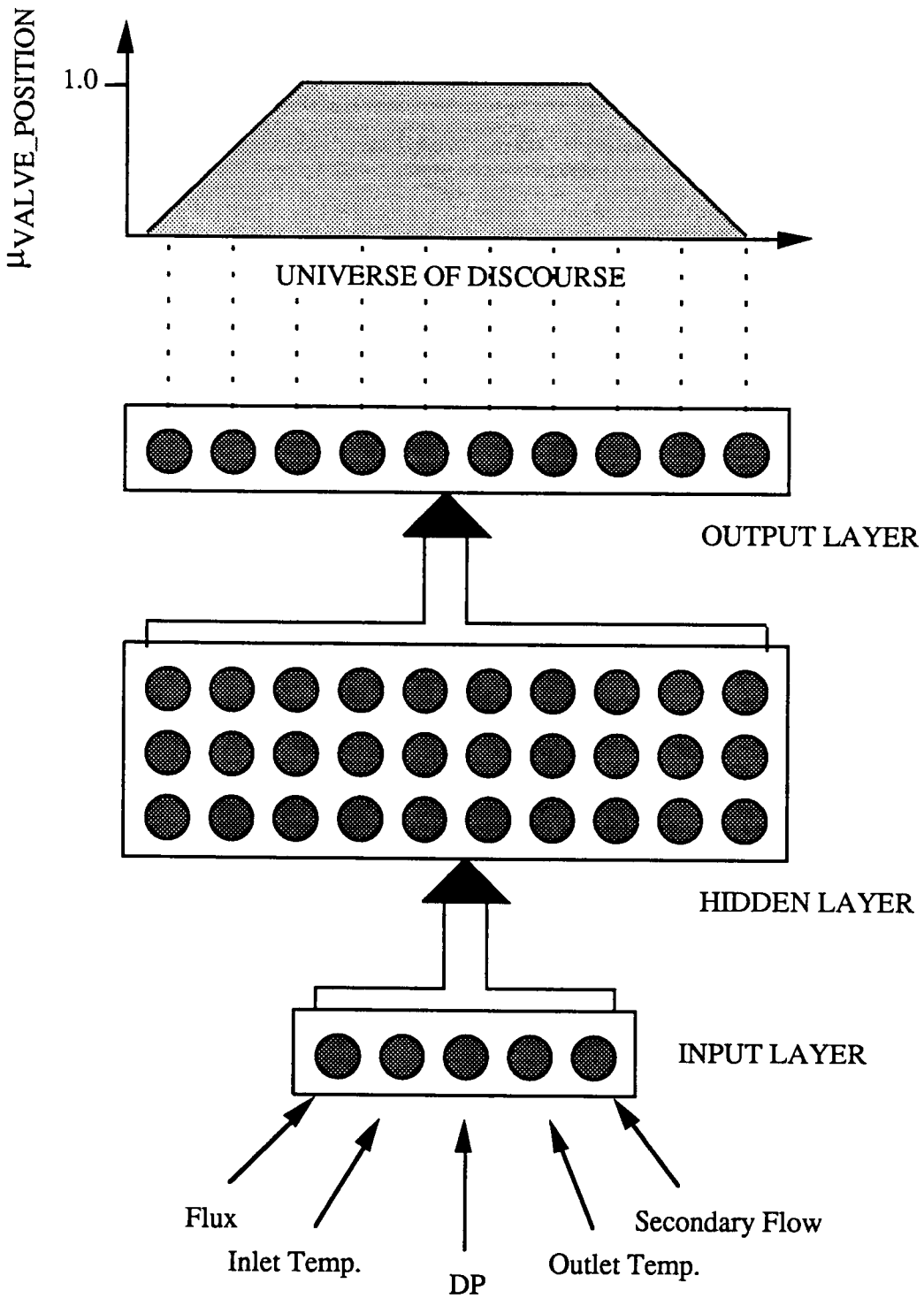

Figure 4.9 Neural network architecture.

the network consists of five nodes, each one receiving information from a particular time series. The output layer consists of ten nodes, corresponding to the coordinates of ten points of the trapezoidal membership function, describing the position of the Secondary Flow Control Valve at the universe of discourse. The optimum number of points was found after continuous experimentation with different combinations of points. Ten points were proven to be sufficient for preserving the accuracy required for the membership function representation. A larger number of points was found to have the same accuracy as ten points, and increased the computational cost significantly. Since for symmetry reasons the same architecture has been chosen for all five networks, a number of trial runs showed that the optimum number of neurons (computing units) in the hidden layer, should be equal to thirty. In order to train the networks the back-propagation algorithm has been utilized, with generalized delta rule and momentum term, as it has been described in Chapter 2. The training procedure was carried out in the Plexi neural environment (Symbolics, 1990) and a maximum of 7,000 training cycles was adequate for attaining a sum square error of 0.02.

Dissemblance Index

The utilization of more than one ANNs for virtual measurements, creates two significant problems. Generally, the outputs of the neural networks will be somewhat different than the membership functions the networks were trained for (Figure 4.10). Moreover, one or two (since overlap of membership functions has been allowed) will represent correct values, while the rest need to be ignored. It is thus important to identify the correct output in order to eliminate the information surplus created by network abundance. The membership functions calculated by the neural networks are treated as *fuzzy numbers*, and the *dissemblance index* (Kaufmann and Gupta, 1991) is computed in order to measure the distance between two fuzzy numbers. These fuzzy numbers are the

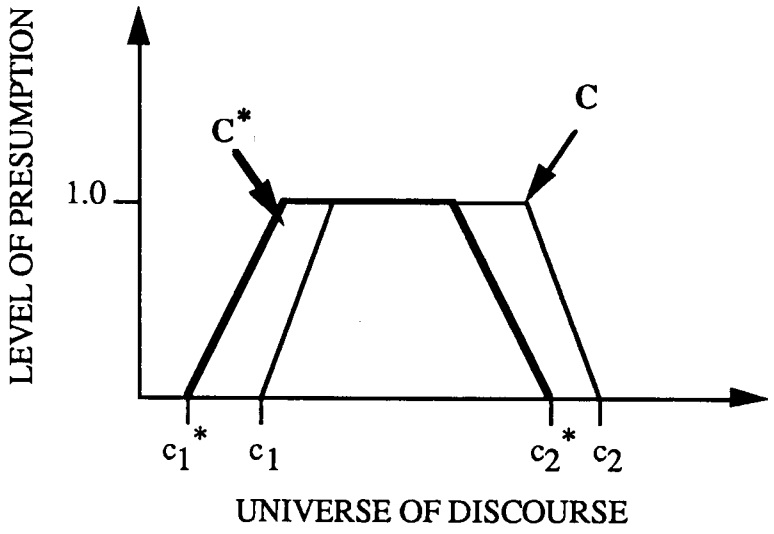

Figure 4.10 Prototype C and predicted C^* trapezoidal fuzzy numbers.

prototype membership function, and the membership function predicted from the network. In this manner, the neural network outputs that are the closest to the set of prototype membership functions are selected as the actual output of the virtual monitoring device at any given time.

Suppose, for example, that the network that recognizes the disc-position fuzzy value **closed** has been trained on the output membership function $\mu_{\textbf{closed}}$, and after training it produces an actual output membership function, $\mu^*_{\textbf{closed}}$. Consider the two membership functions as two fuzzy numbers, call them C and C^*, each with a trapezoidal shape and support on the universe of discourse [0,1] (Figure 4.10). The support of each function is an interval, i.e.,

$$C = [c_1, c_2] \quad \text{and} \quad C^* = [c_1^*, c_2^*]$$

A number $\delta(\mathbf{C}, \mathbf{C}^*) \in [0,1]$, can be computed which is the distance between $\mathbf{C}$ and $\mathbf{C}^*$. The *dissemblance index* of the prototype output, $\mathbf{C}$, and the actual output, $\mathbf{C}^*$, is defined as:

$$\delta(\mathbf{C},\mathbf{C}^*) = \int_{\alpha=0}^{\alpha=1} \delta(\mathbf{C}_\alpha, \mathbf{C}^*_\alpha) d\alpha \tag{4.6}$$

where $\mathbf{C}_\alpha$ and $\mathbf{C}^*_\alpha$ are the α-cuts for the two membership functions $\mathbf{C}$ and $\mathbf{C}^*$, respectively. When the universe of discourse is normalized in the interval [0,1], a value for the *dissemblance index* can be easily derived from Equation (4.6) in terms of the support of each fuzzy value and the α-cuts, $\alpha \in [0,1]$:

$$\delta(\mathbf{C},\mathbf{C}^*) = \frac{1}{2} \int_{\alpha=0}^{\alpha=1} \left(\left| c_1^{(\alpha)} - c_1^{*(\alpha)} \right| + \left| c_2^{(\alpha)} - c_2^{*(\alpha)} \right| \right) d\alpha \tag{4.7}$$

The integral presented in Equation (4.7) is multiplied by 1/2, so that the result of the integration will be normalized in the interval [0,1]. The *dissemblance index* is a number ranging from 0 to 1, representing the distance between two fuzzy numbers. If $\delta(\mathbf{C}, \mathbf{C}^*) \approx 0$ then $\mathbf{C}$ and $\mathbf{C}^*$ are almost identical, on the other hand, if $\delta(\mathbf{C}, \mathbf{C}^*) \approx 1$, then $\mathbf{C}$ and $\mathbf{C}^*$ are totally different.

Since "*almost equal* (≈)" can not be accurately modeled in a computer algorithm, the following technique has been devised to discriminate among different network responses. Every neural network trained on a set of data representing a particular state of the valve (i.e., **closed**, **partially_closed**, etc.), is initially tested on the same set of data. The computed membership functions are then compared with the prototype membership functions, and the *dissemblance index*, ($\mathbf{d\_i_k(j)}$), is calculated for every input vector $\mathbf{j}$ and

network **k**. The computation is performed with a computer algorithm based on the mathematical formalization presented in Equations (4.6) and (4.7). The code has been written in Kernighan and Ritchie C (K&R C) (1978), it is called DISINDEX.C, and a listing of the program is given in the Appendix. Here, it should be noted that all computer algorithms developed for the purposes of this work, have been written in K&R C, and executed in a *SPARC II SUN®* workstation, a *SUN® 386i* workstation, and the *utkux1* mainframe computer of the University of Tennessee Computer Center.

Based on the information supplied from the dissemblance index calculation, the next step is to estimate the maximum value, $\mathbf{max\_d\_i_k}$, of all $\mathbf{d\_i_k}$'s computed from the responses of every network **k**. Thus, a total of five different $\mathbf{max\_d\_i_k}$'s are calculated. These values are then set as the upper limit of the dissemblance index values attained by every network-predicted membership function. This ceiling is incorporated into the logic of the measuring device and the following criterion (called the **DISSEMBLANCE INDEX CRITERION**) is established for every network response:

FOR EVERY NETWORK ***k***

IF $\mathbf{d\_i_k(j)}$ *IS LESS THAN OR EQUAL TO* $\mathbf{max\_d\_i_k}$

THEN **RETAIN NETWORK k PREDICTION**

IF $\mathbf{d\_i_k(j)}$ *IS GREATER THAN* $\mathbf{max\_d\_i_k}$

THEN **DISCARD NETWORK k PREDICTION**

Thus, when the set of networks is tested on a new (previously "unseen") set of input vectors **j***'s, the membership functions calculated from every network will be accepted only if they satisfy the *dissemblance index criterion.* Since all five neural networks will fire when presented with an input vector, it is the *dissemblance index*

criterion which will decide if a particular network response will be accepted, or discarded. Henceforth, only the network responses with their values below or equal to the **max_d_i$_k$** value will be accepted at a given time, and the rest will be ignored. This criterion resolves the issue of information surplus created from multi-network utilization and has been simulated in the algorithm FLAGS.C. The code listing is given in the Appendix.

Time Window for Virtual Measurements

In the proposed work, neural networks are used as filters and classifiers. Neural network memory is by nature *distributed* and *associative*. This leads to reasonable network response when presented with incomplete, noisy, or previously unseen input. This property is referred to as "generalization". Non-linear multi-layer networks (back-propagation networks) learn the features in their hidden layer and combine these to form "intelligent" responses to novel stimuli. It is in the nature of neural memory to associate new patterns with existing ones, in a manner that can implicitly construct the variable **time**. All the training algorithms include the provision for spatial distribution of information at the neuronal interconnections, but there is no mechanism for associating time labels with the corresponding patterns. Therefore, every neural system is by nature **static**, rather than **dynamic**. On the other hand, the notion of measurement is associated with the variation of system parameters in time. This makes imperative the construction of a measuring device that will be inherently dynamic. It is vital therefore, to address the issue of developing a dynamic scheme which will explicitly incorporate the time dimension into the virtual monitoring technique under development.

In order to include the notion of time into the measurement process, a **rule-based** system has been developed that takes a decision based on the outputs of all firing networks

satisfying the *dissemblance index criterion.* Its operation is dynamic, in the sense it takes into consideration the previous and current responses, as well as predictions about the future from every firing neural network. The acceptance, or rejection, of a network response, depends only on the responses of this particular network in the past, the present, and its predicted values about the future. In this approach the rule-based system acts primarily as an **information fusion device.** It "fuses" information pertaining to a single network rather than combining information supplied from different networks. The decision made is a dynamic one, since the variable time is explicitly incorporated into the decision process. *Therefore, although static tools (ANNs) are used for mapping purposes, it is the information fusion tool (rule-based system) which makes the proposed virtual measuring device, a dynamic one.*

The neural networks employed in this work are trained to predict the position of the valve at the next time step. Thus, they offer an indication at time t, of the possible position of the valve at time $t+\Delta t$. In this manner, the virtual measuring device has an indication of the future state of the system, along with the knowledge of the present, as well as the previous states. It seems therefore, appropriate to incorporate all this amount of information into a rule-based system which will be able to draw a conclusion concerning the present state, supported by knowledge of the past and projected estimation about the future state of the system. The development of the rule-based system requires the determination of the optimum number of previous independent network responses required to carry out an accurate prediction. This number of past time steps is necessary to be consulted, so that a decision about the validity of the present network response, will be made. In order to calculate the required number of previous network responses, statistical tools suitable for model selection have been utilized.

The membership functions describing the state of the valve are produced through a set of pre-trained neural networks receiving five time series depicting the state of HFIR at every moment. Non-linear filters are well known for their property to preserve, as well as, analyze all statistical properties of the incoming signals. Apparently, these statistical characteristics are inherent properties of the system generating the input signals (i.e., HFIR). Thus, models containing as independent variables the time series offer a very good estimation of the statistical properties implied in the membership functions calculated from the neural networks.

Let us assume the following representation of a general feed-back model (Upadhyaya, Kitamura and Kerlin, 1980), referred to as the *multi-variate regressive model*

$$\mathbf{Y}_t = \sum_{i=0}^{n} b_i \mathbf{X}_{t-i} + \varepsilon_t \tag{4.8}$$

where $\mathbf{Y}_t$ is the dependent variable (valve-position), b_i are the coefficients to be estimated, $\mathbf{X}_t$ are the independent variables ($\mathbf{X}_{t-i}$, $i \geq 1$, are lagged values of the training signals), and ε_t is the error term independently and identically normally distributed with a zero mean and a constant variance. There exist numerous criteria (or rules of thumb) for adding or deleting explanatory variables from a model. Statistical techniques dealing with this problem fall into three major categories, namely; parsimony criteria, mean square error criteria, and information criteria.

In parsimony criteria, the coefficient of multiple determination, $\mathbf{R^2}$, is a measure of the proportion of the total variance accounted for by the linear influence of the explanatory variables (Judge, et. al. 1985):

$$R^2 = 1 - \frac{(\mathbf{Y} - \mathbf{X}b)'(\mathbf{Y} - \mathbf{X}b)}{(\mathbf{Y} - \bar{\mathbf{Y}})'(\mathbf{Y} - \bar{\mathbf{Y}})} \tag{4.9}$$

Although $\mathbf{R^2}$ is a measure of the goodness of fit, it has a major weakness since it can be maximized by maximizing the number of regressors. Even the corrected (adjusted) coefficient of multiple determination does not include the losses associated with choosing an incorrect model.

A solution to the above problem is provided by considering a criterion based on some estimate of the mean-square prediction error. One such criterion is the *Mallow conditional mean-square error criterion* (Mallow, 1973), and a second one is the *unconditional mean-square error criterion* (Amemiya, 1980). Assume that the regression model used is

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \boldsymbol{\varepsilon} \tag{4.10}$$

where $\mathbf{X}$ is a NxK matrix of K independent variables with N observations each, and b is $Kx1$ vector. Let $\mathbf{X}_1$ be a NxK_1 matrix and $\mathbf{b}_1$ is a $Kx1$ vector such that $K_1 \leq K$. The model selection consists of choosing among the regression models of $\mathbf{y}$ as a dependent variable and all possible subsets of $\mathbf{X}$, namely $\mathbf{X}_1$, as the regressors. The *mean-square error criteria* (MSE) require minimization of the following quantity

$$\sigma^2\left(1 + \frac{K_1}{N}\right) + \frac{1}{N} b_2' X_2' M_1 X_2 b_2 \tag{4.11}$$

where

$$M_1 = I - X_1(X_1'X_1)^{-1}X_1' \tag{4.12}$$

$\mathbf{X_2}$ is $Nx(K\text{-}K_1)$ matrix with $\mathbf{X} = [\mathbf{X_1}|\mathbf{X_2}]$, and b_2 is a $(K\text{-}K_1)x1$ vector with $b' = [b'_1|b'_2]$. The *Prediction Criterion* (PC) is obtained by replacing σ^2 with its unbiased estimate and minimizing the second term in Equation (4.12) with respect to b_2, namely, setting it equal to zero

$$PC = \hat{\sigma}_1^2\left(1+\frac{K_1}{N}\right) \tag{4.13}$$

where

$$\hat{\sigma}_1^2 = (N-K_1)^{-1} y' M_1 y \tag{4.14}$$

Mallows' Criterion (MC) is obtained by estimating, rather than eliminating, the second term as in the previous criterion. The result is to minimize the following expression

$$MC = \frac{2K_1}{N}\frac{y'My}{N-K} + \frac{y'M_1y}{N} \tag{4.15}$$

where

$$M = I - X(X'X)^{-1}X' \tag{4.16}$$

The information criteria seek to incorporate in model selection the divergent consideration of accuracy of estimation and the "best" approximation to reality. Thus, these criteria involve a statistic that incorporates a measure of the precision of the estimate (as MSE does) and a measure of parsimony (as $\mathbf{R^2}$ does). The *Akaike Information Criterion* (AIC) (Akaike, 1973) is based on the entropy criterion of Kullback-Leibler. The Akaike formula minimizes the following quantity

$$AIC = \ln\frac{y'M_1y}{N} + \frac{2K_1}{N} \tag{4.17}$$

According to the analogy discussed previously and the formulation presented above, five linear multi-regression models were initially developed. The dependent variable is the crisp value of the valve-position, and the independent variables are lagged values from each one of the time series used for neural network training. In a direct analogy to the task performed by the neural network; the time lags of the independent time series do not include the present time step value; where the dependent variable is to be predicted at its present value. Thus, the independent time series are time lagging the dependent variable by one time step. The results obtained from the five independent models considering the criteria for calculating the optimum number of explanatory variables are shown in Table 4.1. The left-most column defines the signal characterizing the independent variables of every multi-regression model.

Table 4.1 Results from multi-regression models with lagged variables only.

Indep. Variable	$\mathbf{R^2}$	(PC)	(MC)	(AIC)	Significant No. of lags	D-W
Flux	96%	0.0357	2.017	-7306	3	0.035
DP	92%	0.0739	3.947	-6407	4	0.093
In_Temp.	19%	0.8115	0.226	-3449	1	0.005
Out_Temp	97%	0.0277	1.196	-7619	2	0.046
Sec_Flow	97%	0.0232	0.846	-7838	2	0.096

In the multi-regression model with inlet temperature as the independent variable, the adjusted $\mathbf{R^2}$ criterion for the optimum values of the other criteria was 19.04%. However, its maximum value was 19.08% identifying a different model than the other criteria with 2 lags. The results obtained from the other models presented an excellent accordance among

the maximum/minimum values attained by all four criteria applied. Considering the results given in column 6 at Table 4.1 we see that the optimum number of time lags required for a global representation is in the vicinity of three. Taking into consideration the need for inclusion of all information contained in the five signals (in other words, playing on the safe side), the number of time lags has been chosen to be four. This choice covers the statistical properties of the average pressure variation (DP) time series, as well as incorporates the statistical characteristics of the remaining signals with lower number of significant time lags. The seventh column in Table 4.1 contains the *Durbin-Watson Statistic* (Judge, et. al., 1985), that is used to test the existence of auto-correlation. This statistic is calculated by the equation

$$d = \frac{\sum_{t=2}^{T} (\hat{e}_t - \hat{e}_{t-1})^2}{\sum_{t=1}^{T} \hat{e}^2_t} \tag{4.18}$$

If the value of the statistic is between 2 and 3 (rule of thumb) then the time series is not characterized by auto-correlation properties. Apparently, the signals used share similar auto-correlation characteristics, with a minor difference of the inlet temperature signal.

As a first step in our effort to check the results obtained from the previous multi-regression models, we have developed a single multi-autoregression model. Here, the dependent variable is once again the valve-position but it is regressed on its lagged values. The underlying idea is based on the exploration of the ability of the crisp values of the valve-position signal to predict themselves. The results obtained applying the same criteria as before, are presented in Table 4.2, where one may see that the number of time lags computed from the combination of criteria applied is once again four. This is in agreement

Table 4.2 Results from multi-autoregression model with lagged variables only.

Indep. Variable	R^2	(PC)	(MC)	(AIC)	Significant No. of lags	D-W
Valve_Pos	99%	0.0007	16.64	-12144	4	2.013

with the assumption made that all signals have been generated in the same physical environment carrying therefore similar statistical properties.

In order to reconfirm that the results obtained up to this point exhibit the required consistency, and support the assumption made, the five multi-regressive models described before have been reconstructed. Only, this time the independent variables include the present time steps also. If the assumption of similar statistical behavior is correct, the new set of multi-regressive models should give similar results to those presented in Table 4.1. Indeed, this is the case as it may be seen in Table 4.3.

It becomes apparent from the results presented in Table 4.3 that the assumption made is a valid one. Apart from calculating the same number of time steps as in the previous case, the models attained almost identical values in the minimization/maximization of the information criteria, supporting the argument of similar statistical characteristics. *In fact, the statistical behavior appears to be preserved throughout the duration of the start-up period. Thus, the "optimum" time lag choice is a dynamic one.* Therefore, the number of time steps (four) concluded from the previous six multi-regression models can be considered as a valid choice.

Numerous other multi-regression models have been developed in order to cover all

Table 4.3 Results from multi-regression models with lagged and present variables.

Indep. Variable	R^2	(PC)	(MC)	(AIC)	No. of time steps	D-W
Flux	96%	0.0354	1.669	-7316	3	0.031
DP	92%	0.0727	5.932	-6427	4	0.076
In_Temp.	19%	0.8115	-0.775	-3449	1	0.003
Out_Temp	97%	0.0276	-0.354	-7623	2	0.040
Sec_Flow	97%	0.0218	1.219	-7913	2	0.071

possible cases and validate the results presented previously. Since further discussion on this issue deviates from the scope of this presentation, a short verbal description will be presented in the following, concerning the most important issues addressed. Two multi-regressive models have been developed where lagged values from all five signals have been utilized as independent variables, in an attempt to capture the cross-products among various signals. The results are difficult to be analyzed since the cross-correlation among different signals creates side effects on the number of time lags calculated. As a result a signal may appear with four significant time lags, where another with only one or two, due to the high correlation existing between the two signals. Thus, the information carried by one signal is adequate for accurate model construction and the information encoded in the other signal is dropped. The correlation among different signals is an issue which will become of prime importance in the interpretation of the results obtained from model testing in the next chapter. Finally, it should be mentioned that all of the multi-regressive models used in this work, have been developed at the SAS® statistical environment (SAS Institute Inc., 1985).

Taking advantage of the neural network ability to preserve all the statistical and physical properties existing in the five time series (which actually are characteristic of the device under consideration), the following analogy has been established. The number of time lags required to give the best prediction for the valve position at the present time step, utilizing linear multi-regression models, is equal to the number of previous network responses needed for the rule-based system. The predictability of a particular time series is an inherent statistical property of the time series (or better, of the physical system itself), which is preserved by the neural network. Thus the number of time lags (four) concluded from the regression models, can be used as the optimum number of lagged independent neural network responses, in order a decision to be made concerning the present response of every neural network.

Since the optimum number of past time steps has been found to be four, a rule-based system has been developed (RULE_BASE10.C) which draws a conclusion concerning the validity of the present neural network response. The rule-based system takes into consideration the responses of each network during the previous four time steps, the present time step, as well as the prediction about the next time step. Considering that six consecutive network responses are included in the decision process, it becomes apparent that there are 64 possible dyadic combinations of these estimations. The dyadic representation of the network responses stems out of the satisfaction, or not, of the *dissemblance index criterion* at every time step.

The rule-based system has been designed in order to abide the following general principle (Barr and Feigenbaum, 1982) (called the **DYNAMIC BEHAVIOR CRITERION**):

IF

FOR EVERY NETWORK ***k***

THE DISSEMBLANCE INDEX CRITERION IS SATISFIED

(1) AT THE PRESENT TIME STEP,

(2) AT THE FUTURE TIME STEP,

(3) DURING AT LEAST THREE CONSECUTIVE TIME STEPS

THEN

THE NETWORK ***k*** *ESTIMATION FOR THE PRESENT TIME STEP MUST BE ACCEPTED*

OTHERWISE

THE NETWORK ***k*** *ESTIMATION FOR THE PRESENT TIME STEP MUST BE DISCARDED*

The above principle has been imposed in order to guarantee consistency in the network responses. The first and second prerequisites of the above criterion require the satisfaction of the present and future values of the *dissemblance index criterion*; so that the present response will be accepted. The third requirement necessitates the appearance of definite trends in the network responses. The scope is to exclude the existence of sporadic satisfactions of the *dissemblance index criterion.* The requirement for consistent network satisfaction of the *dissemblance index criterion* is generated from the mathematical definition of the dissemblance index itself (Kaufmann and Gupta, 1991), e.g., for two fuzzy numbers **X** and **Y** in $\Re$ we define

$$(\mathbf{X} = \mathbf{Y}) \Rightarrow (\delta\ (\mathbf{X}, \mathbf{Y}) = 0)$$

The above property guarantees that the dissemblance index between two fuzzy numbers will be zero if **X** and **Y** are the same. Unfortunately, it does not guarantee the opposite. This, in other words means that one could estimate two fuzzy numbers as equal, based on the value of the dissemblance index, when he should not. The consequence of the previous statement is obvious at once. A network response may satisfy the *dissemblance index criterion* even in the case where the prototype and predicted membership functions are not the same. In order to tackle this issue; the prerequisite of consistent network responses has been set, trying to alleviate the problem of excessive satisfaction of the *dissemblance index criterion* by a network. In this case, the rule-based system will not accept the satisfaction of the *dissemblance index criterion* by a particular network; even in the case of two consecutive satisfactory responses (i.e., present and future), but it will search for a third one also. Furthermore, it is obvious that the principle set gives extra weight on the network predictions about the present and the future in order to accept the present response. The reason for this "bias", once again, lies in the deficiency associated with the use of a **semi-norm**, instead of a **norm** as a "*distance-measurement*" tool. Given the network tendency to over-satisfy the *dissemblance index criterion* (due to the "generalization" property), the *dynamic behavior criterion* for the rule-based system has been designed to stabilize the network prejudice towards satisfactory responses.

The rule-based system operates in a dynamic fashion in a sense that it automatically accepts or discards present responses, and then considers the new values when it draws conclusions during future evaluations. As an example, consider the case where the present network evaluation satisfies the *dissemblance index criterion*, but the rule-based system decides to discard this value according to the *dynamic behavior criterion*. At the next time step when it will consider this value as a previous network response, it will use it as not having satisfied the *dissemblance index criterion*. The same applies vice versa. In order to

model the *dynamic behavior criterion* stated above, a total of 11 propositions, in the form of IF-THEN rules (Maus and Keyes, 1991), is adequate to satisfy all possible response scenarios from an independent network. An example of a rule incorporated into the rule-based system reads as follows:

IF

THE NETWORK RESPONSE DID NOT SATISFY THE DISSEMBLANCE INDEX CRITERION AT THE *(PRESENT - 4)* TIME STEP

AND

THE NETWORK RESPONSE DID NOT SATISFY THE DISSEMBLANCE INDEX CRITERION AT THE *(PRESENT - 3)* TIME STEP

AND

THE NETWORK RESPONSE DID NOT SATISFY THE DISSEMBLANCE INDEX CRITERION AT THE *(PRESENT - 2)* TIME STEP

AND

THE NETWORK RESPONSE DID SATISFY THE DISSEMBLANCE INDEX CRITERION AT THE *(PRESENT - 1)* TIME STEP

AND

THE NETWORK RESPONSE SATISFIES THE DISSEMBLANCE INDEX CRITERION AT THE *(PRESENT)* TIME STEP

AND

THE NETWORK RESPONSE WILL SATISFY THE DISSEMBLANCE INDEX CRITERION AT THE *(PRESENT + 1)* TIME STEP

THEN

THE NETWORK RESPONSE FOR THE *PRESENT* TIME STEP IS **ACCEPTED**

It is obvious from the above description that the rules are composed of six *antecedents* in the IF part and one *consequent* in the THEN part (Charniak and McDermott, 1984). This is a straight consequence of the number of a single network responses considered (six), in order to take a decision about the validity of the present network response.

Information Fusion

Utilizing multiple networks creates an environment, where the decision-making procedure is based on multiple network outputs. This problem is partially solved by the *dissemblance index criterion* introduced in the previous section. Unfortunately our inability to define the exact borders of separate classes, introduces fuzziness in the decision process. It should be stated that although the system or device under investigation is a deterministic one, it is our lack of exact knowledge, which makes the measurement procedure fuzzy. The separation imposed on the universe of discourse (Figure 4.8) is artificial, and therefore the results obtained from different firing networks at every time step, must be combined to offer a logical result. The system can not be simultaneously in two mutually exclusive or even neighboring, states. It seems therefore logical to incorporate in the proposed measuring device, a higher "intelligence" component which will be able to resolve ambiguous situations.

The higher "intelligence" component should be capable to combine the results obtained from different firing networks, and produce a unique membership function. The resultant membership function should contain all the information existing in the membership functions produced from independently firing networks. In order to accomplish this task the *dissemblance index criterion* has been utilized once again. The

underlying idea is based on the grade of satisfaction of the *dissemblance index criterion.* Up to this point, a network prediction is initially accepted or not, based on the satisfaction or not, of the *dissemblance index criterion.* Therefore, according to the discussion presented in Chapter 2, the characteristic equation of membership into the set of satisfactory responses is a dyadic one, i.e., for every network **k**,

$$\mu_{SR_k}(d\_i_k(j)) = d\_i_k(j) \quad \text{iff} \quad d\_i_k(j) \leq max\_d\_i_k \tag{4.19a}$$

$$\mu_{SR_k}(d\_i_k(j)) = 0.0 \quad \text{iff} \quad d\_i_k(j) > max\_d\_i_k \tag{4.19b}$$

where the set of satisfactory responses for every network **k** is noted by SR_k. Going one step further, the characteristic equation is considered as a continuous one, assigning continuous grade of membership to the set of *satisfactory_responses.* In this case, the set SR is fuzzy and a continuous membership function is assigned to it, similar to those already discussed. The grade of membership to the set of *satisfactory_responses* depends on how close to $\mathbf{max\_d\_i_k}$ is every network **k** prediction ($\mathbf{d\_i_k(j)}$), for the input vector **j**. The lower the $\mathbf{d\_i_k(j)}$ value is (i.e., closer to zero), the more accurate the prediction of network **k** is, and thus is associated with a higher level of presumption. The membership function assigning a certain degree of belief to the network **k** prediction which has satisfied the *dissemblance index criterion*, is given by the following equation

$$\mu_{SR_k}(d\_i_k(j)) = \frac{1 - \frac{d\_i_k(j)}{\max\_d\_i_k}}{\sum_k \left(1 - \frac{d\_i_k(j)}{\max\_d\_i_k}\right)} \tag{4.20}$$

where the addition in the denominator holds for all possible firing networks **k**. It is apparent from Equation (4.20), that the membership function is normalized into the interval [0,1]. The membership function $\mu_{\mathbf{SR_k}}$ attains its minimum value when $\mathbf{d\_i_k(j)}$ =

max_d_i$_k$ and its maximum value when the denominator is equal to the nominator (i.e., one firing network only). Obviously, the satisfaction of the *dissemblance index criterion* defines the domain of $\mu_{\mathbf{SR_k}}$, for every network **k**, into the interval [0, **max_d_i$_k$**]. The normalization of $\mu_{\mathbf{SR_k}}$ satisfies two purposes. First, is consistent with the definition of membership functions according to Zadeh (1965) and second, the presumption levels associated with different network responses can be used as *weights*.

Equation (4.20) defines the level of presumption of satisfaction of the *dissemblance index criterion* from network **k**. In other words, it gives the grade of success in the satisfaction of the *dissemblance index criterion* from a particular network **k** response. Therefore, the value of the level of presumption can be used as a membership function modifier. The modification of the prototype membership function which has satisfied the *dissemblance index criterion*, is done by multiplying it with the crisp value identifying the presumption level of satisfaction of the *dissemblance index criterion*. The network predicted membership functions are treated as fuzzy numbers, and their product with the level of presumption of their own satisfactory responses is calculated. The multiplication of a fuzzy number **A** having an interval of confidence $A_\alpha = \left[a_1^{(\alpha)}, a_2^{(\alpha)}\right]$ for the level of presumption $\alpha \in [0,1]$, with a crisp number $K \in \Re$ (Kauffman and Gupta, 1991) is defined as follows

$$K * A_\alpha = [K,K](*)\left[a_1^{(\alpha)}, a_2^{(\alpha)}\right] = \left[Ka_1^{(\alpha)}, Ka_2^{(\alpha)}\right] \tag{4.21}$$

and the result of the multiplication is a new membership function. Assuming the fuzzy number $\mathbf{A} \subset \Re$ has the form

$$\mu_A(x) = a_L * x + b_L \qquad x_1 \le x \le x_2 \tag{4.22a}$$

$$\mu_A(x) = a_R * x + b_R \qquad x_2 \le x \le x_3 \tag{4.22b}$$

we may similarly define the multiplication of fuzzy number **A**, $\forall$ x $\in$ $\Re$, with an ordinary number K $\in$ $\Re$ as

$$\mu_{A*K}(x) = \frac{a_L}{K} * x + b_L \qquad K * x_1 \le x \le K * x_2 \tag{4.23a}$$

$$\mu_{A*K}(x) = \frac{a_R}{K} * x + b_R \qquad K * x_2 \le x \le K * x_3 \tag{4.23b}$$

where for notational simplicity $\mu_{\mathbf{SR_k}}(\mathbf{d\_i_k(j)})$ has been substituted by K.

Thus, every membership function predicted by a neural network which has satisfied the *dissemblance index criterion*, will be multiplied by a crisp number ($\mu_{\mathbf{SR_k}}(\mathbf{d\_i_k(j)})$) associated with the degree of satisfaction of the *dissemblance index criterion* by this membership function. Certainly, in the case we have only one successful network response this number will be equal to one, since the numerator and denominator in Equation (4.20) will be equal. Apparently, this is neither the case of interest, nor the issue required the development of the membership function of *satisfactory_responses*. The major purpose of this approach is focused on the case of successful responses from more than one networks, generated from the particular partition (overlapping membership functions) imposed on the universe of discourse. In this case the predicted by the network membership functions will be initially multiplied by the degree of satisfaction of the *dissemblance index criterion*. The resultant membership functions will be added in the universe of discourse, creating a new membership function which will uniquely and unambiguously describe the state of the system at every time step.

The membership functions resulting from the multiplication of the prototype membership functions with the degree of satisfaction of the *dissemblance index criterion* ($\mu_{SR_k}(d\_i_k(j))$) are treated as fuzzy numbers, and may be added in the universe of discourse, creating a new fuzzy number. The addition of two fuzzy numbers **A** and **B**, with A_α and B_α their intervals of confidence for the level of presumption α, $\alpha \in [0,1]$, (Kauffman and Gupta, 1991) is defined as

$$A_\alpha(+)B_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}](+)[b_1^{(\alpha)}, b_2^{(\alpha)}] = [a_1^{(\alpha)} + b_1^{(\alpha)}, a_2^{(\alpha)} + b_2^{(\alpha)}] \tag{4.24}$$

In order to compute the summation of two fuzzy numbers **A** and **B** defined below as

$$\mu_A(x) = a_{1L} * x + b_{1L} \qquad x_1 \le x \le x_2 \tag{4.25a}$$

$$\mu_A(x) = a_{1R} * x + b_{1R} \qquad x_2 \le x \le x_3 \tag{4.25b}$$

$$\mu_B(x) = a_{2L} * x + b_{2L} \qquad x_5 \le x \le x_6 \tag{4.26a}$$

$$\mu_B(x) = a_{2R} * x + b_{2R} \qquad x_6 \le x \le x_7 \tag{4.26b}$$

it is necessary first to compute the intervals of confidence for each level α. Equations (4.25a) and (4.25b) give,

$$\alpha = a_{1L} * a_1^{(\alpha)} + b_{1L} \tag{4.27a}$$

and

$$\alpha = a_{1R} * a_2^{(\alpha)} + b_{1R} \tag{4.27b}$$

Hence, the interval of confidence at the level α is given by

$$A_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] = \left[\frac{\alpha - b_{1L}}{a_{1L}}, \frac{\alpha - b_{1R}}{a_{1R}}\right] \tag{4.28}$$

Similarly, from (4.26a) and (4.26b),

$$\alpha = a_{2L} * b_1^{(\alpha)} + b_{2L} \tag{4.29a}$$

and

$$\alpha = a_{2R} * b_2^{(\alpha)} + b_{2R} \tag{4.29b}$$

Therefore

$$B_\alpha = [b_1^{(\alpha)}, b_2^{(\alpha)}] = \left[\frac{\alpha - b_{2L}}{a_{2L}}, \frac{\alpha - b_{2R}}{a_{2R}}\right] \tag{4.30}$$

The summation of (4.27) with (4.29) gives the interval of confidence of the resultant membership function,

$$A_\alpha(+)B_\alpha = [a_1^{(\alpha)} + b_1^{(\alpha)}, a_2^{(\alpha)} + b_2^{(\alpha)}] = \left[\frac{\alpha - b_{1L}}{a_{1L}}, \frac{\alpha - b_{1R}}{a_{1R}}\right](+)\left[\frac{\alpha - b_{2L}}{a_{2L}}, \frac{\alpha - b_{2R}}{a_{2R}}\right] \Leftrightarrow$$

$$A_\alpha(+)B_\alpha = \left[\frac{\alpha - b_{1L}}{a_{1L}} + \frac{\alpha - b_{2L}}{a_{2L}}, \frac{\alpha - b_{1R}}{a_{1R}} + \frac{\alpha - b_{2R}}{a_{2R}}\right] \tag{4.31}$$

Equation (4.31) gives

$$\mu_{A(+)B}(x) = \frac{a_{1L} * a_{2L}}{a_{1L} + a_{2L}} * x + \frac{b_{1L} * a_{2L} + b_{2L} * a_{1L}}{a_{1L} + a_{2L}} \qquad x_1 + x_5 \le x \le x_2 + x_6 \tag{4.32a}$$

$$\mu_{A(+)B}(x) = \frac{a_{1R} * a_{2R}}{a_{1R} + a_{2R}} * x + \frac{b_{1R} * a_{2R} + b_{2R} * a_{1R}}{a_{1R} + a_{2R}} \qquad x_2 + x_6 \le x \le x_3 + x_7 \qquad (4.32b)$$

Equations (4.32a) and (4.32b) describe the resultant membership function from the addition of membership functions μ_A and μ_B. Apparently, the summation of more membership functions takes place in a similar manner. The modification of the prototype membership function based on the grade of satisfaction of the *dissemblance index criterion*, as well as the addition of the resultant membership functions, are modeled in a computer code called RULE_BASE20.C. The code is executed simultaneously with the RULE_BASE10.C code and its listing is given in the Appendix.

From the previous discussion it becomes apparent that a dynamic scheme has been developed for combining the responses obtained from different firing neural networks. The *dissemblance index* and *dynamic behavior criteria* establish the base line for appropriate network responses. Based on the satisfaction of the *dissemblance index criterion*, a membership function is devised which serves the purpose of fuzzifying the results obtained from the neural networks and the rule-based system. Thus, the network responses themselves are used as a measure of the degree of truth existing into the calculated membership functions. *In this sense the inherent imprecision in the system description has been utilized in order to configure the most appropriate shape and position of the membership functions in the universe of discourse.* The initial choice of membership functions serves only as a library of prototype membership functions. Their position in the universe of discourse, as well as, their final shape are concluded from the statistical and dynamic idiosyncrasies of the system to be modeled. *The decision-making procedure is encoded into the optimization algorithm developed.* The dynamic rearrangement of the

membership functions depends on the physical characteristics of the problem at every instant in time.

The requirement for a dynamic representation of the universe of discourse can be seen through the following example. A ten year old boy can be considered as a very young man, a middle-aged child, or a very old baby, simultaneously. The exact identity (or class), as well as its location in the universe of discourse, depends on the context of the linguistic description imposed. The fuzzy variable **age** can take different fuzzy values described by numerous membership functions. Therefore, the issue of partitioning the universe of discourse remains to be addressed. A dynamic algorithm for configuring the appropriate partition of the universe of discourse resolves this issue, since it calculates the best "fit" at every instant in time. Thus, the optimization algorithm renders the universe of discourse dynamic, through the continuous reconstruction of the membership functions. The number of membership functions reconstructed at the universe of discourse at every time step, is not constant but depends on the statistical behavior of the monitored device.

The statistical characteristics of the original signals have been considered in the development of the rule-based system. In addition, these characteristics are employed in the calculation of the actual shape and position of the prototype membership functions in the universe of discourse. The transformation of the dyadic characteristic equation of the set of responses satisfying the *dissemblance index criterion*, to a continuous membership function, offers the advantage of modifying the prototype membership functions in the universe of discourse, and combine responses supplied from different firing networks. The introduction of the membership function of *satisfactory_responses* comes in full accordance with our effort to describe non-accurate information with fuzzy sets instead of random noise.

The results obtained from the simulation of the position of the Secondary Flow Control Valve are summarized in the following figures. The resultant membership functions, as they have been predicted from the virtual measuring device, are plotted with a frequency of one every ten time steps. Figure 4.11 presents the virtual measurements for the first 200 time steps where the valve is initially identified at the **closed** position and is gradually "re-named" as **partially_closed** and **medium**. These predictions agree with the crisp values of the valve position obtained from HFIR measurements (Figure 4.7). At around 80 time steps after the start-up initiation, the valve has been opened almost half way (Figure 4.7), in order to give the necessary initial boost to the system. The virtual measuring device very accurately perceived this valve position as **medium**, and immediately after it identified the valve position as **partially_closed**, and later **closed**, in accordance to the actual valve position. Following the sharp closure of the valve, a new opening procedure is initiated which is also recorded by the virtual measuring device. After these two abrupt valve-position changes, the valve is temporarily stabilized in the **partially_closed** position according to the characterization assigned from the virtual measuring device. Apparently, the virtual measurements come in full accordance with the crisp values (Figure 4.7).

At the next time interval presented here (times steps 210-400) (Figure 4.12), the valve position is repeatedly characterized as **partially_closed** from the virtual measuring device. This characterization coincides with the crisp values of the valve position which have been measured at around 0.3 (Figure 4.7). The variation in shape and location of the membership functions result from the implementation of the optimization algorithm developed. The resultant membership functions are shifted towards higher values in the universe of discourse, close to the four hundredth time step, as a result of the slight increase of the valve position (Figure 4.7). This brings the crisp value in the support of

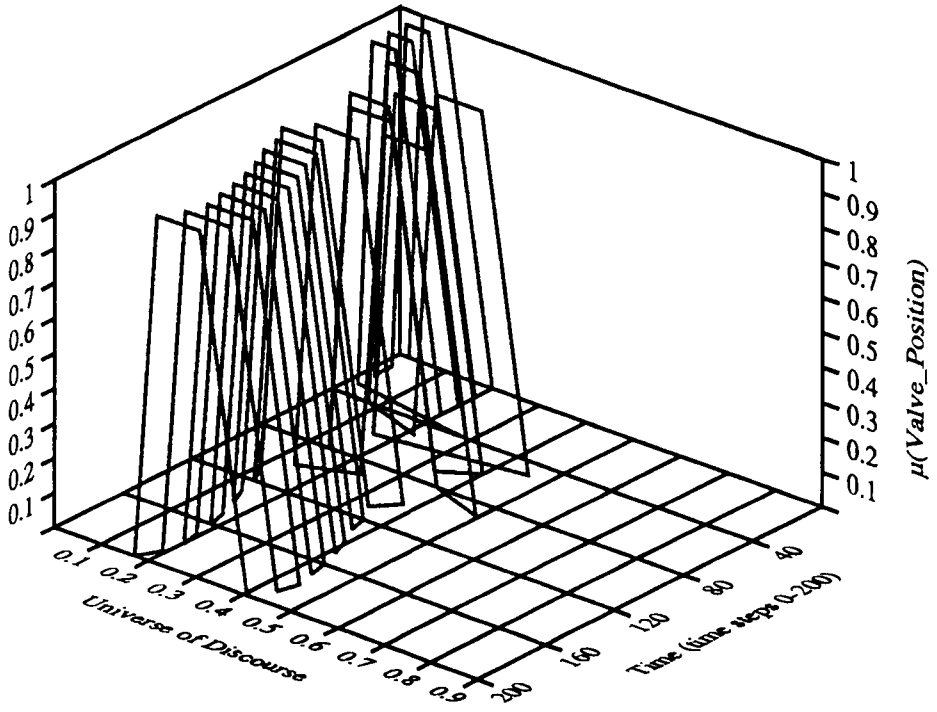

Figure 4.11 $\mu_{VALVE\_POSITION}$ during time steps 0-200.

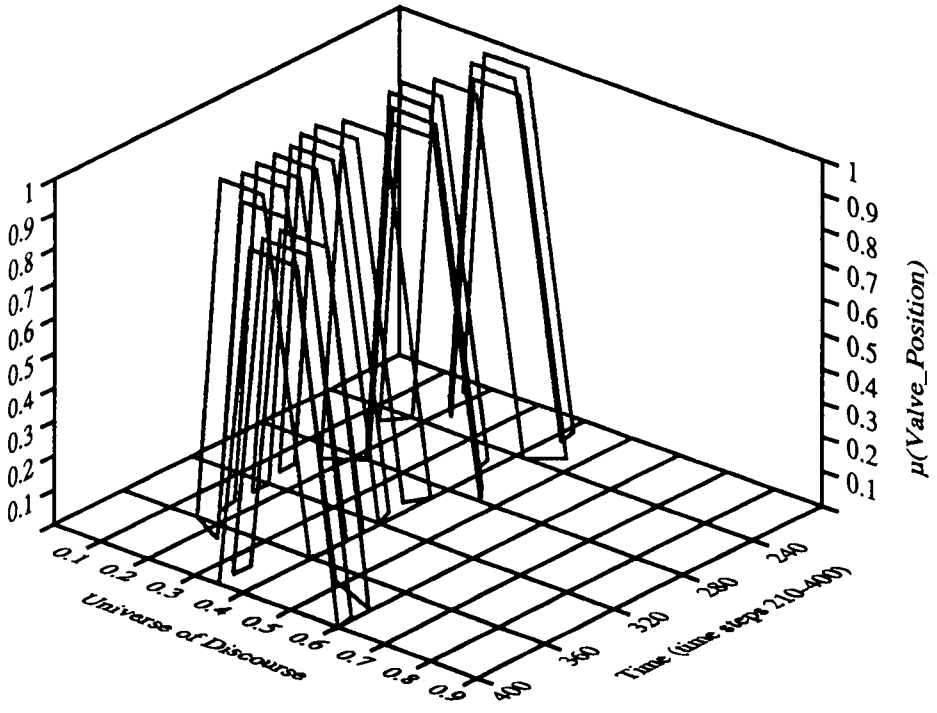

Figure 4.12 $\mu_{VALVE\_POSITION}$ during time steps 210-400.

two adjacent membership functions (**partially_closed** and **medium**), triggering the mechanism for calculating the optimum membership function through the combination of two network responses.

Proceeding to the next time interval (time steps 410-600) (Figure 4.13), the virtual measuring device describes the position of the valve initially as **medium**, and later as **partially_open**. The behavior predicted by our device is identical to the crisp values recorded in the reactor. In Figure 4.7 we may see that a new opening procedure has been initiated at the beginning of this time interval, increasing the opening of the valve to a crisp value close to 0.7 and then stabilizing it at the requested level. The virtual measuring device accurately reiterates the same characterization (**partially_open**) for the same period of time.

At the next time interval (time steps 610-800) (Figure 4.14), the valve position is characterized initially as **partially_open** and is later stabilized to the fuzzy value **open**. This is a straight consequence of further opening the valve during the same time period (Figure 4.7). The operators gradually proceed to a fully-open position by increasing the valve opening through a step increase procedure. This procedure is recognized by the virtual measuring device and is depicted through the calculation of the most appropriate membership function.

The next two time intervals (time steps 810-1000 and 1010-1240) (Figures 4.15 and 4.16) can be analyzed simultaneously. It is apparent from Figure 4.7 that the operators continue increasing the opening of the valve, until it reaches the fully-open position, and then stabilize it there. The virtual measuring device depicts all the attained valve positions as **open** exhibiting an excellent accuracy with respect to the crisp values of the position of

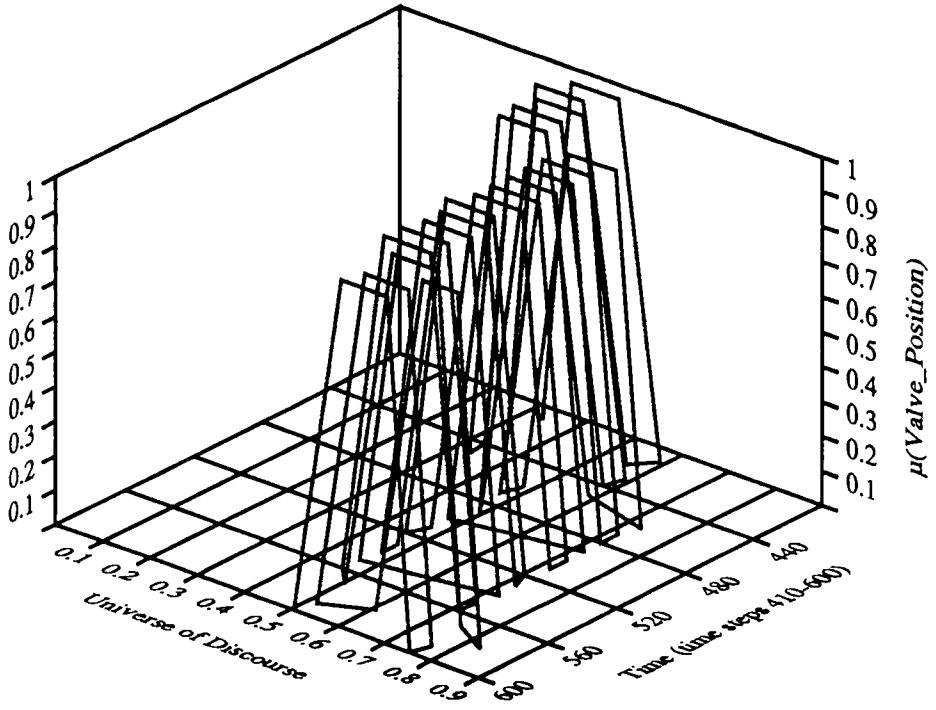

Figure 4.13 $\mu_{VALVE\_POSITION}$ during time steps 410-600.

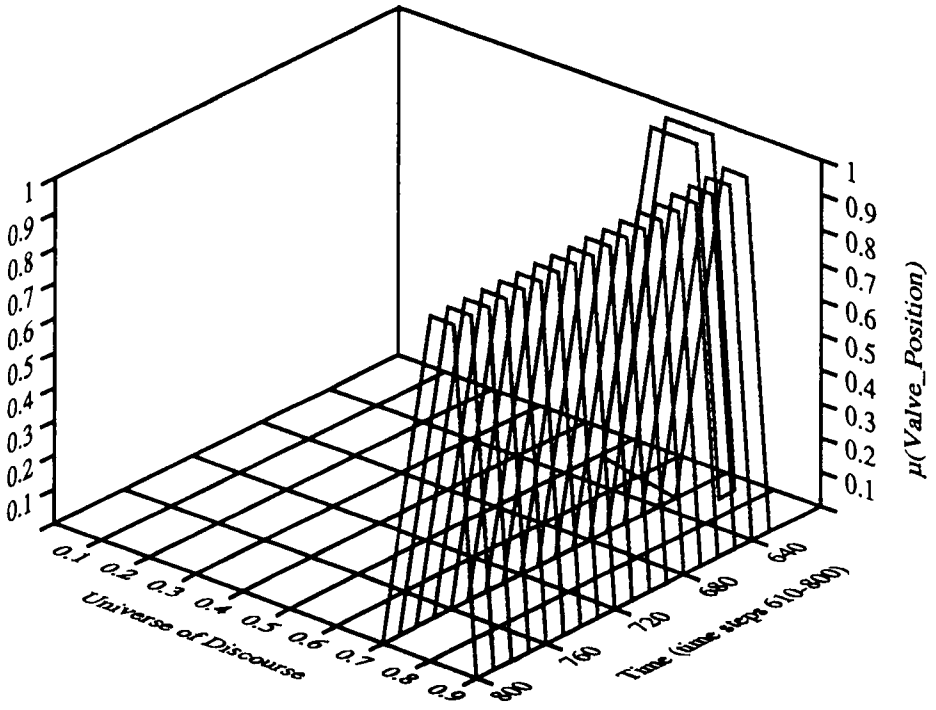

Figure 4.14 $\mu_{VALVE\_POSITION}$ during time steps 610-800.

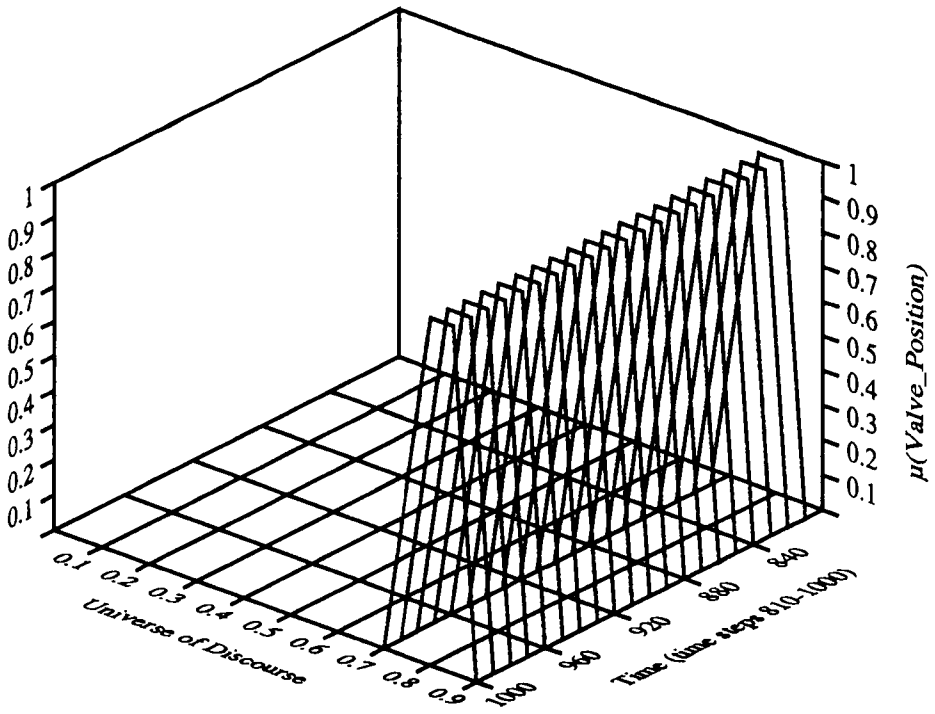

Figure 4.15 $\mu_{VALVE\_POSITION}$ during time steps 810-1000.

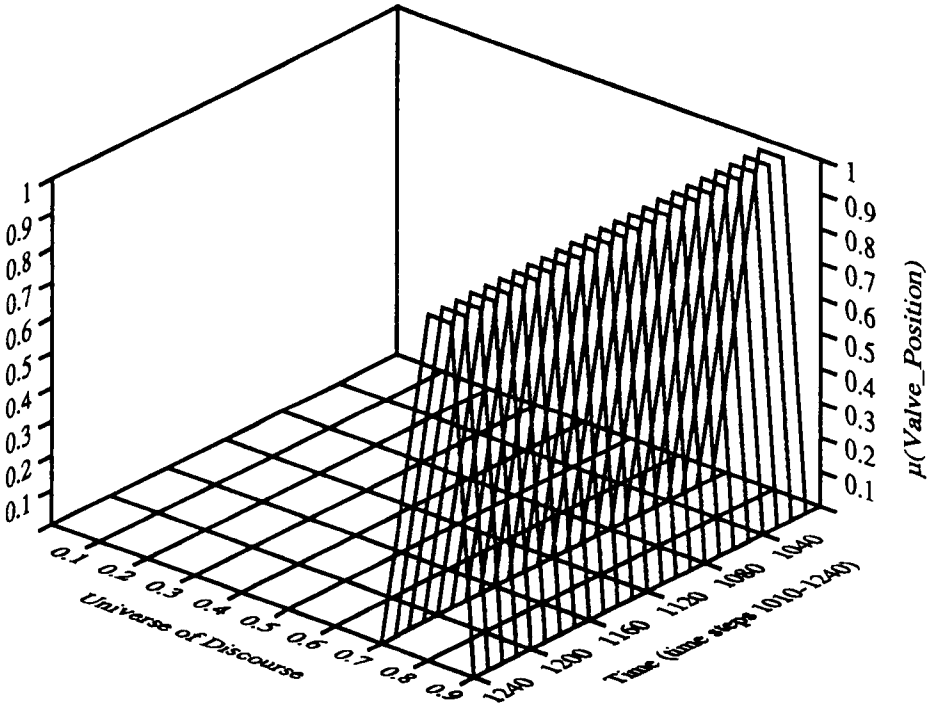

Figure 4.16 $\mu_{VALVE\_POSITION}$ during time steps 1010-1240.

the valve.

A number of points ought to be made from the model behavior described in the previous paragraphs. First, one should notice the excellent accordance between the crisp values of the valve position and the linguistic description assigned by the virtual measuring device. The methodology developed is capable of accurately depicting the exact valve position, following its changes throughout the simulation time span. All the transitions from one state to a different one, were accurately recorded and assigned the most appropriate class identification, based upon the optimization algorithm developed. In addition, the system has been proven capable of following, and accurately identifying, sharp transitions among different system states as those appearing in the first time interval of interest (Figure 4.11).

CHAPTER 5

MODEL TESTING

Introduction

In order to explore the potential and limitations of the virtual measuring device developed in Chapter 4, a number of tests has been devised. A common practice for testing the robustness of a neural network-based system is the application of noise in the original signals used for training the neural networks. This comes in accordance with most "real world" applications, where the existence of noise in measurements acquired from measuring devices is a common phenomenon. Therefore, it becomes imperative to identify the capability of the virtual measuring device to handle noisy signals and recognize system information embedded into noisy environments.

One of the major difficulties encountered in the utilization of gauges for monitoring system parameters, is the recognition of a faulty signal. Quite often in industrial applications there are signals that do not correspond to the actual state of the system. One approach to overcome this problem, is the introduction of more than one measuring device for the same system parameter. Thus, through gauge abundance a faulty signal can be identified applying cross-checking among signals obtained from different measuring devices. The virtual measuring device developed does not have the luxury of more than one signals for the same parameter. Therefore, it is necessary to investigate the abilities of the virtual measuring device to sustain faulty signals and calculate an accurate estimation of

the state of the virtual variable. Susceptibility to faulty signals would significantly affect the accuracy of the predicted membership functions.

Closing the introductory remarks concerning the tests applied to the methodology developed, a final comment on a common testing technique found in the bibliography ought to be made (Takagi and Hayashi, 1991). Researchers in the area of artificial neural networks prefer to withhold a number of patterns from the original signals used to train the networks, in order to use them as test vectors. In the present application this tactic has been avoided, since the results obtained from this type of tests are not representative of the neural network reactions to novel stimuli. Neural networks "learn" patterns - instead of specific vectors - existing into the training set, and thus when presented with a part of the training set not previously "seen", they respond with excellent accuracy. However, this does not constitute satisfactory performance since the testing vectors are already partially known to the trained neural network. Thus, this type of novel stimuli, is not novel at all. By rejecting this approach, different HFIR start-up scenarios obtained during miscellaneous fuel cycles have been used. These scenarios represent completely new patterns, since a number of system parameters behave in dissimilar ways.

10% noise into all training signals

As a first step in the quest for discovering the limitations of the methodology for virtual measurements, we have introduced 10% noise into all five input signals. A computer code called NOISE.C, has been developed for introducing different levels of noise into the input signals, and its listing is given in the Appendix. We tested the system supplying the noisy time series to the set of five neural networks and the results are shown

in Figures 5.1 and 5.2. In Figure 5.1 the responses of the virtual measuring device are plotted with a frequency of appearance equal to one every ten time steps for a time span covering the first 200 steps of the start-up duration. The reason for selecting this particular time span can be seen in Figure 4.7, where the crisp values of the position of the valve are plotted vs. time. It is obvious that the highest activity in the alteration of the valve-position takes place during this time span and thus it is worthwhile to examine the virtual measuring device responses during the same time interval. In Figure 5.1 one may see that the predictions about the valve-position coincide with the crisp measurements obtained from HFIR. The two abrupt openings of the valve at around 80 and 100 time steps after the start-up initiation have been accurately recorded and the valve-position has been identified as **medium**. Considering that the crisp measurements of the position of the valve attain a maximum value close to 0.5 we deduce from Figure 4.8 that the characterization is correct. As it has been expected, the resultant membership functions do not concur exactly to those existing in the library of prototype membership functions, as a result of the activation of the optimization algorithm.

A quick look at Figure 5.2 is enough to show the accuracy of the predictions throughout the start-up period that is depicted here with one membership function every 100 time steps. Almost 700 time steps after the transient initiation the valve-position is characterized as **open**, and the same characterization is repeated thereafter. By taking into account that at the same moment the crisp value of the valve-position becomes 0.8 and later increases to 0.9 (Figure 4.7), it may be concluded that the identification of the state of the valve is accurate. Furthermore, during the time interval between time steps 200 and 700, the state of the valve is initially defined as **medium** and later as **partially_open**, following the transition of the valve-position from one state to another (Figure 4.7).

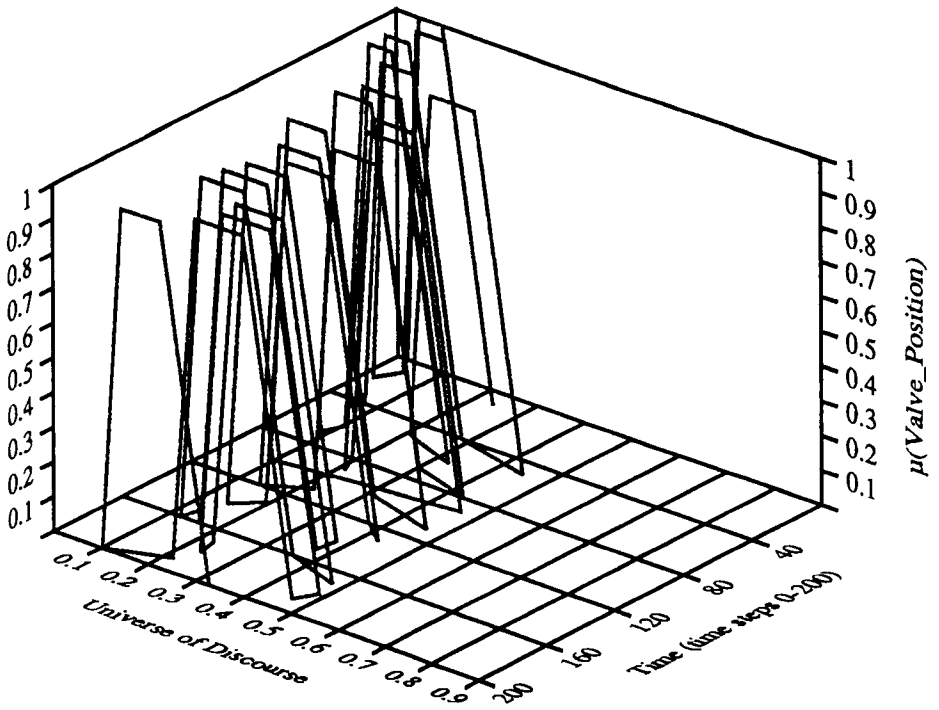

Figure 5.1 Virtual measurements for time steps 0-200 with 10% noise in all signals.

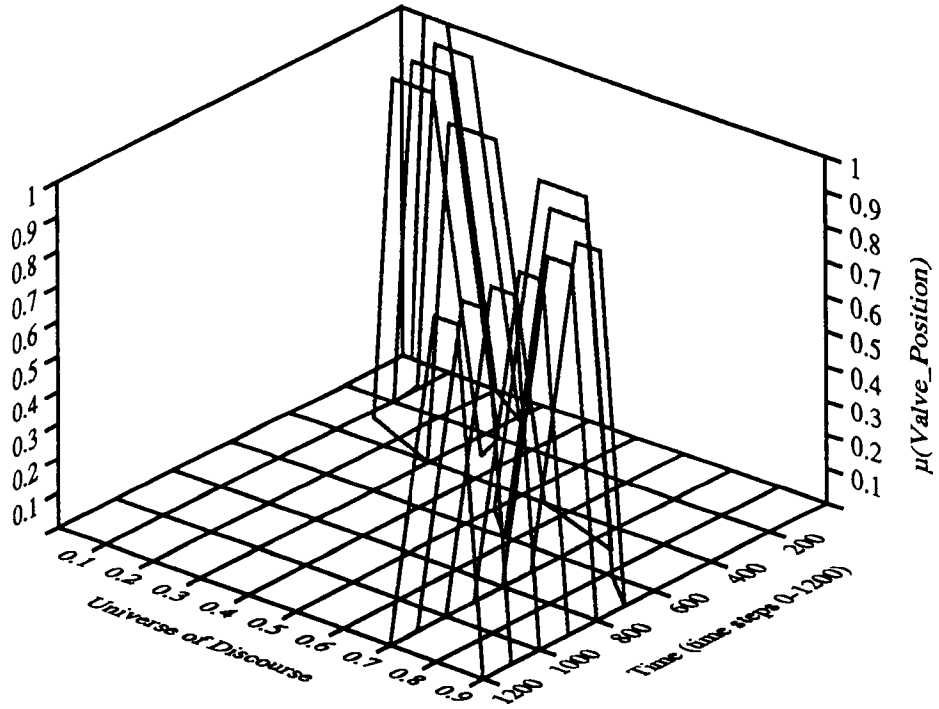

Figure 5.2 Virtual measurements for time steps 0-1200 with 10% noise in all signals.

In order to check the accuracy of the responses obtained from the virtual measuring device under noisy conditions, it is important to compare them with the characterizations assigned to the state of the valve when the testing signals were noise-free. The latter have been shown and analyzed in Chapter 4, in Figures 4.11 to 4.16. The comparison will take place in a way suitable for fuzzy control applications.

The fuzzy control algorithm developed initially by Mamdani (1974) and later improved by King and Mamdani (1977), follows a two stage process. The first stage consists of fuzzification of the crisp values corresponding to system parameter measurements. The resultant fuzzy numbers are combined through the application of fuzzy IF-THEN rules and a fuzzy number is calculated which defines the action suggested by the fuzzy controller. In order to apply the fuzzy action, a de-fuzzification process has to take place. This process is based on the calculation of the centroid of the resultant membership function. Thus, the coordinates of the centroid of the membership function will define the crisp value of the action suggested by the control algorithm. It becomes apparent from the previous discussion that the centroid of the membership functions plays a more important role than the membership functions themselves, in the application of fuzzy control algorithms.

Since the application of the proposed methodology to the area of fuzzy control is of immediate interest, the coordinates of the centroids of the membership functions predicted from the virtual measuring device have been calculated at every time step. Thus, the Euclidean distance between the centroids of the model membership functions and the centroids of the membership functions predicted during testing scenarios, will be the measurement of prediction accuracy provided by the virtual measuring device operating in an "unfriendly" environment.

The centroid, or in other words the center of mass, (or center of gravity) of a system is the point where the sum of all the moments of the particles composing the system is equal to zero (Feynman, Leighton and Sands, 1963). For a 2-D system, we define the moments M_x and M_y with respect to the x-axis and the y-axis, respectively, as

$$M_x = \sum_{i=1}^{n} m_i y_i \tag{5.1}$$

and

$$M_y = \sum_{i=1}^{n} m_i x_i \tag{5.2}$$

where m_i is the mass of each of the n particles located at points $P_1(x_1,y_1),..., P_n(x_n,y_n)$, in a coordinate plane. The result of the following summation

$$m = \sum_{i=1}^{n} m_i \tag{5.3}$$

is called the total mass of the system. The center of mass (or center of gravity) of the system is the point $P = (\bar{x}, \bar{y})$ such that

$$m\bar{x} = M_y \qquad \text{and} \qquad m\bar{y} = M_x \tag{5.4}$$

Let us consider a lamina that is homogeneous, (i.e., has constant density) (Swokowski, 1983). We want to define the center of gravity in a sense analogous to that of systems of particles. We assume that the shape of the lamina is defined by a continuous function f on the closed interval [a,b]. For the homogeneous lamina of area density ρ, we define the moments

$$M_x = \rho \int_a^b \frac{1}{2} f(x) f(x) dx \tag{5.5}$$

and

$$M_y = \rho \int_a^b x f(x) dx \tag{5.6}$$

along with the mass

$$m = \rho \int_a^b f(x) dx \tag{5.7}$$

The center of mass of the lamina $P = (\bar{x}, \bar{y})$ can be found by substituting Equations (5.5) through (5.7) into Equations (5.4), i.e.,

$$\bar{x} = \frac{M_y}{m} = \frac{\rho \int_a^b x f(x) dx}{\rho \int_a^b f(x) dx} \quad \text{and} \quad \bar{y} = \frac{M_x}{m} = \frac{\rho \int_a^b \frac{1}{2} f(x) f(x) dx}{\rho \int_a^b f(x) dx} \tag{5.8}$$

In order to calculate the center of mass of a trapezoidal fuzzy number **A** defined as

$$\mu_A(x) = a_L x + b_L \qquad x_1 \le x \le x_2 \tag{5.9a}$$

$$\mu_A(x) = 1 \qquad x_2 \le x \le x_3 \tag{5.9b}$$

$$\mu_A(x) = a_R x + b_R \qquad x_3 \le x \le x_4 \tag{5.9c}$$

we need to rewrite Equations (5.5) through (5.7) in the form

$$M_x = \rho \left[\int_{x_1}^{x_2} \frac{1}{2} (a_L x + b_L)^2 dx + \int_{x_2}^{x_3} \frac{1}{2} dx + \int_{x_3}^{x_4} \frac{1}{2} (a_R x + b_R)^2 dx \right] \tag{5.10}$$

$$M_y = \rho\left[\int_{x_1}^{x_2} x(a_L x + b_L)dx + \int_{x_2}^{x_3} x dx + \int_{x_3}^{x_4} x(a_R x + b_R)dx\right] \tag{5.11}$$

$$m = \rho\left[\int_{x_1}^{x_2} (a_L x + b_L)dx + \int_{x_2}^{x_3} dx + \int_{x_3}^{x_4} (a_R x + b_R)dx\right] \tag{5.12}$$

and substitute the results of the integrations into Equations (5.8). Carrying out the above integrations we get

$$\bar{x} = \frac{\frac{1}{3}a_L(x_2^3 - x_1^3) + \frac{1}{2}b_L(x_2^2 - x_1^2) + \frac{1}{2}(x_3^2 - x_2^2) + \frac{1}{3}a_R(x_4^3 - x_3^3) + \frac{1}{2}b_R(x_4^2 - x_3^2)}{\frac{1}{2}a_L(x_2^2 - x_1^2) + b_L(x_2 - x_1) + (x_3 - x_2) + \frac{1}{2}a_R(x_4^2 - x_3^2) + b_R(x_4 - x_3)} \tag{5.13}$$

and

$$\bar{y} = \frac{\frac{1}{6}a_L^2(x_2^3 - x_1^3) + \frac{1}{2}b_L^2(x_2^2 - x_1^2) + \frac{1}{2}a_L b_L(x_2^2 - x_1^2) + \frac{1}{2}a_R(x_3 - x_2)}{\frac{1}{2}a_L(x_2^2 - x_1^2) + b_L(x_2 - x_1) + (x_3 - x_2) + \frac{1}{2}a_R(x_4^2 - x_3^2) + b_R(x_4 - x_3)} +$$

$$+ \frac{\frac{1}{6}a_R^2(x_4^3 - x_3^3) + \frac{1}{2}b_R^2(x_4^2 - x_4^2) + \frac{1}{2}a_R b_R(x_4^2 - x_3^2)}{\frac{1}{2}a_L(x_2^2 - x_1^2) + b_L(x_2 - x_1) + (x_3 - x_2) + \frac{1}{2}a_R(x_4^2 - x_3^2) + b_R(x_4 - x_3)} \tag{5.14}$$

The algebraic formulation developed for the calculation of the coordinates of the centers of gravity of the membership functions has been included in a computer algorithm called CENTROID.C. The listing of the code is given in the Appendix.

The centers of mass of the membership functions predicted by the model under normal operating conditions (Chapter 4) have been calculated, and the results are to be compared with those calculated during the tests. For the first test discussed here, the results of the comparison are shown in Figure 5.3. The Euclidean distance is calculated between the centers of gravity of the membership functions predicted during normal operating conditions and those with noisy input signals.

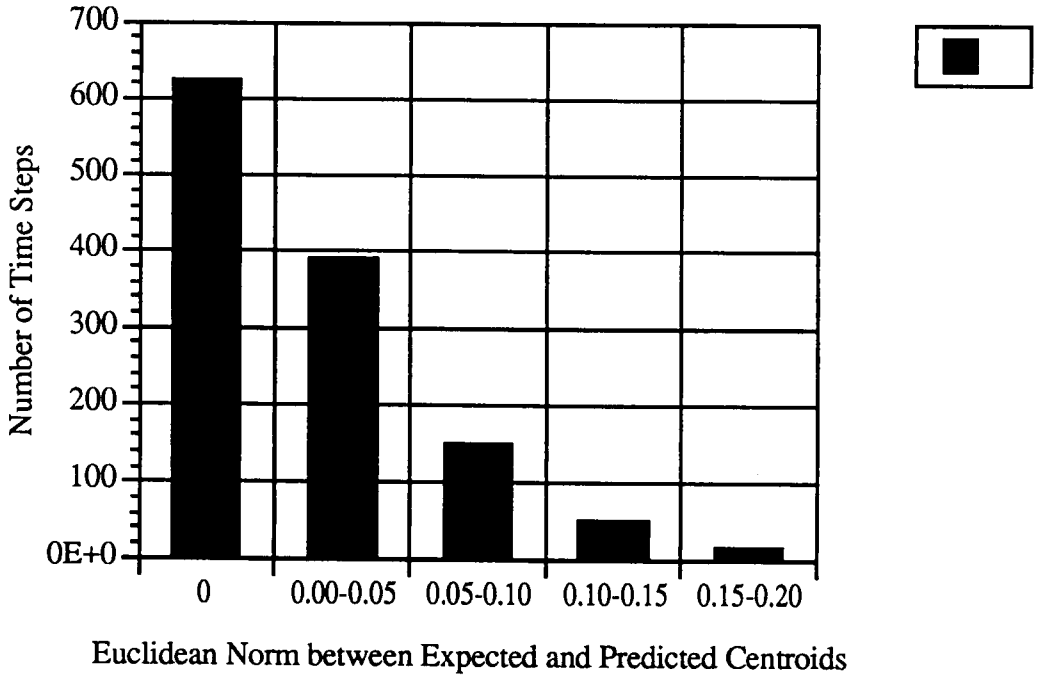

Figure 5.3 Distribution of the Euclidean norms between expected and predicted centroids with 10% noise in all input signals.

In Figure 5.3 we see that more than half of the membership functions have been predicted with 100% accuracy. In addition, one third of them has been calculated with a norm, between expected and predicted values, equal to or less than 0.05. For the remaining time steps the membership functions have been estimated with an Euclidean distance from the expected values, equal to or less than 0.20. The average Euclidean distance for the total start-up duration has been computed equal to 0.0237. The results

presented exhibit an excellent noise tolerance and render the system noise robust. The application of 10% noise into all input signals had a negligible effect on the system response as it can be seen from the average Euclidean norm between the centroids of the noise-free and "noisy" membership functions. This result is in accordance to our expectations, since the neural networks have been trained with noisy signals. We have shown in Chapter 4 that all five signals used for training the neural networks were already embedded into significant amount of noise. This is a result of the utilization of actual reactor signals, instead of simulation data. Thus, the neural networks were already "familiar" with noise characteristics and "learned" the underlying signal patterns, ignoring the extra amount of noise introduced in the input signals.

Partial information elimination

One of the main points behind the development of the virtual measuring device, is its ability to measure values associated with non-directly measurable parameters. In order to perform this task, the virtual measuring device depends on information supplied from other system parameters and infers a linguistic description of the state of the system under investigation. Thus, it seems important to explore the abilities of the virtual measuring device to execute its task when it is supplied with partially correct information. Failure of one or more measuring devices is a rather common phenomenon in industrial applications. As a consequence false information may be recorded concerning the status of the system. The problem during this situation is that the measuring device records totally random values without a particular pattern, and thus it becomes very difficult to detect and isolate the faulty signal. In case the recordings were constant repetition of the same values the task of false signal detection would be a trivial one. In order to simulate this situation as realistically as

possible, we substituted each one of the five time series with a signal corresponding to 100% random noise. One of the original signals is completely dropped and a random (created by a random number generator) time series replaces it. We use this random signal along with the other four original signals in order to test the abilities of the virtual monitoring device to successfully operate with partially appropriate information. Figure 5.4 gives a schematic representation of a typical random signal used for the purposes of this study.

As a first step towards the search of the fault-tolerance capabilities of the virtual measuring device, we have substituted the average neutron flux signal (Figure 4.2) with a totally random one, generated from the random number generator modeled in the algorithm NOISE_SIG.C. We have supplied the virtual measuring device with vectors containing the other four signals in their original form and the random noise in the place of the average neutron flux. The results are plotted, in a similar to the previous case manner, in Figures 5.5 and 5.6.

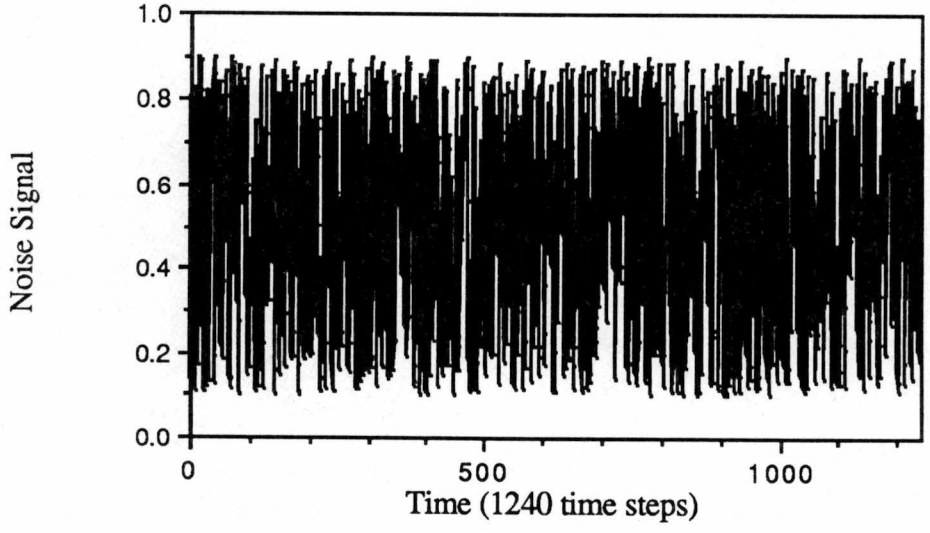

Figure 5.4 Typical random signal used for substituting the training signals.

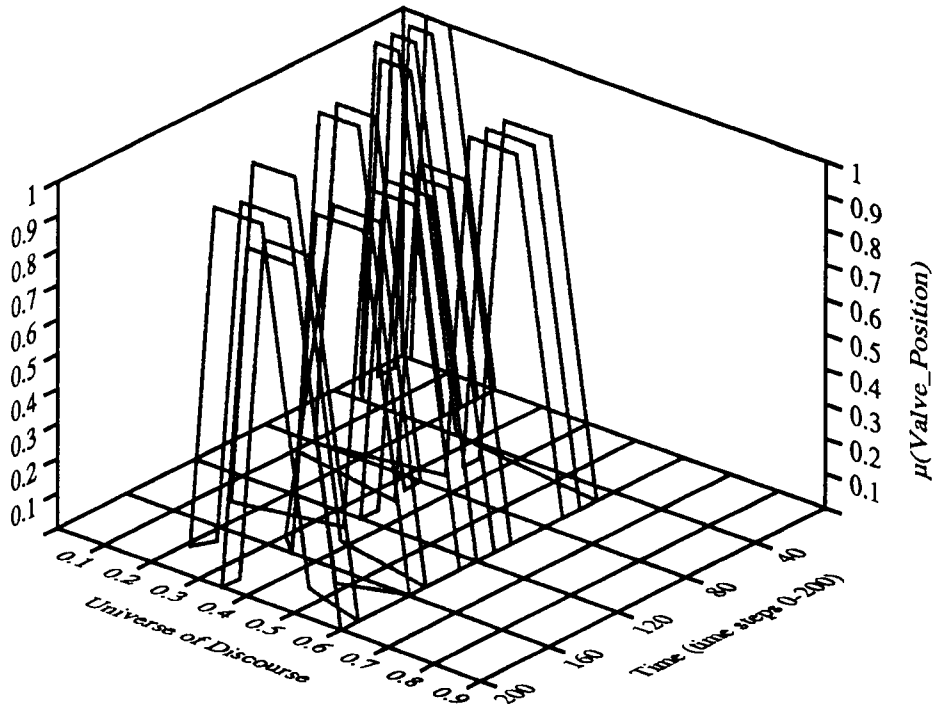

Figure 5.5 Virtual measurements for time steps 0-200 when the average flux signal has been substituted with 100% noise.

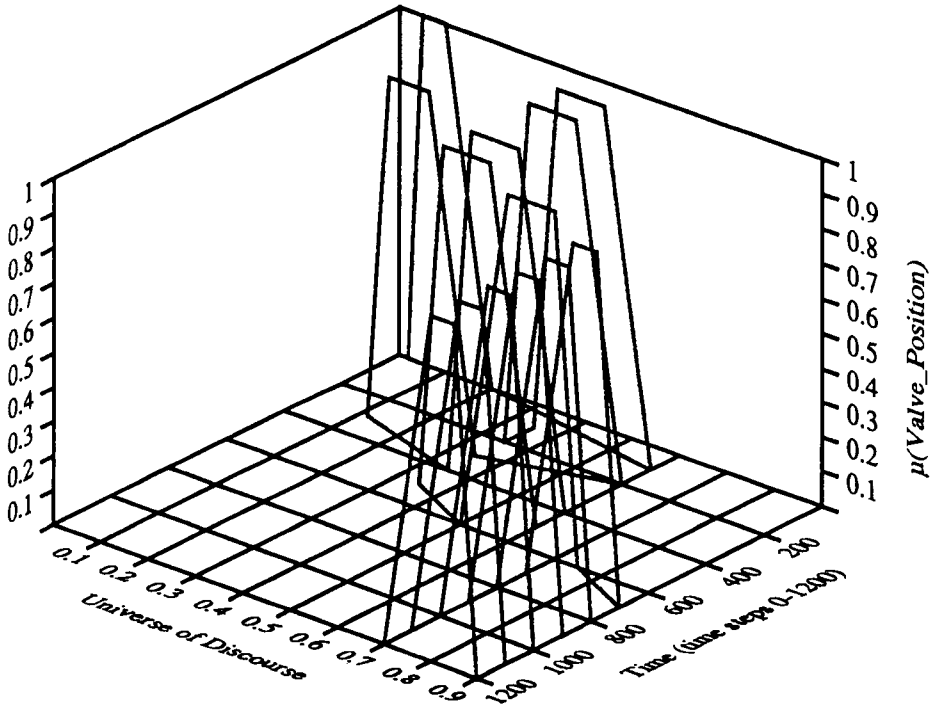

Figure 5.6 Virtual measurements for time steps 0-1200 when the average flux signal has been substituted with 100% noise.

In the next test we have omitted the average pressure variation signal (DP) (Figure 4.3) and used a random signal similar to the one shown in Figure 5.4. The resultant membership functions are shown in Figures 5.7 and 5.8 as they have been recorded from the virtual measuring device. Figure 5.7 presents the first 200 indications with a frequency of one every ten time steps, where Figure 5.8 shows the total simulation period with a membership function drawn every 100 time steps.

A third step in the similar direction is the replacement of the average inlet-temperature signal (Figure 4.4) with a random one, where the rest four signals used in their original form. The same testing procedure has been repeated and the measurements acquired are exhibited in Figures 5.9 and 5.10.

Resetting the average inlet-temperature signal in its original form, the average outlet-temperature signal (Figure 4.5) is substituted by a random one. The recordings obtained in the form of membership functions are shown in Figures 5.11 and 5.12.

Finally, the secondary-flow time series (Figure 4.6) is replaced by a random signal. The measurements of the fuzzy variable **VALVE_POSITION** are given in Figures 5.13 and 5.14.

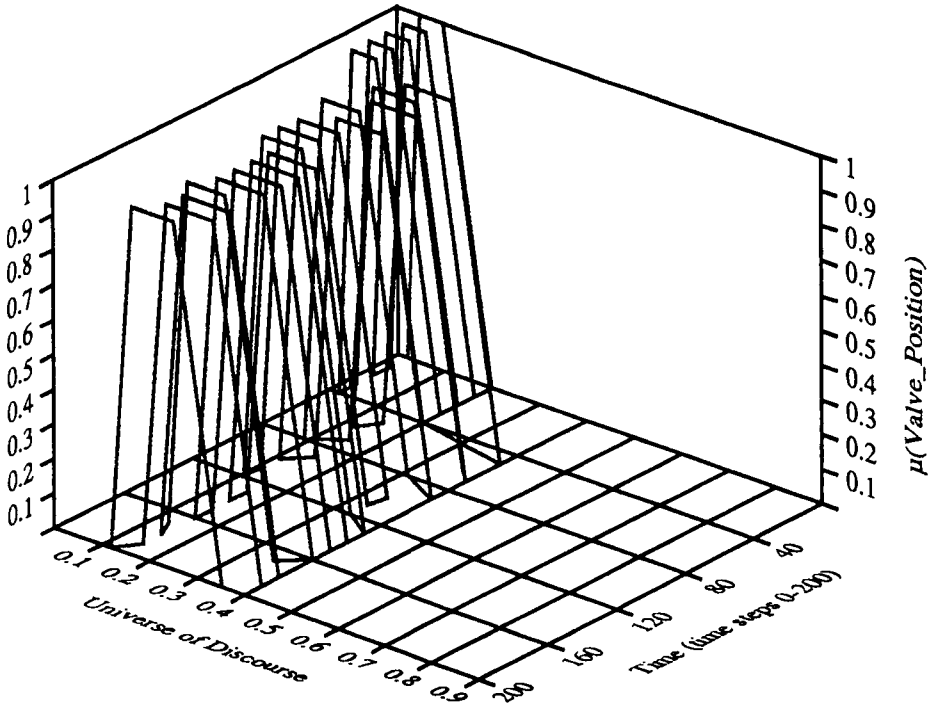

Figure 5.7 Virtual measurements for time steps 0-200 when the average DP signal has been substituted with 100% noise.

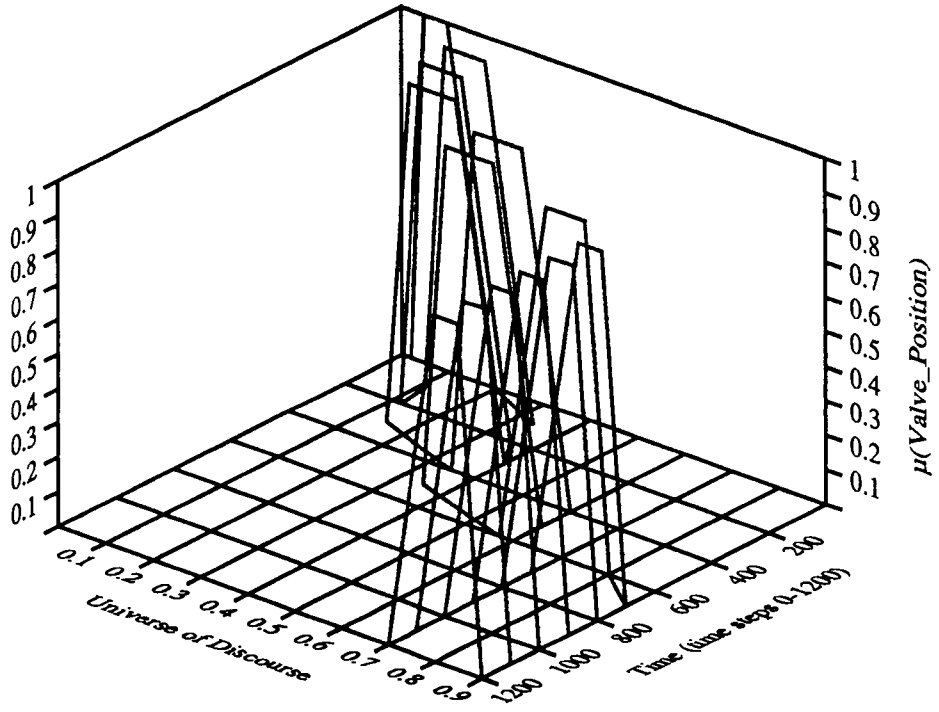

Figure 5.8 Virtual measurements for time steps 0-1200 when the average DP signal has been substituted with 100% noise.

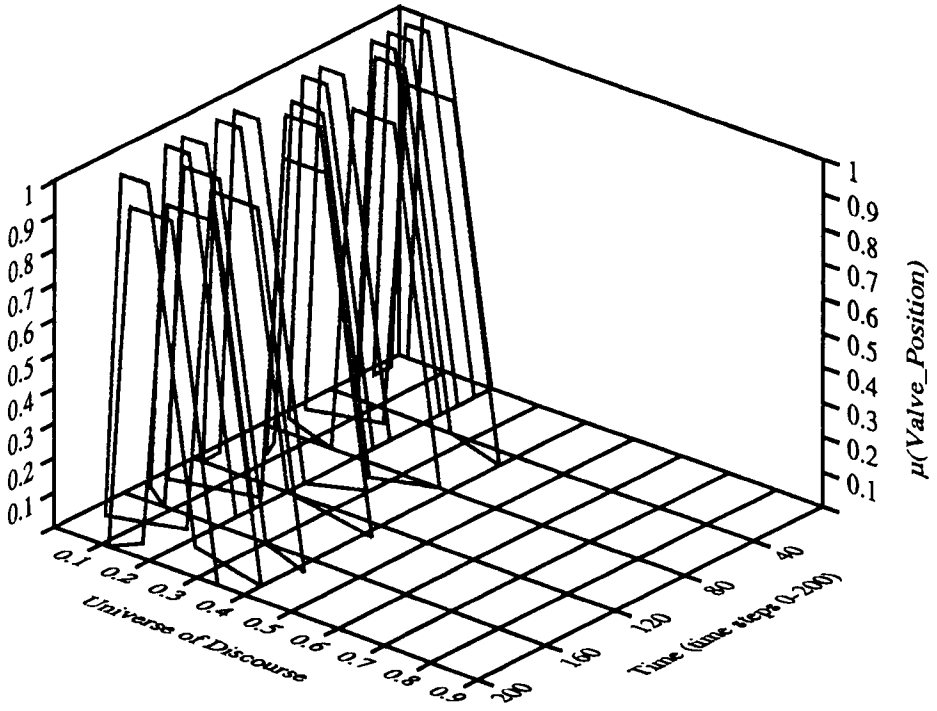

Figure 5.9 Virtual measurements for time steps 0-200 when the average inlet-temperature signal has been substituted with 100% noise.

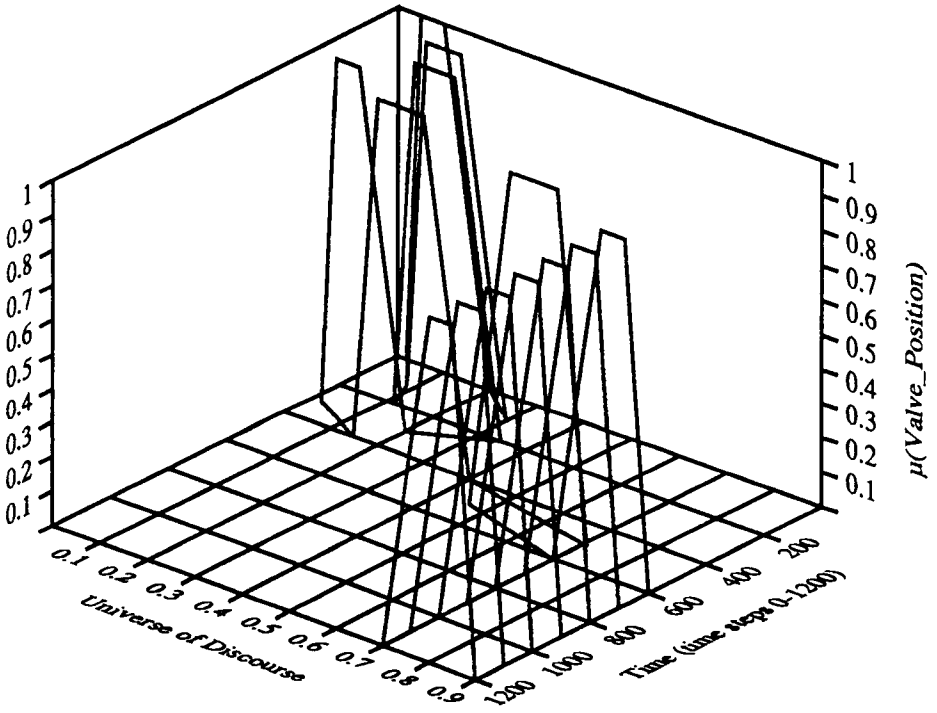

Figure 5.10 Virtual measurements for time steps 0-1200 when the average inlet-temperature signal has been substituted with 100% noise.

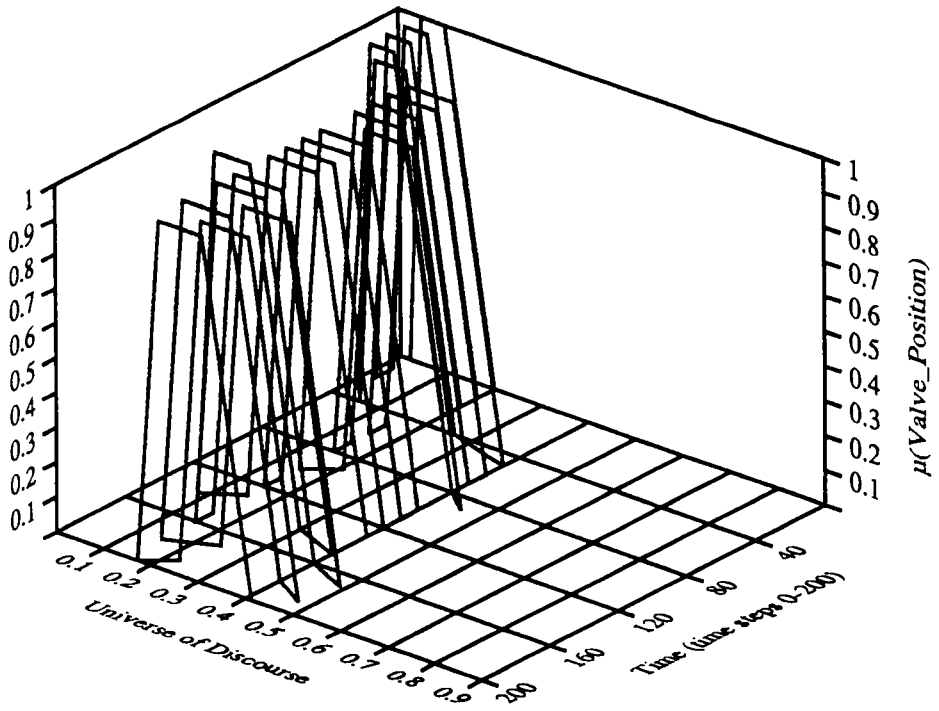

Figure 5.11 Virtual measurements for time steps 0-200 when the average outlet-temperature signal has been substituted with 100% noise.

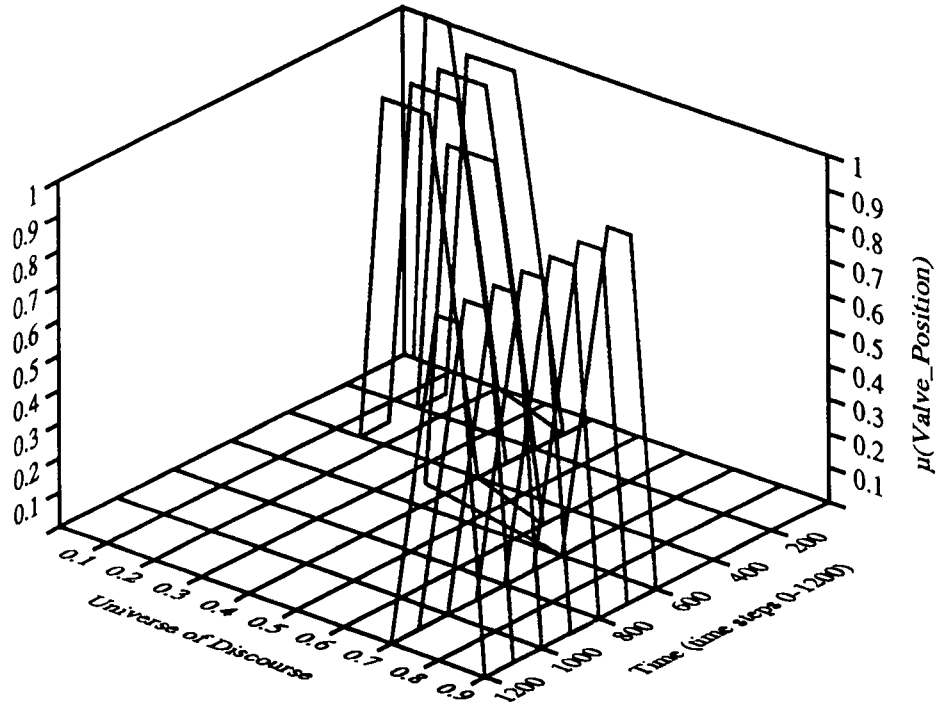

Figure 5.12 Virtual measurements for time steps 0-1200 when the average outlet-temperature signal has been substituted with 100% noise.

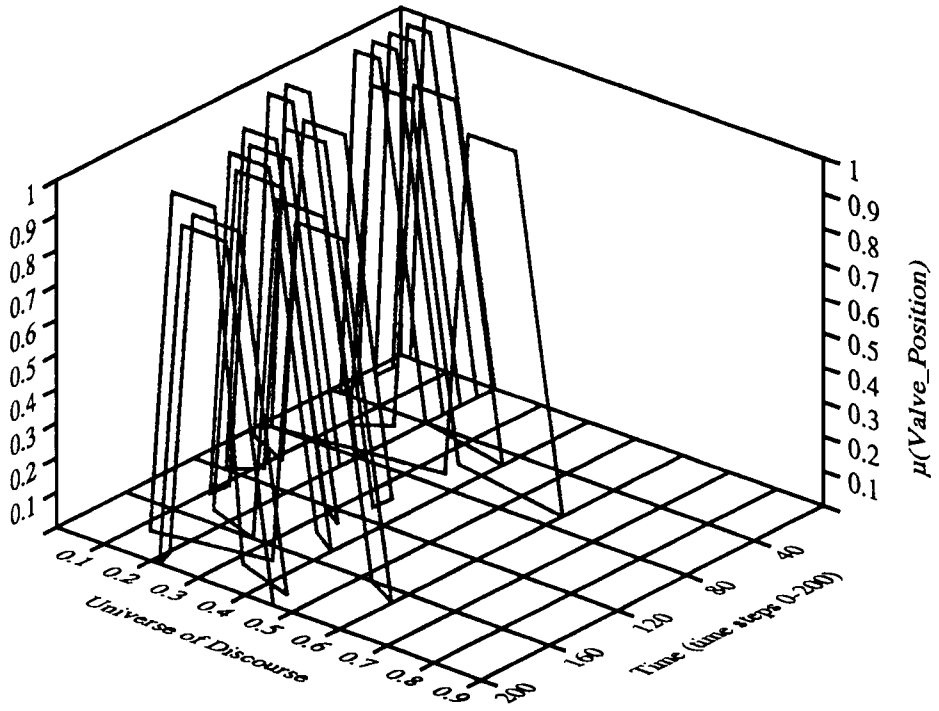

Figure 5.13 Virtual measurements for time steps 0-200 when the average secondary-flow signal has been substituted with 100% noise.

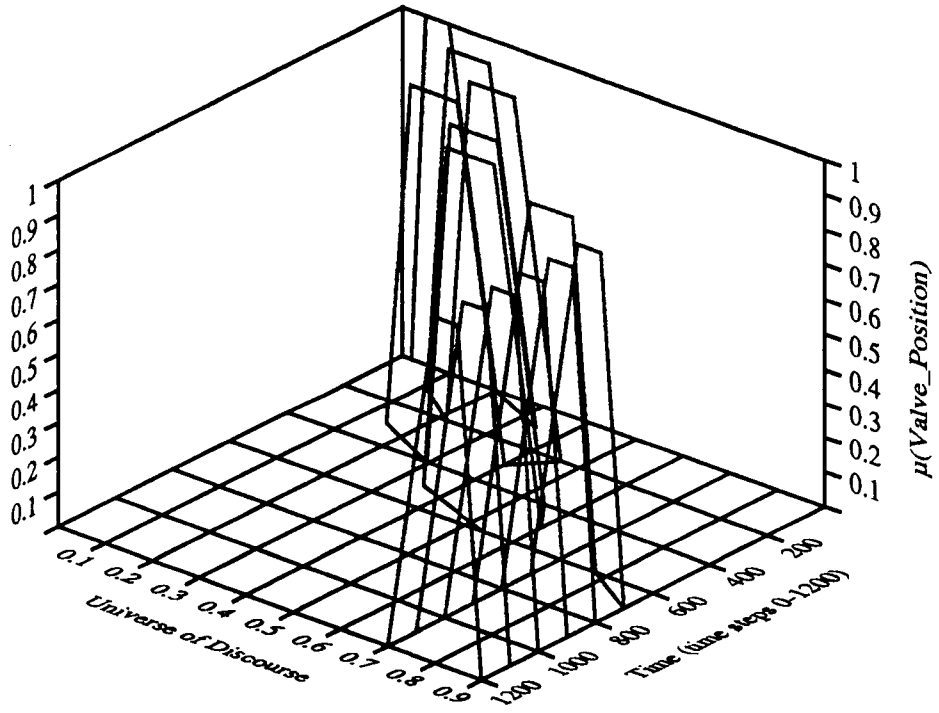

Figure 5.14 Virtual measurements for time steps 0-1200 when the average secondary-flow signal has been substituted with 100% noise.

The process of calculating the Euclidean norms between the centroids of the expected membership functions and the centroids of the predicted by the measuring device fuzzy values is repeated. The Euclidean norm serves the purpose of establishing a reference value for the degree of accuracy accompanying the virtual measurements. As before the Euclidean distance is calculated for the total start-up period and the results are given in Figures 5.15 to 5.19, for the five scenarios analyzed here.

The calculated norms are separated into classes associated with different levels of accuracy. This segmentation offers a better overall description of the distribution of errors associated with the measurement process.

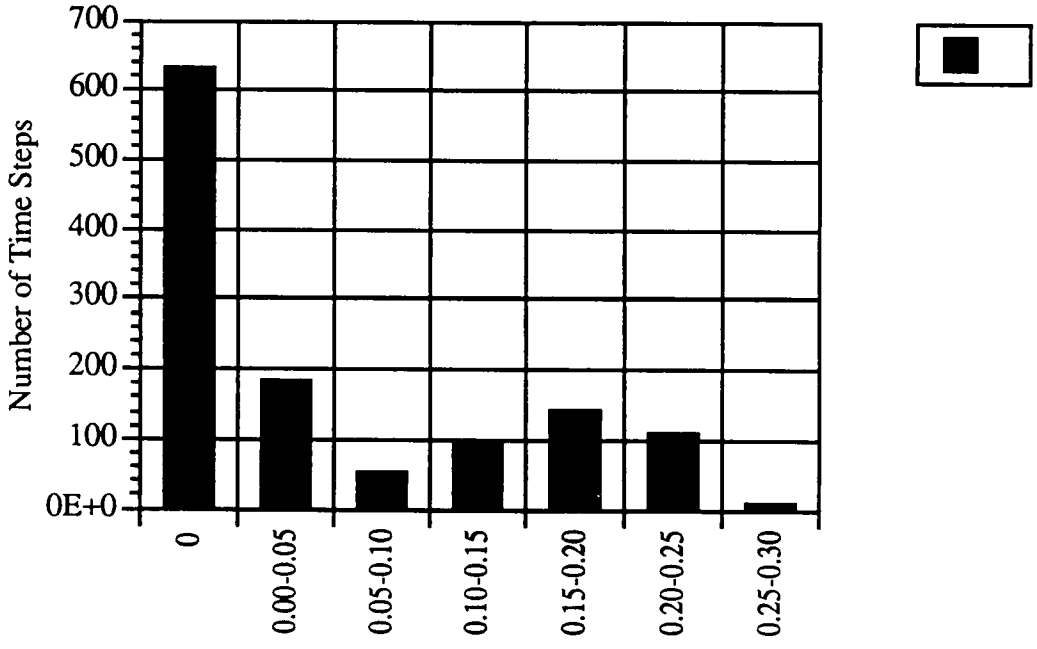

Figure 5.15 Distribution of the Euclidean norms between expected and predicted centroids when the average flux signal has been eliminated.

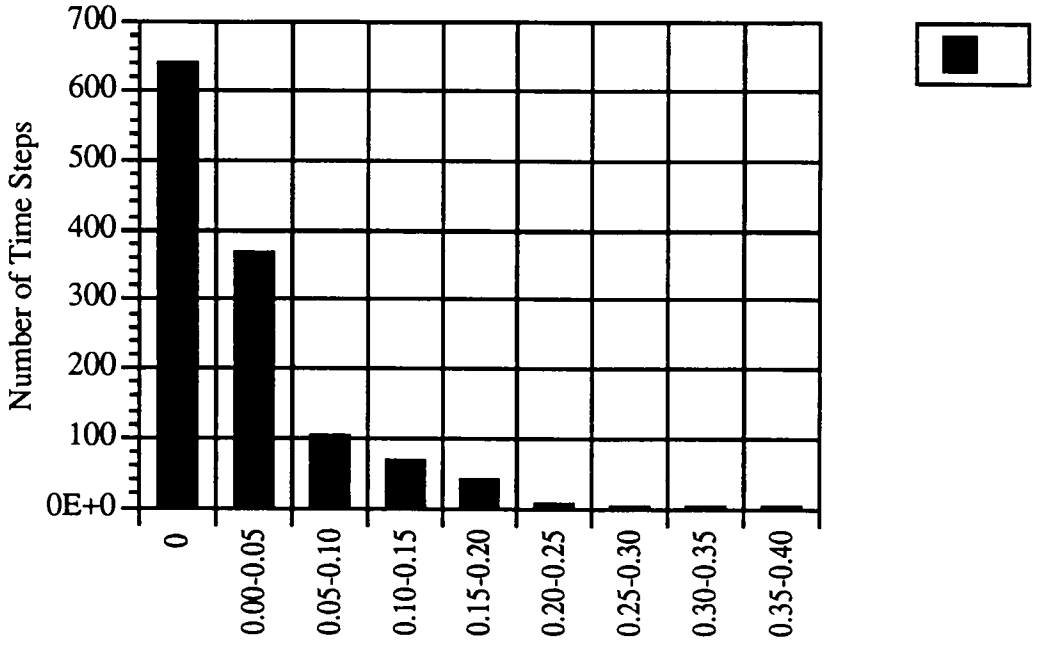

Figure 5.16 Distribution of the Euclidean norms between expected and predicted centroids when the average DP signal has been eliminated.

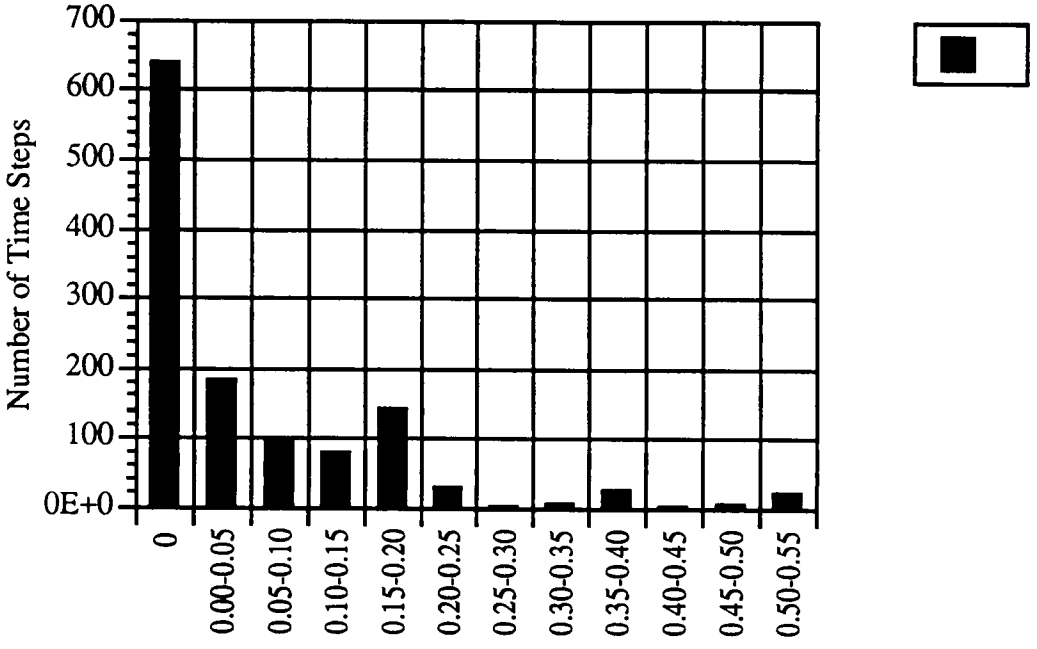

Figure 5.17 Distribution of the Euclidean norms between expected and predicted centroids when the average inlet-temperature signal has been eliminated.

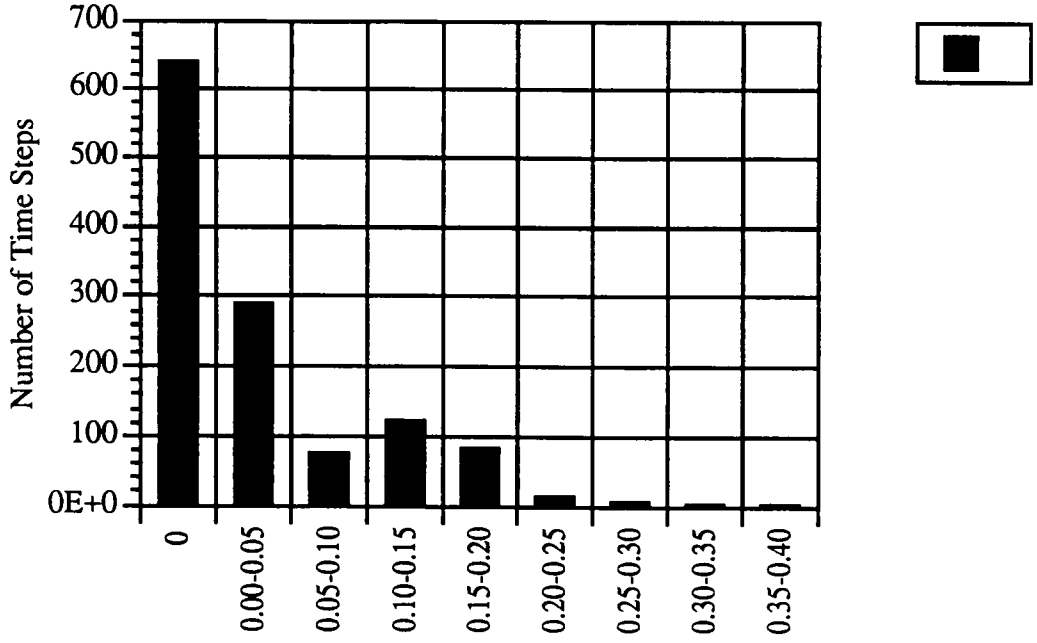

Figure 5.18 Distribution of the Euclidean norms between expected and predicted centroids when the average outlet-temperature signal has been eliminated.

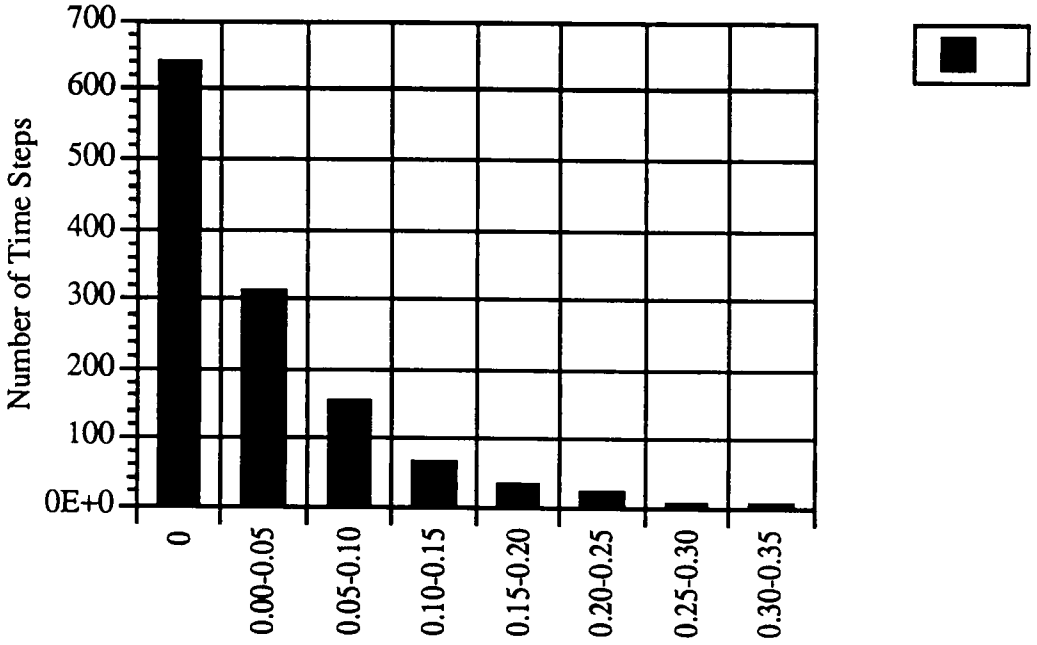

Figure 5.19 Distribution of the Euclidean norms between expected and predicted centroids when the secondary-flow signal has been eliminated.

Figures 5.15 through 5.19 demonstrate the excellent robustness of the virtual monitoring technique to faulty signals. In all five cases more than half of the membership functions have been predicted with 100% accuracy (zero Euclidean norm), where the remaining predictions are associated with an error spread over an area between zero and 0.55, in the worst case scenario (Figure 5.17). In the remaining four cases the maximum Euclidean norms have been computed in the area between 0.40 (Figures 5.16 and 5.18) and 0.30 (Figure 5.15). *Henceforth, the virtual measuring methodology developed, is capable of providing accurate predictions of the position of the valve, even in the case where one fifth of the required input information has been substituted by 100% random noise.*

As a consequence of the above observation, one may conclude that the proposed algorithm is capable of recovering sufficient part of the information eliminated from its input patterns. The missing information is substituted by similar information existing in the remaining four signals. This behavior justifies the assumption made previously, that all training signals have been generated in the same physical environment carrying therefore similar (but not identical) information. This assumption has already been proven valid from the results obtained from the multi-regressive models in Chapter 4, and the present behavior comes to verify our expectations.

It is important to explain these results from the point of view of the measuring device. First of all the deterioration of the Euclidean norms with respect to the addition of 10% noise in all input signals, is in accordance to our expectations. In the tests discussed here, the amount of information eliminated from the input vectors (one fifth for every case) is by far larger compared to the amount of noise added in the previous case. Nevertheless, the addition of noise does not eliminate any of the information carried originally by the

signal, but mainly distorts it with high frequency components. Henceforth, the distribution of the Euclidean norms covers a larger area than before, and it may be considered as following the χ^2 distribution. It is worth noticing that in case the present results exhibited similar accuracy to those obtained before, then significant questions could be raised concerning the "memorization" by the neural networks of the training signals.

A closer look to Figure 5.17 where the results are drawn when the average inlet-temperature signal has been eliminated reveals an interesting point. We observe that it is the only case where the distribution of the Euclidean norms has been spread in a larger area than in the other cases attaining a maximum value close to 0.55. As means of measuring the relative prediction accuracy among the different cases, we have calculated the average Euclidean norm for the five tests and plotted their values in Figure 5.20. The average Euclidean norms exhibit a striking difference between the case where the average inlet-temperature signal has been eliminated and the rest of them. Although, the average Euclidean norms computed for the other four tests are around 0.035, the substitution of the average inlet-temperature signal by random noise, increases the average Euclidean norm to 0.064. Therefore, the elimination of this signal has more significant impact on the measurement process than the substitution of any of the other time series. In other words, the information about HFIR encoded in the average inlet-temperature signal can not be sufficiently recovered from the remaining four signals. This behavior implies a low correlation between the average inlet-temperature and the rest signals and appears to be characteristic of the methodology developed. Henceforth, it remains to be examined if it is consistent with the behavior of the system under investigation.

In order to prove consistency, we have computed the degree of dependence among the five signals used as input to the virtual measuring device. The correlation between two

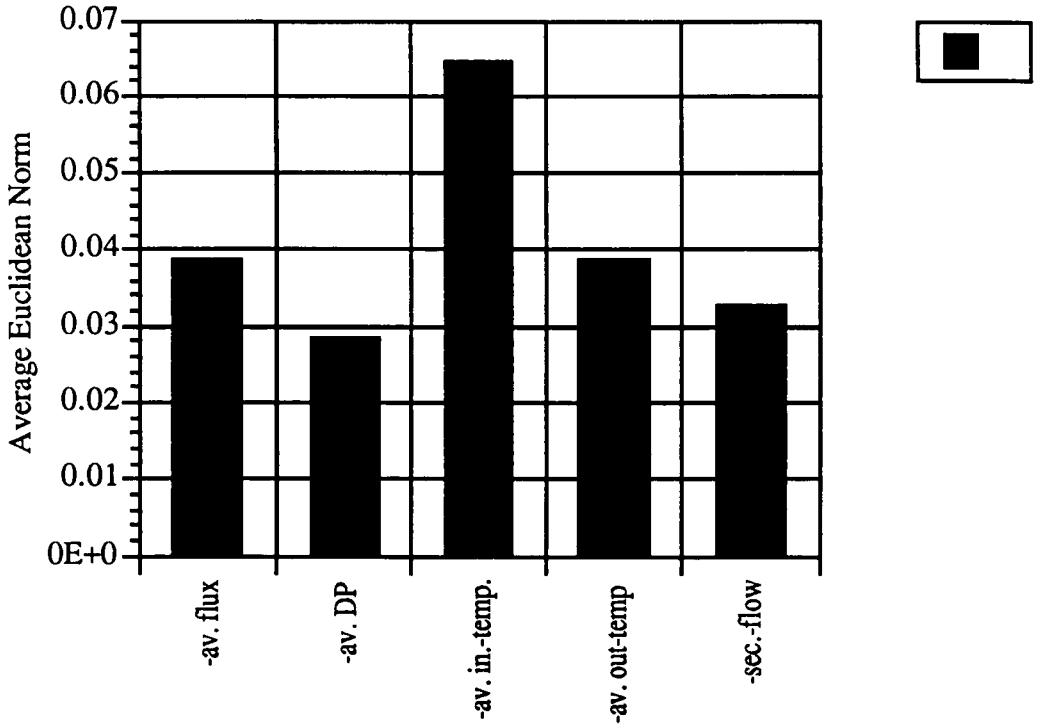

Figure 5.20 Values of the average Euclidean norm for different signal eliminations.

different signals x and y is quantified by the *correlation coefficient*, r_{xy}, (Bendat, 1958) which is defined by the equation

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left[\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2\right]}} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n}\left[\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i\right]}{\sqrt{\left[\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right]}\sqrt{\left[\sum_{i=1}^{n} y_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} y_i\right)^2\right]}} \quad (5.15)$$

where x_i and y_i are measurements of the x and y signals, and $\bar{x}$ and $\bar{y}$ are the mean values of the x and y time series. In this case there exist five signals and thus the *correlation matrix, C*, describing the correlation among the five time series, has to be calculated. Once

again the SAS® statistical environment has been utilized and the results are shown in the following matrix

$$C = \begin{bmatrix} & \textit{flux} & \textit{DP} & \textit{in.}-\textit{temp.} & \textit{out.}-\textit{temp.} & \textit{sec}-\textit{flow} \\ \textit{flux} & 1.0 & 0.9361 & 0.3574 & 0.9878 & 0.9955 \\ \textit{DP} & & 1.0 & 0.5568 & 0.9615 & 0.9341 \\ \textit{in.}-\textit{temp.} & & & 1.0 & 0.4952 & 0.3575 \\ \textit{out}-\textit{temp.} & & & & 1.0 & 0.9857 \\ \textit{sec}-\textit{flow} & & & & & 1.0 \end{bmatrix}$$

The numbers given in the above matrix come in full agreement with the qualitative behavior of the virtual measuring device. Four out of the five signals have a correlation coefficient close to one, which actually means that the signals are highly dependent. The exception in this pattern is the average inlet-temperature signal which exhibits a relatively lower correlation with all four signals. This means that the degree of dependence between this time series and the rest is very low. Thus, the elimination of this signal from the input patterns presented to the measuring device, results in the extinction of certain pieces of information which *can not* be substituted from the remaining information supplied from the other signals. Apparently, this does not happen when one of the other four signals is replaced by random signals, since the high correlation existing among them, suggests that the information carried by each one is very similar (if not identical) to the information existing in the remaining time series.

The above behavior is suggested from the results presented in the correlation matrix, and is an inherent statistical property of the signals used for input patterns. On the other hand we see that the methodology for virtual measurements, exhibits identical behavior to the one suggested by the correlation matrix. *This is sufficient to reassure that the model developed has incorporated all the statistical properties existing into the original training signals.* The utilization of artificial neural networks enabled the analysis and "learning" of the statistical properties embedded into the original signals. These statistical properties are preserved during the mapping of measurable system parameters - in the form time series - to the space of human linguistics (universe of discourse). Thus, the fuzzy values predicted by the neural networks are sufficiently equipped with the statistical properties of the original input signals.

The membership functions predicted from the set of artificial networks are initially accepted (or discarded), and later combined in order to calculate the resultant membership functions. This procedure is based on the *dynamic behavior criterion* which has been imposed in the form of the rule-based system developed. Since the *dynamic behavior criterion* has been designed considering the statistical characteristics of the training signals as they have been found from multi-regression models, the rule-based system does not affect the statistical properties of the original signals since it "behaves" in a manner dictated by them. Thus, the resultant membership functions contain all the statistical information existing in the original signals. In case the construction principle of the rule-based system had been arbitrarily imposed, the statistical properties of the monitored parameter would have been affected and the measuring device would lack sensitivity with respect to the specifics of the system. *It is the preservation of the statistical characteristics of the original signals through the utilization of non-linear filters and statistical-dependent rule-based*

systems, which renders the virtual measuring device sensitive to the characteristics of the monitored system.

In order to further investigate the capabilities of the proposed methodology, the training signals have been eliminated in pairs. Initially, the average flux and DP signals have been substituted with 100% noise. The results from testing the model are shown in Figures 5.21 and 5.22 for the first 200 time steps and the total duration of the start-up, respectively. In the second test the average flux along with the average inlet-temperature time signatures have been omitted and in their place signals similar to those shown in Figure 5.4 have been placed. The measurements of the virtual monitoring device are drawn in Figures 5.23 and 5.24. Going one step further, the outlet-temperature signal has been dropped along with the average flux and the linguistic measurements of the fuzzy variable **VALVE_POSITION** are shown in Figures 5.25 and 5.26 for the time duration of the simulation process. The last combination of double signal elimination containing the average flux signal is this with the secondary-flow time series. The response of the virtual measuring device when these two input signals have been replaced by a pair of random time series is shown in Figures 5.27 and 5.28.

Since all possible combinations containing the average flux signal have been covered the next stage is to eliminate the average DP signal with each one of the remaining three signals. Initially, we have left out the average inlet-temperature signal along with the average DP measurements. The results are shown in Figures 5.29 and 5.30. Next, we have omitted the average DP and outlet-temperature signals and replaced them with random time signatures. The measurements supplied by the proposed technique are plotted in Figures 5.31 and 5.32. The last combination contains the elimination of the average DP signal is this with the secondary-flow time series. Schematic representation of the recorded

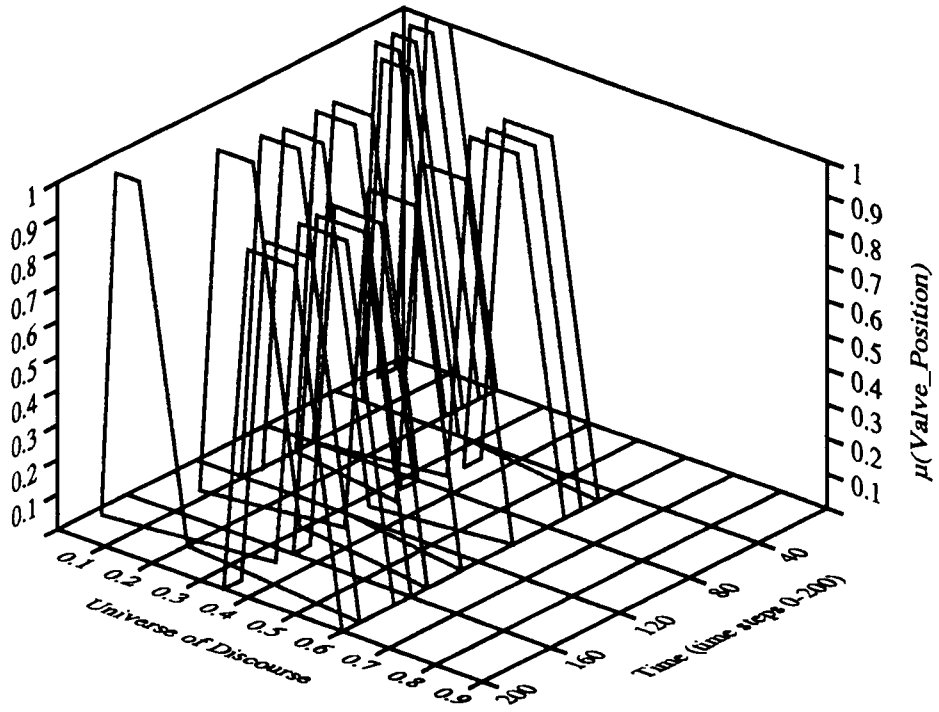

Figure 5.21 Virtual measurements for time steps 0-200 when the average flux and DP signals have been substituted with 100% noise.

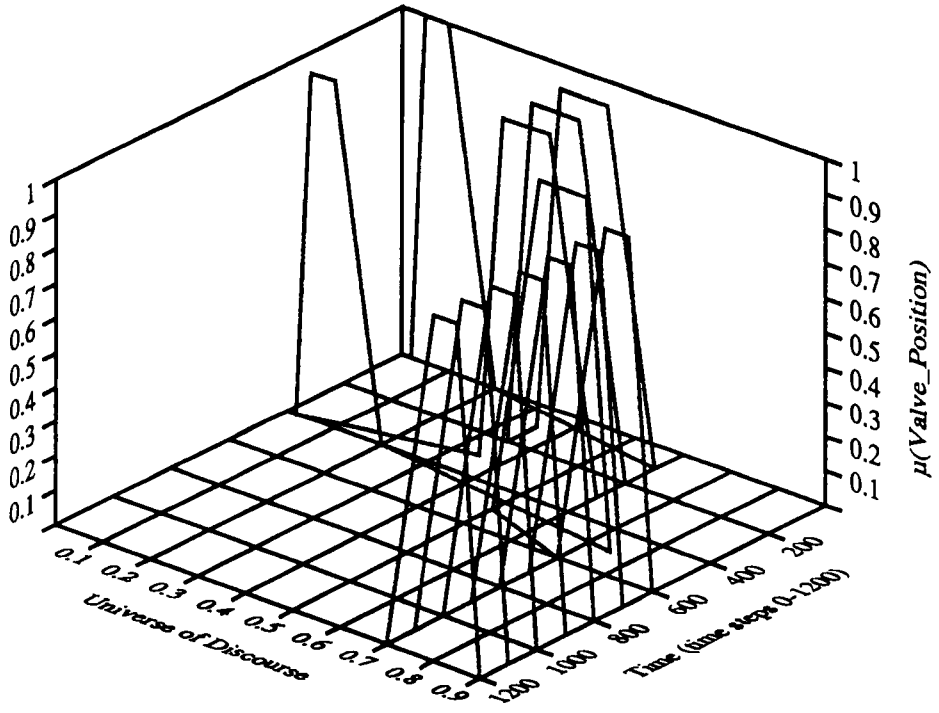

Figure 5.22 Virtual measurements for time steps 0-1200 when the average flux and DP signals have been substituted with 100% noise.

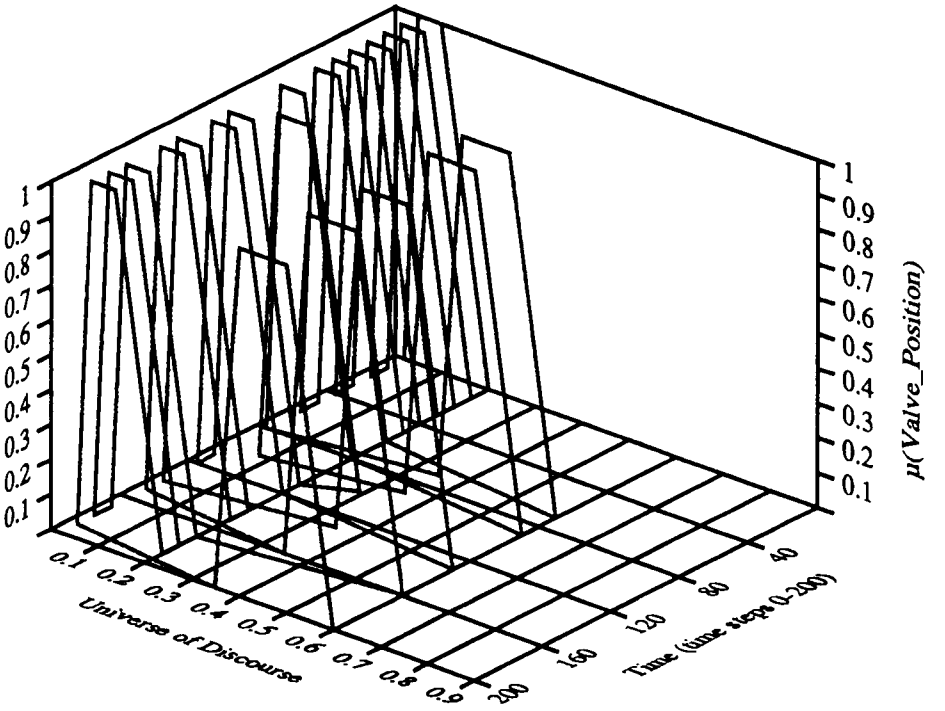

Figure 5.23 Virtual measurements for time steps 0-200 when the average flux and inlet-temperature signals have been substituted with 100% noise.

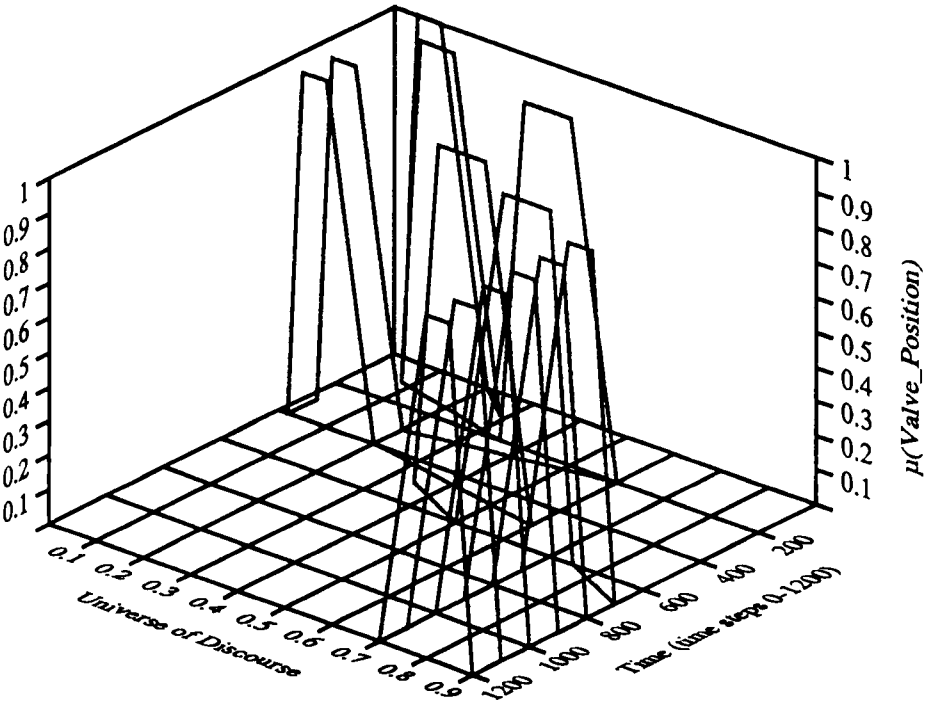

Figure 5.24 Virtual measurements for time steps 0-1200 when the average flux and inlet-temperature signals have been substituted with 100% noise.

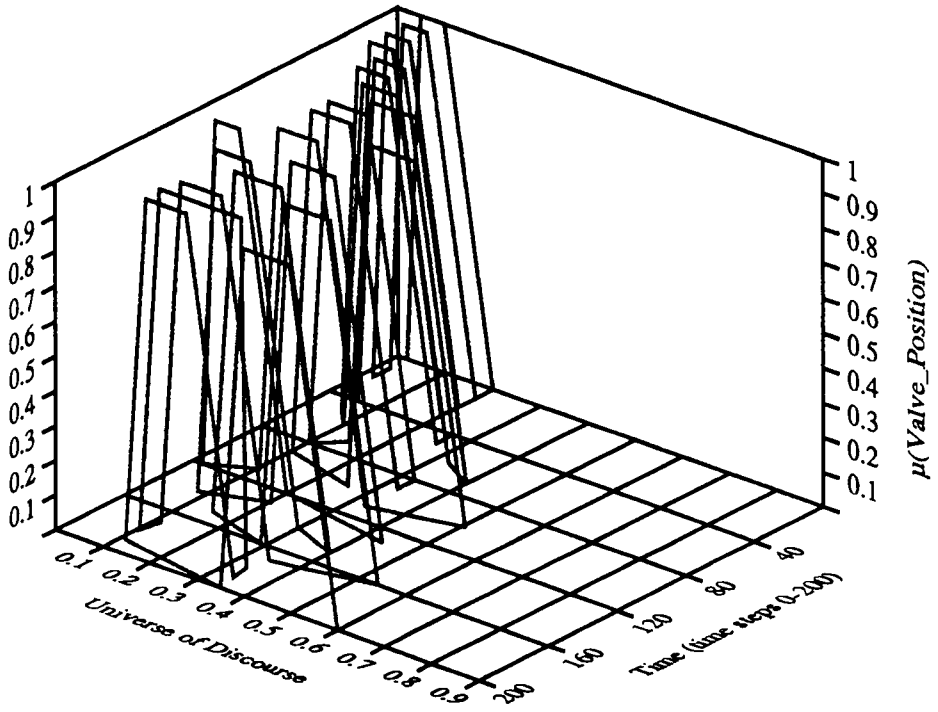

Figure 5.25 Virtual measurements for time steps 0-200 when the average flux and outlet-temperature signals have been substituted with 100% noise.

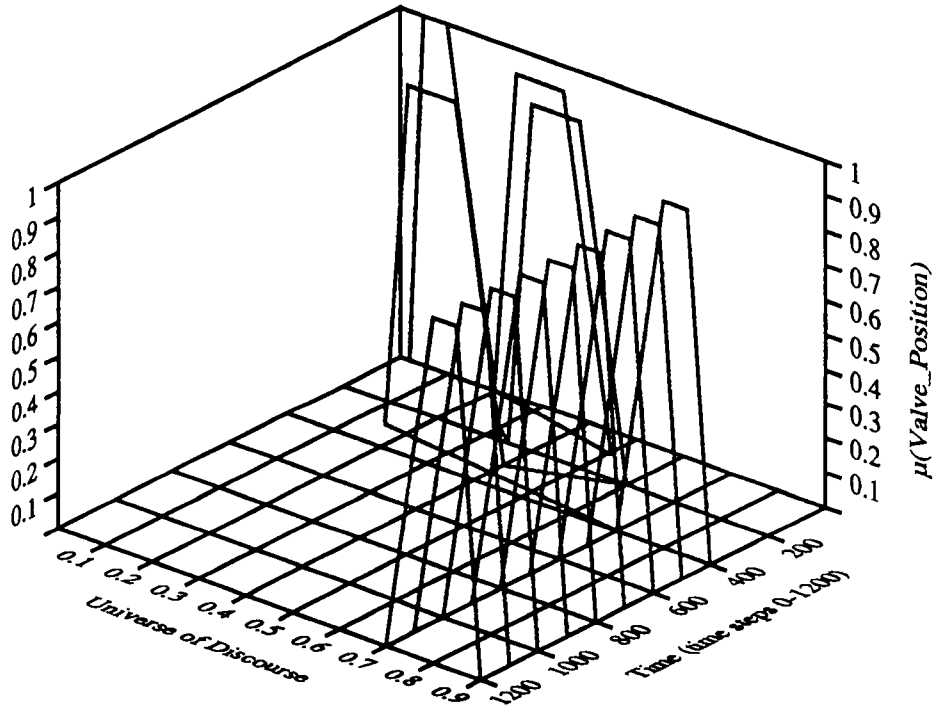

Figure 5.26 Virtual measurements for time steps 0-1200 when the average flux and outlet-temperature signals have been substituted with 100% noise.

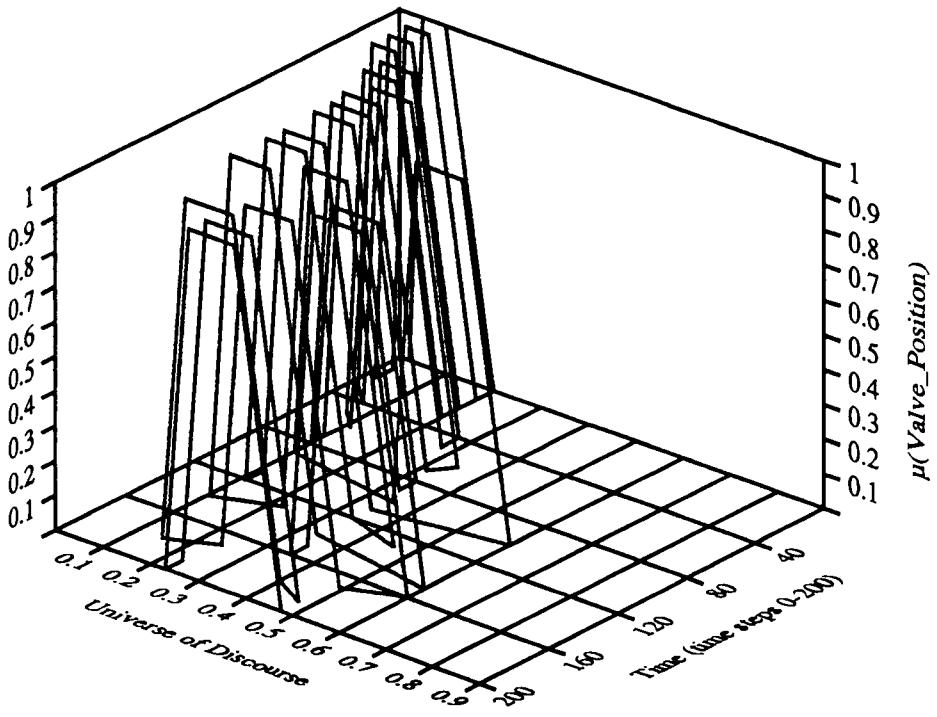

Figure 5.27 Virtual measurements for time steps 0-200 when the average flux and secondary-flow signals have been substituted with 100% noise.

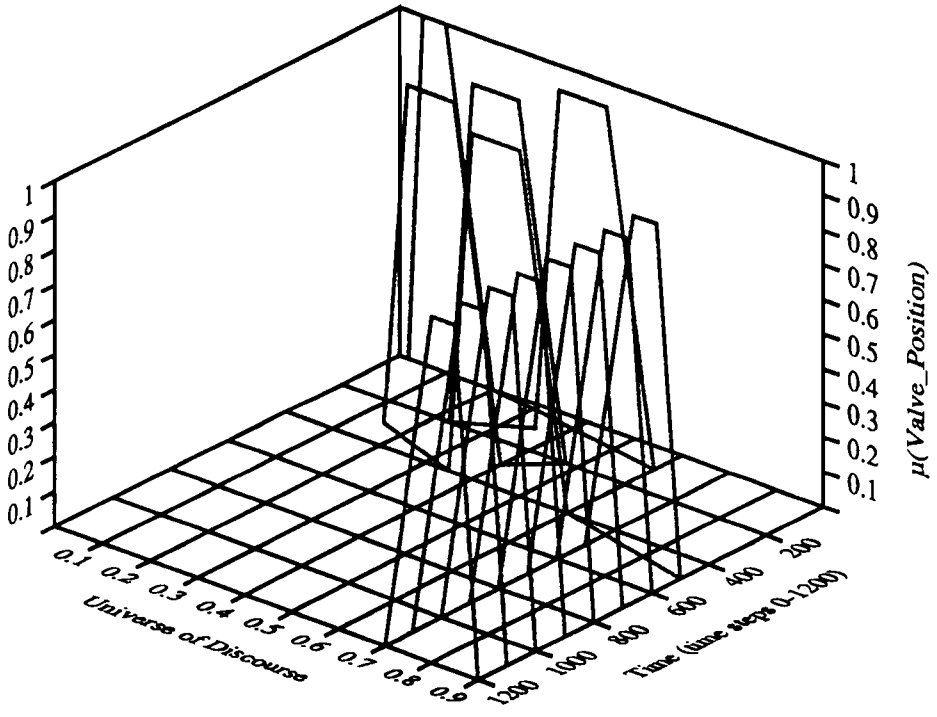

Figure 5.28 Virtual measurements for time steps 0-1200 when the average flux and secondary-flow signals have been substituted with 100% noise.

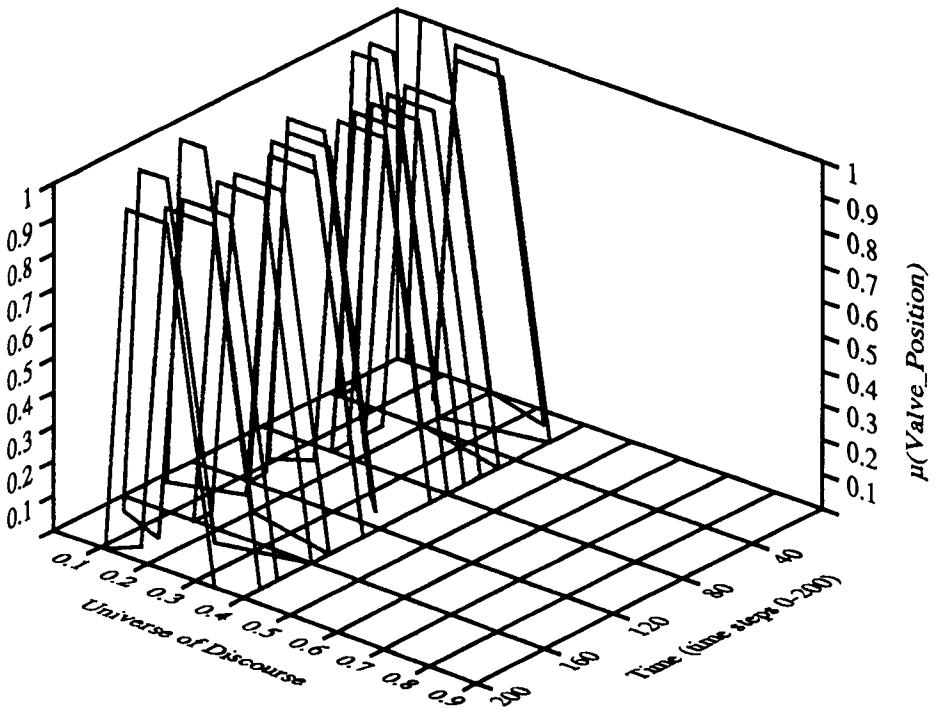

Figure 5.29 Virtual measurements for time steps 0-200 when the average DP and inlet-temperature signals have been substituted with 100% noise.

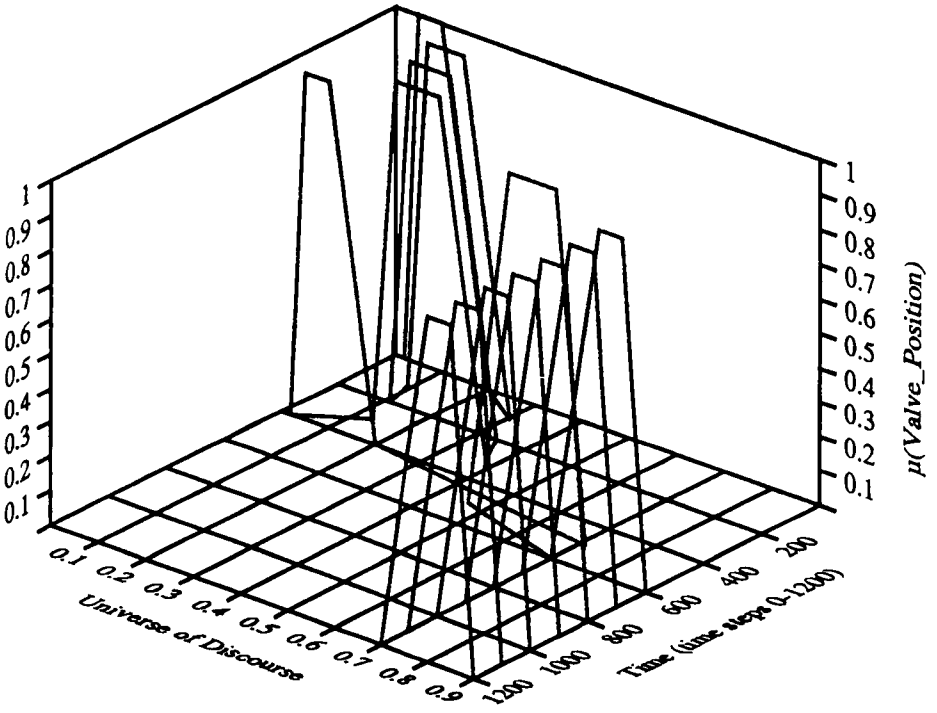

Figure 5.30 Virtual measurements for time steps 0-1200 when the average DP and inlet-temperature signals have been substituted with 100% noise.

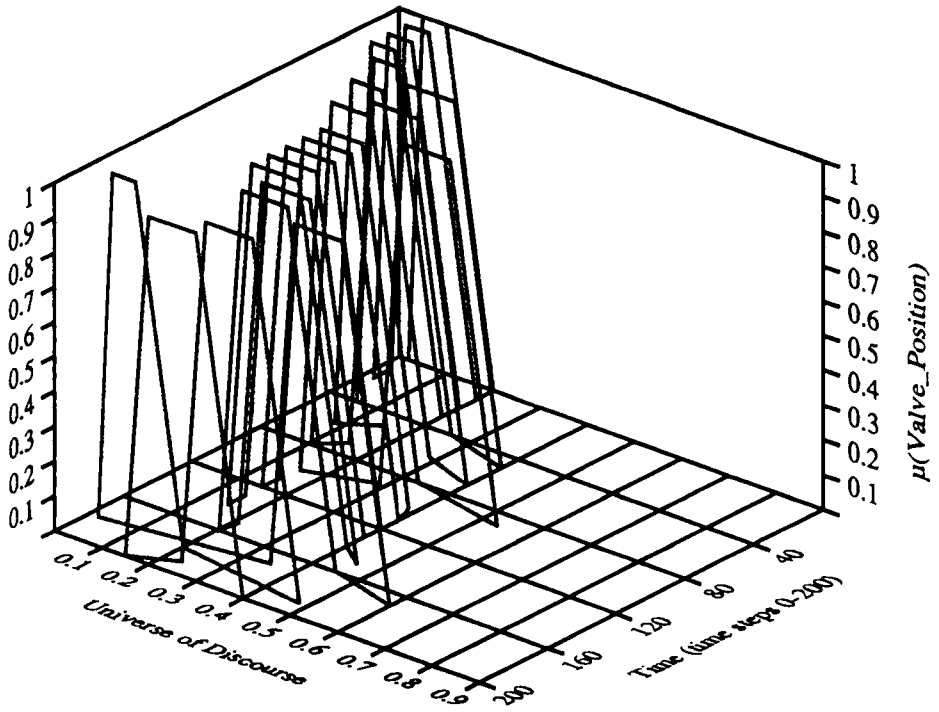

Figure 5.31 Virtual measurements for time steps 0-200 when the average DP and outlet-temperature signals have been substituted with 100% noise.

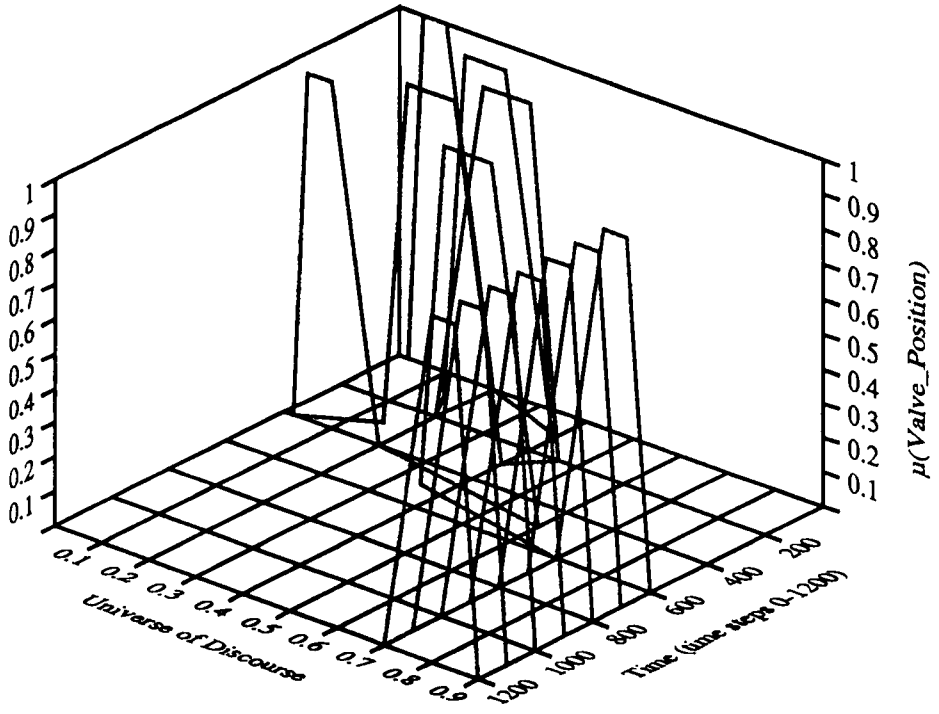

Figure 5.32 Virtual measurements for time steps 0-1200 when the average DP and outlet-temperature signals have been substituted with 100% noise.

by the virtual monitoring device values, is given in Figures 5.33 and 5.34.

The next two combination are the last two containing the average inlet-temperature signals. The third pair of this class consists of the simultaneous replacement of the average inlet and outlet-temperature signals by two random time series. The virtual measurements obtained are shown in Figures 5.35 and 5.36. The remaining secondary-flow signal has been substituted along with the average inlet-temperature signal by 100% noise and the linguistic evaluations of the measured parameter are given in Figures 5.37 and 5.38.

The last dyadic combination is that of the average outlet-temperature and the secondary flow input time series. Two totally random signals were placed in their position and the predicted fuzzy values associated with the fuzzy variable **VALVE_POSITION** are sketched in Figures 5.39 and 5.40.

In order to further examine the results obtained from the previous tests, we have calculated the Euclidean distances between the centroids of the expected and predicted membership functions. The results are shown in Figures 5.41 to 5.50.

A close look to the plotted results is sufficient to show that the predictions of the measuring device are consistent with the actual system states although the input signals have been severely "damaged". The measuring algorithm is still capable to recognize the patterns encoded into the heavily altered input vectors and perform successful evaluations. The information remaining in the unchanged signals seems to be enough for the measuring device to function properly and provide appropriate readings. In addition, it should be stressed that during this testing stage the measuring device provided information during the whole simulation period and never failed to give a response - even a partially incorrect one.

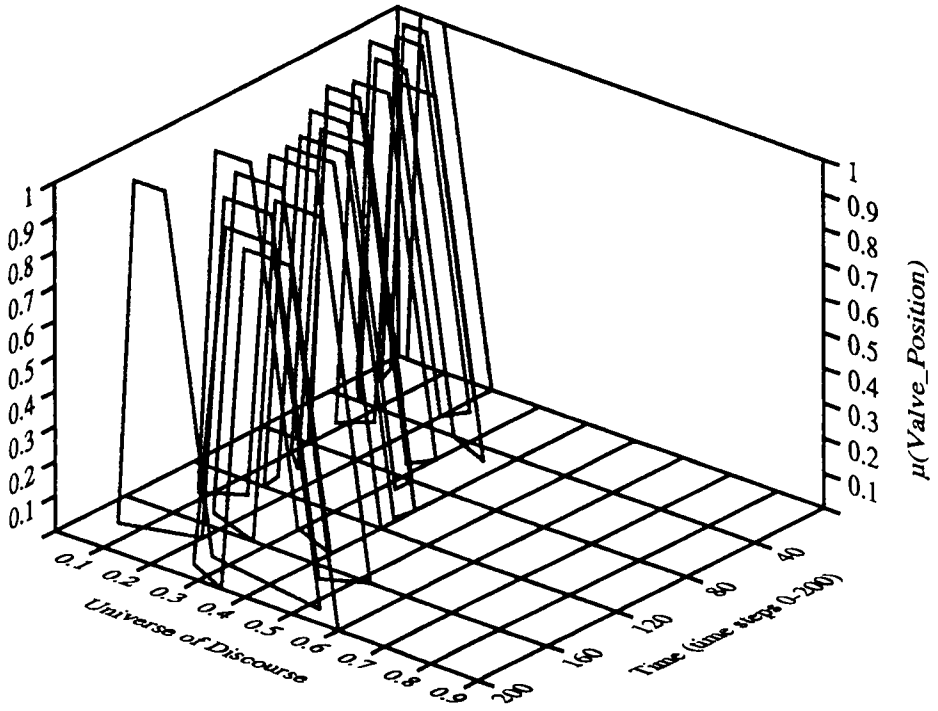

Figure 5.33 Virtual measurements for time steps 0-200 when the average DP and secondary-flow signals have been substituted with 100% noise.

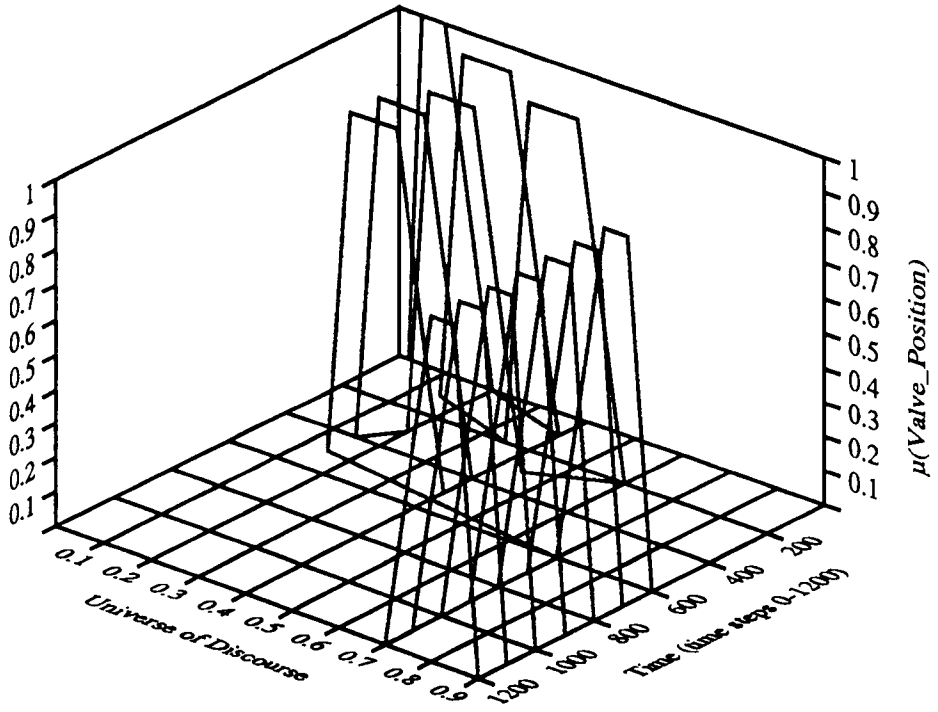

Figure 5.34 Virtual measurements for time steps 0-1200 when the average DP and secondary-flow signals have been substituted with 100% noise.

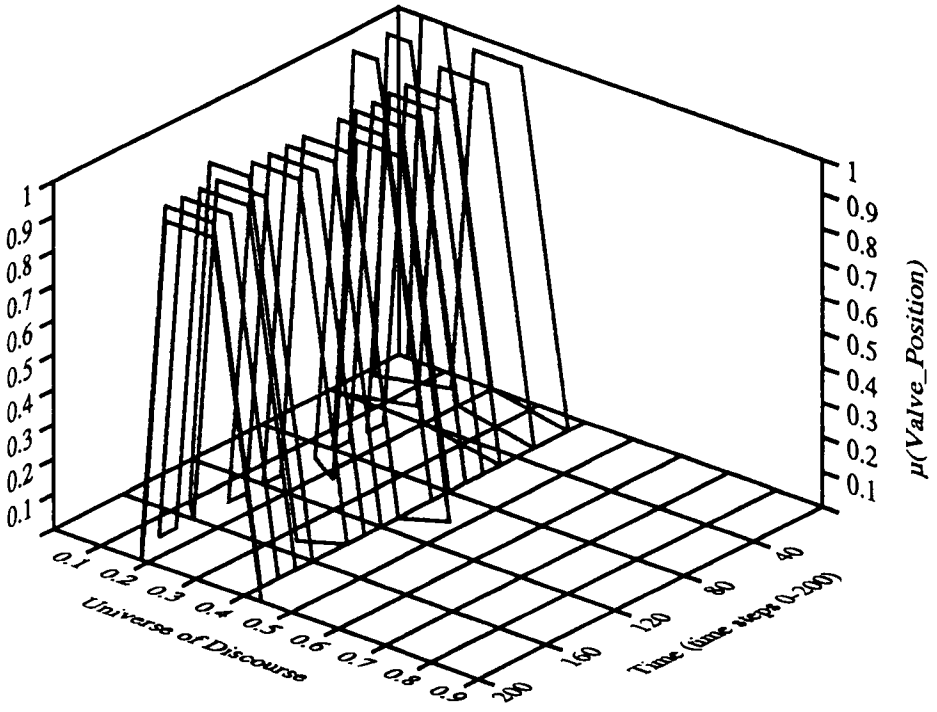

Figure 5.35 Virtual measurements for time steps 0-200 when the average inlet-temperature and outlet-temperature signals have been substituted with 100% noise.

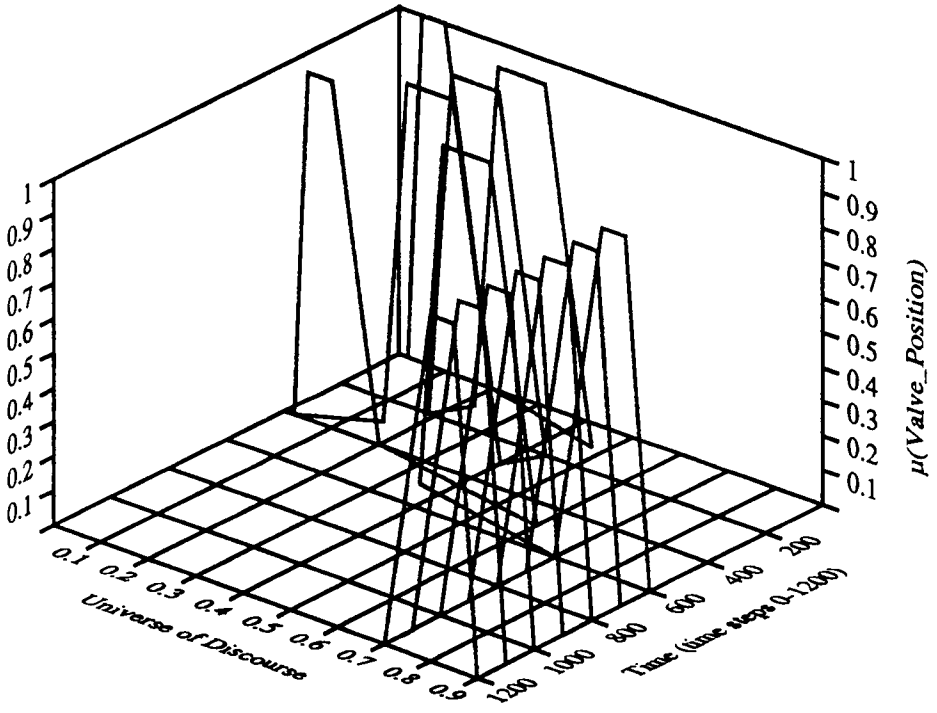

Figure 5.36 Virtual measurements for time steps 0-1200 when the average inlet-temperature and outlet-temperature signals have been substituted with 100% noise.

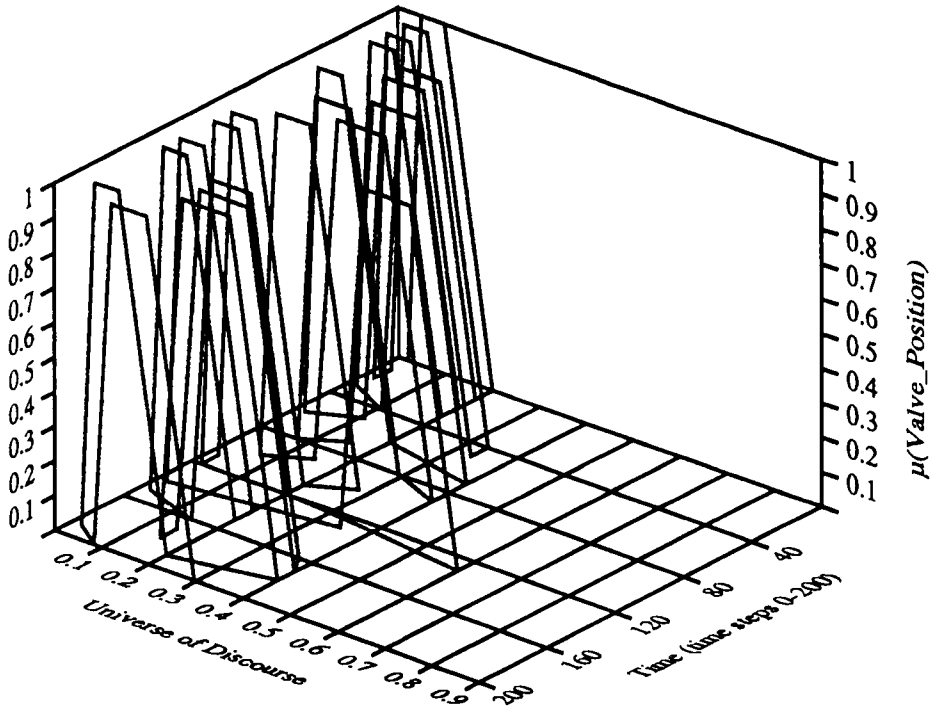

Figure 5.37 Virtual measurements for time steps 0-200 when the average inlet-temperature and secondary-flow signals have been substituted with 100% noise.

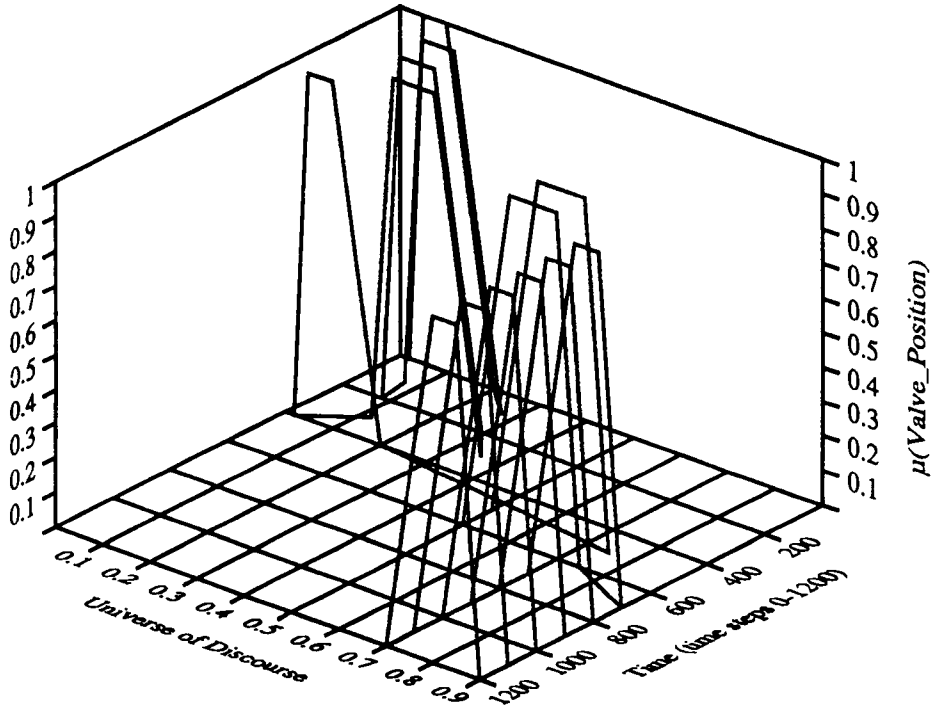

Figure 5.38 Virtual measurements for time steps 0-1200 when the average inlet-temperature and secondary-flow signals have been substituted with 100% noise.

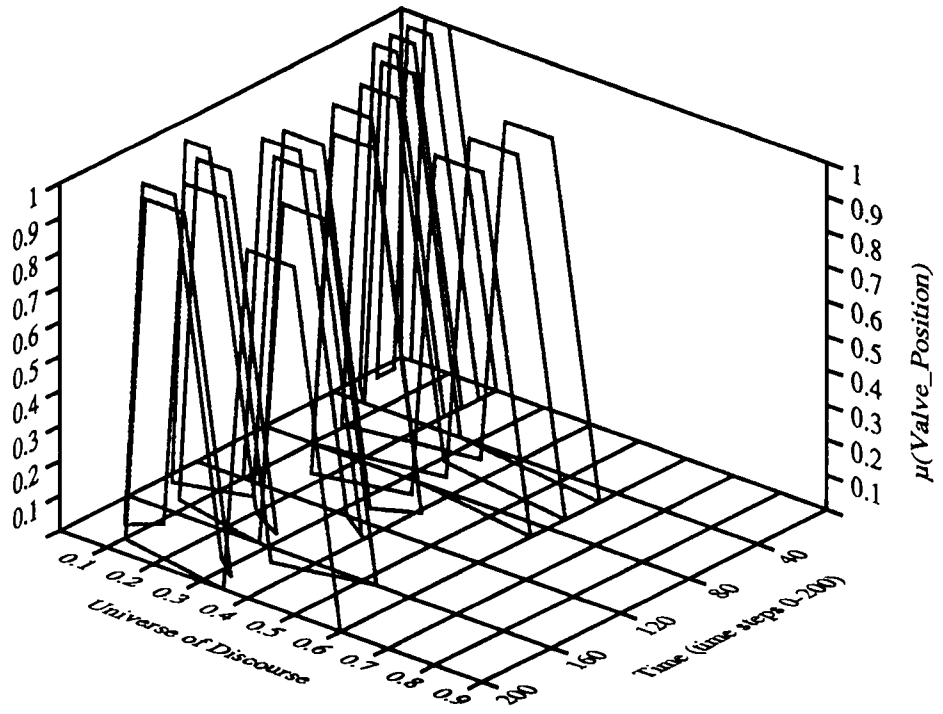

Figure 5.39 Virtual measurements for time steps 0-200 when the average outlet-temperature and secondary-flow signals have been substituted with 100% noise.

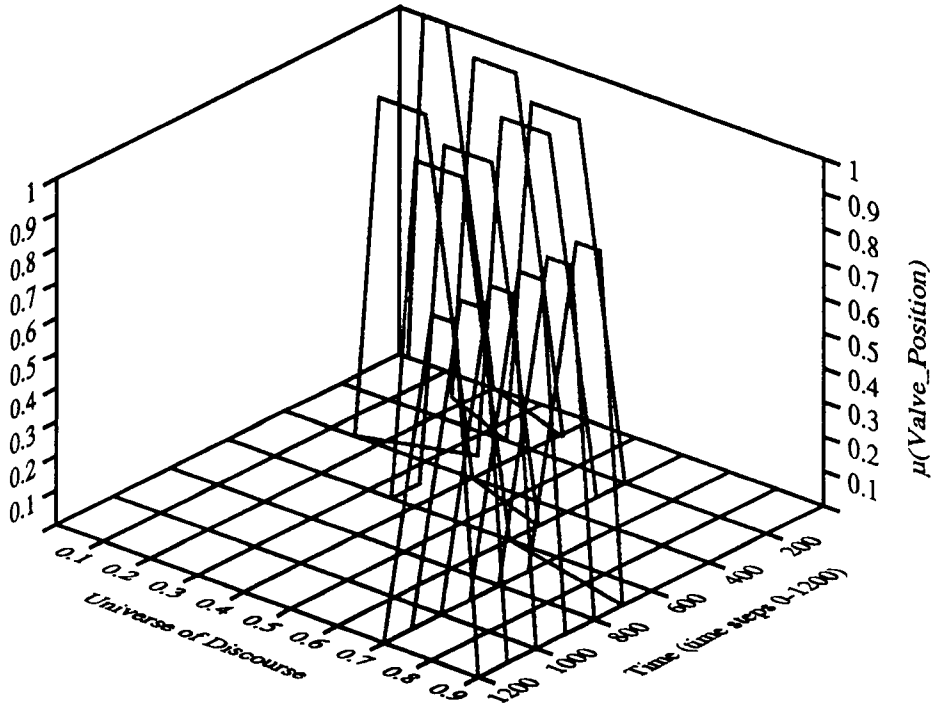

Figure 5.40 Virtual measurements for time steps 0-1200 when the average outlet-temperature and secondary-flow signals have been substituted with 100% noise.

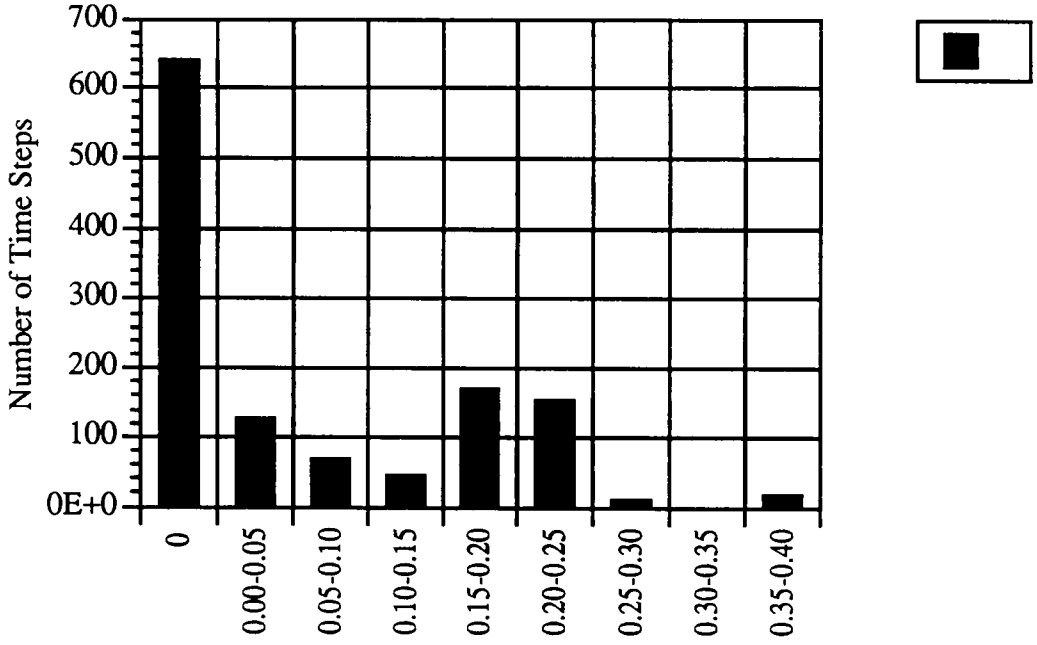

Figure 5.41 Distribution of the Euclidean norms between expected and predicted centroids when the average flux and DP signals have been eliminated.

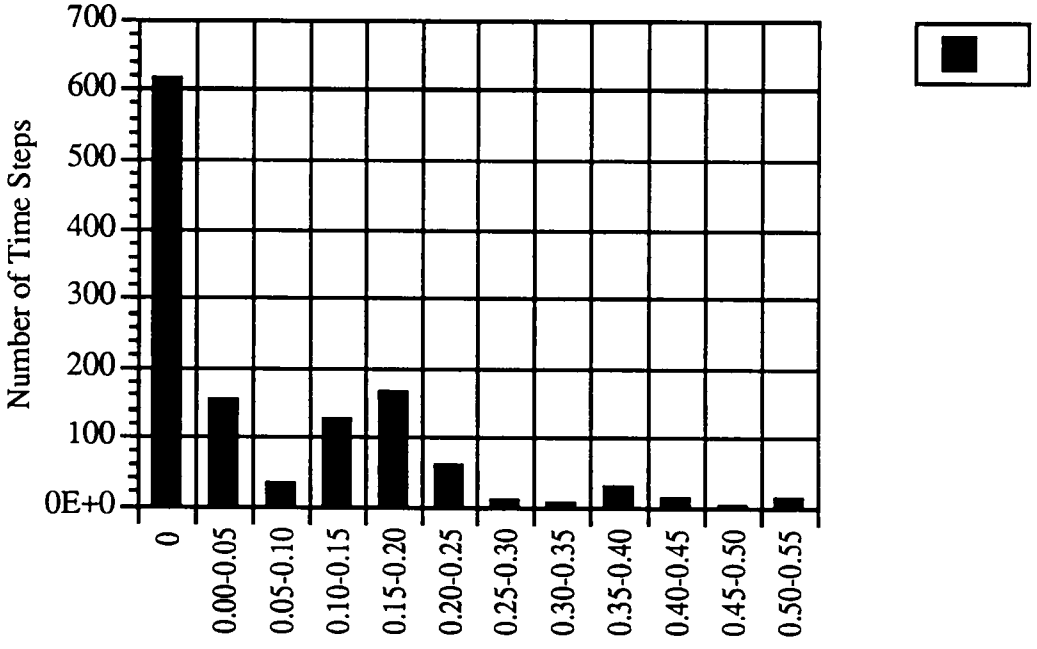

Figure 5.42 Distribution of the Euclidean norms between expected and predicted centroids when the average flux and inlet-temperature signals have been eliminated.

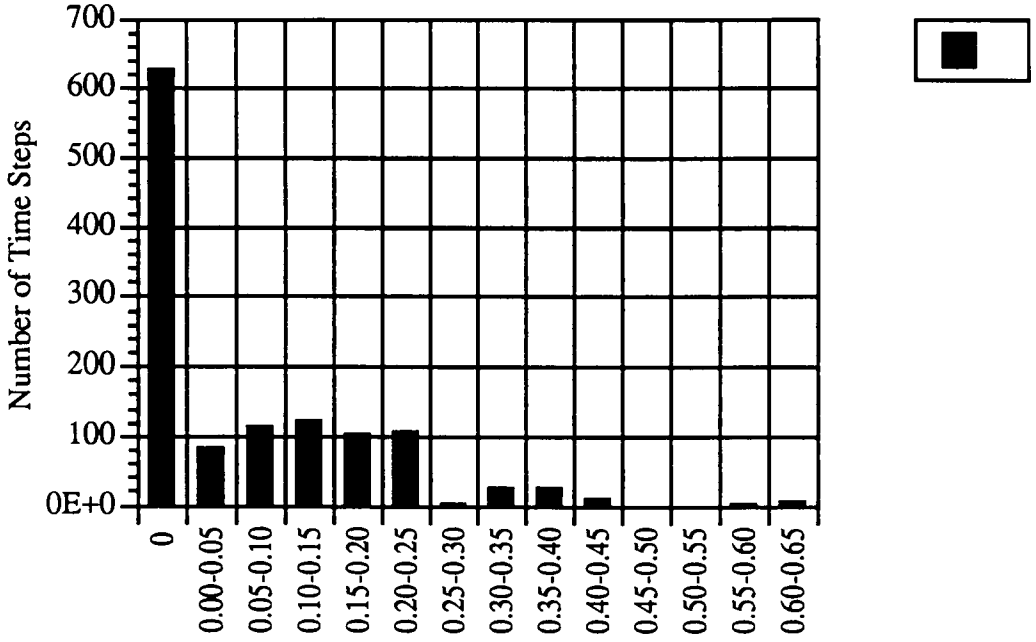

Figure 5.43 Distribution of the Euclidean norms between expected and predicted centroids when the average flux and outlet-temperature signals have been eliminated.

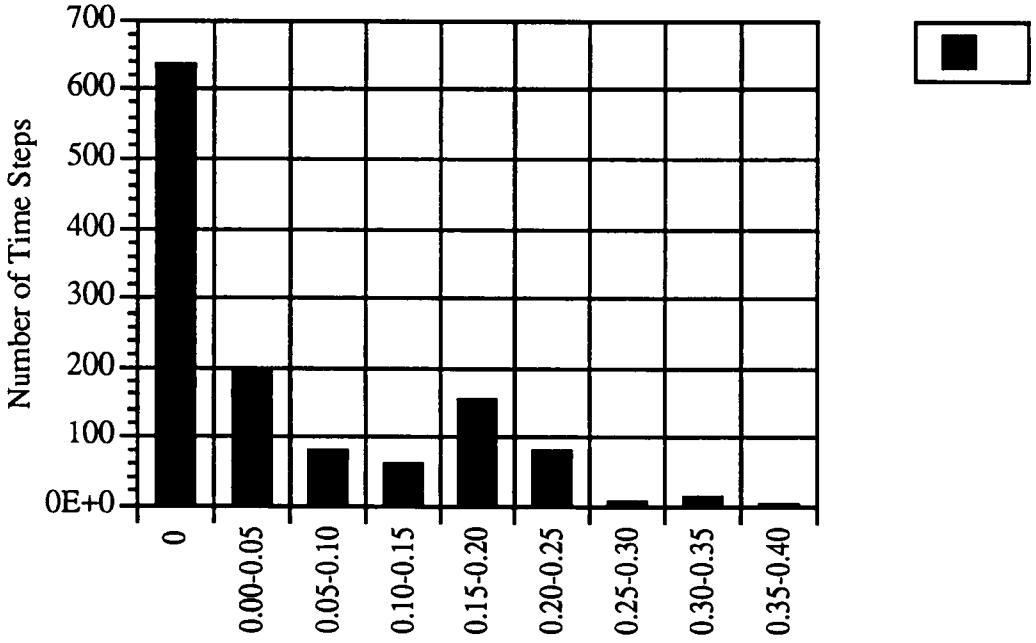

Figure 5.44 Distribution of the Euclidean norms between expected and predicted centroids when the average flux and secondary-flow signals have been eliminated.

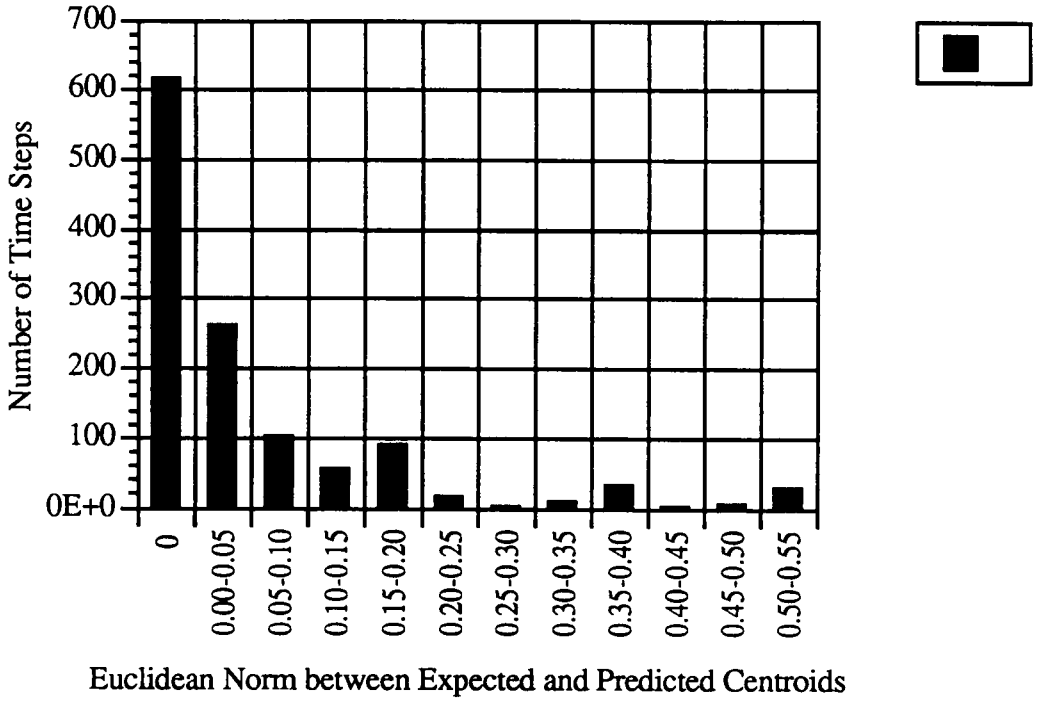

Figure 5.45 Distribution of the Euclidean norms between expected and predicted centroids when the average DP and inlet-temperature signals have been eliminated.

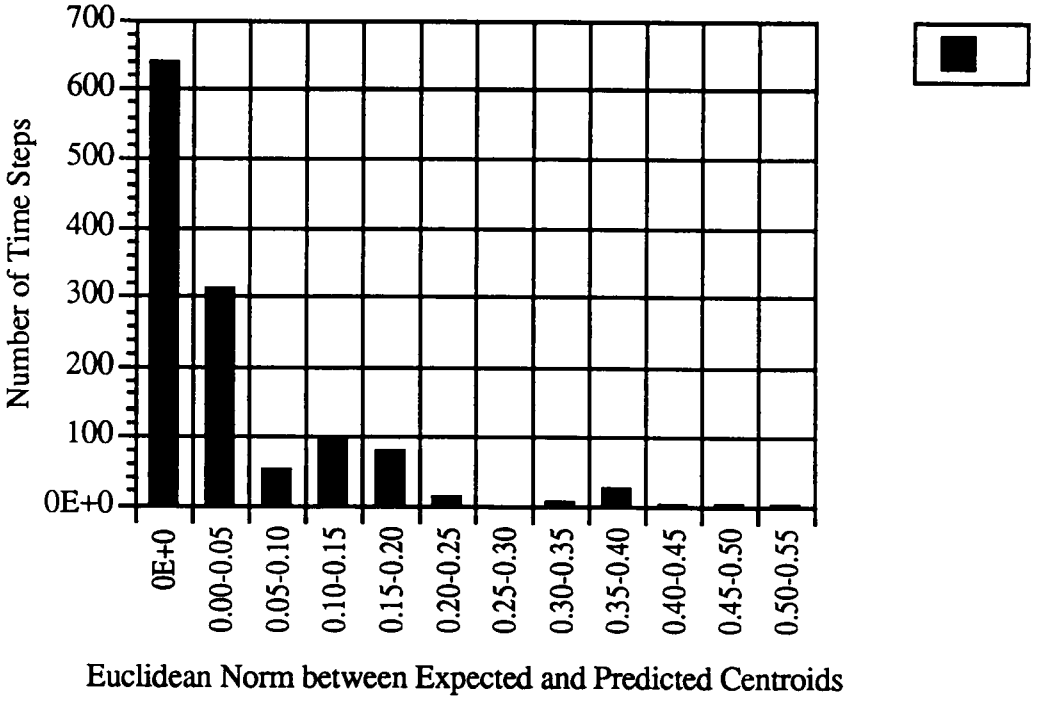

Figure 5.46 Distribution of the Euclidean norms between expected and predicted centroids when the average DP and outlet-temperature signals have been eliminated.

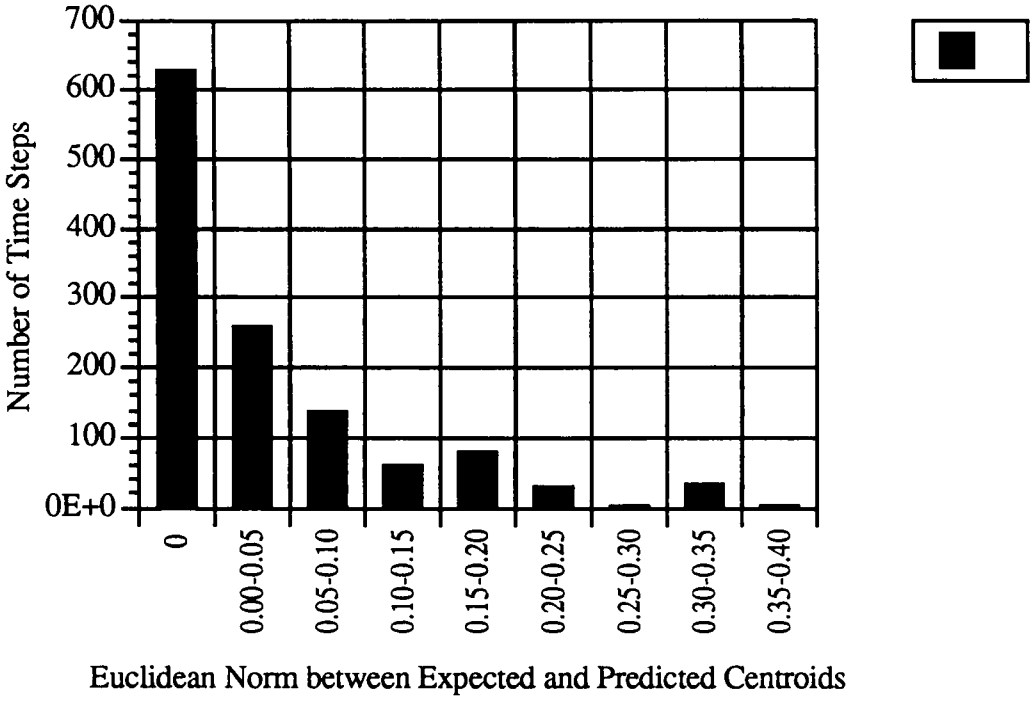

Figure 5.47 Distribution of the Euclidean norms between expected and predicted centroids when the average DP and secondary-flow signals have been eliminated.

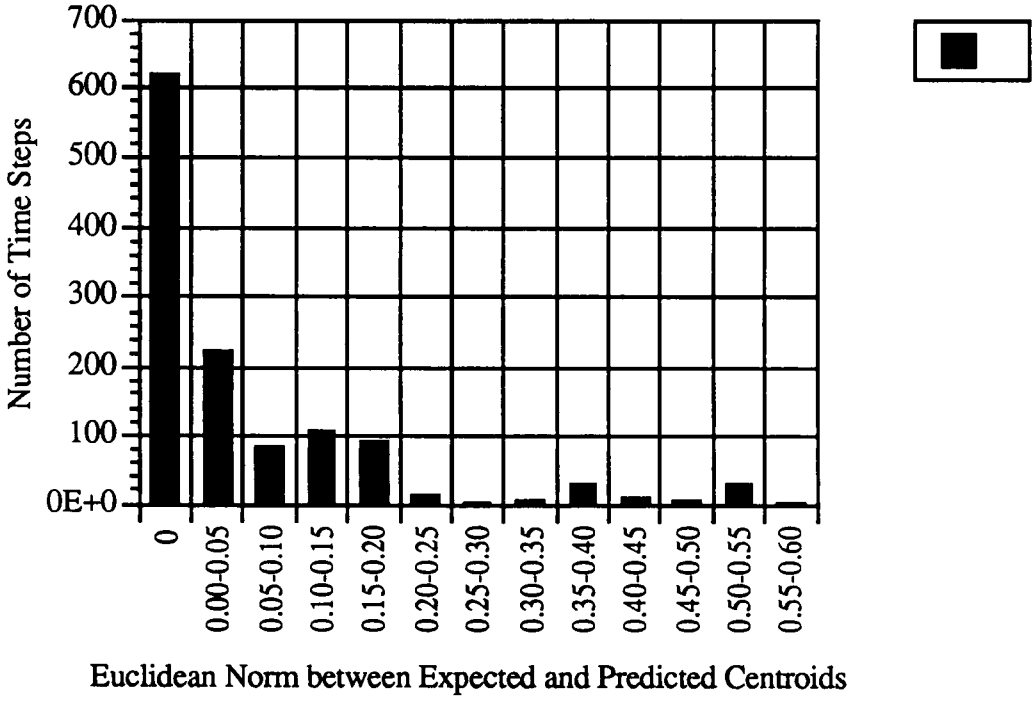

Figure 5.48 Distribution of the Euclidean norms between expected and predicted centroids when the average inlet and outlet-temperature signals have been eliminated.

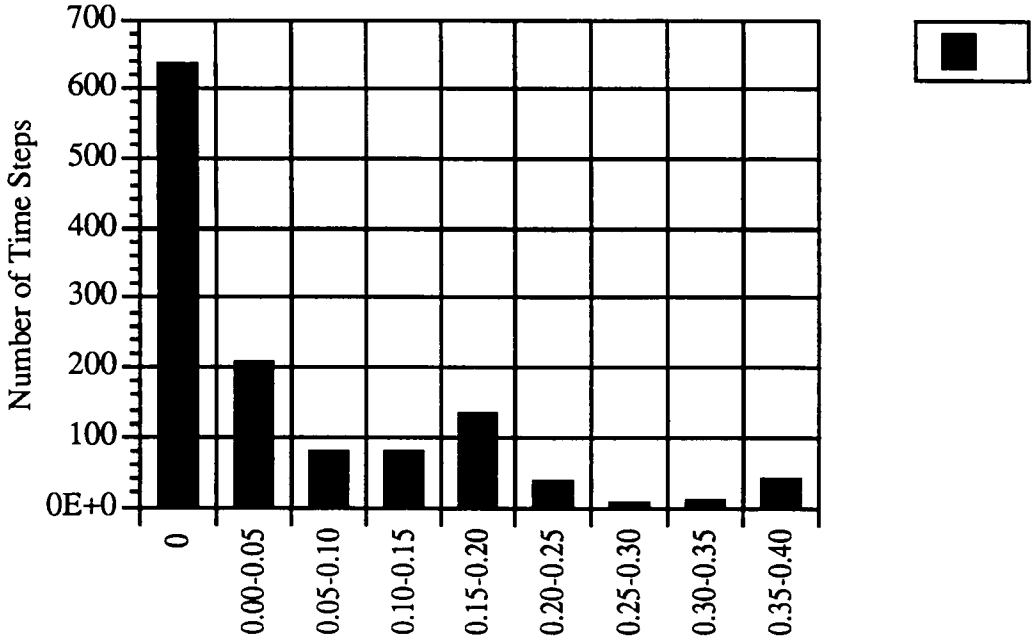

Figure 5.49 Distribution of the Euclidean norms between expected and predicted centroids when the average inlet-temp. and secondary-flow signals have been eliminated.

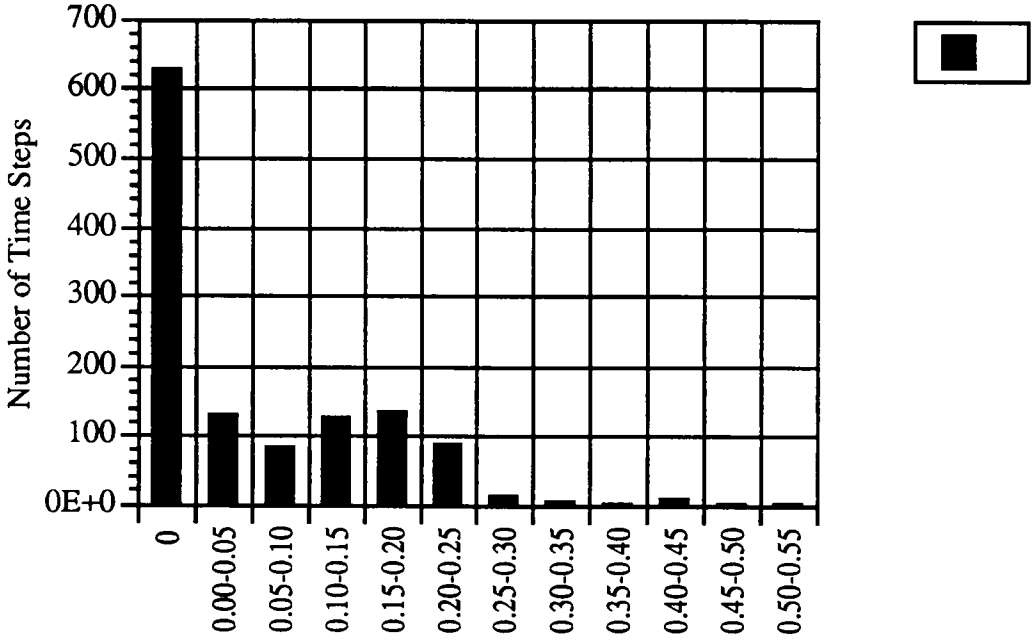

Figure 5.50 Distribution of the Euclidean norms between expected and predicted centroids when the average outlet-temp. and secondary-flow signals have been eliminated.

This behavior demonstrates the excellent fault tolerance characterizing the methodology developed for performing measurements of virtual variables with linguistic meaning. Apart from that, we see that there is a significant expansion of the distribution of the Euclidean distances, compared to the results obtained previously. This was expected since the amount of information dropped out of the input patterns has been doubled. Thus, the virtual measuring methodology developed exhibits the required sensitivity to the type of information supplied.

A closer look to Figures 5.42, 5.45, 5.48, and 5.49 reveals an interesting point. The distribution of the Euclidean norms deviates from the distribution shown in the rest of the graphs. The peak in the origin becomes sharper and a second peak appears in the middle of the x-axis. The common element among all these figures is the omittance of the average inlet-temperature signal from the input vectors. Certainly, a complete comparison between the Euclidean norms calculated during every testing case, and the correlation coefficients, can not be performed. There is no way of knowing *a priori* the exact amount of common information carried from separate signals, and as a consequence we can not predict the exact effect their substitution with random signals may have in the measurement process.

The correlation coefficient is a measure of the dependence implied from the joint probability distribution of two signals. The effect of the elimination of two signals may have on the output, can only be deduced from the joint probability distribution of three signals. A hypothetical way of establishing a correlation between two signals and a number of other time series would be by creating a new time series composed by both signals and then projecting the resultant time series on the other time series. Apparently, this is not feasible because composition of two dependent signals will create a new time

series with different statistical characteristics than the original signals. Therefore, this discussion is limited to a qualitative point of view based on indications provided from the results obtained in the previous tests.

Summarizing the above, one could say that the low correlation existing between the average inlet-temperature signal and the other training signals serves to accurately explain the behavior of the measuring device when random noise replaces an original input signal. When two or more signals are exchanged with random time series the information provided by the correlation matrix is not adequate for a satisfactory explanation of the results. The correlation coefficient in this case, acts only as a qualitative indication of appropriate behavior and does not pose the behavior itself. Apparently, the virtual monitoring methodology follows these indications and behaves properly.

As a last step in the same direction, three of the original signals have been replaced by random noise and the results are shown in Figures 5.51 and 5.52. In this case the average flux, inlet and outlet-temperature signals have been omitted. The Euclidean norms between the expected and predicted membership functions are shown in Figure 5.53.

In Figure 5.53 it is shown that there exists a significant deterioration in the accuracy of the prediction and the results can be considered valid only to some extend. The peak in the origin of the Euclidean norm diagram, has become very sharp and the area covered by the distribution has been significantly increased.

A point ought to be made about the magnitude of the peak in the origin of the Euclidean norm diagrams, that remains significantly high even during extreme tests. The reasoning for this behavior can be found in the second half of the start-up period, where the

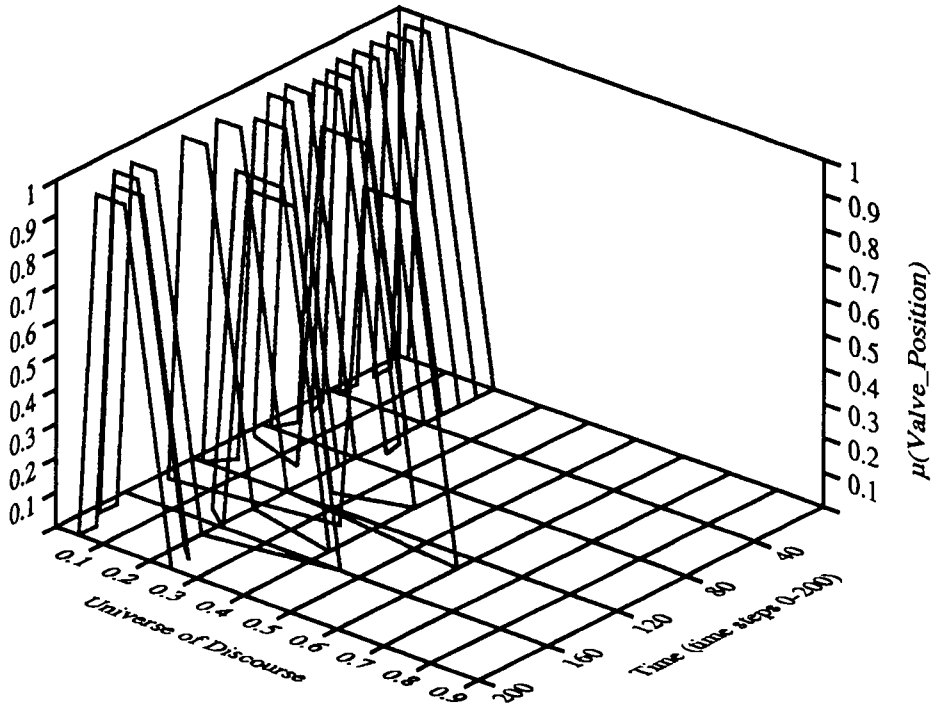

Figure 5.51 Virtual measurements for time steps 0-200 when the average flux, inlet, and outlet-temperature signals have been substituted with 100% noise.

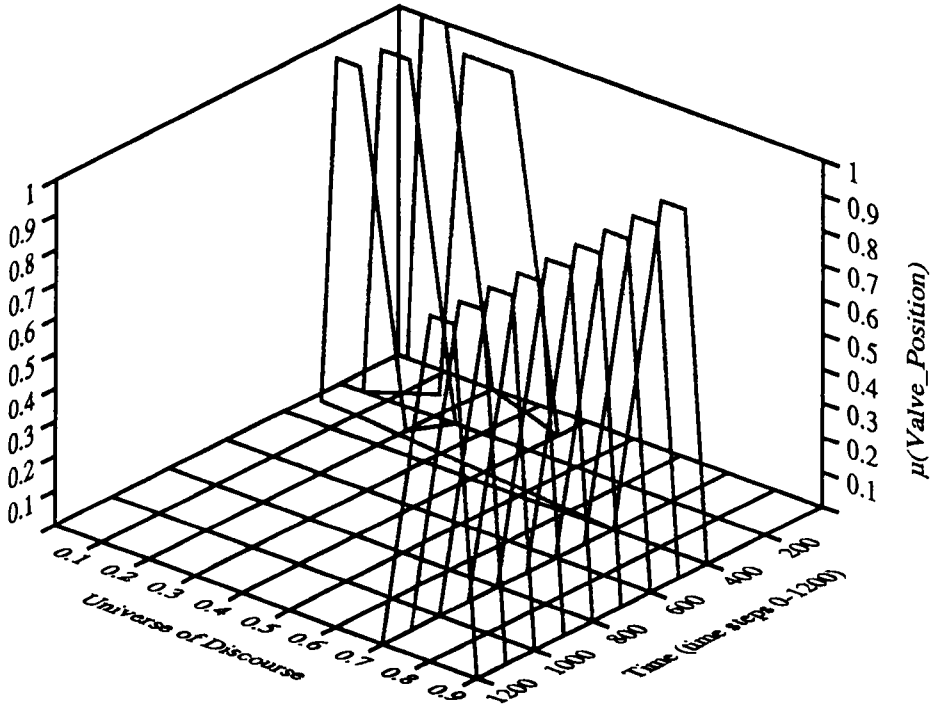

Figure 5.52 Virtual measurements for time steps 0-1200 when the average flux, inlet, and outlet-temperature signals have been substituted with 100% noise.

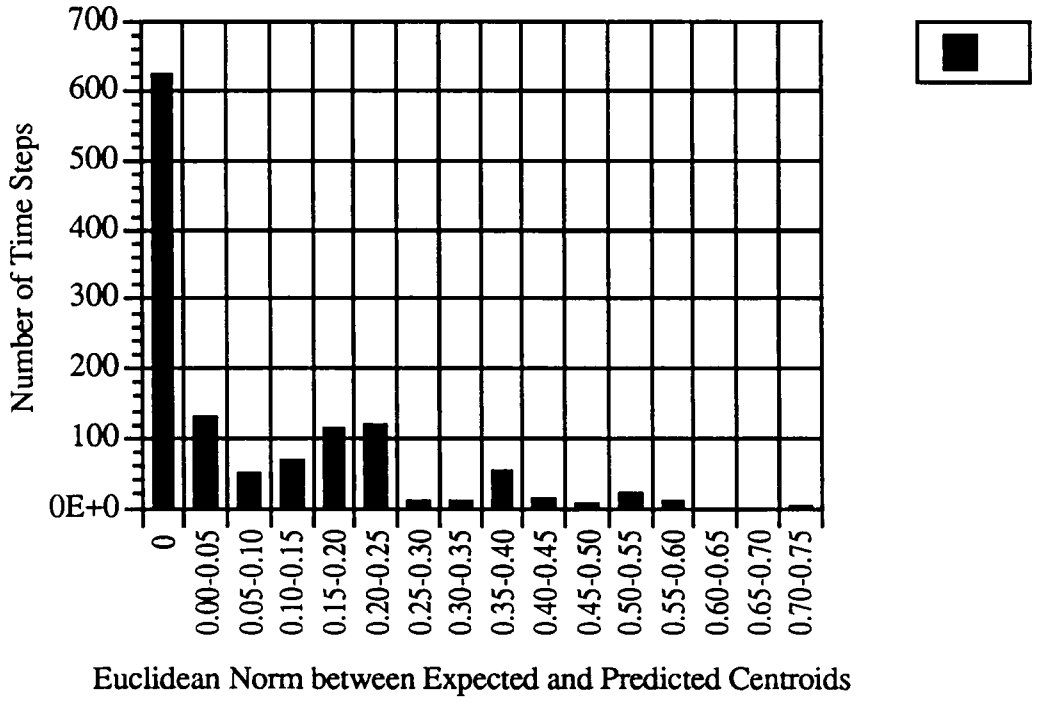

Figure 5.53 Distribution of the Euclidean norms between expected and predicted centroids when the average flux, inlet, and outlet-temperature signals have been eliminated.

valve is shown to initially approach the fully-open position and later stabilize at this level. During this time period the conditions of the system do not rapidly change and essentially remain constant after a certain time. Therefore, the vectors describing this state of the system are all located in the same area at the Euclidean space of the input vectors. Modification of the input vectors affects (through the replacement of original signals with random noise) only partially the position of these vectors and as a result the system is capable of distinguishing them. In other words, the measuring device is capable of accurately predicting the exact state of the valve - even when three fifths of the input information has been eliminated - when the system is in a steady-state operation (Figure 5.53). This should be expected since during normal operation the changes in the state of the system are extremely slow and the measuring device can accurately follow them with partial input information only. The remaining original information in the distorted input vectors is sufficient to describe the state of the system. Apparently, this does not hold

when the system is subjected to transients and all the system parameters are required to give an accurate estimation of the rapid changes taking place.

As a final step, we have examined the virtual measuring device behavior during two separate HFIR operation scenarios. No comparison can be made between the results demonstrated previously and those here, since they correspond to different system states. Hence, the following comments are focused on a qualitative description of the measuring device's general behavior. The first scenario consists of one hour of normal operation (100% power, fully-open valve-position) during a different fuel cycle than the one used for training the neural networks and developing the optimization algorithm. The system proved to be absolutely successful and predicted with excellent accuracy the exact valve-position (**open**) during the whole testing period, although a number of input parameters were different due to the change of the fuel cycle.

The second HFIR start-up scenario belongs to a different fuel cycle than before and the maximum power attained by the system is only 14%. Besides, the duration of the start-up is only a fraction of the previous one and the input signals behave significantly different from the time series used for model developing. In spite of all these drawbacks the virtual measuring device has achieved an overall success rate close to 80% at its estimations. This performance is well within satisfactory limits, considering the duration of the start-up, as well as the power level and the differences in the fuel cycles and the input signals. The measurement accuracy could be significantly increased by normalizing the data in an area pertaining to the nominal power level. Nevertheless, this start-up scenario can be incorporated into the training vectors rendering it part of the data bank of the measuring device. Apparently, this does not correspond to a case of interest, since this scenario is to be used for testing purposes only.

CHAPTER 6

APPLICATION ENVIRONMENTS

Overview

Our discussion up to this point has concentrated on the development of a methodology for measuring system parameters with linguistic meaning. The necessity for developing software-based virtual measuring devices was shown and analyzed in Chapter 4, and their capabilities demonstrated in Chapter 5. A virtual measurement instrument is not only capable of performing evaluations of system parameters in a qualitative form suitable for fuzzy control applications, it also performs its task in a time frame faster than real time. It seems therefore appropriate at this point to examine the specifics of the application environment required for the implementation of the proposed methodology. Hence, the virtual measuring technique is seen from the point of view of the anticipatory systems, and the possible role it may play in the anticipatory paradigm is further discussed. Furthermore, the applicability of the proposed technique for prediction of undesirable events in a nuclear power plant is investigated for different transient scenarios.

Anticipatory paradigm

In the anticipatory paradigm a control action is taken based not only on the current condition of the system but also on an estimate of what the system may be doing in the

future (Rosen, 1985b and Casti, 1988). The ideas underlying anticipatory control processes were first investigated by Robert Rosen (1985a, 1985b) in the field of mathematical biology. In conventional control techniques, the response of the system is mainly reactive, i.e. the operator or the system itself reacts according to the present behavior of the system (Bellman, 1961). This description may be formulated in a mathematical representation as follows

$$\frac{dx_i}{dt}\Big|_t = f_i\left(x_1(t),\ldots,x_n(t)\right) \tag{6.1}$$

where x_i's are the state variables of the system. It is well known that this approximation of complexity by simplicity is local and temporary. Generally, the stability of a system can be examined through a set of quantities of the form

$$u_{ij}\left(x_1(t),\ldots,x_n(t)\right) = \frac{\partial}{\partial x_i}\left(\frac{dx_i}{dt}\right) \tag{6.2}$$

where i,j = 1,2,...,n and u_{ij} are control laws. Unfortunately, this applies mostly to systems for which linear, time-invariant, models can be designed and operated in accordance with the stability conditions of mathematical system theory. For complex systems non-linearities and operator/system interactions have primacy and hence, general conditions or procedures that simultaneously meet robustness, stability, and good dynamical response, are mostly not available (Narendra and Parthasarathy, 1990). Difficulties with the practical implementation of various control algorithms is a well known fact for engineering practice particularly for large scale systems. In addition, it can be demonstrated mathematically that only a relatively small class of systems can be effectively described by the tools of mathematical system theory (Rosen, 1985a).

Contrary to that philosophy, the anticipatory paradigm provides a model where the system behavior at a particular time t, is based on previous and future measurements of the state of the system (Rosen, 1985b). Taking this into consideration, Equations (6.2) may be modified to include the change of x_i at time t as a function of the values of all available x_i's up to time t and their anticipated values in the future (t + Δt). When time advances to (t + Δt), the system evaluates the validity of its earlier predictions and proceeds to re-evaluate them. Generally, it may be asserted that a prominent part of human behavior is based on the activation of predictive models (Ivakhnenko and Lapa, 1967). The fundamental building blocks of an anticipatory system are shown in Figure 6.1, where; **S** is the actual system, **M** is a predictive model, **C** is the controller, and **E** is the environment.

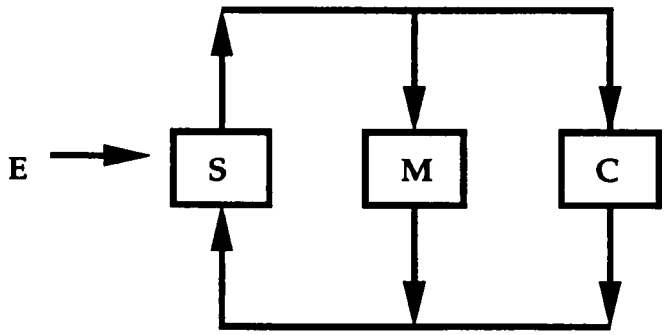

Figure 6.1 Constituents of an anticipatory system.

In the anticipatory paradigm, a model **M** is used to predict the future states of the system. If **S** and **M** are started at the same time, t = 0, in equivalent states, and if real time is allowed to run for a fixed interval, t, then **M** will have proceeded further along its course, than **S**. In this way the behavior of **M** predicts the behavior of **S** (Ragheb and Tsoukalas, 1988). That is, the state of **M** at time t describes the state of **S** at a later time t+Δt. Therefore, by consulting the information supplied by **M** at a particular instant in time, we acquire knowledge about the state of **S** in the near future.

From the above discussion, we see that the anticipatory paradigm offers a natural application environment for the proposed technique. The virtual measuring device has been designed and developed in order to incorporate the predictive characteristics of the **M** block in Figure 6.1. The model receives at time t, a number of signals describing the state of the system **S** at the present time step. It uses this information to predict the state of a particular system parameter at time $t + \Delta t$. The rule-based diagnostician imitates the role of the **C** block by drawing a conclusion about the present state of a particular system component. In addition, the virtual measurements are in a linguistic form suitable for human cognition and application of fuzzy control algorithms. Apparently, the same process can be expanded so that the network prediction will cover the time $t+2\Delta t$. In this case the rule-based system will provide an estimation for time $t+\Delta t$, and the whole measuring model will constitute the **M** predictive block whose response will be used by the **C** constituent.

It should be clarified that although in the application presented the prediction pertains to a non-measurable parameter this need not be the case. The reason we chose to demonstrate the proposed technique with non-directly monitored parameters is that they present a more challenging problem, as it was discussed in the introduction of Chapter 4. Apparently, the same methodology can be used for predicting future states of physically monitored system parameters.

Transient Identification

Another area of application of the proposed methodology is the prediction of events with undesirable consequences, such as the identification of developing transients in a nuclear power plant. The major objective of a transient identification system, is to provide

sufficient information to the human operators for assisting them in taking action to mitigate the transient consequences. This identification process is a difficult task since many transients are fast evolving and present similar indications. In order to examine the applicability of the proposed methodology we have applied it for distinguishing among different transients taking place in a nuclear power plant. (Ikonomopoulos, et. al. 1991a, 1991b). In the proposed integration ANNs are pre-trained using data from a nuclear power plant simulator. A pre-trained neural network is the receiver of several on-line time series corresponding to vital plant parameters. It primarily acts as a filter and classifier. The noise of the incoming series is filtered, and the output of the system is a membership function representing a particular transient. The membership function encodes only the necessary information for transient identification, since its shape is **unique** and corresponds to a particular accident scenario (Ikonomopoulos, Tsoukalas and Uhrig, 1991b).

Among different postulated transients that may occur in a Westinghouse four-loop pressurized water reactor (PWR), the most difficult to distinguish are those categorized as *Condition IV - Limiting Faults* (Collier, 1987). Although the probability of occurrence of this type of transients is very low, they have been extensively studied since their consequences may include the release of significant amounts of radioactive materials to the reactor containment. Two of the transients that are categorized as *Condition IV - Limiting Faults* are:

1. *Rod Cluster Control Assembly Ejection (RCCAE)*, and
2. *Steam Generator Tube Leak (SGTL).*

RCCAE is defined as the mechanical failure of a control rod mechanism pressure housing resulting in the ejection of a rod cluster control assembly and its drive shaft.

Immediately after initiation, there is a significant amount of positive reactivity insertion into the system as a result of the removal of the control rods. The sudden control assembly ejection will automatically trip the system. After the reactor trip the major system parameters will sharply decrease to shutdown levels, mitigating therefore the consequences of the transient.

A SGTL transient is caused by a failure of a single steam generator tube. The leakage of reactor primary coolant into the secondary side will cause a decrease in the primary pressure. On the secondary side there is a steam flow/feedwater flow mismatch as feedwater flow to the affected steam generator is reduced due to the additional break flow which is being supplied to the unit. Continued loss of reactor coolant inventory leads to a reactor trip signal generated by low pressurizer pressure. Later the primary system temperature will decrease following the feedback from the negative temperature coefficient of reactivity.

Both transients exhibit very similar behavior in almost all of the system parameters throughout their development periods. In fact, the scenarios used have been simulated at the Watts Barr NPP simulator and the RCCAE transient has only a 46 second faster development than the SGTL transient. Apart from that, the behavior of all the vital system parameters could be perceived as identical. The data received is normalized in the interval 0 to 1 and sampled every 5 seconds with a total time span of 180 seconds. In order to represent the two transients, five significant parameters were chosen as inputs to the ANN as shown in Table 6.1. All five time series represent average values of the corresponding parameters of the four loop system. The signals of these five parameters are used to train an ANN with five input nodes at the input layer and two sets of four output nodes at the output layer. The output is a trapezoidal membership function labeling a specific scenario.

Table 6.1 ANN inputs and outputs.

INPUT TIME SERIES	OUTPUT MEMBERSHIP FUNCTIONS
primary pressure	
hot leg temperature	μ_{RCCAE}
primary coolant flow	μ_{SGTL}
steam pressure	
steam flow	

The membership function μ_{RCCAE} is defined by a trapezoid with peak coordinates {(0, 0), (0.1, 1), (0.2, 1), (0.3, 0)}, where the μ_{SGTL} is depicted by the coordinates {(0.2, 0), (0.3, 1), (0.4, 1), (0.5, 0)}. It is apparent from these geometrical schemes that the two membership functions have a significant overlapping which was introduced in order to represent the large similarity of indications exhibited by the two transients. The area occupied by both membership functions in the universe of discourse is the same since they are analyzed over the same time span. In order to incorporate the time lag existing in the development of the RCCAE over the SGTL transient the μ_{RCCAE} membership function was set in the left most side of the universe of discourse. Utilizing that approach, there is a distinct representation of the time difference in the development of the transients in the position of the corresponding membership functions. The shape of the membership function which has been assigned to each transient is unique and therefore there is a sharp distinction between different accidents even in the extreme case considered here, where they exhibit very similar behavior.

The neural network used is a three-layer network consisting of an input layer, a Kohonen layer and a Grossberg outstar layer, and the training algorithm is counter-propagation (Wasserman, 1989). All layers have been constructed and trained using the

Plexi neural environment in a SPARC workstation. The neurons in the input layer serve only as fan-out points and do not perform any computation. The input layer of the network consists of five nodes, each one receiving an input from a particular time series. Two sets of four nodes compose the output layer, each one representing the coordinates of the peaks of each of the two trapezoids. A total of 72 nodes compose the Kohonen layer (each one corresponding to a different input vector) and the number of epochs was 10,000. In order to test the ability of the network to predict the membership functions of the RCCAE and the SGTL transients, different levels of noise were introduced in all five input signals.

First we introduced 10% noise to all of the input signals and tested the trained network. When the test vectors corresponding to the RCCAE transient were applied, the results obtained from the network are exhibited in Figure 6.2.

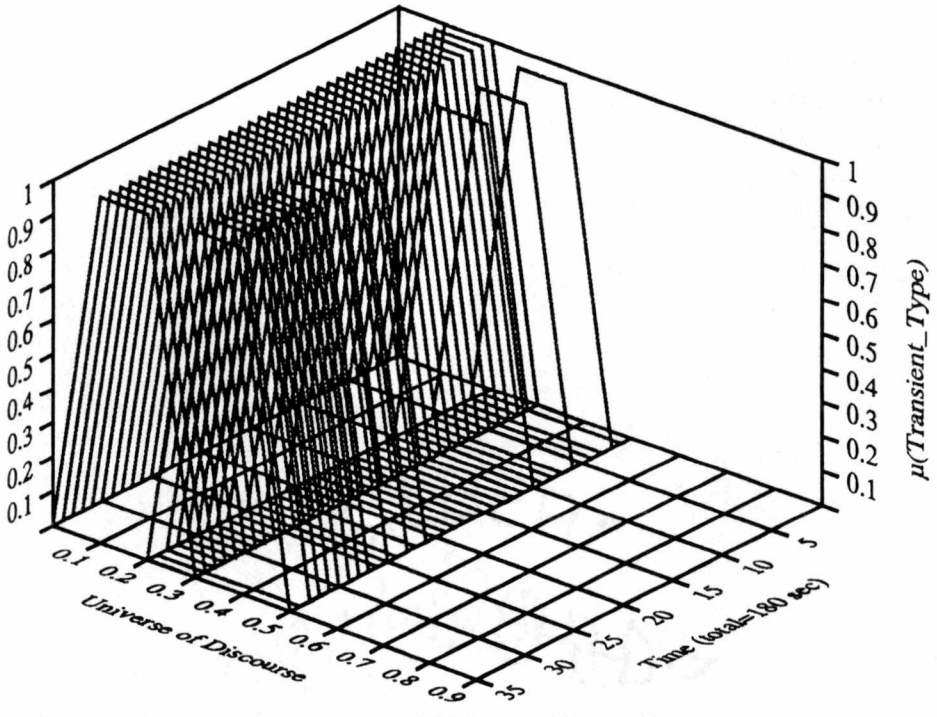

Figure 6.2 Network derived membership functions with 10% noise at all input signals.

The four output neurons corresponding to μ_{RCCAE} gave output at all 36 time steps, denoting therefore that there is a complete match between the 36 time steps corresponding to the RCCAE transient and the 36 noisy test vectors introduced to the system. On the other hand the four neurons corresponding to μ_{SGTL} gave output in 14 time steps. The majority of the time steps where μ_{SGTL} was predicted belongs to the very end of the transient where the system is shut-down and all of its parameters obtain identical values (become almost zero). It was therefore expected that the neurons corresponding to SGTL fire along with those corresponding to RCCAE. At this particular time the input vectors of both transients are almost the same, and hence occupy the same positions in the hypersphere.

As a next step, the noise level in all five input signals was increased to 20% and the network was tested, applying again the RCCAE transient scenario first. The results were identical to those shown in Figure 6.2 for the previous case. The neurons corresponding to the RCCAE fault fired in all 36 time steps and those corresponding to the SGTL accident lighted at exactly the same 14 time steps as before. Finally, the 36 noisy vectors corresponding to the SGTL transient were supplied to the network. This time the results (Figure 6.3) were not as in the previous case. The SGTL transient was predicted in all but two of the scenario time steps. At time steps 24 and 25 the network did not calculate the μ_{SGTL} shape at all. On the other hand the neurons trained to represent the RCCAE fault, were activated at 20 time steps though most of them appeared in the very end of the transient as it was expected for the reasons described previously.

It is interesting to notice that although the quality of information obtained with 20% noise in all input signals for the RCCAE transient is the same as in the case of 10% noise, this does not hold also for the SGTL transient. The insertion of noise into the input vectors

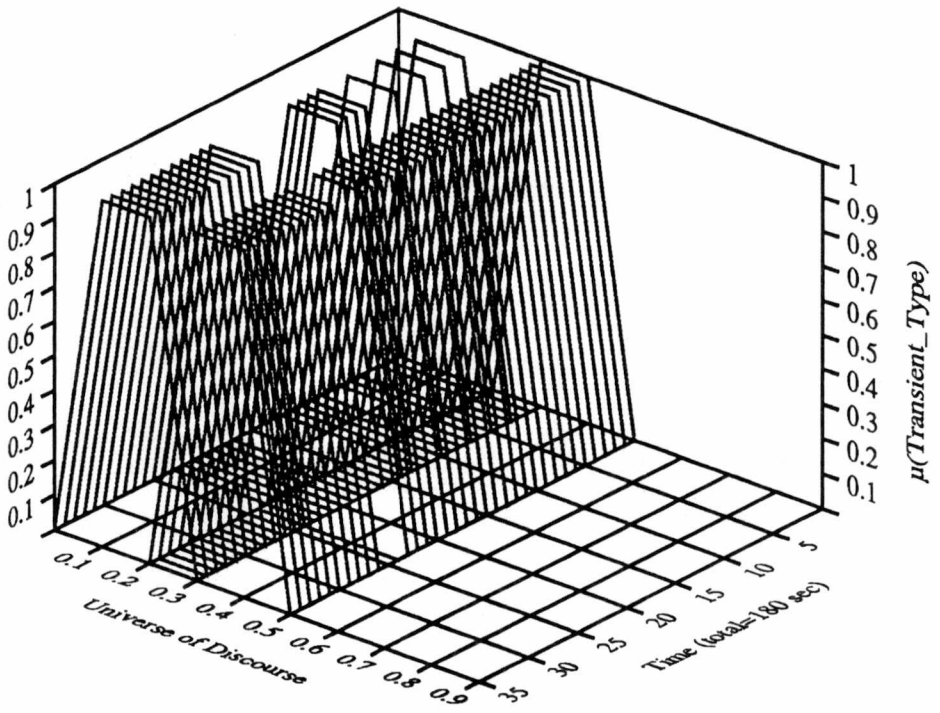

Figure 6.3 Network derived membership functions with 20% noise at all input signals.

caused them to randomly distribute eventually capturing a weight vector (Khanna, 1990). This random distribution moved input vectors to occupy random positions inside the hypersphere.

It should be mentioned that the transient identification problem has also been approached using a set of ANNs trained with the back-propagation algorithm (Ikonomopoulos, Uhrig and Tsoukalas, 1991a). The results obtained exhibit very similar characteristics to those presented here, and suggest a similar approach to the NPP transient identification problem. Further application of the proposed methodology on the same set of data, does not seem appropriate at the present stage. Nevertheless, the choice of a single neural network does not obligate the introduction of the *dissemblance index criterion.* The responses calculated from the neural network (Figures 6.2 and 6.3) are sufficient for

distinguishing between the two transients and furthermore identify the exact type of every transient very early in the transient development. Thus, the utilization of the rule-based system is not necessary in this particular application. The reason for this lies in the type of numerical data used in this problem. The time series used for training the neural network have been provided by a simulator and are only an approximation to the real behavior of the nuclear reactor. The transient initiations are accompanied by distinctive parameter changes that partially correspond to real applications. Contrary to the data used previously for the model development, the signals used this time are noise-free and describe transitions from one state to another in a smooth, almost linear, manner. Thus, the problem is over-simplified, and the application of neural network methodologies alone is sufficient for resolving the classification issues raised.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

A methodology for monitoring complex systems employing artificial neural networks and fuzzy logic has been developed. It employs the notion of a virtual measuring device, i.e., a software based instrument for the "measurement" of user-specified dynamic variables with operational significance. The function of such devices can be modified by changing their software, not hardware, (e.g., re-train the neural networks to "measure" *transient-type*) a promising feature for application specific monitoring tasks. The building blocks of the measuring technique are graphically presented in Figure 7.1. Neural networks are employed for mapping a set of complex, temporal, input patterns to a simplified set of membership functions. The produced membership functions are used as input to a rule-based system which draws a conclusion about the validity of the network responses based on the statistical characteristics of the original training signals. The notion of time is explicitly incorporated into the decision-making procedure rendering the measuring process a dynamic one.

The issue of adjacent membership function overlapping in the universe of discourse, due to the inherent inability to define exact borders of class membership, is resolved with the implementation of an optimization algorithm. It treats the membership functions estimated from the rule-based diagnostician as fuzzy numbers and combines them according to the degree of satisfaction of the ***dissemblance index criterion.*** The optimization algorithm produces a membership function which uniquely and

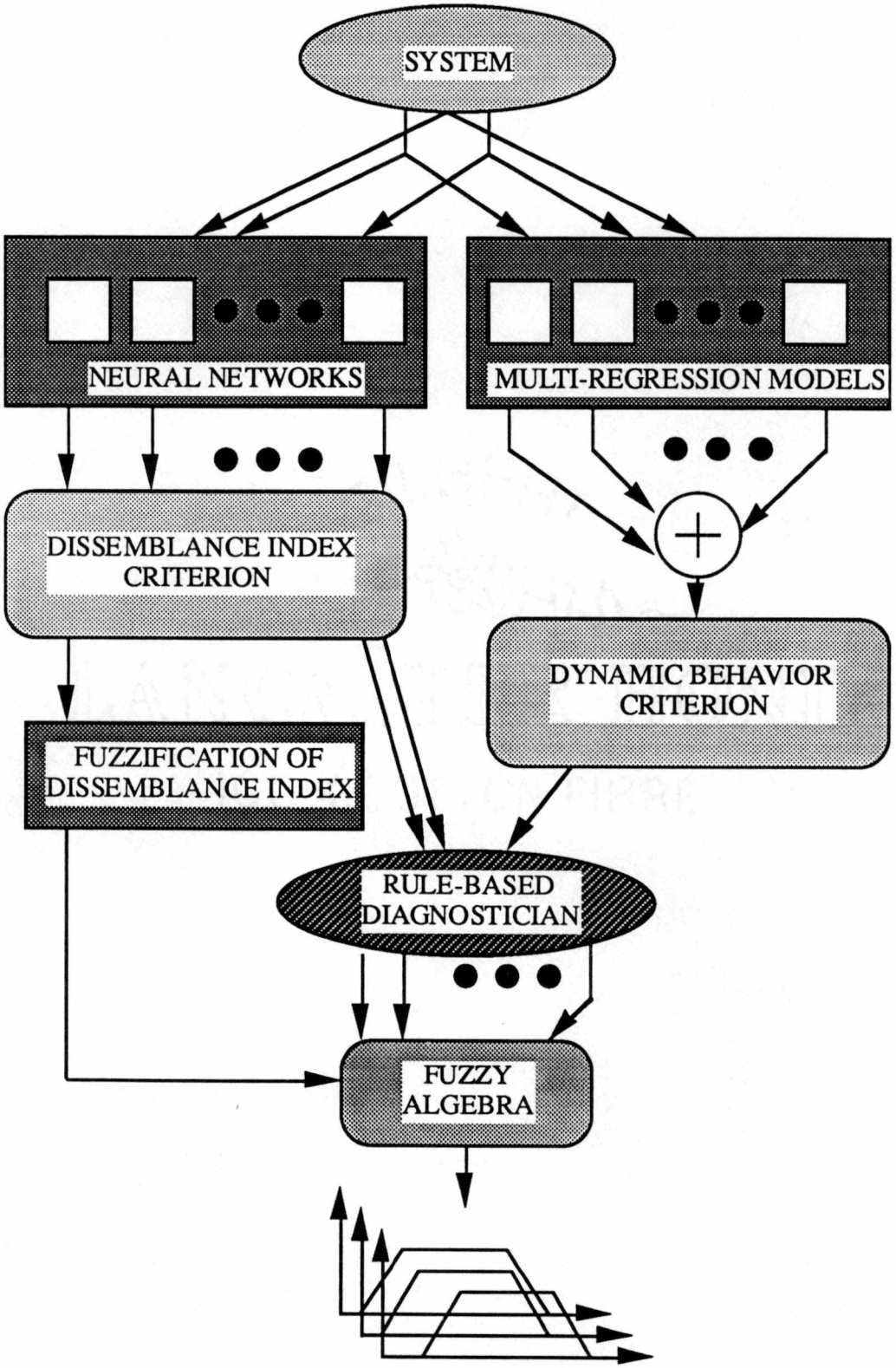

Figure 7.1 Software-based virtual monitoring device.

unambiguously represents the values of variables that are fuzzy, such as valve-position and transient_type.

The representation in the form of membership functions is sufficient to preserve the necessary information for distinguishing between different states of the monitored system variable. On the basis of the results demonstrated previously it is worth mentioning the noise as well as fault tolerances of the overall methodology. The ability of the measuring device to pick up the necessary information from signals embedded in 10% noise is unique. The excellent predictions supplied from the virtual monitoring technique when 100% noise replaces one of the original signals, exhibit the unique tolerance to fault-signals characterizing the software-based methodology. Additionally, the model behavior during substitution of different input signals by totally random time series, suggests that the methodology developed is capable of not only recognizing the statistical properties of the device to be monitored, but also preserving them.

Typically, a control algorithm consists of a set of IF/THEN rules associating *situation* or *antecedent* propositions, s_j, with *action* or *consequent* propositions, a_j, through an inference procedure referred to as *classical modus ponens*, where each s_j, a_j is a proposition of the form "*X* is A." This formalization guarantees the truth of a consequent given that the antecedent of the syllogism is true. However, these type of logical rules fail to give an answer when the satisfaction of the antecedent clause is partially valid, i.e., the prerequisite is only partially satisfied. Zadeh (Dubois and Prade, 1980) proposed what became widely known as *generalized modus ponens* which is actually an extension of the classical modus ponens. An example of this type of syllogism is given below

$s_j \rightarrow a_j$: IF the tomato is *red* THEN the tomato is *ripe*.

$s_j' \rightarrow a_j'$: IF the tomato is *partially_red* THEN the tomato is *partially_ripe.*

In generalized modus ponens the antecedent is true only to some degree (e.g., *partially_red*) and hence it is desired to compute the degree to which the consequent is true. Fuzzy sets is the natural environment for this type of computations, since fuzzy variables (e.g., tomatoes) can take fuzzy values (e.g., *red, partially_red*) which uniquely quantify the degree of membership to a class. In fuzzy control the fuzziness of the antecedents eliminates the need for a precise match with the input. Each rule is fired to a degree that is a function of the degree of match between the antecedents and the input.

Two major problems are encountered, however, in the application of fuzzy reasoning, (a) the design of the membership functions, and (b) the lack of adaptability for possible changes in the reasoning environment. The work presented here addresses primarily the first issue employing neural networks as membership function generators. The membership functions calculated from independent firing networks belong to a set of prototype membership functions which sufficiently describe the universe of discourse. The *dissemblance index* and *dynamic behavior criteria,* along with the fuzzification of the degree of satisfaction of the *dissemblance index criterion*, establish an optimization algorithm for modifying the prototype membership functions. The resultant membership functions encode all the necessary information for an accurate system description. The optimization algorithm is suited for adjusting the membership functions, in a way relevant to possible changes in the input environment (i.e., partial elimination of input information).

Besides the above, it should be mentioned that the measurement process takes place in a time frame faster than real time. The neural networks used as receivers of time series and generators of membership functions perform this task in a predictive mode, since the

calculated fuzzy values describe the state of the system at the next time step. Hence, this type of measuring processes are suitable for operating in an anticipatory environment. The controller will have at its disposal not only information pertaining to the present state of the system, but also accurate estimations about the future. In this manner, the operating environment will become **active** rather than **reactive**.

Finally, the methodology for developing virtual measuring devices capable of producing *virtual measurement values* may be summarized in the following general algorithm (Ikonomopoulos, Tsoukalas and Uhrig, 1992b):

1. Determine the number of partitions necessary to adequately cover the range of the device.
2. Select the number of physically measurable variables that will be the input of the virtual measurement device.
3. Train every neural network to recognize a single state of the monitored fuzzy variable.
4. Calculate the maximum dissemblance index value for every network response and pose the *dissemblance index criterion.*
5. Develop multi-regression models to identify the statistical properties characterizing the training signals and define the *dynamic behavior criterion.*
6. Design an appropriate logic using the *dissemblance index* and *dynamic behavior criteria,* in order to develop the rule-based diagnostician.
7. Fuzzify the degree of satisfaction of the *dissemblance index criterion* in order to modify the prototype membership functions.
8. Combine adjacent membership functions to select a fuzzy number that will be the actual output value of the instrument at any given time.

Between steps 2 and 3 in the above algorithm, the calculation of the correlation matrix among the input time series is strongly recommended. In case one of the signals exhibits low degree of dependence from the remaining signals, a similar signal should be used as input also. Thus, possible elimination of this signal will not greatly affect the measurement process.

The initial partition of the universe of discourse should be articulate enough to adequately identify all possible system states. A well posed optimization algorithm may determine the most appropriate partition at every time step. In case someone decides to expand the application environment of the monitoring device, he only needs to further train the neural networks to the new input patterns. In this case steps 4, 5, and 6 in the above algorithm should be repeated.

One of the most crucial issues in the synthesis of anticipatory systems is the determination of the anticipatory step or the "look-ahead time." In the present thesis it is taken to be equal to one time step and the neural networks predict the state of the system at the next time step. More research is required with a bigger set of data for the determination of the "look-ahead time." Two possible routes can dilate the prediction time. We may either train the networks to predict the condition of the system at time $t+2\Delta t$, or modify the logic incorporated in the rule-based diagnostician so that it will draw a conclusion about the future time step instead of the present. In the first scenario there is a need to examine the predictability of the system by applying the model selection criteria discussed in Chapter 4. In this case the logic incorporated into the rule-based diagnostician remains the same and a decision is made about the state of the system at $t+\Delta t$. In the second scenario, a modification needs to be made in the time window of the optimization algorithm in order to draw a conclusion about the next time step. Apparently, a procedure is needed for

verifying the current network prediction so that it can be used for prediction of the future state of the system.

An area of further inquiry is to decide the most appropriate algorithm and architecture for neural network training. The back-propagation algorithm has established itself as the most promising tool for classification purposes but still there is no formal model for selecting the optimum number of layers and neurons required for every application. The network architecture depends on the experience of the modeller who mainly follows his intuition and practical rules of thumb. The counter-propagation algorithm used for nuclear power plant transient identification, resolves the issue of information surplus created from the utilization of more than one neural networks. On the other hand, it is difficult to train networks applying this algorithm when the number of input vectors becomes significantly large and the Euclidean distance among them is very small.

In this research it has been shown that the virtual measuring device is capable of responding to novel stimuli when this is within the context of the set of vectors it was trained on. Further research needs to be done in examining the possibility of "on-line" training of neural networks when presented with totally "unknown" information. The proposed algorithm assumes that the modeller has an a priori knowledge of all possible scenarios that may take place. The resolution of this issue requires the implementation of a procedure responsible for classifying input vectors as "previously-unseen." These vectors can then be used for training further the existing networks or new ones.

The procedure implemented in this work for combining responses supplied from different firing networks is based on the application of fuzzy algebra. The tools used are

the multiplication between fuzzy and crisp numbers and the addition of fuzzy numbers. It would be worth investigating whether fuzzy IF-THEN rules could perform the same task. In other words, it needs to be further examined whether a reliable conclusion could be drawn about the validity of the current network prediction based on information supplied from crisp measurements of other system parameters.

LIST OF REFERENCES

LIST OF REFERENCES

AKAIKE, H. (1973). Information theory and an extension of the maximum likelihood principle. *Second International Symposium on Information Theory*, Petrov, B. N. and Csaki, F., Eds., Budapest: Akademai Kiado, 267-281.

AMEMIYA, T. (1980). Selection of regressors. *International Economic Review*, Vol. **21**, No. 2, 331-354.

BARR, A. & FEIGENBAUM, E. (1982). *The handbook of artificial intelligence*. Volume II, Stanford: Heuristech Press.

BELLMAN, R. E. (1961). *Adaptive Control Process - A Guide Tour*. Princeton: Princeton University Press.

BELLMAN, R. E. & ZADEH, L. A. (1969). Decision making in a fuzzy environment. *ERL-69-8*. Electronics Research Laboratory, University of California, Berkeley.

BENDAT, J. (1953). *Principles and applications of random noise theory*. New York: John Wiley & Sons.

BRYSON, A. E. & HO, Y. C. (1969). *Applied optimal control*. New York: Blaisdell.

CASTI, L. J. (1988). *Alternate realities: mathematical models of nature and man*. New York: John Wiley & Sons.

CHARNIAK, E. & McDERMOTT, D. (1984). *Introduction to artificial intelligence*. Reading: Addison-Wesley Publishing Co.

COLLIER, J. (1987). *Introduction to Nuclear Power*. New York: Hemisphere Publishing Corporation.

CYBENKO, G. (1989). Approximation by superposition of a sigmoidal function. *Mathematics of control, signals and systems*. **2**, 303-314.

DUBOIS, D. & PRADE, H. (1980). *Fuzzy sets and systems, theory and applications.* Boston: Academic Press.

DYATLOV, A. (1991). How it was: an operator's perspective. *Nuclear Engineering International.* **November 1991**, 43-50.

FEYNMAN, R., LEIGHTON, R. & SANDS, M. (1963). *The Feynman lectures on Physics.* Reading: Addison-Wesley.

GABOR, D., WILBY, W. P. & WOODCOCK, R. (1961). A universal non-linear filter, predictor and simulator which optimizes itself by a learning process. *Proceedings IEE*, **1088**, 422-438.

GABOR, D. (1967). Predicting Machines. *Scientia*, **XCVII-5-6**, 113-117.

GAINES, B. R., (1976a). Foundations of fuzzy reasoning. *Int. J. Man-Machine Studies*, **8**, 623-668.

GAINES, B. R., (1976b). On the complexity of causal models. *IEEE Transactions on Systems Man and Cybernetics*, **SMC-6**, 56-59.

GAINES, B. R., (1977). Sequential fuzzy system identification. *Proceedings of IEEE Conference in Decision Control*, New Orleans, 1309-1314.

GAINES, B. R. & COHOUT, L. J. (1977). The fuzzy decade: a bibliography of fuzzy systems and closely related topics. *Int. J. Man-Machine Studies*, **9**, 1-68.

GOTTINGER, H. W. (1983). *Coping with Complexity.* Dordrecht: D. Reidel Publishing Company

HABETS, P. & PEIFFER, K. (1973). Classification of stability-like concepts and their study using vector Lyapunov functions. *Journal of Mathematical Analysis and Applications*, **44**, 537-570.

HORNIK, K., STINCHCOMBE, M. & WHITE, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, **3**, 551-560.

HERTZ, J., KROGH, A. & PALMER, R. (1991). *Introduction to the theory of neural computation*. Redwood City: Addison-Wesley Publishing Company.

HOPFIELD, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, **USA 79**, 2554-2558.

IKONOMOPOULOS, A., UHRIG, R. E. & TSOUKALAS, L. H. (1991a). Fuzzy logic - artificial neural network integration for transient identification. *Proceedings of AI91: Frontiers in Innovative Computing for the Nuclear Industry*, Jackson, WY, 217-226.

IKONOMOPOULOS, A., TSOUKALAS, L. H., & UHRIG, R. E. (1991b). Fuzzy logic - artificial neural network integration for transient identification. *Intelligent Engineering Systems Through Artificial Neural Networks*, Dagli, C. H., Kumara, S. R. T. & Shin, Y. C., Eds., New York: ASME Press, 251-257.

IKONOMOPOULOS, A., TSOUKALAS, L. H., MULLENS, J. A. & UHRIG, R. E. (1992a). Monitoring nuclear reactor systems using neural networks and fuzzy logic. *Proceedings of the 1992 ANS Topical Meeting on Advances in Reactor Physics*, **2**. Charleston, S. C., 140-151.

IKONOMOPOULOS, A., TSOUKALAS, L. H. & UHRIG, R. E. (1992b). Diagnostics and control of pressurized reactors using artificial neural networks. *Proceedings of Intelligent Information Systems,* SPIE Conference, Orlando, FL.

IVAKHNENKO, A. G. & LAPA, V. G. (1967). *Cybernetics and Forecasting Techniques*. New York: American Elsevier Publishing Company.

JUDGE, G. G., GRIFFITHS, W. E., LUTKEPOHL, H. & LEE, T. C. (1985). *The theory and practice of econometrics*. New York: John Wiley and Sons.

KALMAN, R. E. (1957). *Analysis and Synthesis of Linear Systems Operating on Randomly Sampled Data*. New York: McGraw Hill.

KALMAN, R. E., FALB, P. L. & ARBIB, M. A. (1969). *Topics in Mathematical Systems Theory*. New York: McGraw Hill.

KAUFMANN, A. & GUPTA, M. M. (1988). *Fuzzy mathematical models in engineering and management science*. Amsterdam: North-Holland.

KAUFMANN, A. & GUPTA, M. M. (1991). *Introduction to fuzzy arithmetic*. New York: Van Nostrand Reinhold.

KERNIGHAN, B. & RITCHIE, D. (1978). *The C Programming Language*. New Jersey: Prentice-Hall.

KHANNA, T. (1990). *Foundations of neural networks*. Boston: Addison-Wesley Publishing Company.

KING, P. J. & MAMDANI, E. H. (1977). The application of fuzzy control systems to industrial processes. *Automatica*, **13**, 235-242.

LAPEDES, A. & FARBER, R. (1987). Nonlinear signal processing using neural networks: prediction and system modeling. *Los Alamos National Laboratory report LA-UR-87-2662*.

LAPEDES, A. & FARBER, R. (1988). How neural nets work. *Evolution, learning and cognition*. Lee, C. Y. Ed., Singapore: World Scientific.

LUKASIEWICZ, J. (1930). Philosophical remarks on many-valued systems of prepositional logic, *Polish Logic 1920-1939*. Oxford: Clarendon Press.

MAMDANI, , E. H. (1974). Application of fuzzy algorithms for the control of a dynamic plant. *Proceedings of IEE, Control and Science*, **121**, 1585-1588.

MALLOWS, C. L. (1973). Some comments on C_p. *Technometrics*, **15**, 661-676.

MAUS, R. & KEYES, J. (1991). *Handbook of expert systems in manufacturing*. New York: McGraw Hill, Inc.

MCCLELLAND, J. & RUMELHART, D. (1986). *Explorations in parallel distributed processing*. Cambridge: The MIT Press.

MINSKY, M. L. & PAPERT, S. A. (1969). *Perceptrons*. Cambridge: The MIT Press.

MIZUMOTO, M. (1988). Fuzzy controls under various fuzzy reasoning methods. *Information Science*, **45**, 129-151.

NARENDRA, S. K. & PARTHASARATHY, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, Vol. **1**, No. 1, 4-27.

PARKER, D. B. (1985). *Learning-logic*. Report **TR-47**, Cambridge: MIT Center for Research in Computational Economics and Management Science

PATTEE, H. H. (1977). Dynamic and linguistic modes of complex systems. *Intl. J. General Systems*, Vol. **3**, 259-266.

PATTEE, H. H. (1986). Universal principles of measurement and language functions in evolving systems. *Complexity, language, and life: mathematical approaches*. Casti, L. J. & Karlqvist, A. Eds., Berlin: Springer-Verlag.

PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A. & VETTERLING, W. T. (1986). *Numerical Recipes*. Cambridge: Cambridge University Press.

RAGHEB, M. & TSOUKALAS, L. H. (1988). Monitoring performance of devices using coupled probability-possibility methods. *International Journal of Expert Systems*, **1**, 217-229.

ROSENBLATT, F. (1962). *Principles of Neurodynamics*. New York: Spartan.

ROSEN, R. (1978). *Fundamentals of measurement and representation of natural systems*. New York: North Holland.

ROSEN, R. (1985a). Organisms as casual systems which are not mechanisms: an essay into the nature of complexity. *Theoretical Biology and Complexity*. Rosen, R. Ed. Orlando: Academic Press.

ROSEN, R. (1985b). *Anticipatory Systems*. New York: Pergamon Press.

ROSEN, R. (1987). On the scope of syntactics in mathematics and science: the machine metaphor. *Real brains, artificial minds*. Casti, L. J. & Karlqvist, A., Eds., New York: North Holland.

RUMELHART, D. E., HINTON, G. E. & WILLIAMS, R. J. (1986). Learning representations by back-propagating errors. *Nature* **323**, 533-536.

RUSSELL, B. (1923). Vagueness. *Australian Journal of Philosophy*, **1**, 84-92.

SCHETZEN, M. (1970). *The Volterra and Wiener theories of non-linear systems*. New York: John Wiley and Sons.

SAS INSTITUTE INC. (1985). *SAS® user's guide: basics, version 5 edition*. Cary: SAS Institute Inc.

SUNG, C. H. & PRIEBE E. C. (1988). Temporal Pattern Recognition. *Transactions of IEEE International Conference on Neural Networks*, San Diego, CA, **1**, 689-696.

SWOKOWSKI, E. (1983). *Calculus with analytic geometry*. Boston: PWS Publishers.

SYMBOLICS, Inc. (1990). *Plexi user's manual*. Burlington: Symbolics, Inc.

TAKAGI, H. & HAYASHI, I. (1991). NN-driven fuzzy reasoning. *International Journal of Approximate Reasoning*, **5**, 191-212.

TSOUKALAS, L. H., IKONOMOPOULOS, A. & UHRIG, R. E. (1991a). Hybrid expert system - neural network methodology for transient identification. *Proceedings of the American Power Conference*, Chicago, IL, 1206-1211.

TSOUKALAS, L. H. & IKONOMOPOULOS, A. (1991b). Modeling complex systems with artificial neural networks. *Intelligent Engineering Systems Through Artificial Neural Networks*, Dagli, C. H., Kumara, S. R. T. & Shin, Y. C., Eds., New York: ASME Press, 581-587.

TSOUKALAS, L. H. & IKONOMOPOULOS, A. (1992). Uncertainty modeling in anticipatory systems. *Analysis and Management of Uncertainty: Theory and Applications*, Ayyub, B. M., Gupta, M. M. & Kanal, L. N., Eds., New York: Elsevier North Holland.

UPADHYAYA, B. R., KITAMURA, M. & KERLIN, T. W. (1980). Multivariate signal analysis algorithms for process monitoring and parameter estimation in nuclear reactors. *Annals of Nuclear Energy*, Vol. **7**, 1-11.

WASSERMAN, J. (1989). *Neural computing, theory and practice*. New York: Van Nostrand Reinhold.

WERBOS, P. (1974). *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. Ph.D. Thesis, Harvard University.

WERBOS, P. J. (1988). Generalization of backpropagation with application to recurrent gas market model. *Neural Networks*, **1**, 339-356

WHITE, H. (1990). Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings. *Neural Networks*, **3**, 535-549.

WIELAND, A. & LEIGHTON, R., (1987) Geometric analysis of neural network capabilities. *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, 385-392.

WIENER, N. (1958). *Nonlinear problems in random theory*. New York: Wiley.

YANASE, M. M. (1985). Fuzziness and probability. *Annals of the Japan Association for Philosophy of Science*, **6**, 219-226.

ZADEH, L. A. (1953). Optimum nonlinear filters. *Journal of Applied Physics*, **24**, 396-401.

ZADEH, L. A. (1965). Fuzzy sets. *Information and Control*, **8**, 338-353.

ZADEH, L. A. (1972). A fuzzy set-theoretic interpretation of linguistic hedges. *Journal of Cybernetics*, **2**, No. 3, 4-34.

ZADEH, L. A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems Man and Cybernetics*, **SMC-3**, 28-44.

ZADEH, L. A. (1976). A fuzzy-algorithmic approach to the definition of complex or imprecise concepts. *Int. Journal Man-Machine Studies*, **8**, 249-291.

ZADEH, L. A. (1979). A theory of approximate reasoning. *Machine Intelligence*, Hayes, J., Michie, D. and Mikulich, L., J. Eds., **9**, 43-80.

ZADEH, L. A. (1988). Fuzzy logic. *IEEE Computer*, **April 1988**, 83-93.

ZADEH, L. A. (1992). The calculus of fuzzy if/then rules. *AI Expert*, **March 1992**, 23-27.

ZIMMERMAN, H. (1985). *Fuzzy set theory and its applications*. Boston: Kluweer - Ninjhof Publishing.

APPENDIX

```
/*-------------------------------------------------------------------------------------------*/
                                /* DISINDEX.C */
/*-------------------------------------------------------------------------------------------*/

/*      AUTHOR:     Andreas Ikonomopoulos
                    306 Pasqua Engineering Bldg.
                    Department of Nuclear Engineering
                    The University of Tennessee, Knoxville
                    Knoxville, TN 37996-2300

        PURPOSE:    This program calculates the dissemblance index between the
membership functions calculated by the ANNs and the prototype membership functions.
The prototype membership functions are declared as external arrays in the global
declaration part of the program. The input files to the program must be named using as 5th
character one of the numbers 1 to 5 corresponding to a particular network.

        LANGUAGE:       K&R C (SunOS 4.1)                                   */

/*-------------------------------------------------------------------------------------------*/

#include <stdio.h>
#define SIZE 10
FILE *infile;
FILE *outfile;
char buffin[40], buffout[40];
memfunction();
double mf1[SIZE] = {0.025, 0.03, 0.035, 0.04, 0.045, 0.1, 0.13, 0.15, 0.17, 0.19};
double mf2[SIZE] = {0.15, 0.16, 0.17, 0.18, 0.19, 0.30, 0.33, 0.35, 0.37, 0.39};
double mf3[SIZE] = {0.35, 0.36, 0.37, 0.38, 0.39, 0.50, 0.53, 0.55, 0.57, 0.59};
double mf4[SIZE] = {0.50, 0.53, 0.55, 0.57, 0.59, 0.70, 0.71, 0.72, 0.73, 0.74};
double mf5[SIZE] = {0.70, 0.73, 0.75, 0.77, 0.79, 0.85, 0.86, 0.87, 0.88, 0.89};


main()
{
```

```
printf ("input filename: ");
scanf ("%s", buffin);
printf ("output filename: ");
scanf ("%s", buffout);

if ((infile = fopen (buffin, "r")) == NULL)
{
        printf ("Cannot open input file %s\n", buffin);
        exit(1);
}

if ((outfile = fopen (buffout, "w")) == NULL)
{
        printf ("Cannot open output file %s\n", buffout);
        exit(1);
}

switch(buffin[5])
{
        case '1' :
        memfunction(mf1, SIZE);
        break;
        case '2' :
        memfunction(mf2, SIZE);
        break;
        case '3' :
        memfunction(mf3, SIZE);
        break;
        case '4' :
        memfunction(mf4, SIZE);
        break;
        case '5' :
        memfunction(mf5, SIZE);
        break;
        default:
```

```
            printf("This is not an appropriate file specification.\n");
        }
        fclose(infile);
        fclose(outfile);
}

memfunction(mf, n)
double mf[];
int n;
{
        int i, ch;
        double x[SIZE];
        double bogus, n_bogus, disin;

        disin = 0.0;

        while ((ch = getc(infile)) != EOF && ch != 0)
        {
                for (i = 0; i < n; i++)
                {
                        fscanf(infile, "%lf", &x[i]);
                        bogus = x[i] - mf[i];
                        n_bogus = (bogus < 0.0) ? -bogus : bogus;
                        disin += n_bogus;
                        n_bogus = bogus = 0.0;
                }
                fprintf(outfile, "%lf\n", disin);
                disin = 0.0;
        }
}
```

```
/*-------------------------------------------------------------------------------------------*/
                                    /* FLAGS.C */
/*-------------------------------------------------------------------------------------------*/

/*      AUTHOR:     Andreas Ikonomopoulos
                    306 Pasqua Engineering Bldg.
                    Department of Nuclear Engineering
                    The University of Tennessee, Knoxville
                    Knoxville, TN 37996-2300

        PURPOSE:    This program considers the files created from the DISINDEX.C
program as input files and creates files where the responses during the time steps satisfying
the dissemblance index criterion are set equal to the dissemblance index values and those
which do not equal to zero. The results obtained from testing the network on the training
file are used as reference file so that the maximum dissemblance index value will be
calculated and used as basis.

        LANGUAGE:       K&R C (SunOS 4.1)                                   */

/*-------------------------------------------------------------------------------------------*/

#include <stdio.h>
#define SIZE 1240
#define FLAG_ON 1.0
#define FLAG_OFF 0.0
FILE *infile1;
FILE *infile2;
FILE *outfile;
double maximum();
flagset();
char buffin1[40], buffin2[40], buffout[40];
int n_steps;

main()
{
```

```
        double maxim;

        printf ("Filename of the file to be flagged: ");
        scanf ("%s", buffin1);
        printf ("Filename of the network results on itself testing: ");
        scanf ("%s", buffin2);
        printf ("Filename for the file where flags will be set: ");
        scanf ("%s", buffout);

        if ((infile1 = fopen (buffin1, "r")) == NULL)
        {
                printf ("Cannot open input file %s\n", buffin1);
                exit(1);
        }

        if ((infile2 = fopen (buffin2, "r")) == NULL)
        {
                printf ("Cannot open input file %s\n", buffin2);
                exit(1);
        }

        if ((outfile = fopen (buffout, "w")) == NULL)
        {
                printf ("Cannot open output file %s\n", buffout);
                exit(1);
        }

        maxim = maximum();
        flagset(maxim);
        fclose(infile1);
        fclose(infile2);
        fclose(outfile);
}
```

```
/*-------------------------------------------------------------------------------------------*/
/*      This function calculates the maximum value of the dissemblance index for the
membership functions corresponding to the training file. The computed maximum value is
the limit value for the dissemblance indices calculated during novel stimuli. */
/*-------------------------------------------------------------------------------------------*/

double maximum()
{
        int i, ch;
        double ar[SIZE];
        double max;

        max = 0.000001;

        for (i = 0; (ch = getc(infile2)) != EOF && ch != 0; i++)
        {
                fscanf(infile2, "%lf", &ar[i]);
                if (ar[i] >= max)
                        max = ar[i];
                n_steps++;
        }
        printf("%lf %d\n", max, n_steps - 2);
        return max;
}

/*-------------------------------------------------------------------------------------------*/
/*      This function resets the dissemblance index values during novel stimuli based on
the satisfaction or not of the dissemblance index criterion.
/*-------------------------------------------------------------------------------------------*/

flagset(max)
double max;
{
        int i, n_flags_on;
        double ar[SIZE];
```

```
        n_flags_on = 0;

        for (i = 0; i < SIZE; i++)
        {
                fscanf(infile1, "%lf", &ar[i]);
                if (ar[i] <= max)
                {
                        fprintf(outfile, "%lf\n", ar[i]);
                        n_flags_on++;
                }
                else
                        fprintf(outfile, "%lf\n", FLAG_OFF);
        }
        printf("Out of %d time steps of the total HFIR", SIZE);
        printf("start-up file,\n");
        printf("%d time steps should", n_steps - 2);
        printf("be accepted from NN%c, and\n", buffin2[3]);
        printf("%d time steps passed the", n_flags_on);
        printf("dissemblance index criterion.\n", n_flags_on);
}
```

```
/*-------------------------------------------------------------------------------------------*/
                                /* RULE_BASE10.C */
/*-------------------------------------------------------------------------------------------*/

/*      AUTHOR:     Andreas Ikonomopoulos
                    306 Pasqua Engineering Bldg.
                    Department of Nuclear Engineering
                    The University of Tennessee, Knoxville
                    Knoxville, TN 37996-2300

        PURPOSE:    This program considers the dissemblance index values computed
previously from the FLAGS.C program. It uses responses supplied from independent
ANNs during 6 consecutive time steps (4 from the past, 1 present, and 1 future) to infer the
present time step value, according to the dynamic behavior criterion. It dynamically resets
the present value and considers the new value in the next time step evaluation. The process
uses the 64 possible dyadic combinations of 6 numbers based on the satisfaction or not of
the dissemblance index criterion. The values considered from the rule-based system are
either zero or the actual dissemblance index values.

        LANGUAGE:       K&R C (SunOS 4.1)                                   */

/*-------------------------------------------------------------------------------------------*/

#include <stdio.h>
#define SIZE 1240
#define FLAG_ON 1.0
#define FLAG_OFF 0.0
#define NO_FILES 5
FILE *infile[5];
FILE *outfile;
char buffin[5][40], buffout[40];

main()
{
        int i, j, n_flags_on;
```

```
double ar[NO_FILES][SIZE];
char ar_name;

n_flags_on = 0;

for (j = 0; j < NO_FILES; j++)
{
        printf ("Input Filename: ");
        scanf ("%s", buffin[j]);
        if ((infile[j] = fopen (buffin[j], "r")) == NULL)
        {
                printf ("Cannot open input file %s\n", buffin);
                exit(1);
        }
}

printf ("Output Filename: ");
scanf ("%s", buffout);
if ((outfile = fopen (buffout, "w")) == NULL)
{
        printf ("Cannot open output file %s\n", buffout);
        exit(1);
}

for (j = 0; j < NO_FILES; j++)
{
        for (i = 0; i < SIZE; i++)
                fscanf(infile[j], "%lf", &ar[j][i]);
}

for (i = 0; i < (SIZE - 5); i++)
{
/*      fprintf(outfile, "%d\t", i);*/
        for (j = 0; j < NO_FILES; j++)
        {
```

```
if (ar[j][i] == 0.0 && ar[j][i+1] == 0.0
&& ar[j][i+2] == 0.0 && ar[j][i+3] == 0.0 &&
ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (4)     000011 */
}
else if (ar[j][i] == 0.0 && ar[j][i+1] == 0.0
&& ar[j][i+2] == 0.0 && ar[j][i+3] != 0.0
&& ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (8)     000111 */
}
else if (ar[j][i] == 0.0 && ar[j][i+1] == 0.0
&& ar[j][i+2] != 0.0 && ar[j][i+3] != 0.0
&& ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (16)    001111 */
}
else if (ar[j][i] == 0.0 && ar[j][i+1] != 0.0
&& ar[j][i+2] == 0.0 && ar[j][i+3] != 0.0
&& ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (24)    010111 */
}
else if (ar[j][i] == 0.0 && ar[j][i+1] != 0.0
&& ar[j][i+2] != 0.0 && ar[j][i+3] != 0.0
&& ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (32)    011111 */
}
```

```
else if (ar[j][i] != 0.0 && ar[j][i+1] == 0.0
&& ar[j][i+2] == 0.0 && ar[j][i+3] != 0.0
&& ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (40)    100111 */
}
else if (ar[j][i] != 0.0 && ar[j][i+1] == 0.0
&& ar[j][i+2] != 0.0 && ar[j][i+3] != 0.0
&& ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (48)    101111 */
}
else if (ar[j][i] != 0.0 && ar[j][i+1] != 0.0
&& ar[j][i+2] == 0.0 && ar[j][i+3] != 0.0
&& ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (56)    110111 */
}
else if (ar[j][i] != 0.0 && ar[j][i+1] != 0.0
&& ar[j][i+2] != 0.0 && ar[j][i+3] == 0.0
&& ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (60)    111011 */
}
else if (ar[j][i] != 0.0 && ar[j][i+1] != 0.0
&& ar[j][i+2] != 0.0 && ar[j][i+3] != 0.0
&& ar[j][i+4] != 0.0 && ar[j][i+5] != 0.0)
{
        fprintf(outfile, "%lf ", ar[j][i+4]);
        n_flags_on++;           /* Rule (64)    111111 */
}
```

```
                else
                {
                        fprintf(outfile, "%lf ", FLAG_OFF);
                        ar[j][i+4] = FLAG_OFF;
                }
        }
        fprintf(outfile, "\n");
}

printf("Out of %d time steps of the total HFIR start-up file,\n", 5*SIZE);
printf("%d time steps passed the dissemblance index and\n", n_flags_on);
printf("dynamic behavior criteria.\n");

for (j = 0; j < NO_FILES; j++)
        fclose(infile[j]);
fclose(outfile);
}
```

```
/*-------------------------------------------------------------------------------------------*/
                                /* RULE_BASE20.C */
/*-------------------------------------------------------------------------------------------*/

/*      AUTHOR:     Andreas Ikonomopoulos
                    306 Pasqua Engineering Bldg.
                    Department of Nuclear Engineering
                    The University of Tennessee, Knoxville
                    Knoxville, TN 37996-2300

        PURPOSE:  This program considers the responses provided by the
RULE_BASE10.C rule-based system. It initially fuzzifies the set of satisfactory responses
of the dissemblance index criterion for every neural network. It multiplies the evaluated
from RULE_BASE10.C membership functions concerning the present time step with the
degree of presumption associated with the satisfaction of the dissemblance index criterion.
Consequently it combines the resultant membership functions from different ANNs
applying fuzzy addition on the modified membership functions. The prototype
membership functions along with the maximum dissemblance index values are supplied to
the code as manifest constants.

        LANGUAGE:       K&R C (SunOS 4.1)                                   */

/*-------------------------------------------------------------------------------------------*/

#include <stdio.h>
#define MEM_FUN 8
#define NO_NET 5
#define SIZE 1240
#define base_di1 0.019077
#define base_di2 0.021207
#define base_di3 0.022715
#define base_di4 0.011487
#define base_di5 0.028231
FILE *infile;
FILE *outfile;
```

```
char buffin[40], buffout[40];
calculate_new1();
calculate_new2();
calculate_new3();
double mf1[MEM_FUN] = {33.34, -0.6668, 0.02, 0.05, -10.00, 2.00, 0.10, 0.20};
double mf2[MEM_FUN] = {20.00, -3.00, 0.15, 0.20, -10.00, 4.00, 0.30, 0.40};
double mf3[MEM_FUN] = {20.00, -7.00, 0.35, 0.40, -10.00, 6.00, 0.50, 0.60};
double mf4[MEM_FUN] = {10.00, -5.00, 0.50, 0.60, -20.00, 15.00, 0.70, 0.75};
double mf5[MEM_FUN] = {10.00, -7.00, 0.70, 0.80, -20.00, 18.00, 0.85, 0.90};
double aL, aR, bL, bR;
double x1L, x2L, x1R, x2R;

main()
{
        int i, j;
        double a[NO_NET][SIZE];
        double d_i1, d_i2, d_i3, d_i4, d_i5;

        printf ("Input Filename: ");
        scanf ("%s", buffin);
        printf ("Output Filename: ");
        scanf ("%s", buffout);

        if ((infile = fopen (buffin, "r")) == NULL)
        {
                printf ("Cannot open input file %s\n", buffin);
                exit(1);
        }

        if ((outfile = fopen (buffout, "w")) == NULL)
        {
                printf ("Cannot open output file %s\n", buffout);
                exit(1);
        }
```

```
for (i = 0; i < (SIZE - 5); i++)
{
        for (j = 0; j < NO_NET; j++)
                fscanf(infile, "%lf", &a[j][i]);
}

for (i = 0; i < (SIZE - 5); i++)
{
        fprintf(outfile, "%d\t", i);

/*-------------------------------------------------------------------------------------------------*/
                        /* Only one network has responded */
/*-------------------------------------------------------------------------------------------------*/

                if (a[0][i] != 0.0 && a[1][i] == 0.0 && a[2][i] == 0.0
                && a[3][i] == 0.0 && a[4][i] == 0.0 )
                {
                        d_i1 = a[0][i];
                        calculate_new1(mf1, d_i1, base_di1);
                }
                else if (a[0][i] == 0.0 && a[1][i] != 0.0 && a[2][i] == 0.0
                && a[3][i] == 0.0 && a[4][i] == 0.0 )
                {
                        d_i2 = a[1][i];
                        calculate_new1(mf2, d_i2, base_di2);
                }
                else if (a[0][i] == 0.0 && a[1][i] == 0.0 && a[2][i] != 0.0
                && a[3][i] == 0.0 && a[4][i] == 0.0 )
                {
                        d_i3 = a[2][i];
                        calculate_new1(mf3, d_i3, base_di3);
                }
                else if (a[0][i] == 0.0 && a[1][i] == 0.0 && a[2][i] == 0.0
                && a[3][i] != 0.0 && a[4][i] == 0.0 )
                {
```

```
            d_i4 = a[3][i];
            calculate_new1(mf4, d_i4, base_di4);
        }
        else if (a[0][i] == 0.0 && a[1][i] == 0.0 && a[2][i] == 0.0
        && a[3][i] == 0.0 && a[4][i] != 0.0 )
        {
            d_i5 = a[4][i];
            calculate_new1(mf5, d_i5, base_di5);
        }

/*-------------------------------------------------------------------------------------------------*/
                    /* Two networks have responded */
/*-------------------------------------------------------------------------------------------------*/

        else if (a[0][i] != 0.0 && a[1][i] != 0.0 && a[2][i] == 0.0
        && a[3][i] == 0.0 && a[4][i] == 0.0 )
        {
            d_i1 = a[0][i];
            d_i2 = a[1][i];
            calculate_new2(mf1, d_i1, base_di1,
            mf2, d_i2, base_di2);
        }
        else if (a[0][i] == 0.0 && a[1][i] != 0.0 && a[2][i] != 0.0
        && a[3][i] == 0.0 && a[4][i] == 0.0 )
        {
            d_i2 = a[1][i];
            d_i3 = a[2][i];
            calculate_new2(mf2, d_i2, base_di2,
            mf3, d_i3, base_di3);
        }
        else if (a[0][i] == 0.0 && a[1][i] == 0.0 && a[2][i] != 0.0
        && a[3][i] != 0.0 && a[4][i] == 0.0 )
        {
            d_i3 = a[2][i];
            d_i4 = a[3][i];
```

```
            calculate_new2(mf3, d_i3, base_di3,
            mf4, d_i4, base_di4);
        }
        else if (a[0][i] == 0.0 && a[1][i] == 0.0 && a[2][i] == 0.0
        && a[3][i] != 0.0 && a[4][i] != 0.0 )
        {
            d_i4 = a[3][i];
            d_i5 = a[4][i];
            calculate_new2(mf4, d_i4, base_di4,
            mf5, d_i5, base_di5);
        }

/*-------------------------------------------------------------------------------------------------*/
                    /* Three networks have responded */
/*-------------------------------------------------------------------------------------------------*/

        else if (a[0][i] != 0.0 && a[1][i] != 0.0 && a[2][i] != 0.0
        && a[3][i] == 0.0 && a[4][i] == 0.0 )
        {
            d_i1 = a[0][i];
            d_i2 = a[1][i];
            d_i3 = a[2][i];
            calculate_new3(mf1, d_i1, base_di1,
            mf2, d_i2, base_di2, mf3, d_i3, base_di3);
        }
        else if (a[0][i] == 0.0 && a[1][i] != 0.0 && a[2][i] != 0.0
        && a[3][i] != 0.0 && a[4][i] == 0.0 )
        {
            d_i2 = a[1][i];
            d_i3 = a[2][i];
            d_i4 = a[3][i];
            calculate_new3(mf2, d_i2, base_di2,
            mf3, d_i3, base_di3, mf4, d_i4, base_di4);
        }
        else if (a[0][i] == 0.0 && a[1][i] == 0.0 && a[2][i] != 0.0
```

```
&& a[3][i] != 0.0 && a[4][i] != 0.0 )
{
        d_i3 = a[2][i];
        d_i4 = a[3][i];
        d_i5 = a[4][i];
        calculate_new3(mf3, d_i3, base_di3,
        mf4, d_i4, base_di4, mf5, d_i5, base_di5);
}

/*-------------------------------------------------------------------------------------------------*/
        /* More than three networks have responded */
/*-------------------------------------------------------------------------------------------------*/

else if (a[0][i] != 0.0 && a[1][i] != 0.0 && a[2][i] != 0.0
&& a[3][i] != 0.0 && a[4][i] == 0.0 )
{
        a[0][i] = 0.0;
        d_i2 = a[1][i];
        d_i3 = a[2][i];
        d_i4 = a[3][i];
        calculate_new3(mf2, d_i2, base_di2,
        mf3, d_i3, base_di3, mf4, d_i4, base_di4);
}
else if (a[0][i] == 0.0 && a[1][i] != 0.0 && a[2][i] != 0.0
&& a[3][i] != 0.0 && a[4][i] != 0.0 )
{
        a[1][i] = 0.0;
        d_i3 = a[2][i];
        d_i4 = a[3][i];
        d_i5 = a[4][i];
        calculate_new3(mf3, d_i3, base_di3,
        mf4, d_i4, base_di4, mf5, d_i5, base_di5);
}
else if (a[0][i] != 0.0 && a[1][i] != 0.0 && a[2][i] != 0.0
&& a[3][i] != 0.0 && a[4][i] != 0.0 )
```

```
            {
                    a[0][i] = 0.0;
                    a[4][i] = 0.0;
                    d_i2 = a[1][i];
                    d_i3 = a[2][i];
                    d_i4 = a[3][i];
                    calculate_new3(mf2, d_i2, base_di2,
                    mf3, d_i3, base_di3, mf4, d_i4, base_di4);
            }
            else
            {
                    fprintf(outfile, "%lf %lf %lf %lf %lf %lf %lf %lf\n",
                    aL, bL, x1L, x2L, aR, bR, x1R, x2R);
            }
    }

/*-------------------------------------------------------------------------------------------------*/

    fclose(infile);
    fclose(outfile);
}

/*-------------------------------------------------------------------------------------------------*/
/*      This function computes the resultant membership function when only one response
has satisfied the criteria set. */
/*-------------------------------------------------------------------------------------------------*/

calculate_new1(memf1, di1, base_disin1)
double memf1[];
double di1, base_disin1;
{
    int i;
    double bogus_ratio[MEM_FUN];
    double bogus_sum;
    double weight[MEM_FUN];
```

```
        double bogus_memf1[MEM_FUN];

        bogus_sum = 0.0;
        for (i = 0; i < MEM_FUN; i++)
        {
                bogus_memf1[i] = memf1[i];
                weight[0] = 0.0;
                bogus_ratio[0] = 0.0;
        }

        bogus_ratio[0] = 1.0 - (di1 / base_disin1);
        if (bogus_ratio[0] == 0.0)
                bogus_ratio[0] = 0.000001;
        bogus_sum = bogus_ratio[0];
        weight[0] = bogus_ratio[0] / bogus_sum;

        bogus_memf1[0] = bogus_memf1[0] / weight[0];
        bogus_memf1[2] = bogus_memf1[2] * weight[0];
        bogus_memf1[3] = bogus_memf1[3] * weight[0];
        bogus_memf1[4] = bogus_memf1[4] / weight[0];
        bogus_memf1[6] = bogus_memf1[6] * weight[0];
        bogus_memf1[7] = bogus_memf1[7] * weight[0];

        aL = bogus_memf1[0];
        bL = bogus_memf1[1];
        aR = bogus_memf1[4];
        bR = bogus_memf1[5];
        x1L = bogus_memf1[2];
        x2L = bogus_memf1[3];
        x1R = bogus_memf1[6];
        x2R = bogus_memf1[7];

        fprintf(outfile, "%lf %lf %lf %lf %lf %lf %lf %lf\n",
        aL, bL, x1L, x2L, aR, bR, x1R, x2R);
}
```

```
/*-------------------------------------------------------------------------------------------*/
/*      This function computes the resultant membership function when two responses
have satisfied the criteria set. */
/*-------------------------------------------------------------------------------------------*/

calculate_new2(memf1, di1, base_disin1, memf2, di2, base_disin2)
double memf1[], memf2[];
double di1, di2;
double base_disin1, base_disin2;
{
        int i;
        double bogus_ratio[MEM_FUN];
        double bogus_sum;
        double weight[MEM_FUN];
        double bogus_memf1[MEM_FUN];
        double bogus_memf2[MEM_FUN];

        bogus_sum = 0.0;
        for (i = 0; i < MEM_FUN; i++)
        {
                bogus_memf1[i] = memf1[i];
                bogus_memf2[i] = memf2[i];
                weight[i] = 0.0;
                bogus_ratio[i] = 0.0;
        }

        bogus_ratio[0] = 1.0 - (di1 / base_disin1);
        bogus_ratio[1] = 1.0 - (di2 / base_disin2);
        for (i = 0; i < 2; i++)
        {
                if (bogus_ratio[i] == 0.0)
                        bogus_ratio[i] = 0.000001;
                bogus_sum += bogus_ratio[i];
        }
        weight[0] = bogus_ratio[0] / bogus_sum;
```

```
        weight[1] = bogus_ratio[1] / bogus_sum;

        bogus_memf1[0] = bogus_memf1[0] / weight[0];
        bogus_memf1[2] = bogus_memf1[2] * weight[0];
        bogus_memf1[3] = bogus_memf1[3] * weight[0];
        bogus_memf1[4] = bogus_memf1[4] / weight[0];
        bogus_memf1[6] = bogus_memf1[6] * weight[0];
        bogus_memf1[7] = bogus_memf1[7] * weight[0];

        bogus_memf2[0] = bogus_memf2[0] / weight[1];
        bogus_memf2[2] = bogus_memf2[2] * weight[1];
        bogus_memf2[3] = bogus_memf2[3] * weight[1];
        bogus_memf2[4] = bogus_memf2[4] / weight[1];
        bogus_memf2[6] = bogus_memf2[6] * weight[1];
        bogus_memf2[7] = bogus_memf2[7] * weight[1];

        aL = (bogus_memf1[0] * bogus_memf2[0]) /
        (bogus_memf1[0] + bogus_memf2[0]);
        bL = (bogus_memf1[1] * bogus_memf2[0] +
        bogus_memf2[1] * bogus_memf1[0]) / (bogus_memf1[0] + bogus_memf2[0]);
        aR = (bogus_memf1[4] * bogus_memf2[4]) /
        (bogus_memf1[4] + bogus_memf2[4]);
        bR = (bogus_memf1[5] * bogus_memf2[4] +
        bogus_memf2[5] * bogus_memf1[4]) / (bogus_memf1[4] + bogus_memf2[4]);
        x1L = bogus_memf1[2] + bogus_memf2[2];
        x2L = bogus_memf1[3] + bogus_memf2[3];
        x1R = bogus_memf1[6] + bogus_memf2[6];
        x2R = bogus_memf1[7] + bogus_memf2[7];

        fprintf(outfile, "%lf %lf %lf %lf %lf %lf %lf %lf\n",
        aL, bL, x1L, x2L, aR, bR, x1R, x2R);
}
```

```
/*-------------------------------------------------------------------------------------------*/
/*      This function computes the resultant membership function when three responses
have satisfied the criteria set. */
/*-------------------------------------------------------------------------------------------*/

calculate_new3(memf1, di1, base_disin1, memf2, di2, base_disin2,
memf3, di3, base_disin3)
double memf1[], memf2[], memf3[];
double di1, di2, di3;
double base_disin1, base_disin2, base_disin3;
{
        int i;
        double bogus_ratio[MEM_FUN];
        double bogus_sum;
        double weight[MEM_FUN];
        double bogus_memf1[MEM_FUN];
        double bogus_memf2[MEM_FUN];
        double bogus_memf3[MEM_FUN];
        double two[MEM_FUN];

        bogus_sum = 0.0;
        for (i = 0; i < MEM_FUN; i++)
        {
                bogus_memf1[i] = memf1[i];
                bogus_memf2[i] = memf2[i];
                bogus_memf3[i] = memf3[i];
                weight[i] = 0.0;
                bogus_ratio[i] = 0.0;
        }

        bogus_ratio[0] = 1.0 - (di1 / base_disin1);
        bogus_ratio[1] = 1.0 - (di2 / base_disin2);
        bogus_ratio[2] = 1.0 - (di3 / base_disin3);
        for (i = 0; i < 3; i++)
        {
```

```
        if (bogus_ratio[i] == 0.0)
                bogus_ratio[i] = 0.000001;
        bogus_sum += bogus_ratio[i];
}
weight[0] = bogus_ratio[0] / bogus_sum;
weight[1] = bogus_ratio[1] / bogus_sum;
weight[2] = bogus_ratio[2] / bogus_sum;

bogus_memf1[0] = bogus_memf1[0] / weight[0];
bogus_memf1[2] = bogus_memf1[2] * weight[0];
bogus_memf1[3] = bogus_memf1[3] * weight[0];
bogus_memf1[4] = bogus_memf1[4] / weight[0];
bogus_memf1[6] = bogus_memf1[6] * weight[0];
bogus_memf1[7] = bogus_memf1[7] * weight[0];

bogus_memf2[0] = bogus_memf2[0] / weight[1];
bogus_memf2[2] = bogus_memf2[2] * weight[1];
bogus_memf2[3] = bogus_memf2[3] * weight[1];
bogus_memf2[4] = bogus_memf2[4] / weight[1];
bogus_memf2[6] = bogus_memf2[6] * weight[1];
bogus_memf2[7] = bogus_memf2[7] * weight[1];

bogus_memf3[0] = bogus_memf3[0] / weight[2];
bogus_memf3[2] = bogus_memf3[2] * weight[2];
bogus_memf3[3] = bogus_memf3[3] * weight[2];
bogus_memf3[4] = bogus_memf3[4] / weight[2];
bogus_memf3[6] = bogus_memf3[6] * weight[2];
bogus_memf3[7] = bogus_memf3[7] * weight[2];

two[0] = (bogus_memf1[0] * bogus_memf2[0]) /
(bogus_memf1[0] + bogus_memf2[0]);
two[1] = (bogus_memf1[1] * bogus_memf2[0] +
bogus_memf2[1] * bogus_memf1[0]) / (bogus_memf1[0] + bogus_memf2[0]);
two[4] = (bogus_memf1[4] * bogus_memf2[4]) /
(bogus_memf1[4] + bogus_memf2[4]);
```

```
        two[5] = (bogus_memf1[5] * bogus_memf2[4] +
        bogus_memf2[5] * bogus_memf1[4]) / (bogus_memf1[4] + bogus_memf2[4]);
        aL = (two[0] * bogus_memf3[0]) / (two[0] + bogus_memf3[0]);
        bL = (two[1] * bogus_memf3[0] + bogus_memf3[1] * two[0]) /
        (two[0] + bogus_memf3[0]);
        aR = (two[4] * bogus_memf3[4]) / (two[4] + bogus_memf3[4]);
        bR = (two[5] * bogus_memf3[4] + bogus_memf3[5] * two[4]) /
        (two[4] + bogus_memf3[4]);
        x1L = bogus_memf1[2] + bogus_memf2[2] + bogus_memf3[2];
        x2L = bogus_memf1[3] + bogus_memf2[3] + bogus_memf3[3];
        x1R = bogus_memf1[6] + bogus_memf2[6] + bogus_memf3[6];
        x2R = bogus_memf1[7] + bogus_memf2[7] + bogus_memf3[7];

        fprintf(outfile, "%lf %lf %lf %lf %lf %lf %lf %lf\n",
        aL, bL, x1L, x2L, aR, bR, x1R, x2R);
}
```

```
/*-------------------------------------------------------------------------------------------*/
                                    /* NOISE.C */
/*-------------------------------------------------------------------------------------------*/

/*      AUTHOR:     Andreas Ikonomopoulos
                    306 Pasqua Engineering Bldg.
                    Department of Nuclear Engineering
                    The University of Tennessee, Knoxville
                    Knoxville, TN 37996-2300

        PURPOSE:    This program introduces 10% noise to all input signals supplied to
the virtual measuring device. The random number generator is the one presented at the "C:
Step-by-Step" book by M. Waite and S. Prata, pp. 442-444. The user has the choice of
supplying different or the same seed at the program initialization. In this way the same or
different random series of numbers can be generated. The total number of random
numbers generated is 1240. The numbers calculated are normalized in the area 0.1 to 0.9.

        LANGUAGE:       K&R C (SunOS 4.1)                                   */

/*-------------------------------------------------------------------------------------------*/

#include <stdio.h>
#define SCALE 32768.0
#define SIZE 1240
#define NO_NET 5
extern int srand1();
extern double rand1();
FILE *infile;
FILE *outfile;
char buffin[40], buffout[40];

main()
{
        int i, j;
        int seed;
```

```
double ar[NO_NET][SIZE];

printf ("Input Filename: ");
scanf ("%s", buffin);
printf ("Output Filename: ");
scanf ("%s", buffout);
printf("Please enter your choice of seed: ");
scanf("%d", &seed);

if ((infile = fopen (buffin, "r")) == NULL)
{
        printf ("Cannot open input file %s\n", buffin);
        exit(1);
}

if ((outfile = fopen (buffout, "w")) == NULL)
{
        printf ("Cannot open output file %s\n", buffout);
        exit(1);
}

for (i = 0; i < SIZE; i++)
{
        for (j = 0; j < NO_NET; j++)
                fscanf(infile, "%lf", &ar[j][i]);
}

srand1(seed);
for (i = 0; i < SIZE; i++)
{
        for (j = 0; j < NO_NET; j++)
        {
                ar[j][i] += 0.1 * rand1()/SCALE;
```

```
/*-------------------------------------------------------------------------------------------*/
/*      Normalization of noisy signals. */
/*-------------------------------------------------------------------------------------------*/

                        if (ar[j][i] < 0.1)
                        {
                                if (ar[j][i] > 0.0)
                                        ar[j][i] = ar[j][i] + 0.1;
                                else
                                        ar[j][i] = - ar[j][i];
                        }
                        else if (ar[j][i] > 0.9)
                        {
                                if (ar[j][i] > 1.0)
                                        ar[j][i] = 0.9;
                                else
                                        ar[j][i] = ar[j][i] - 0.1;
                        }
                        fprintf(outfile, "%lf ", ar[j][i]);
                }
                fprintf(outfile, "\n");
        }
        fclose(infile);
        fclose(outfile);
}

/*-------------------------------------------------------------------------------------------*/
/*      Random number generator functions. */
/*-------------------------------------------------------------------------------------------*/

static int randx =1;
double rand1()
{
        randx = (randx * 25173 + 13849) % 65536;
        return((short) randx);
```

```
}

int srand1(x)
unsigned x;
{
        randx = x;
}
```

```
/*-------------------------------------------------------------------------------------------*/
                                /* CENTROID.C */
/*-------------------------------------------------------------------------------------------*/

/*      AUTHOR:     Andreas Ikonomopoulos
                    306 Pasqua Engineering Bldg.
                    Department of Nuclear Engineering
                    The University of Tennessee, Knoxville
                    Knoxville, TN 37996-2300

        PURPOSE:    This program calculates the centroids of the resultant membership
functions as they have been computed from RULE_BASE20.C. The calculation of the
coordinates of the centroids is based on the calculation of the x and y moments as well as
the "mass" of the area surrounded by a trapezoid membership function. The algorithm can
be used for a variety of geometrical schemes although it has been developed for trapezoidal
membership functions.

        LANGUAGE:       K&R C (SunOS 4.1)                                   */

/*-------------------------------------------------------------------------------------------*/

#include <stdio.h>
#include <math.h>
#define SIZE 1240
FILE *infile;
FILE *outfile;
char buffin[40], buffout[40];

main()
{
        int i, counter;
        double aL[SIZE], bL[SIZE];
        double aR[SIZE], bR[SIZE];
        double x1L[SIZE], x2L[SIZE];
        double x1R[SIZE], x2R[SIZE];
```

```
double mass[SIZE];
double center_of_mass_x[SIZE];
double center_of_mass_y[SIZE];
double moment_Mx[SIZE], moment_My[SIZE];

printf ("Input Filename: ");
scanf ("%s", buffin);
printf ("Output Filename: ");
scanf ("%s", buffout);

if ((infile = fopen (buffin, "r")) == NULL)
{
        printf ("Cannot open input file %s\n", buffin);
        exit(1);
}

if ((outfile = fopen (buffout, "w")) == NULL)
{
        printf ("Cannot open output file %s\n", buffout);
        exit(1);
}

for (i = 0; i < (SIZE - 5); i++)
        fscanf(infile, "%d%lf%lf%lf%lf%lf%lf%lf",
        &counter, &aL[i], &bL[i], &x1L[i], &x2L[i],
        &aR[i], &bR[i], &x1R[i], &x2R[i]);

for (i = 0; i < (SIZE - 5); i++)
{
        fprintf(outfile, "%d\t", i);

        mass[i] = aL[i] * (pow(x2L[i],2.0) - pow(x1L[i],2.0)) / 2.0 +
                bL[i] * (x2L[i] - x1L[i]) + (x1R[i] - x2L[i]) + aR[i] *
                (pow(x2R[i],2.0) - pow(x1R[i],2.0)) / 2.0 + bR[i] * (x2R[i] -
                x1R[i]);
```

```
            moment_My[i] = aL[i] * (pow(x2L[i],3.0) - pow(x1L[i],3.0)) / 3.0 +
                    bL[i] * (pow(x2L[i],2.0) - pow(x1L[i],2.0)) / 2.0 +
                    (pow(x1R[i],2.0) - pow(x2L[i],2.0)) / 2.0 + aR[i] *
                    (pow(x2R[i],3.0) - pow(x1R[i],3.0)) / 3.0 + bR[i] *
                    (pow(x2R[i],2.0) - pow(x1R[i],2.0)) / 2.0;

            moment_Mx[i] = pow(aL[i],2.0) * (pow(x2L[i],3.0) - pow(x1L[i],3.0)) /
                    6.0 + aL[i] * bL[i] * (pow(x2L[i],2.0) - pow(x1L[i],2.0)) / 2.0 +
                    pow(bL[i],2.0) * (x2L[i] - x1L[i]) / 2.0 +(x1R[i] - x2L[i]) / 2.0 +
                    pow(aR[i],2.0) * (pow(x2R[i],3.0) - pow(x1R[i],3.0)) / 6.0 +
                    aR[i] * bR[i] * (pow(x2R[i],2.0) - pow(x1R[i],2.0)) / 2.0 +
                    pow(bR[i],2.0) * (x2R[i] - x1R[i]) / 2.0;

            if (mass[i] != 0.0)
            {
                    center_of_mass_x[i] = moment_My[i] / mass[i];

                    center_of_mass_y[i] = moment_Mx[i] / mass[i];
            }
            else
            {
                    center_of_mass_x[i] = 0.0;

                    center_of_mass_y[i] = 0.0;
            }

            fprintf(outfile, "%lf\t%lf\n", center_of_mass_x[i],
            center_of_mass_y[i]);
    }
    fclose(infile);
    fclose(outfile);
}
```

```
/*-------------------------------------------------------------------------------------------*/
                                /* NOISE_SIG.C */
/*-------------------------------------------------------------------------------------------*/

/*      AUTHOR:     Andreas Ikonomopoulos
                    306 Pasqua Engineering Bldg.
                    Department of Nuclear Engineering
                    The University of Tennessee, Knoxville
                    Knoxville, TN 37996-2300

        PURPOSE:    This program substitutes one of the input signals with a random time
series. The random number generator is the one presented at the "C: Step-by-Step" book
by M. Waite and S. Prata, pp. 442-444. The user has the choice of supplying different or
the same seed at the program initialization. In this way the same or different random series
of numbers can be generated. The total number of random numbers generated is 1240.
The numbers calculated are normalized in the area 0.1 to 0.9. In case more than one
original signals have to substituted by random time series, the user has to do it on a
repetitive mode.

        LANGUAGE:       K&R C (SunOS 4.1)                                   */

/*-------------------------------------------------------------------------------------------*/

#include <stdio.h>
#define SCALE 32768.0
#define SIZE 1240
#define NO_NET 5
extern int srand1();
extern double rand1();
FILE *infile;
FILE *outfile;
char buffin[40], buffout[40];

main()
{
```

```
int i, j;
int seed, signal;
double ar[NO_NET][SIZE];

printf ("Input Filename: ");
scanf ("%s", buffin);
printf ("Output Filename: ");
scanf ("%s", buffout);
printf("Please enter your choice of seed: ");
scanf("%d", &seed);
printf("Please enter your choice of signal to be randomized: ");
scanf("%d", &signal);

if ((infile = fopen (buffin, "r")) == NULL)
{
        printf ("Cannot open input file %s\n", buffin);
        exit(1);
}

if ((outfile = fopen (buffout, "w")) == NULL)
{
        printf ("Cannot open output file %s\n", buffout);
        exit(1);
}

for (i = 0; i < SIZE; i++)
{
        for (j = 0; j < NO_NET; j++)
                fscanf(infile, "%lf", &ar[j][i]);
}

srand1(seed);
for (i = 0; i < SIZE; i++)
{
        ar[signal][i] = rand1()/SCALE;
```

```
/*-------------------------------------------------------------------------------------------*/
/*      Normalization of noisy signals. */
/*-------------------------------------------------------------------------------------------*/

                if (ar[signal][i] < 0.0)
                {
                        ar[signal][i] = -ar[signal][i];
                        if (ar[signal][i] < 0.1)
                                ar[signal][i] = ar[signal][i] + 0.1;
                        else if (ar[signal][i] > 0.9)
                                ar[signal][i] = ar[signal][i] - 0.1;
                }
                else if (ar[signal][i] < 0.1)
                        ar[signal][i] = ar[signal][i] + 0.1;
                else if (ar[signal][i] > 0.9)
                        ar[signal][i] = ar[signal][i] - 0.1;
                for (j = 0; j < NO_NET; j++)
                        fprintf(outfile, "%lf ", ar[j][i]);
                fprintf(outfile, "\n");
        }
        fclose(infile);
        fclose(outfile);
}

/*-------------------------------------------------------------------------------------------*/
/*      Random number generator functions. */
/*-------------------------------------------------------------------------------------------*/

static int randx =1;
double rand1()
{
        randx = (randx * 25173 + 13849) % 65536;
        return((short) randx);
}
```

```
int srand1(x)
unsigned x;
{
        randx = x;
}
```

VITA

Andreas Ikonomopoulos was born in Athens, Greece in 1965. He completed his High School education in 1980 and entered the Lyceum from which he graduated in 1983. The same year he successfully passed the National Examination and joined the Physics Department at the University of Athens. As a junior student at the University he was attracted by the field of theoretical high energy physics. After a year of studying particle trajectories in high resolution films he decided that lower energy particles were of more interest to him. In 1986 he started working at the Nuclear Research Center in Athens, where a year later he completed his diploma thesis on the application of noise techniques in nuclear reactor monitoring. In September 1987 he graduated with honors from the Physics Department with a specialization in Nuclear Technology and became partner in a software development firm while he was pursuing acceptance for graduate studies abroad. In August 1988 he was enrolled at the University of Illinois at Urbana-Champaign having been awarded a University Scholarship. Research on the application of best-estimate (thermohydraulic) models in nuclear reactor transient simulation earned him an M.S. degree from the Nuclear Engineering Department in May 1990. Subsequently, he joined the Department of Nuclear Engineering at the University of Tennessee, Knoxville in June 1990. He has held numerous research assistantships during his graduate studies and he is the recipient of four scholastic awards for academic achievements from the Greek government. Mr. Ikonomopoulos has 17 publications in the fields of best estimate models, artificial neural networks, expert systems, fuzzy logic and their applications to nuclear reactor safety, time series analysis, and system monitoring. He has been elected member of the ΦΚΦ and ΑΝΣ honor societies.