1994

Spline network modeling and fault classification of a heating ventilation and air-conditioning system

Mathew Scaria Chackalackal
Iowa State University

Order Number 9518364

Spline network modeling and fault classification of a heating ventilation and air-conditioning system

Chackalackal, Matthew Scaria, Ph.D.

Iowa State University, 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Spline network modeling and fault classification of a heating ventilation and air-conditioning system

by

Mathew Scaria Chackalackal

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Electrical Engineering and Computer Engineering
Major: Electrical Engineering (Communications and Signal Processing)

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1994

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Humankind is becoming increasingly aware of environmental issues. At the same time, ironic as it is, human beings are demanding higher comfort levels at the work-place, home and in modes of transit. The car has to cool down or warm up to one's comfort in the shortest time possible. Hardly a day passes without hearing about the ergonomic design of some car. Ergonomics is the science of maximizing comfort in an affordable manner. Yet ergonomics fails to economize the natural resources on earth. Industries are beginning to take interest in environmental concerns. This includes systems that are fuel efficient. The trend in the future will be to maximize both fuel efficiency and comfort. There definitely must be trade offs. A built-in fault diagnostics system is essential to maintain the fuel effeciency and the comfort levels that the system was designed for. In this dissertation, a real life system is modeled and a fault classification approach is devised. The techniques are kept as simple as possible, in an attempt to make it practical. In many cases, it is the simplest of techniques that produce the best results.

Quality and reliability seem to be the key words in industry. Neither can be guaranteed for a system that has no fully automated fault diagnostics. This research is aimed at the design of fault diagnostics for any general system that does a lot of steady state operation and, is a step in the direction of a new generation of pro-

cess control systems. Modifications, if necessary, can be made to adapt to highly dynamic systems. A Heating, Ventilation and Air-conditioning (HVAC) system is a good candidate for this research. The state-of-the-art HVAC systems do not have any built-in fault diagnostics, though there is a lot of on-going research in this area. Further, there was access to the Charles L. Schwab HVAC test loop in the Mechanical Engineering Department at Iowa State University. Most of the work done in improvement of HVAC systems has focused on system optimization and not as much on Fault Detection and Identificaiton (FDI) (Usoro and Schick 1985). Optimization is process specific. An optimized HVAC system could be optimized with respect to energy efficiency and human comfort. Faults in a system could lead to extreme non-optimality! Thus ability to detect faults, localize the faults, and rectify the faults are essential components of a truly optimal system.

Degradation in the components of the system could result in increased energy consumption and discomfort to the clients. The degradation of components could lead to their breakdown or the ultimate breakdown of the entire system. The fault diagnosis system also informs the controller about abnormal changes in the system, so that the system can adapt to its current conditions. This would ensure total quality control for the system.

A working model of the system is essential to detect system abnormalities. A more than tolerable deviation of the system output variables from those predicted by the model is a fault indication. A popular approach to fault detection is using a mathematical process model (Wilsky 1976 and Isermann 1984). The process variables are related to process coefficients in the mathematical model, making it possible to locate the faults in the system (Patton and Clark 1989). Many of the process control

systems one encounters are highly complex and nonlinear, and is often difficult to come up with a reasonably accurate model.

Neural networks are being increasingly used in building process models for complex systems. The process of building working models is also termed system identification. The ability of the neural networks to learn the nature of the system, based on a large training set of input and output variables, makes it possible to develop neural network models of complex systems with little knowledge of the actual system. Further it has been proved that a two layer neural network (one hidden layer) can approximate any arbitrary non-linear function (Cybenko 1989). But depending on the complexity of the system, the hidden layer of the neural network could be infinitely large. To circumvent this, one can add more layers with fewer nodes. Hajnal (1987) proved that problems that required an exponential number of nodes in a two-layer network can be implemented with a polynomial number of nodes in a three layer network. It is yet to be proved whether the same rule applies to a three layer network (Hush and Horne 1993). There are no hard and fast rules to determine the exact number of layers or the number of nodes per layer. Thus converging to a satisfactory neural network model of the system may involve trial and error. This could be time consuming, and is one of the motivations for the proposed network in Chapter 4.

There are many fault detection schemes described in the literature and a brief review is given later in this chapter. A fault diagnostic approach based on deviation from the neural network model, common sense rules, hardware redundancy, and fuzzy logic is proposed and implemented in this dissertation.

The research involves both experimental and theoretical work. The ultimate

objective is to develop a reliable, yet feasible fault diagnostics in a real system. In the next section, the HVAC system is described, followed by a review of existing neural network and fault diagnostic techniques.

Charles L. Schwab HVAC Test Loop

The HVAC system used for the research was the Charles L. Schwab test loop in the Mechanical Engineering Department at Iowa State University. A simplified schematic of the part of the loop that is relevant to the research is shown in Figure 1.1. This is a single zone air handling unit. The main components of the system are hot water to air heat exchanger, steam to hot water exchanger, supply fan, water pump, air ducts, dampers that allow mixing of air, and numerous sensors. Thus, there are a number of components that are potential candidates for a fault classification scheme.

In the proposed experimental set up, the air flow loop is run as a closed loop. This implies that no air enters or leaves the loop. The air handling unit consists of a filter, hot water coil, and a variable feed supply fan. The hot water flowing through the hot water to air exchanger (HWX) heats the incoming air. The heated air is then driven through the loop by the supply fan. The warm air then flows over a chilled water coil. The chilled water flow can be adjusted manually. This is used to simulate different load conditions in the loop. The cool air then returns to the HWX. Normally, the air coming out of HWX is maintained at a set point. External to this loop is a steam to water heat exchanger (SWHX). The hot water that flows into HWX is heated in the SWHX. Steam that flows into the SWHX can be controlled using a pneumatic valve. The steam pressure is maintained at a steady level and is

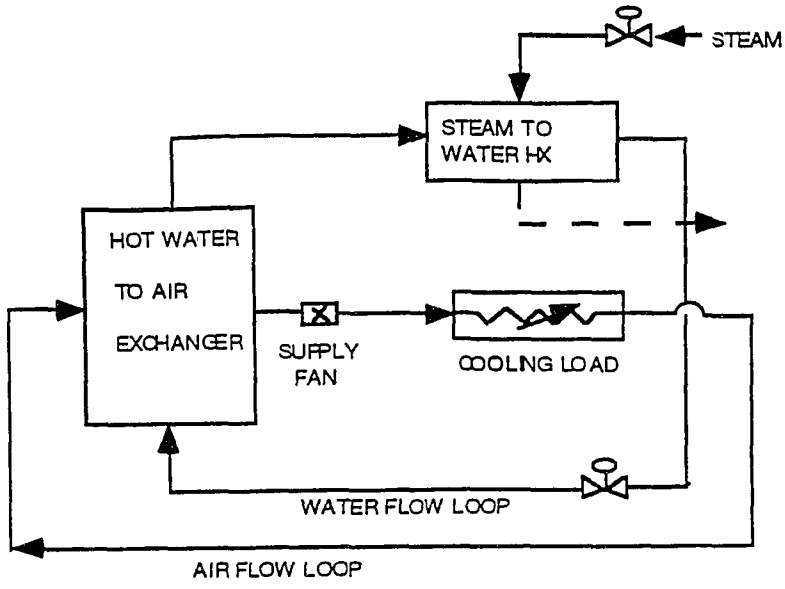

Figure 1.1: Simplified schematic of HVAC system.

generated at the Iowa State Physical Plant. The hot water flow through the HWX can be controlled manually.

The controlled variable for the system is the temperature of the heated air on the downside of the supply fan. The objective is to maintain a certain set point temperature for this controlled variable. The control variables are the fan speed, water flow rate and steam flow. These variables are adjusted, taking into consideration the temperature of the incoming air, to obtain the set point. The variables are adjusted manually, or by the system controller. A data acquisition system provides the air flow rate, water flow rate through HWX, input and output temperatures to HWX, the incoming air temperature, and the conditioned air temperature readings, sampled at 1 Hz.

Neural Networks and Modeling

Neural networks have proven the ability to implement Boolean logic funtions (Morgan and Scofield 1991 and Muroga 1971), to partition the pattern space for classificaton problems (Lippmann 1981 and Makhoul et al. 1989), and to implement non-linear transformations for functional approximation probelms (Cybenko 1989). One of the more popular networks is the Multilayer Perceptron (MLP) (Hush and Horne 1993). This neural network is a system which has layers of nodes that are interconnected between the inputs and the outputs. The interconnections have weights associated with them. The weights are chosen during the training process (back-propagation). The back-propagation training algorithm is an iterative gradient algorithm designed to minimize the mean square error between the actual output of the multilayer perceptron and the desired output (Lippmann 1987). The network is said to converge when the error has been minimized to within tolerable limits for a training set consisting of input-output pairs. Applications of back-propagation based MLPs include speech synthesis and recognition, visual pattern recognition, analysis of sonar signals, defense applications, medical diagnosis, and learning in control systems. Applications specific to control systems include controller modeling, process identification and modeling, and inverse process modeling (Samad 1991).

Neural networks have universal approximation capabilities in the sense that they can be used to model arbitrarily complex, highly nonlinear, multidimensional functions (Hornik et al. 1989). This makes neural networks an attractive choice for modeling highly nonlinear complex systems. Further, not much knowledge of the actual system is required. All the network needs is sufficient numbers of input/output patterns that cover the domain of interest. The objective of the system model is

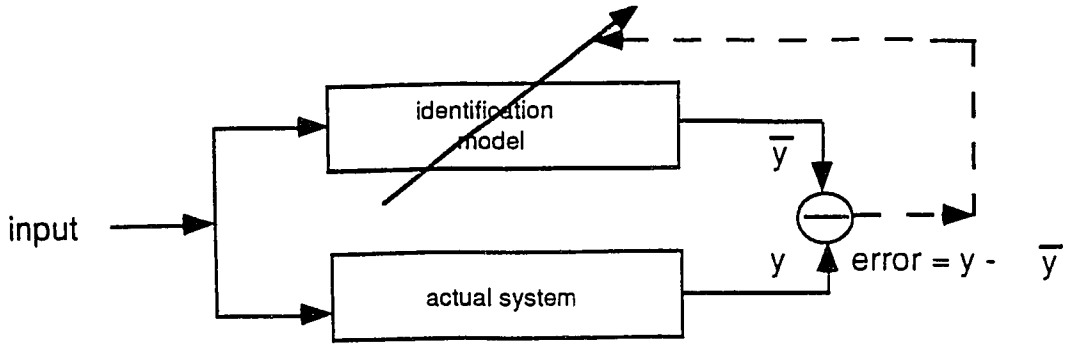

Figure 1.2: System identification.

to duplicate the output of the system when working under normal conditions. The process of system modeling or identification is shown in Figure 1.2.

There are two classes of neural networks that could be used for system modeling. i) Static neural networks: A static neural network is one in which the output is a function of the current input only. The node equations are memoryless. The MLP is the most widely used static network. The static neural network is capable of approximating any functional relationship between input "x" and output "y" given by

$$y(k) = F(x(k)).$$

The static network can be used to model the steady state characteristics of a system. ii) Dynamic neural networks: Dynamic neural networks or recurrent neural networks are systems with memory. Their node equations are often described by differential or difference equations. The dynamic neural network is theoretically capable of approximating any functional relationship given by

$$y(k) = F(x(k), x(k-1), x(k-2), ..., x(k-p), y(k-1), y(k-2), ..., y(k-q)).$$

The dynamic network can be used to model the steady state as well as the transient characteristics of dynamic systems. All the reported work in modeling dynamical systems are mainly theoretical, or on simple systems. Dynamic modeling of systems holds great promise.

Bhat et al. (1990) discusses the use of neural nets for modeling nonlinear chemical systems. Their paper discusses the results on steady state modeling of a nonlinear chemical reactor, and the dynamic model that predicts the dynamic response of the pH in a tank into which flows sodium hydroxide and acetic acid. For their steady state modeling a simple chemical reaction was used. The underlying reactor kinetics are assumed to be unknown. The network had four inputs and three outputs. The number of hidden layers was arbitrarily set at nine. It took presentation of 10,000 input/output patterns before convergence occurred. The MLP network that was trained using backpropagation did an excellent job in learning the governing chemical reaction. For the dynamic modeling example, the authors studied the dynamic response of pH in a stirred tank reactor. Two input streams, one containing sodium hydroxide and the other acetic acid, flow in at rates F_1 and F_2. To model the pH response a back-propagation MLP net was used. Of the flow rates F_1 was held constant. At any given time the inputs to the net were current and past four pH and F_2 values, and the future five F_2 values. The pH was predicted one to five steps into the future. They had a data base of pH and flow rates. They repeatedly submitted the network to the contents of the network until it converged. The input and output layers had fifteen and five neurons repectively. The only hidden layer had five nodes, stating thereby the simplicity of the dynamic system. The net gave excellent predictions on convergence and is reported to have performed better than

an ARMA model.

Nguyen and Widrow (1990) show in their paper how a neural network can learn on its own accord to control a nonlinear dynamic system. An emulator, which is a multilayered neural network, learns to identify the system characteristics. Basically, the emulator is a neural network model of the dynamic system. Another neural network then learns to control the emulator. This self-trained controller is then trained to control the actual dynamic system. The system is the backing of a trailer truck to a loading dock.

For the plant identification part, it is assumed that all the states of the plant are directly observable. A neural network with as many outputs as there are states, and as many inputs as there are states plus plant inputs, is created. They used some empirical rules to determine the number of layers and the number of nodes in each layer. The network is trained to predict the succeeding states, given the current states and the steering signal (plant input). The actual value of the next state is used to train the neural network. It took about 20,000 back-ups of the truck before the net was trained. The emulation and control worked well on a simulated truck. In this technique, one has to know the exact state trajectory, and the network is basically trained to follow it. Further, the states have to be completely observable.

Narendara and Parthasarathy (1990) discuss the identification and control of dynamical systems using neural networks. They defined four models that can describe the various nonlinear models. In the first model, the output at any instant is a linear function of the weighted sum of past output values and a nonlinear function of the present and past input functions. In the second model, the output is a nonlinear function of the past output values and a linear fuction of the weighted sum of present

and past input values. In the third model, the output function is the sum of two nonlinear fuctions. The first sum is a nonlinear function of all the past output values. The second sum is a nonlinear function of all the present and past input values. In the fourth model, the output is a nonlinear function of the past output values, present and past output values. These models can be of any order, depending on the system being modeled. The authors also give the neural network structures to identify the four different models.

The authors identified plants known to be governed by certain difference equations, though the exact nature of the nonlinearity was unknown. For simple difference equations, it took as many as 100,000 time steps to identify the function. Theroetically, it sounds exciting. Practically, one is not working with simple dynamic systems, and the model that pertains to the system of interest may be unknown. Thus, when modeling a dynamic system using neural networks, one is adding to the list of uncertainties that already exist when working with neural networks. The uncertainties referred to are those associated with number of layers, number of nodes per layer, learning rate of neural network (which for back-propagation is anywhere from 0.1 to 0.9), number of training samples and finally whether the neural network is going to converge. The proposed network is easier to train, and there are no empirical rules to depend on. This network will be discussed in Chapter 3. Further, there has been no mention of modeling of HVAC systems using neural networks, to the best of the author's knowledge.

Fault Diagnostics

"Total quality control" is an extremely common phrase in the industry. Industries are under pressure to come up with products that are reliable, unsurpassed in quality, safe, and by all means economical to the user. To guarantee all this there must be some means of supervising the process. The process supervisor should be able to detect faults, evaluate the extent of the fault, raise an alarm if necessary and perform fault diagnosis (localize and estimate the nature of fault(s) and inform the user and the sytem controller). If this can be done successfully, it would permit lives and machinery to be saved, the machine to be run at an optimum, and increased longevity of the machine.

Kao (1985) showed that the sensor errors alone in the air-handling unit of an HVAC system increase the annual energy requirements up to fifty percent. In an HVAC system, in addition to the sensors, fans, pumps, pneumatic valves, air ducts, dampers and heat converters are some of the parts that could develop defects. In a world that is becoming environmentally conscious, such wastage of energy from malfunctioning of parts will not go unnoticed much longer. In addition to energy losses, HVAC system defects could lead to human discomfort and the breakdown of the machine or machine parts. Thus a reliable, practical and economic fault diagnostic system is a must!

Willsky (1976) surveyed a number of methods for the detection of abrupt changes in stochastic dynamical systems. The author concentrates on the design of failure sensitive filters for the detection of a wide variety of changes in linear time-invariant systems. A reasonably accurate model of the system is essential. Iserman (1984) illustrates with the aid of process models as well as estimation and decision methods,

that it is possible to monitor non-measurable variables such as process states, process parameters and characteristic quantities. The technique is to detect, locate and estimate faults with the help of mathematical models and measurable input and output quantities. Limit and trend (derivative) checking of measurable quantities can be performed to monitor changes in the process. Spectral analysis could highlight fault signatures. If process faults are indicated by internal non-measurable process state variables, it might be possible to estimate these state variables from measurable signals using a known process model. Once again, there is the need for a working mathematical model of the system.

There has been an abundance of research geared towards fault diagnostics in HVAC systems. The approaches include steady-state input/output relationships, energy balances, and dynamic system models. But according to sources at Johnson Contols, a pioneer in the HVAC and controls industry, all of these look good on paper. For an HVAC industry, where little has changed in the design for the HVAC systems or the controls in a few decades, it will take an economical, easy to implement, and reliable fault diagnostic system to overcome their inertia. Some of the HVAC fault diagnostics related work is summerized below.

Haberl and Claridge (1987) developed relationships between environmental, building load and occupany variables as inputs, and fuel consumption as output. Input/output correlations were compared with the reference period of normal operation to detect problems.

Anderson et al. (1989) also added statistical analysis and a rule-based approach to detect faults. The statistical analysis included redundancy checks for sensor failures, and checks for differnces between measurements and predictions. Predictions

were made using historical data. If the differences were greater than a tolerable limit, it was considered as the indication of a fault.

Liu and Kelly (1989) compared actual performance with an optimum calculated by simulation for specific outside and inside zone temperatures and checked flow rates, supply air temperature, temperature of the mixed return and outdoor air, and proper sequencing of valves and acutators. Pape et al. (1991) used input/ouput relationships involving electric power. They developed an optimal control strategy that yields the optimal set of control variables that minimizes power consumption at any time. The power consumption of the entire HVAC system is represented by a quadratic relation for the total power in terms of the control variables, loads, and ambient conditions. Any deviation from optimal power consumption is the indication of faults in the system.

Iserman (1989) used dynamic models and parameter estimation for diagnosing faults in a pump and a heat exchanger. The nonlinear system model for the pump had nine process coefficients and deviations in these coefficients were related to nineteen different faults. This method requires extensive amounts of data, high sampling rates and is not practical at a system level.

The proposed research is based on a steady state neural network model of the system. Sensor redundancy, isolation of defects using the model, and fault classification using fuzzy neural networks, will be combined to form the proposed fault detection scheme. Steady state fault detection can detect many of the faults in HVAC systems (Norford 1992). Faults internal to the motor and pumps might need vibrational analysis, but they are less likely to fail than other parts.

Experimental Work

The experimental part of this research was extremely time consuming and to a great extent, frustrating. But that is to be expected when one works with real-life systems. One has trouble duplicating the same results, however hard one tries to duplicated all of the conditions. Further, the system takes almost forty five minutes to "settle down", and the reading can be taken only after it reaches that equilibrium. This means, it takes that long to take a single reading. It took nearly six months of operating the system before it was possible to get the system tailored to operate in the manner necessary for collecting experimental data. The supply fan can be operated at several differnt speeds. The air flow rate is dependent on the fan speed. But the fan does not maintain a steady speed. Thus, a strobe meter was used to measure the exact fan speed. Further, it is almost impossible to have the dampers, that control the air flow in the duct, twice in the same exact position. To get around this, the decision was made to have the system work in a closed loop. Thus, one does not have to worry about the dampers. The steam pressure, controlled by the physical plant, was never steady. Steam could be trapped in the hot water pipe, thereby affecting the flow rate and heat transfer. This steam had to be released on a regular basis using escape valves. There is a world of difference between the real system and simulated models.

The experimental work involved collecting data for all possible realizations. These include numerous combinations. The input variables to the system are inlet air temperature, fan speed, steam flow, the inlet water temperature to the hot water to air exchanger, and the water flow rate. The output variables are the outlet air temperature and the outlet water temperature. The control variables are the fan

speed, water flow, and steam flow. The temperature of the out flowing air stream is the only controlled variable. The ranges for the operation of fan speed, air temperature, water flow rate, and steam flow rate have been selected. Readings of the outlet air temperature for all possible variations and combinations of the above mentioned variables were taken. In addition to these five readings, the corresponding air flow rates, and inlet and outlet water temperatures are also recorded. Once all the readings were taken, different fault situations were introduced, and corresponding readings recorded. An example of a fault scenario could be change in fan speed from its set point, which could be easily realized.

A good model is essential for a fault detection scheme. For this reason, this research is mainly focussed on developing a steady state model of the HVAC system. A new network is used to model the system. This network adheres to the simplicity constraints. The model sounds the alarm in case of any system discrepancies. A fuzzy neural approach is used to identify the faulty component. Chapter 2 analyses the conventional neural networks, in light of their limitations. Chapter 3 discusses picecewise polynomial approximation. Chapter 4 introduces the new network, and illustrates the modeling results. Chapter 5 discusses fuzzy neural techniques, and their application to fault classification in HVAC systems. Chapter 6 includes a brief summary of the research achievements, and suggestions for future research.

CHAPTER 2. NEURAL NETWORKS AND PROPOSED NETWORKS

Neural networks are parallel interconnections of neurons. They consist of several layers, with varying number of neurons in each layer. Each neuron, in all but the first layer, is usually connected to all the neurons in the preceding and succeeding layers. Weights associated with the interconnections are assigned during the neural network training phase. The neurons are assigned activation functions that could be linear, sigmoidal, gaussian, or any other function the user assigns. The input to each neuron is the sum of neuron outputs from the preceding layer, multiplied by the respective connection weights. The output of each neuron depends on the activation function. Figure 2.1 is the schematic of a three layer Multilayer Perceptron (MLP). MLPs are the most widely used neural networks. The first and last layers are referred to as input and output layers respectively. All the other layers are hidden layers. The network in Figure 2.1 has one hidden layer. The input layer neurons have unity gain activation functions.

Neural network applications can be broadly classifed into pattern recognition and functional approximation. For both these cases, the neural network is trained to map given input variables to certain output variables. The user decides on the number of layers, and the number of neurons per layer by trial and error. The

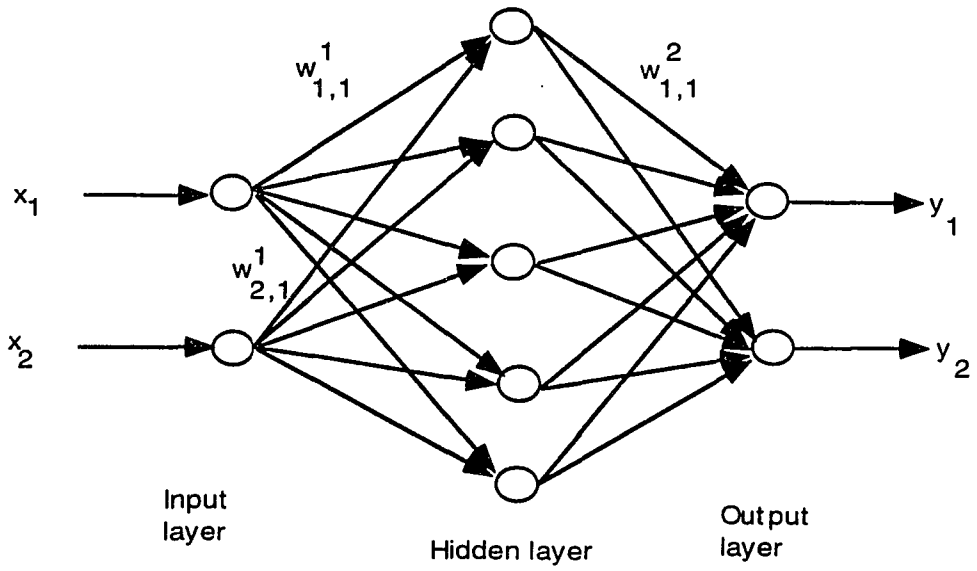

Figure 2.1: Three layer MLP.

number of neurons in the input and output layers depends on the number of input and output variables, respectively. More layers give the network greater freedom. The network becomes more powerful with the increase in the number of neurons per layer. The network size is usually picked by trial and error. It is also left to the user to decide the activation functions for the neurons in different layers. For functional approximation problems, the activation functions of the output layer neurons are normally chosen to be linear. The user is once again at a loss, when it comes to selecting the interconnection weights. The common approach is to set the weights to small random numbers. The parameters that uniquely specify a neural network model are the number of layers, number of neurons per layer, and the interconnection weights.

After picking an arbitrary set of parameters for the neural network, the network is ready for the training phase. The input variables are submitted to the network and the network output obtained. This output is compared with the expected ouput called the target vector, and the error vector is computed. The error vector is used to adjust the interconnection weights in an iterative process called backpropagation. In this gradient descent technique, the weights are adjusted so as to minimize the sum squared error between the network output and the target vector for all the input/output pairs. The weights are adjusted in the direction of steepest descent with respect to the error. The weight adjustments are proportional to the weights' effect on the sum squared error. The network is said to be trained, when the sum squared error drops below a threshold value. This network can be used to generate ouput vectors when presented with input vectors.

Training a MLP

The output of the jth neuron in layer $l+1$ is given by

$$y_j^{l+1} = f[(\sum_{i=1}^{N} y_i^l w_{i,j}^l) - c_j^{l+1}]. \tag{2.1}$$

Here N is the number of neurons in layer l, f is the activation function of jth neuron in layer $l+1$, $w_{i,j}^l$ is the weight of interconnection from the ith neuron in layer l to the jth neuron in layer $l+1$, and c_j^l is the bias or threshold of the the jth neuron in layer l. The biases facilitate better representation of input-output relationship. Given any input vector, the output vector of the network is obtained by stepping through the network layer by layer. The output of any neuron is obtained using Equation 2.1.

Training stages

All the interconnection weights are set to small random numbers in the range [-1,1]. The number of layers and the number of nodes are to some extent picked arbitrarily. In case the network fails to converge, the user picks a different network architecture. The network is said to converge when the sum squared error over all the input/output training pairs drops below the user picked threshold level. The input to the jth neuron in layer $l+1$ is given by

$$x_j^{l+1} = \sum_{i=1}^{N} y_i^l w_{i,j}^l - c_j^{l+1}. \tag{2.2}$$

Assuming sigmoidal activation function for all the neurons, the output of jth neuron in the $l+1$ layer (not the input layer) is given by

$$y_j^{l+1} = \frac{1}{1+e^{-x_j^{l+1}}}. \tag{2.3}$$

For nodes in the input layer (l =1), the output is given by

$$y_j^1 = x_j^1, \tag{2.4}$$

where x_j^1 is the jth component of the input vector. The output of the neurons in the second layer are obtained using Equation 2.1. The outputs of the neurons in the higher layers are calculated sequentially, until the the output of the neurons in the output layer (L) is obtained. This is the ouput of the network for the initial set of weights.

The least mean square error of the network for all input/output cases is calculated using (Pal 1992)

$$E(\mathbf{w}) = \sum_c \sum_j (y_{c,j}^L(\mathbf{w}) - d_{c,j})^2, \tag{2.5}$$

where $y^L_{c,j}(\mathrm{w})$ is is the output of jth neuron in the output layer for input-output case c, and $d_{c,j}$ is the target output vector for the jth neuron for input-output case c. Here w is the current weight vector. Thus, the error is a function of the weight vector. The objective is to find the weights responsible for the error and adjust them in order to minimize the error. The gradient descent method is one of the popular techniques used to adjust the weights.

In the gradient descent method, one starts with a random weight vector. Each weight in the vector is updated to

$$w^l_{i,j}(k) = w^l_{i,j}(k-1) + \Delta w^l_{i,j}(k), \tag{2.6}$$

where

$$\Delta w^l_{i,j}(k) = -\epsilon \frac{\partial E(\mathrm{w})}{\partial w^l_{i,j}(k-1)} + \alpha \Delta w^l_{i,j}(k-1) - dec.w^l_{i,j}(k-1). \tag{2.7}$$

In Equation 2.7, the positive learning rate ϵ controls the descent in the error space, dec is the decay coefficient, $0 \leq \alpha \leq 1$ is the momentum constant, and k is the iteration number. The decay coefficeint is used to remove weights that do not play any role in error reduction. The learning rate controls the speed at which the network converges to the desired optimal solution. The momentum rate, if appropriately chosen, reduces the possibility of the network getting stuck in local minima. For any given weight vector, the error is calculated using Equation 2.5. The weights are updated using Equation 2.7. The error is once again calculated for the new set of weights. This process is continued until the error drops below the user picked value, when the network is said to converge. The partial derivatives in this equation are calculated using the popular error back propagation method (Hush and Horne 1993). It may take many passes of the input data before the network converges.

Drawbacks of Neural Networks

There are many practical concerns when working with neural networks. The biggest concern is choosing the network size. For a given problem, one does not know the optimum size for the neural network. This refers to the number of layers and the number of neurons per layer. Choosing the network size is critical. Increasing the number of neurons per layer could result in overfitting. In this situation, the training points are well fit, but there exist oscillations between training points. On the other hand, if the number of neurons are too few, it could result in underfitting. In one approach, one starts with the smallest possible network, and gradually increases the size until the performance levels off. Each network would be trained separately. Another approach is to start with a large network, and then destroy weights and nodes that do not contribute, using a pruning technique (Hush and Horne 1993). These techniques are extremely time consuming and may never produce the optimum network.

Another major concern is the length of the training process, and the size of the training input/output set. The training set consists of all the input/output numerical pairs that are used to train the network. Normally, the network is trained until convergence. This happens when the error drops below a threshold value. The training duration, and the training set size depend on the complexity of the function being approximated. The convergence can be hastened by picking the right learning rate. A lucky guess is the best way of picking the right learning rate. The backpropagation is a gradient search algorithm , that could get trapped in local minima. The momentum coefficient could reduce the chances of the network getting trapped in local minima. But once again the optimum momentum rate is in the range [0,1]. With all

these aids, the convergence of the network is not always guaranteed.

The function being approximated do not appear to have any bearing on the weights of the artificial neural network. For this reason, the weights are initially set to small random numbers. The final weight vector depends on the initial set. Training would be a lot faster if the initial weights are set closer to the actual weights. But there are no rules for deciding the weights, that have strong theoretical backing.

In attempts to hasten the training process, the backpropagation algorithm is steadily becoming more complex. A variety of parameters, that can take a wide range of values, have to be picked by the user. This adds to the long list of uncertainties associated with training the network. Considering the amount of a learning the human brain does every day, the logical explanation is that the learning process has to be simplistic. It has remained elusive so far. To shed more light to this discussion, the operation of the MLP is analyzed in the following section.

Analyzing the MLP

Neural networks attempt to mimic the human brain. But none among the numerous networks that have become popular in the last decade, come anywhere close to emulating the human brain. Researchers have accepted the limitations of neural networks, and are content with exploiting the numerous capabilities of neural networks. The MLP is analyzed to provide the motivation for the proposed network.

Figure 2.2 illustrates a single input/single output neuron. The activation function is sigmoidal and the output of the neuron is given by

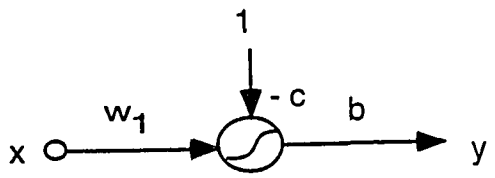

Figure 2.2: Single input perceptron.

$$y = \frac{b}{1 + e^{-(w_1 x - c)}}, \tag{2.8}$$

where c is the threshold, and w_1 is the weight. Figure 2.3 gives the plot of output versus input for $b = 1$. The threshold and the weight determines how much the function is shifted in the x-direction. Figure 2.4 shows the output for $b = 2$. Here b is a constant that stretches the sigmoid function.

It has been proved that a neural network with one hidden layer can approximate any arbitrary non-linear function. In functional approximation applications, the neurons of the hidden and output layers have non-linear and linear activation functions respectively. Figure 2.5 illustrates a typical neural network for a single input/single output system. There are N neurons in the hidden layer. The activation functions for the hidden layer are sigmoidal and are given by Equation 2.3. The output of this network is given by

$$y = a_1 f_1(x) + a_2 f_2(x) + \ldots + a_N f_N(x). \tag{2.9}$$

Here a_i is the connection weight from the ith neuron in the hidden layer to the output

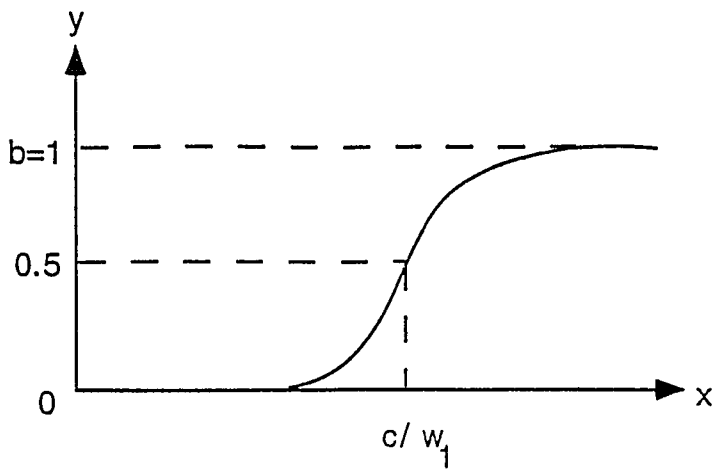

Figure 2.3: Plot of output versus input for b=1.

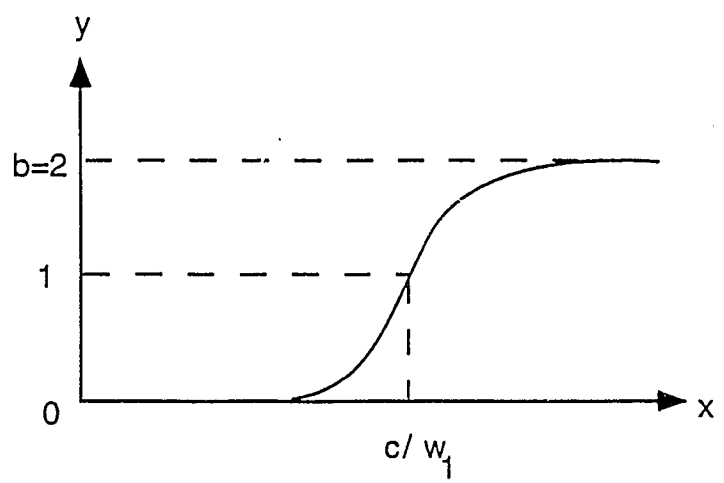

Figure 2.4: Plot of output versus input for b =2.

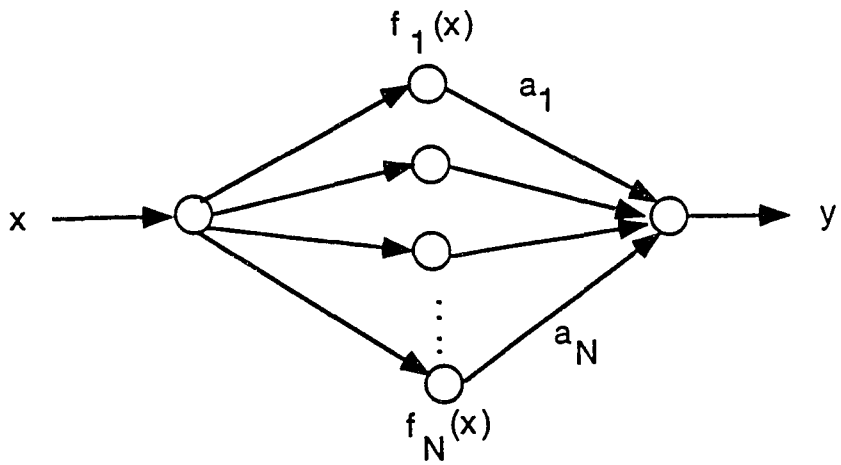

Figure 2.5: A typical functional approximation type MLP.

neuron. Also

$$f_i(x) = \frac{1}{1 + e^{-(w_i x + c_i)}}, \tag{2.10}$$

where w_i is the weight of the connection from the input neuron to the ith neuron in the hidden layer, and c_i is the bias of the ith neuron in the hidden layer. Thus the output is a linear combination of the functions $f_1(x)$, $f_2(x)$, ..., and $f_N(x)$. From Figure 2.3, it is clear that $f_i(x)$ is a sigmoid function centered at $x = c_i/w_i$. The the output is a linear combination of sigmoidal functions shifted to different abscissas in the input domain. The sigmoidal functions can be considered as basis functions, with infinite support. Thus this MLP performs piecewise sigmoidal interpolation, with sigmoidal functions acting as basis functions.

Artificial Neural Networks versus Biological Networks

Artificial neural networks are far from emulating their biological counterparts. Equation 2.9 illustrates in a simple fashion the neural network philosophy. One could visualize the neural network operation, as trying to fit shifted and stretched sigmoidal functions in a smooth fashion to the functional surface. In Figure 2.6 an unknown function that is approximated using neural netwoks is shown(Hush and Horne 1993). The final network is also shown in the figure. By trial and error, the network converged to a two layer network with two nodes in the hidden layer. This is obvious from visual examination of the function. Hush and Horne (1993) also model the same function using a neural network with gaussian activation functions. It took a network with five nodes in the hidden layer and gaussian activation functions to approximate the same function. Thus, it clearly indicates that choosing the right activation function can simplify the network. Further, it illustrates how neural networks work. The network is trying to fit functions to the domain of interest. The higher the complexity, the more the number of nodes in the hidden layer. In the example shown in Figure 2.6, it took two sigmoidal functions and hence the two nodes in the hidden layer.

In the MLPs, generally all the nodes in the hidden layer have the same activation function. This keeps the learning process as simple as possible. The sigmoidal and the radial basis functions are among the most popular activation functions. An analogy is trying to unscrew nuts in varying sizes and shapes with sockets of different sizes, but of a single shape. The task would be so much easier if one had the socket of the right shape and size. Likewise, rather than limiting the network to a certain kind of function, it definitely can approximate better if it has a few functions to choose from.

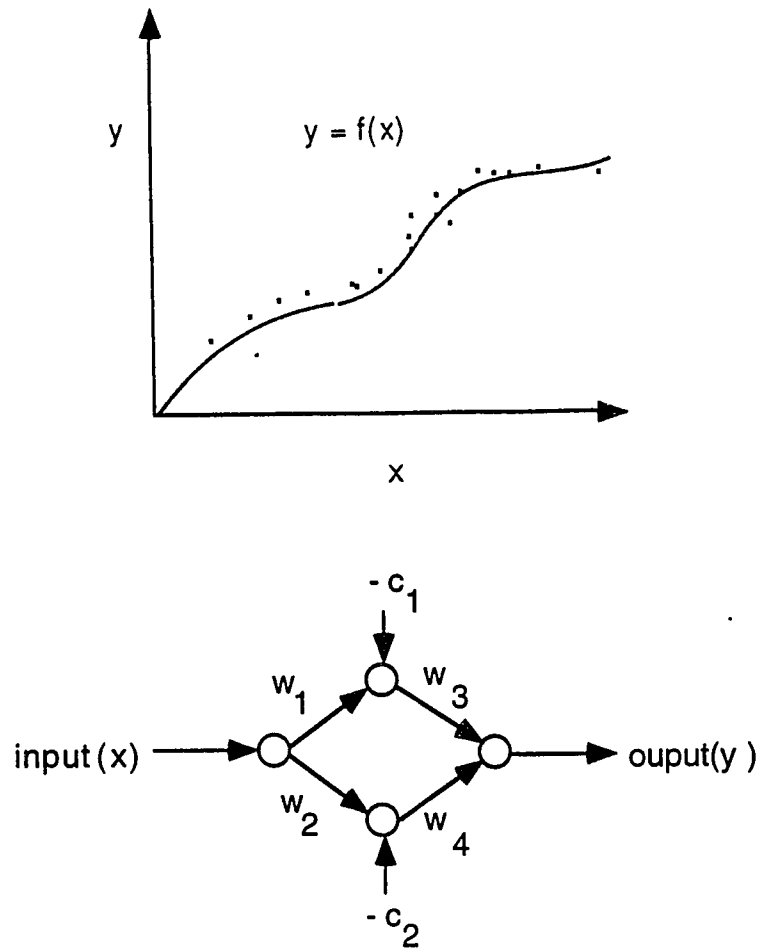

Figure 2.6: Approximated function and the corresponding neural network

During the training process, there do not exist any rules that help the user to pick the size of the network, and the weights of the interconnections. The neural network has no knowledge of the function that it is trying to approximate. It could be argued that the lack of knowledge and choice of activation functions are two of the striking differences between the artificial and biological networks.

A blindfolded person, equipped with sockets of all sizes but a single shape, trying to unsrew a nut of unknown size and shape, offers a fitting analogy to the artificial neural network learning process. Being constrained to a certain type of activation function and lack of any information regarding the function of interest, explains to a great extent the numerous iterations during the training process. Like the blindfolded person, the only way out is through trial and error.

How does the human cognitive mechanism work? Here is a suggestion that compares the learning of President Clinton's face by a human brain and an artificial neural network. The human being knows that it is the face of a caucasian male. Further, there probably are functions defined for human faces based on race. Now all the learning mechanism has to do is to fine tune the facial parameters to that of Mr. Clinton's. A neural network, on the other hand, does not even know what it has been presented with. This is one of the reasons why the network could take forever and ultimately fail to converge. Thus, for an artificial neural network to better emulate human performance, it should be provided with the nature of the function being approximated. The functions that could be associated with human faces could be piecewise polynomial functions (splines).

Proposed Network

Figure 2.7 gives the same function that was approximated in Figure 2.6. Here, the function is approximated by four straight lines. The domain of the input variable has been divided into four regions. This is made possible by picking five break points (x_0, x_1, x_2, x_3, and x_4 are the break points). The break points are picked such that the interval defined by any consecutive break points can be approximated by a linear function. In Figure 2.8 is shown, the corresponding network that approximated the function. The network has four neurons corresponding to each one of the regions. Depending on the input range, the input selector switch directs the input to the right neuron. This switch also connects just this neuron to the output. The activation function of each neuron is linear and is given by,

$$y = w_i x + c_i \qquad i = 1, 2, \ldots, n. \tag{2.11}$$

Here 'n' is the number of regions, 'w_i' is the connection weight, and 'c_i' is the threshold. The connection weight is the slope of the region the particular neuron represents. The threshold is the y-intercept. Using the above example as a motivation, the first neural network is proposed. This network is a finite element-type network, and hence the name.

Finite element-type network

The entire input domain is divided into different regions that are decided during the training process. In each range, the function has to be approximately linear. The size of the regions vary, and are dependent on the nature of the function. In a certain region, if the function is highly non-linear, the range size will be small. To each

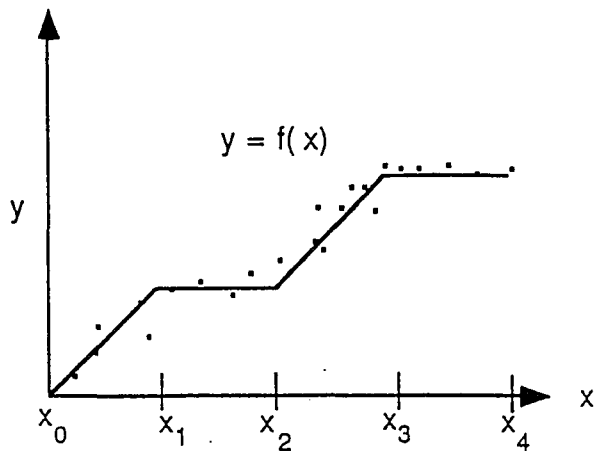

Figure 2.7: Finite element modeling

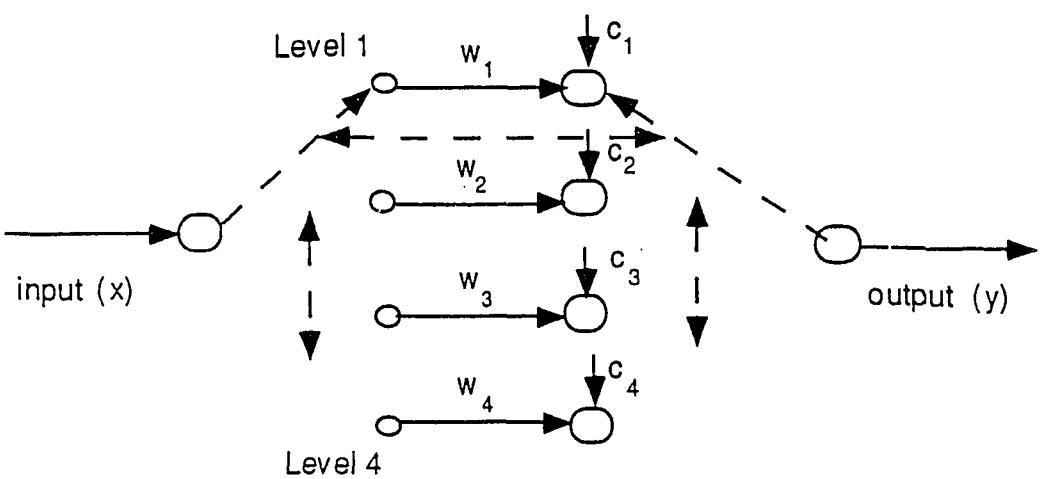

Figure 2.8: Finite element-type network.

region is assigned a neuron that approximates the function in that interval. There will be as many inputs to each neuron as there are dimensions to the function being approximated.

In the first stage of training, the ranges are selected. In the second stage, the weights and the thresholds are picked. This is basically a linear regression problem. There will be as many neurons as there are regions in the functional domain. The number of elements for this network could be larger than a MLP, but there is no comparison to the reduction in complexity of the training process. Convergence of the network is also guaranteed. There are discontinuities at boundaries of the various ranges. This network can be used where smoothness of the approximating function is not a major concern. Smoother fits can be obtained using the network proposed in the following section.

Piece-wise functional network

This network is an extension of the finite-element type network. Ideally, in this network the domain of interest is divided into regions based on its functional nature. Figure 2.9 illustrates this for the one-dimensional case. The highly non-linear function is a combination of sigmoid, gaussian, linear, and constant functions. x_0, x_1, x_2, x_3, and x_4 are the break points. The break points are picked such that the interval defined by any consecutive break points can be approximated by a non-linear function. The function can be made smooth. The learning stages, that this network goes through, could possibly closely match the human learning process. They are listed below:

1) It identifies the regions based on the functional characteristic. This could be an

almost impossible task.

2) To each region is assigned a neuron. Each neuron has as many connections as there are dimensions to the function being approximated.

3) There is a range selector switch that assigns the input functions to the correct region in the functional domain.

4) Training involves determining the connection weights and the threshold for each neuron.

Let y = f(x) be the function to be modeled. Then according to the proposed model

$$\begin{aligned} y &= \sum_{i=1}^{n} a_i f_i(x) \\ a_i &= 1 \qquad x_{i-1} \leq x \leq x_i \\ a_i &= 0 \qquad \textit{otherwise.} \end{aligned} \tag{2.12}$$

y = $a_i f_i(x)$ is the output of the ith neuron, and n is the number of regions. Thus for any input only one of the neurons is active. In other words, $f_i(x)$ has non-zero support in only the ith region.

The approach taken, in the proposed network, is to break down the functional domain into several regions. In each region, the system being modeled can be approximated with desired accuracy by a single function. Dividing the functional space into regions could become extremely tedious, when the input dimension is greater than one. Techniques from mathematical morphology could be used for approximate classification of domain of interest into functional regions. But, once again things are being made too complicated. A simplistic approach would pick regions that are small enough, where lower order polynomials can do the approximation. A similar approach is taken in curve and surface fitting using splines. Chapter 3 discusses in

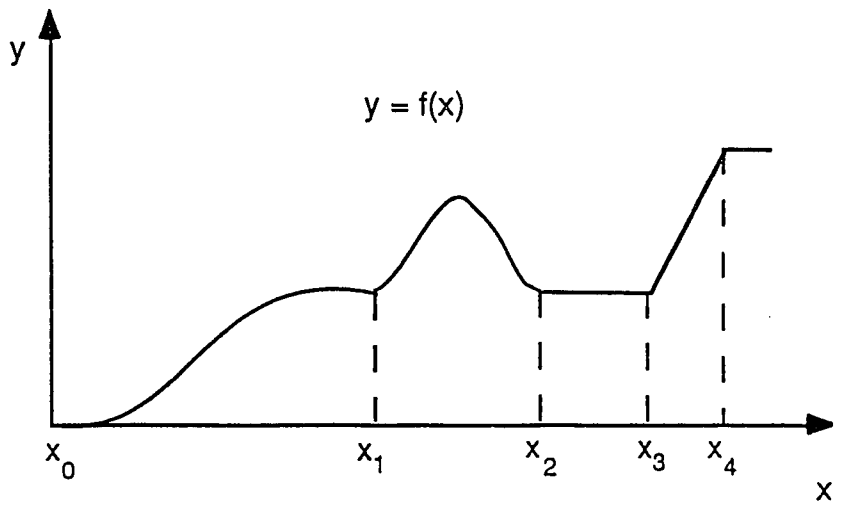

Figure 2.9: One-dimensional illustration of the piece-wise functional model.

detail the theory of splines.

CHAPTER 3. CURVE FITTING WITH SPLINES

Polynomials are among the most important class of functions used for representation of input/output relationships. If the function is approximated over a large interval of the input domain, the approximating polynomials could tend to be unacceptably large. This could result in oscillations of the fitted curve. An alternative is to subdivide the input domain into small intervals. Lower order polynomials are then used to approximate in each of the subintervals. The polynomial pieces can be forced to blend smoothly to give the composite function,

$$s(x) = \sum_j a_j p_j(x). \tag{3.1}$$

This smooth composite function is called a piecewise polynomial function or a spline. The polynomial pieces could be linear, quadratic, cubic, or even higher order polynomials. The piece-wise cubic polynomial functions or cubic splines are the most popular. They are the compromise between accuracy and computational complexity. The discussion will begin with a treatise on general curve fitting with polynomials (Lancaster and Salkauskas 1986).

Polynomial Interpolation

A polynomial is said to belong to the class P_N, if its degree is N or less. In the simplest case of polynomial interpolation, we are given $N + 1$ x-values and the

corresponding y-values. The objective is to find the polynomial f(x) that satisifies,

$$f(x_i) = y_i, \qquad i = 0, 1, \ldots, N. \tag{3.2}$$

Here there is one-on-one mapping between the sets $\{x_0, x_1, \ldots, x_N\}$ and $\{y_0, y_1, \ldots, y_N\}$. It has been proved that there exists a unique polynomial f(x) in the class P_N, such that equation 3.2 is satisfied. This polynomial takes the form,

$$f(x) = a_N x^N + a_{N-1} x^{N-1} + \ldots + a_1 x + a_0. \tag{3.3}$$

Lagrange's method for interpolation

This is a different approach taken to determine the polynomial that satisfies Equation 3.2. Here $N + 1$ different but simpler interpolation problems are solved. These are then combined to form the complete solution. The $N + 1$ primitive set of y values,

$$\{1, 0, \ldots, 0\}, \{0, 1, 0, \ldots, 0\}, \ldots, \{0, \ldots, 0, 1\} \tag{3.4}$$

are taken. Polynomials L_0, L_1,..., L_N are obtained as solutions to the interpolation of $\{x_0, x_1, \ldots, x_N\}$ with each one of the primitive sets in the order given. The polynomials L_0, L_1,..., L_N are known as the fundamental Lagrange polynomials or the cardinal functions. The polynomial that satisfies Equation 3.2 is given by

$$f(x) = y_0 L_0(x) + y_1 L_1(x) + \ldots + y_N L_N(x) = \sum_{i=0}^{N} y_i L_i(x). \tag{3.5}$$

The Lagrange polynomials belong to the class P_N and exhibit the properties

$$L_i(x_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases} \tag{3.6}$$

Further they are linearly independent and hence form a basis for P_N. Since all the Lagrange polynomials belong to P_N, it follows that $f(x)$ is also in P_N. From Equation 3.6, L_j has N distinct zeros at all x_i, except for $i = j$. Thus $L_i(x)$ can be represented as

$$\begin{aligned} L_i(x) &= k(x - x_0)(x - x_1)\dots(x - x_N) \\ &= k \prod_{j=0, j\neq i}^{N} (x - x_j), \end{aligned} \tag{3.7}$$

for a real number k. To solve for k, use

$$L_i(x_i) = 1 = k \prod_{j=0, j\neq i}^{N} (x_i - x_j). \tag{3.8}$$

From Equations 3.7 and 3.8,

$$L_i(x) = \frac{\prod_{j=0, j\neq i}^{N}(x - x_j)}{\prod_{j=0, j\neq i}^{N}(x_i - x_j)}, \quad i = 0, 1, \dots, N. \tag{3.9}$$

Hermite interpolation

In Hermite interpolation, in addition to the functional values at the data points, slopes are also provided. Thus, one works with $\{x_0, x_1, \dots, x_N\}$, $\{y_0, y_1, \dots, y_N\}$, and $\{y'_0, y'_1, \dots, y'_N\}$. Hermite interpolation involves obtaining f(x) which satisfies

$$f(x_i) = y_i, \qquad f'(x_i) = y'_i, \qquad i = 0, 1, \dots, N. \tag{3.10}$$

There are $2N + 2$ degrees of freedom. Thus $f(x)$ belongs to the class P_{2N+1}. It has been proved that there exists a unique polynomial in this class that satisfies Equation 3.10. To address cubic splines, only the case $N = 1$ need be considered. Thus we have two data points and four conditions, and hence the degree of the polynomial

is three. To solve this problem, four cardinal functions H_0, K_0, H_1, and K_1 are required. These cardinal functions are cubic and easy to obtain. The interpolating Hermite polynomial is given by

$$f(x) = y_0 H_0(x) + y_0' K_0(x) + y_1 H_1(x) + y_1' K_1(x). \tag{3.11}$$

Linear Splines

Linear splines are piecewise linear functions. As the name suggests, linear splines are continuous, but lack smoothness. ξ[a,b] denotes the class of functions that are continuous at every point of [a,b]. ξ^1[a,b] denotes the class of functions that are continuous and have a continuous derivative at every point of [a,b]. ξ^N[a,b] denotes the class of functions that are continuous and have continuous derivatives of orders $0, 1, 2, \ldots, N$ on [a,b]. Linear splines belong to the class ξ^0[a,b]. The first derivative of linear splines have discontinuities at points known as knots. Knots are the points at which linear interpolation segments start and terminate.

Here one is trying to determine the linear spline, $L_N(x)$, that satisfies Equation 3.2. $\{x_0, x_1, \ldots, x_N\}$ and $\{y_0, y_1, \ldots, y_N\}$ are the knot sequence (K) and the ordinates respectively. At the outset, a cardinal basis is chosen using Equation 3.6. Solving,

$$l_i(x_k) = \delta_{ik}, \qquad k = 0, 1, \ldots, N; \quad i = 0, 1, \ldots, N, \tag{3.12}$$

the cardinal functions $l_i(x)$ $(i = 0, 1, \ldots, N)$ are obtained. Here δ_{ik} is the Kronecker delta and is given by

$$\delta_{ik} = \begin{cases} 1, & i = k, \\ 0, & i \neq k. \end{cases} \tag{3.13}$$

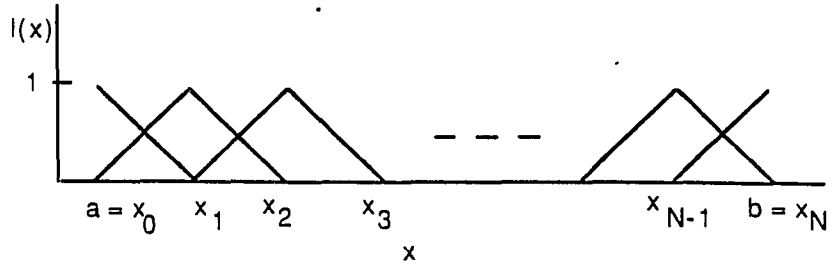

Figure 3.1: Cardinal functions that make up the linear spline.

These cardinal functions are the tent functions given in Figure 3.1. The expressions for the tent functions are

$$l_0(x) = \begin{cases} \frac{x-x_1}{x_0-x_1}, & a = x_0 \le x \le x_1, \\ 0, & x_1 \le x \le b = x_N. \end{cases}$$

$$l_j(x) = \begin{cases} 0, & a \le x \le x_{j-1}, \\ \frac{x-x_{j-1}}{x_j-x_{j-1}}, & x_{j-1} \le x \le x_j, \\ \frac{x-x_{j+1}}{x_j-x_{j+1}}, & x_j \le x \le x_{j+1}, \\ 0, & x_{j+1} \le x \le b. \end{cases}$$

$$l_N(x) = \begin{cases} 0, & a \le x \le x_{N-1}, \\ \frac{x-x_{N-1}}{x_N-x_{N-1}}, & x_{N-1} \le x \le b. \end{cases} \tag{3.14}$$

Here j can take the values $1, 2, \ldots, N-1$. Also $x_0 = \mathrm{a}$ and $x_N = \mathrm{b}$.

From Equation 3.5, the linear spline has the form

$$\begin{aligned} L_N(x) &= l_0(x)y_0 + \ldots + l_N(x)y_N \\ &= \mathrm{l}(x)^T\mathbf{y}, \end{aligned} \tag{3.15}$$

where $\mathrm{l}(x)^T = [l_0(x), \ldots, l_n(x)]$ and $\mathbf{y}^T = [y_0, \ldots, y_N]$. The values of the functions

that are being interpolated show up in the above equation. The values **y** control the form of the linear spline $L_N(x)$.

Least squares approximation by splines

In this case we pick $M \leq N$ knots. The knot set is given by $k_0 < k_1 < \ldots < k_M$ and k_0 = a and k_N = b. Here the interior knots may have no relation to the points $x_1, x_2, \ldots, x_{N-1}$. Thus, given are $\{x_0, x_1, \ldots, x_N\}$ and $\{y_0, y_1, \ldots, y_N\}$. The user picks the M knots. The linear spline is given by

$$L_M(x) = \sum_{i=0}^{M} c_i l_i(x). \tag{3.16}$$

Here c_i's are the spline coefficients, and $l_i(x)$ is given by Equation 3.14. The approximation error at data point x_j is

$$E = \sum_{i=0}^{M} c_i l_i(x_j) - y_j. \tag{3.17}$$

The sum of the squares of the error at all the data points is given by

$$E(L) = \sum_{j=0}^{N} \{\sum_{i=0}^{M} c_i l_i(x_j) - y_j\}^2. \tag{3.18}$$

From Figure 3.1, it is clear that for any j, only three of the $l_i(x_j)s$ are going to be nonzero. This is due to the fact that the support of $l_i(x_j)$ covers only three knots. The $c_i^{'}s$ that minimize the error in Equation 3.18 are obtained from

$$\frac{\partial E(L)}{\partial c_k} = 2 \sum_{j=0}^{N} \{\sum_{i=0}^{M} c_i l_i(x_j) - y_j\} l_k(x_j) = 0 \quad for \quad k = 0, \ldots, M. \tag{3.19}$$

This in turn yields

$$\begin{bmatrix} a_{00} & \cdots & a_{0M} \\ \vdots & & \\ a_{M0} & \cdots & a_{MM} \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} b_0 \\ \vdots \\ b_M \end{bmatrix}, \tag{3.20}$$

where

$$a_{ki} = \sum_{j=0}^{M} l_k(x_j) l_i(x_j), \qquad k, i = 0, 1, \ldots, M,$$

$$b_k = \sum_{j=0}^{M} y_j l_k(x_j), \qquad k = 0, 1, \ldots, M.$$

The matrix $[a_{ki}]$ in Equation 3.20 is tridiagonal due to the nature of the support of the tent functions. This makes the inversion of $[a_{ki}]$ easy to calculate the coefficients. The knots can be uniformly spaced through the input domain. If it is necessary to optimize the number of knots, one could start with the two outer knots and a center knot. The sum squared error to the right and left of the knots is calculated. Another knot is inserted in the region with the greater error. This process is repeated until desired accuracy is achieved. This algorithm was proposed by Ichida *et al* (1976).

Piecewise Cubic Functions and Splines

It is possible to have piecewise cubic functions that have one or two continuous derivatives whereever it is defined. Then the cubic functions belong to the classes $\xi^1[a, b]$ and $\xi^2[a, b]$ respectively. Let K be the set of knots satisfying $a = k_0 < k_1 < \ldots < k_N = b$. If the function values and first derivatives are known at the knots, then Hermite interpolation technique discussed earlier can be applied to each segment $[k_{i-1}, k_i]$ for $i = 1, 2, \ldots, N$. The slope at k_i is the same for the two cubic segments

meeting at that point, namely $[k_{i-1}, k_i]$ and $[k_i, k_{i+1}]$. This ensures continuity of the first derivative at all the interior knots. Likewise, if the second derivatives are forced to be equal at all interior knots, one obtains piecewise cubic functions that have two continuous derivatives. Given $N+1$ knots, the functional values, and slopes at the knots, it requires $2N+2$ cardinal functions to perform Hermite interpolation. The cardinal functions Φ_i and Ψ_i are piecewise cubic polynomials with a small support. The interpolant can be written using Equation 3.11 as

$$S(x) = \sum_{i=0}^{N} \Phi_i(x) y_i + \sum_{i=0}^{N} \Psi_i(x) m_i. \tag{3.21}$$

The following properties

$$\Phi_i(x_j) = \delta_{ij}, \quad \Phi_i'(x_j) = 0, \quad \Psi_i(x_j) = 0, \quad \Psi_i'(x_j) = \delta_{ij}, \tag{3.22}$$

aid in the construction of the cardinal functions. The cardinal functions for i =1,..N-1, with $h_i = x_i - x_{i-1}$ are given by

$$\Phi_i(x) = \begin{cases} 0, & x < x_{i-1}, \\ -(2/h_i^3)(x - x_{i-1})^2(x - x_i - h_i/2), & x_{i-1} \le x < x_i, \\ (2/h_{i+1}^3)(x - x_i + h_{i+1}/2)(x - x_{i+1})^2, & x_i \le x < x_{i+1}, \\ 0, & x \ge x_{i+1}. \end{cases} \tag{3.23}$$

and

$$\Psi_i(x) = \begin{cases} 0, & x < x_{i-1}, \\ (h_i^2)^{-1}(x - x_{i-1})^2(x - x_i), & x_{i-1} \le x < x_i, \\ (h_{i+1}^2)^{-1}(x - x_i)(x - x_{i+1})^2, & x_i \le x < x_{i+1}, \\ 0, & x \ge x_{i+1}. \end{cases} \tag{3.24}$$

For $i = 0$, only the first two definitions in Equations 3.23 and 3.24 apply. For $i = N$, only the last two definitions in Equations 3.23 and 3.24 apply. The two cardinal

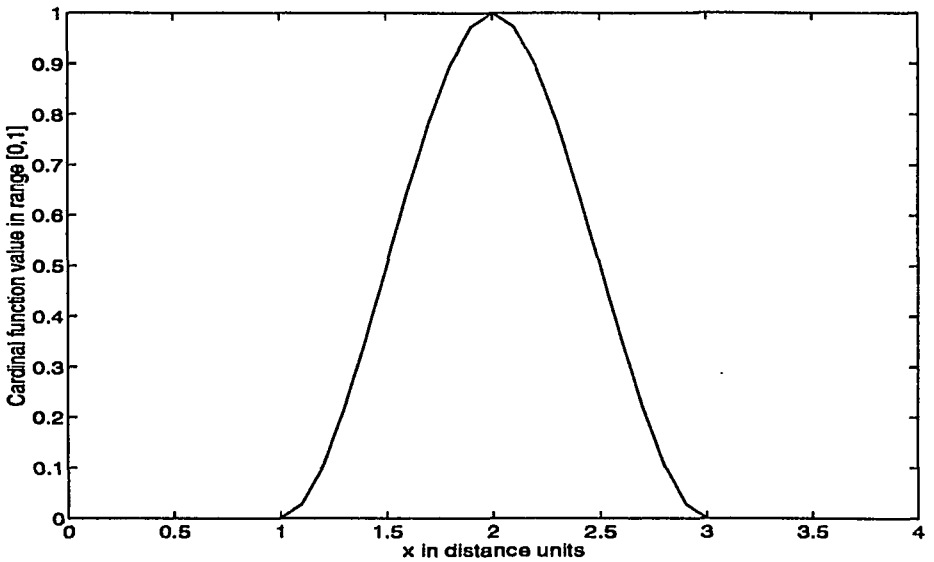

Figure 3.2: Cardinal function Φ_i for $x_i = 2$.

functions are illustrated in Figures 3.2 and 3.3 respectively. The piecewise cubic function given by Equation 3.21 has a single continuous derivative. Piecewise cubic polynomials that have two continuous derivatives are referred to as cubic splines and are discussed in the following section.

Cubic Splines

For the same knot sequence K, the cubic spline S(x) is a cubic polynomial in every interval $[k_{i-1}, k_i]$ $(i = 1, 2, \ldots, N)$, such that it has two continuous first and second derivatives at every interior knot. Thus $S(x)$ belongs to $\xi^2[a, b]$. To construct the cubic spline, one starts with Equation 3.21. This equation has $2N+2$ paramerers. The condition that S(x) takes the values $y_0, y_1, \ldots, y_N$ at $k_0, k_1, \ldots, k_N$ respectively applies $N + 1$ constraints. The slopes would supply the remain $N + 1$ constraints.

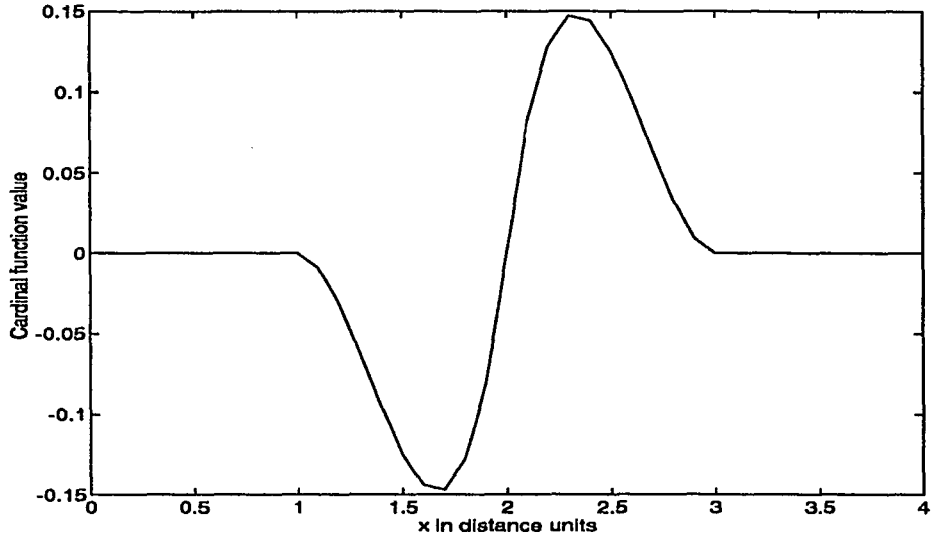

Figure 3.3: Cardinal function Ψ_i for $x_i = 2$.

But the constraint that the second derivative of S(x), S"(x), be continous at all the interior nodes facilitates expressing the slopes explicitly in terms of the functional values. Thus the spline can be determined based on just the functional values, using Equation 3.21. Let $\{k_0, k_1, \ldots, k_N\}$, $\{y_0, y_1, \ldots, y_n\}$, and $\{m_0, m_1, \ldots, m_N\}$ be the knots, functional values at the knots, and the corresponding slopes respectively. It is assumed that the $x_i = k_i$ for $i = 1, 2, \ldots, N$. The second derivative of S(x) at an interior knot x_j approaching from the left and right are given by

$$S"(x_k^-) = \lim_{x \to x_k^-} S"(x) \quad and \quad S"(x_k^+) = \lim_{x \to x_k^+} S"(x)$$

respectively. But by the definition of cubic splines, the second derivative approaching from the left and right should be the same. Thus,

$$S"(x_k^-) \quad - S"(x_k^+) = 0. \tag{3.25}$$

Combining this equation with Equation 3.21,

$$\sum_{i=0}^{N} y_i[\Phi''(x_k^-) - \Phi''(x_k^+)] + \sum_{i=0}^{N} m_i[\Psi''(x_k^-) - \Psi''(x_k^+)] = 0.$$

Due to the small support of the cardinal functions, the summations only have three terms each (refer Figures 3.2 and 3.3). The above equation reduces to

$$\sum_{i=k-1}^{k+1} y_i[\Phi''(x_k^-) - \Phi''(x_k^+)] + \sum_{i=k-1}^{k+1} m_i[\Psi''(x_k^-) - \Psi''(x_k^+)] = 0.$$

This is solved using equations 3.23 and 3.24. It is found that the $N+1$ slopes satisfy the $N-1$ equations

$$\frac{1}{h_k} m_{k-1} + 2\left(\frac{1}{h_k} + \frac{1}{h_{k+1}}\right) m_k + \frac{1}{h_{k+1}} m_{k+1} = 3\frac{y_k - y_{k-1}}{h_k^2} + 3\frac{y_{k+1} - y_k}{h_{k+1}^2}, \tag{3.26}$$

for $k = 1, 2, \ldots, N-1$ and $h_k = x_{k+1} - x_k$. Two more conditions are needed for unique solution of the slopes. One way of specifying these will lead to the natural cubic spline, to be discussed later. A general way of expressing the two conditions are

$$\begin{aligned} 2m_0 + \mu_0 m_1 &= c_0 \quad \textit{and} \\ \lambda_N m_{N-1} + 2m_N &= c_N. \end{aligned} \tag{3.27}$$

The user picks μ_0, λ_N, c_0, and c_N. Establishing the notations

$$\begin{aligned} &\lambda_k = h_{k+1}/(h_k + h_{k+1}), && \mu_k = 1 - \lambda_k, \\ &c_k = 3\lambda_k[(y_k - y_{k-1})/h_k] + 3\mu_k[(y_{k+1} - y_k)/h_{k+1}], && k = 1, 2, \ldots, N-1, \end{aligned} \tag{3.28}$$

equations 3.26 and 3.27 can be written in the form

$$\begin{bmatrix} 2 & \mu_0 & 0 & \dots & 0 & 0 & 0 \\ \lambda_1 & 2 & \mu_1 & \dots & 0 & 0 & 0 \\ 0 & \lambda_2 & 2 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & \\ 0 & 0 & 0 & \dots & 2 & \mu_{N-2} & 0 \\ 0 & 0 & 0 & \dots & \lambda_{N-1} & 2 & \mu_{N-1} \\ 0 & 0 & 0 & \dots & 0 & \lambda_N & 2 \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ \vdots \\ m_{N-2} \\ m_{N-1} \\ m_N \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-2} \\ c_{N-1} \\ c_N \end{bmatrix}. \quad (3.29)$$

On setting the second derivatives equal to zero at the endpoints, the natural spline is obtained. Thus the curvatures are zero at the end points. Thus the natural cubic spline S(x) also satisfies the end conditions S"(x_0) = S"(x_N) = 0. The cardinal splines obtained from Equation 3.21 with the slopes obtained from Equation 3.29 have large support and hence is not computationally suitable. B-splines are non cardinal spline basis functions having minimal support.

B-splines

Similar to the tent functions for linear splines, one can can generate basis functions for cubic splines. They are called B-splines. The requirement that the cubic spline be twice differentiable requires the B-spline to have a support of four consecutive intervals.

Theorem (Lancaster and Salkauskas 1986): Let K be a knot sequence satisfying $k_0 < k_1 \dots < k_M$. For $j = 2, 3, \dots, M-2$, there exists a choice of non-zero ordinates f_{j-1}, f_j, f_{j+1} such that the natural cubic spline with knot sequence K that

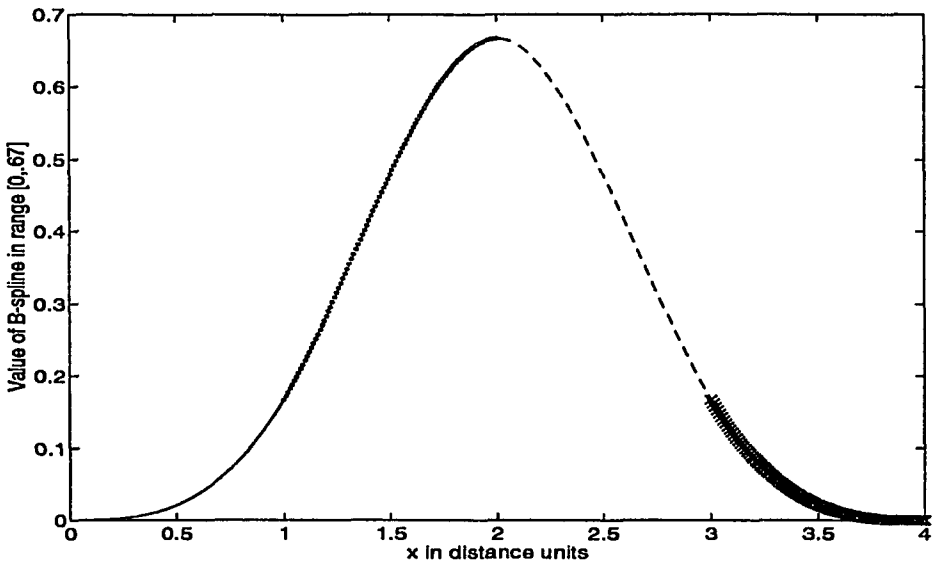

Figure 3.4: A B-spline.

satisfies

$$S(k_i) = \begin{cases} f_i, & i = j-1, j, j+1, \\ 0, & otherwise, \end{cases}$$

vanishes outside the interval (k_{j-2}, k_{j+2}) (and has zero slope at k_{j-2}, k_{j+2}). Such a natural cubic spline is termed B-spline and denoted by B_{j-2}. A B-spline that starts at knot k_0, and terminates at k_4 is denoted by B_0 and is illustrated in Figure 3.4. The B-spline consists of four polynomials joined together as shown in Figure 3.4.

The B-spline is obtained by setting $N = 4$ in Equation 3.29. Also from the definition, $m_0 = m_4 = f_0 = f_4 = 0$, and use

$$\mu_0 = 1, \quad c_0 = 3[(f_1 - f_0)/h_1], \quad \lambda_N + 1, \quad c_N = 3[(f_N - f_{N-1})/h_N].$$

Using these end conditions and Equations 3.27, 3.28, and 3.29, we obtain

$$\begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & 2 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 2 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 2 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ m_1 \\ m_2 \\ m_3 \\ 0 \end{bmatrix} = 3 \begin{bmatrix} -\frac{1}{h} & \frac{1}{h} & 0 & 0 & 0 \\ -\frac{1}{2h} & 0 & \frac{1}{2h} & 0 & 0 \\ 0 & -\frac{1}{2h} & 0 & \frac{1}{2h} & 0 \\ 0 & 0 & -\frac{1}{2h} & 0 & \frac{1}{2h} \\ 0 & 0 & 0 & -\frac{1}{h} & \frac{1}{h} \end{bmatrix} \begin{bmatrix} 0 \\ f_1 \\ f_2 \\ f_3 \\ 0 \end{bmatrix}. \quad (3.30)$$

The first and last rows of Equation 3.30 gives

$$m_1 = 3f_1/h, \quad m_3 = -3f_3/h. \quad (3.31)$$

The remaining three rows combine to give

$$m_1 = \frac{3}{14h}\left\{\frac{14}{4}f_2 + f_1 - f_3\right\}, m_3 = \frac{3}{14h}\left\{-\frac{14}{4}f_2 + f_1 - f_3\right\}. \quad (3.32)$$

From equations 3.31 and 3.32, we obtain

$$f_1 = 1/4f_2, \quad f_3 = 1/4f_2, \quad f_2 = arbitrary.$$

If the values are picked to be $f_2 = 2/3$, $f_1 = f_3 = 1/6$, the B-spline is said to be normalized. Then,

$$f_2 + 1/4f_2 + 1/4f_2 = 1.$$

This means that for the B-spline in Figure 3.4,

$$B_0(k_1) = 1/6, B_0(k_2) = 2/3, \quad and \quad B_0(k_3) = 1/6.$$

The B-spline that was obtained is only a basis function. But using this basis function, all the B-splines that are non-zero in the range $[k_0 = a, k_N = b]$ can be obtained. All the other B-splines can be constructed by shifting B_0 to the right knot. Thus if

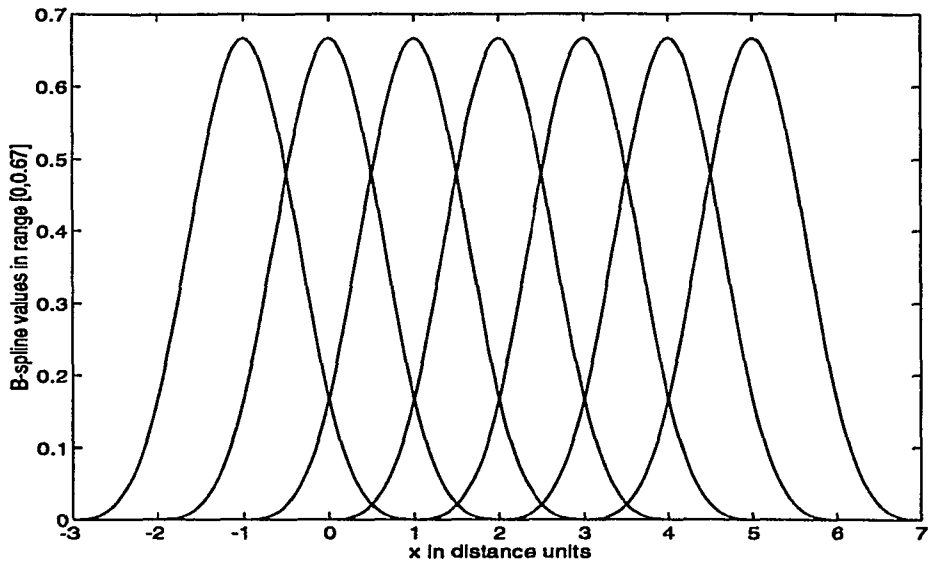

Figure 3.5: B-splines with support in [0,4].

B_0 is shifted one knot to the right, we obtain B_1. This is provided that all the knots are equally spaced. Some of the key points to be noted are listed below:

1. For the given knot sequence K = $\{k_0, k_1, \ldots, k_N\}$, the set of B-splines that have non-zero support in [a,b] are B_{-3},B_2,...,and B_{N-1}. Thus for N+1 knots, there are N+3 B-splines that have non-zero support. Figure 3.5 shows the B-splines $(B_{-3}, B_{-2}, \ldots, B_1, B_2, B_3)$ with non-zero support for N=4.

2. At any point in [a,b], there are at most four B-splines that have non-zero values. This is due to the fact that any point falls in the support of a maximum of four splines. For any point x in $[k_i, k_{i+1}]$, the non-zero B-splines are $B_{i-3}, B_{i-2}, B_{i-1}$, and B_i.

3. Further, since the B-splines are normalized, the sum of the values of the B-splines

at this point sum to unity. This is given by,

$$\sum_{i=-3}^{N-1} B_i(x) = 1.$$

4. The cubic spline for the knot sequence K can be written as

$$S(x) = \sum_{i=-3}^{N-1} a_i B_i(x) \tag{3.33}$$

The a_i's are the spline coefficents and they uniquely determine the function being approximated.

5. Cubic B-splines are of order four. The support of a B-spline of order n is n+1 knots. The nth order B-spline, whose support is $[k_i, k_{i+n}]$, is denoted by $B_{i,n}$.

6. The most efficient way of computing of B-splines of order n is to start from a B-spline of order one, and use the recurrence relation

$$B_{i,n}(x) = \frac{x - k_i}{k_{i+n-1} - k_i} B_{i,n-1}(x) + \frac{k_{i+n} - x}{k_{i+n} - k_{i+1}} B_{i+1,n-1}(x). \tag{3.34}$$

Here $i = -3, \ldots, N-1$ and $n = 1, 2, 3, 4$ Thus B-splines of order two are obtained from B-splines of order one. It is also clear from Equation 3.34 that the B-splines are a function of the knots alone. The definition of the B-spline of order one is given by

$$B_{i,1} = \begin{cases} 0, & x < k_i, \\ 1, & k_i \leq x < k_{i+1}, \quad i = -3, -2, \ldots, N+3, \\ 0, & k_{i+1} \leq x. \end{cases} \tag{3.35}$$

Note: When the order is not specified, the splines are assumed to be cubic splines. $B_i(x)$ is a cubic spline.

Constructing an interpolating cubic spline

The objective is to construct an interpolating cubic spline S(x) given

$$S(k_j) = y_j, \quad j = 0, 1, \ldots, N$$

. The cubic spline has B-splines as basis functions. Thus from Equation 3.33

$$S(x) = \sum_{i=-3}^{N-1} a_i B_i(k_j) = y_j, \quad j = 0, 1, \ldots, N. \tag{3.36}$$

The values of $B_i(k_j)$ can be computed using Equation 3.34. It is to be noted that for the jth Equation the non-zero B-splines are B_{j-3}, B_{j-2}, and B_{j-1}. In Equation 3.36, there are $N + 1$ equations and $N + 3$ unknown coefficients. The remaining two conditions are obtained from the end conditions, $S"(k_0) = S"(k_N) = 0$. From Equation 3.33

$$S(k_0) = a_{-3}B_{-3}(k_0) + a_{-2}B_{-2}(k_0) + a_{-1}B_{-1}(k_0)$$

and

$$S(k_N) = a_{N-3}B_{N-3}(k_N) + a_{N-2}B_{N-2}(k_N) + a_{N-1}B_{N-1}(k_N).$$

Then the last two equation required for solving 3.36 are

$$S"(k_0) = a_{-3}B_{-3}"(k_0) + a_{-2}B_{-2}"(k_0) + a_{-1}B_{-1}"(k_0) = 0, \tag{3.37}$$

$$S"(k_N) = a_{N-3}B_{N-3}"(k_N) + a_{N_2}B_{N-2}"(k_N) + a_{N-1}B_{N-1}"(k_N) = 0.$$

The required second derivatives are given by (Lancaster and Salkauskas 1986)

$$B_{-3}"(k_0) = \frac{6}{(k_1 - k_{-1})(k_1 - k_{-2})},$$

$$
\begin{aligned}
B_{-2}''(k_0) &= \frac{-6}{k_1 - k_{-1}} \left\{ \frac{1}{k_2 - k_{-1}} + \frac{1}{k_1 - k_{-2}} \right\}, \\
B_{N-1}''(k_0) &= \frac{6}{(k_1 - k_{-1})(k_2 - k_{-1})}, \\
B_{N-3}''(k_N) &= \frac{6}{(k_{N+1} - k_{N-1})(k_{N+1} - k_{N-2})}, \\
B_{N-2}''(k_N) &= \frac{6}{k_{N+1} - k_{N-1}} \left\{ \frac{1}{k_{N+2} - k_{N-1}} + \frac{1}{k_{N+1} - k_{N-2}} \right\}, \\
B_{N-1}''(k_N) &= \frac{6}{(k_{N+1} - k_{N-1})(k_{N+2} - k_{N-1})}
\end{aligned}
\tag{3.38}
$$

Six arbitrary nodes $k_{-3}, k_{-2}, k_{-1}, k_{N+1}, k_{N+2}, k_{N+3}$ have to be appended, three each to either end of the knot sequence. These knots have to satisfy $k_{-3} < k_{-2} < \ldots < k_{N+3}$. The coefficients of Equation 3.36 can also be determined by method of least squares.

CHAPTER 4. MODELING WITH SPLINE NETWORKS

Chapter 3 discussed modeling of one-dimensional functions using splines. The next step is to merge spline techniques with the proposed neural network, given by Equation 2.12, in Chapter 2. Equation 2.9 gives the neural network model for the one-dimensional approximation of a single input - single output system. Figure 2.5 gives the schematic for the corresponding neural network architecture. Equation 3.33 gives the expression for the cubic B-spline approximation, S(x), of a one-dimensional function. The similarities between Equations 2.9 and 3.33 are striking. In the former equation, the a_i's are the connection weights from the the output of the ith neuron to the neuron in the output layer. In the latter equation, the a_i's are the B-spline coefficients. The $f_i(x)$'s in Equation 2.9 are sigmoidal functions in comparison to the B-spline basis functions in Equation 3.33.

One-dimensional Spline Network

Figure 4.1 shows the architecture of this network. The activation functions of the neurons in the hidden layer are the B-spline basis functions. The B-spline basis functions are obtained as solutions of Equation 3.34. The connection weights from the input layer neuron to the hidden layer neurons are unity. The connection weights from the hidden layer neurons to the the output layer are obtained as solutions of

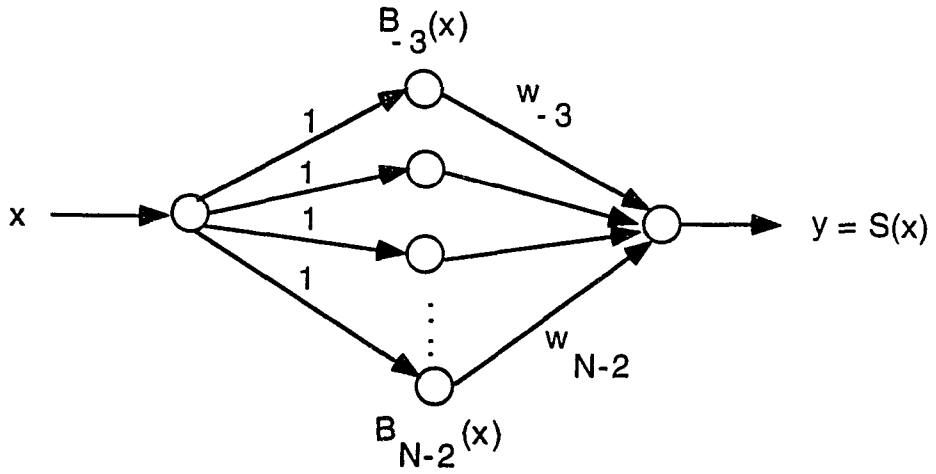

Figure 4.1: Single input- single output spline network.

Equations 3.33 and 3.37. The neuron in the output layer peforms the summation operation. The number of neurons in the hidden layer are a function of the number of knots, and are picked by the user. The number of nodes in the hidden layer are picked by trial and error, whereas the number of knots in the B-spline are picked by the user. Picking the knots could be made a rather simple process. Yet obtaining the weights for the neural network is no trivial process. The neural network could even fail to converge. On the other hand, the B-spline coefficients can be obtained by solving linear Equations 3.33 and 3.37. Thus the spline network retains non-linear characteristics of the neural network but does not possess its ambiguities.

Design of the network

The design involves picking the number of neurons in the hidden layer, the basis functions, and the connection weights. Let there be N data points with abscissas and

ordinates given by $\{x_0, x_1, \ldots, x_{N-1}\}$ and $\{y_0, y_1, \ldots, y_{N-1}\}$ respectively. The range of the abscissa is given by $[a = x_0, b = x_{N-1}]$. The knot sequence $\{k_0 = a, k_1, \ldots, k_{N-1} = b\}$ is also picked with the first and last knots of the sequence corresponding to the first and last abscissas respectively. The interior knots have to be in ascending order and do not have to coincide with the abscissas. The knots do not have to be equally spaced.

The first step is to pick the basis functions. All the basis functions that have finite support in [a,b] have to be included. If the order of B-splines is n (degree = $n-1$), then the number of B-spline basis functions is

$$K = N + n - 2. \tag{4.1}$$

Thus, K is the number of nodes in the hidden layer. Extra knots have to be provided to the knot sequence so that all the B-splines have support within this knot range. The updated knot sequence is given by $\{k_{-n+1}, k_{-n+2}, \ldots, k_N, \ldots, k_{N+n-2}\}$. The B-spline basis functions that have support in [a,b] are given by $\{B_{-n+1}, B_{-n+2}, \ldots, B_{N-2}\}$. The B-spline B_{-n+1} starts at the knot k_{-n+1} and has a range $[k_{-n+1}, k_1]$. Thus it has five knots in its support. A B-spline of order n is computed using the recurrence formula

$$B_{i,n}(x) = \frac{x - k_i}{k_{i+n-1} - k_i} B_{i,n-1}(x) + \frac{k_{i+n} - x}{k_{i+n} - k_{i+1}} B_{i+1,n-1}(x). \tag{4.2}$$

The nature of the B-spline is going to solely depend on the positioning of the knots. If all the knots are equally spaced, all the B-splines are shifted versions of one B-spline.

The next step is to determine the weights (w_i) of the network. It is to be noted

that the weights are the coefficients in the equation

$$S(x) = \sum_{i=-3}^{N-2} w_i B_i(x) \tag{4.3}$$

There are K weights and N data points, and cubic B-splines are selected for basis functions. Substituting the N data points into this equation, we obtain N linear equations with $(N + n - 2)$ unknowns. The $n - 2$ additional equations required for solving for the weights are obtained by using the end conditions. For a cubic B-spline, which is used most in this research, the two equations are given by

$$S"(k_0) = a_{-3}B_{-3}"(k_0) + a_{-2}B_{-2}"(k_0) + a_{-1}B_{-1}"(k_0) = 0,$$
$$S"(k_N) = a_{N-3}B_{N-3}"(k_N) + a_{N_2}B_{N-2}"(k_N) + a_{N-1}B_{N-1}"(k_N) = 0. \tag{4.4}$$

All the notations have been explained in Chapter 3. Thus, obtaining the weights involves solving linear equations, which is fairly straight forward. Closer the knots, better the approximation. The number of knots picked in the range [a,b] should be the same as the number of data points used. To keep the method simple, one could use uniformly spaced knots corresponding to the abscissas.

Illustration of one-dimensional spline network with different activation functions

{1,2,3,4,5,6,7} and {12,14.2,16.6,20.5,26,31,37.4} are the absissas and ordinates respectively for a given set of data points. There are N = 7 data points. The knot sequence is chosen to be {-2,-1,0,1,2,...,7,8,9,10}. Cubic B-splines are used as activation functions. Thus there are $(K = N + n - 2 = 9)$ nine neurons in the hidden layer. The cubic B-spline activation function that has support [1,5] is given in Figure

4.2. The result of approximation with the spline network using cubic spline activation functions is given in Figure 4.3. The root mean square error over ten points not used in the interpolation is given by

$$E = \sqrt{\sum_{j=1}^{10} (\sum_{i=-2}^{6} w_i B_i(x_j) - y_j)^2} = 0.02^o C. \tag{4.5}$$

The result of using linear spline activation functions is given in Figure 4.4. Here the order of the B-spline is two. The knot sequence for this case is {0,1,2,...,7,8}. The root mean square error over the same ten points is

$$E = 0.1466^o C.$$

Figure 4.5 gives the result of with fewer data points, and cubic spline activation functions. In this case the abscissas are {1,3,5,7}. The root mean square error over the same ten points is E = $0.4268^o C$. Figure 4.6 gives the result for linear spline activation functions for the updated data set. The root mean square error is E= $0.4531^o C$. The technique of selecting the number of hidden layer neurons (knots) will be to start with two neurons, and keep adding neurons until the required accuracy is obtained.

Higher Dimensional Spline Networks

Three-dimensional spline basis functions take the place of one-dimensional spline basis functions, when it comes to surface approximation. As the name suggests surface splines are three dimensional stuctures. Interpolation with surface splines are computationally very expensive. Surface interpolation by tensor product method, which is computationally less intensive, is used for the spline network. This technique

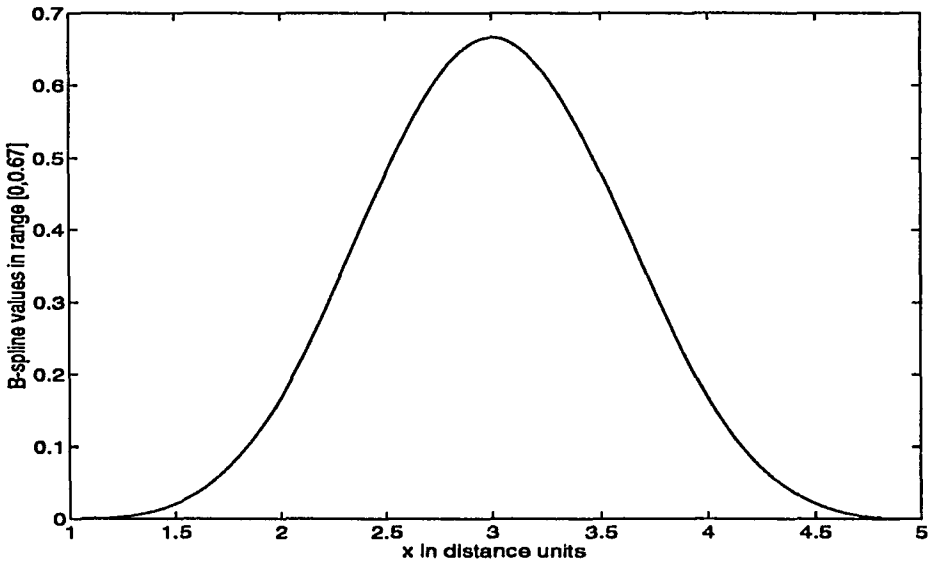

Figure 4.2: B-spline (B_1).

is both accurate and computationally attractive. Further, this technique can be easily extended to higher dimensional approximations. The technique for surface approximation is explained below. Higher dimensional approximation follow the same rules.

The data points are assumed to lie on a rectangular lattice in the xy plane and functional values, z = f(x,y), are assigned to each point. Let there be M points in the x-direction and N points in the y-direction. The set of data points are given by $\{(x_1,y_1),(x_2,y_2),\ldots,(x_M,y_N)\}$. The goal of the approximation is to determine the surface, S(x,y), that satisfies

$$S(x_i,y_j) = z_{ij}, \tag{4.6}$$

for $i = 1,2,\ldots,M$, and $j = 1,2,\ldots,N$. The rectangular lattice is said to be uniform if the x_i's and the y_i's are uniformly spaced. The surface spline S(x,y) is defined to

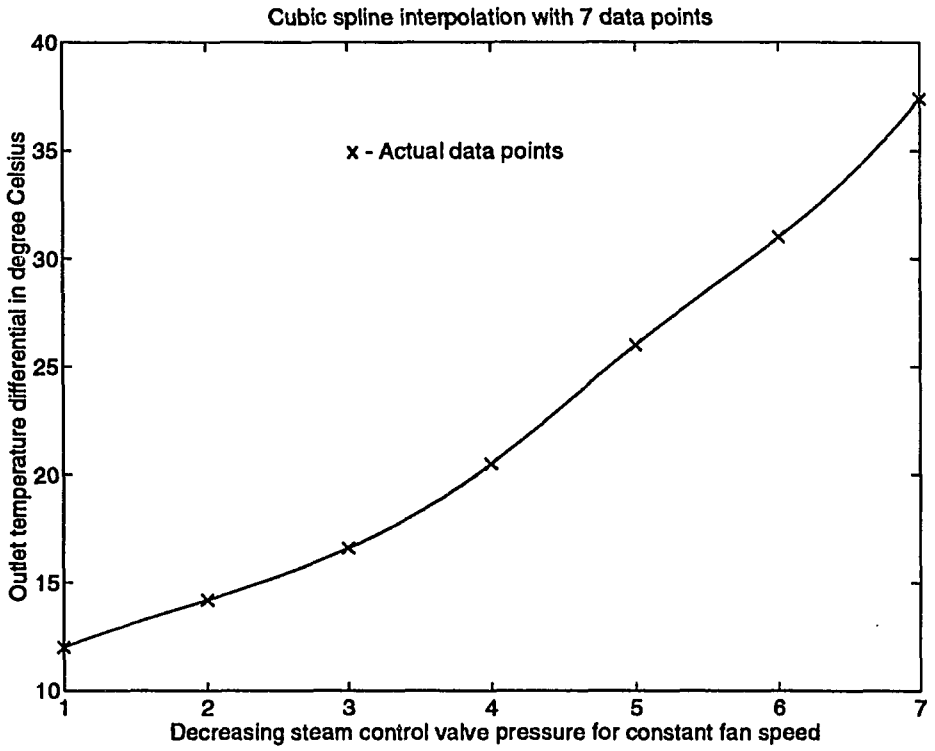

Figure 4.3: Spline network approximation. Fan speed = 350 rpm, steam control valve pressure decreases from 6 psi to 4.5 psi in steps of 0.25 psi.

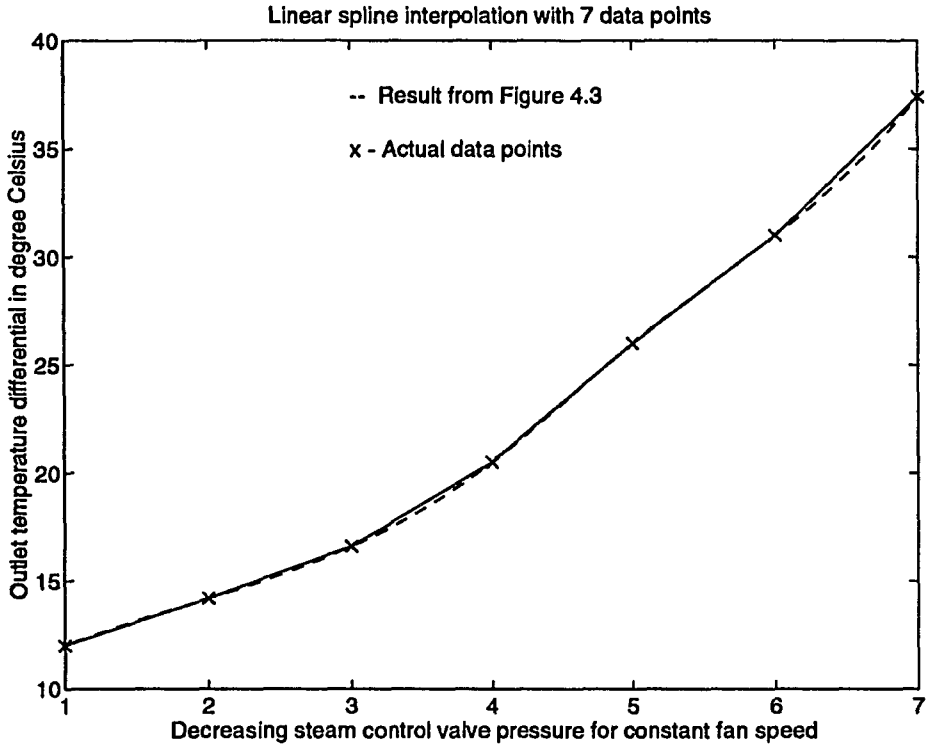

Figure 4.4: Approximation with linear spline activation functions. Fan speed = 350 rpm, steam control valve pressure decreases from 6 psi to 4.5 psi in steps of 0.25 psi.

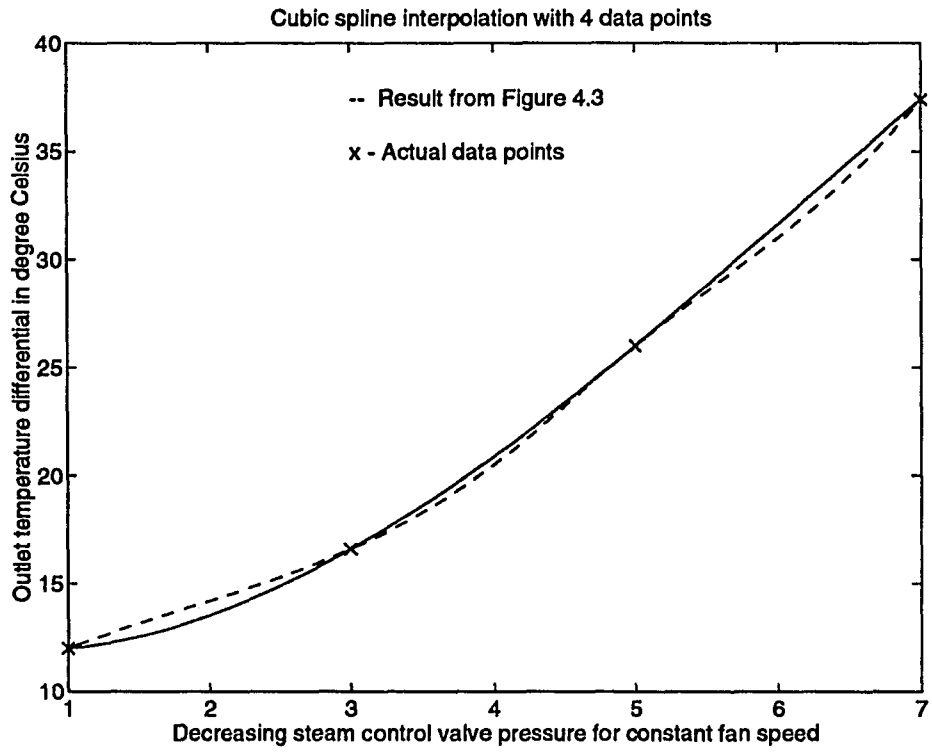

Figure 4.5: Approximation with fewer data points. Fan speed = 350 rpm, steam control valve pressure decreases from 6 psi to 4.5 psi in steps of 0.25 psi.

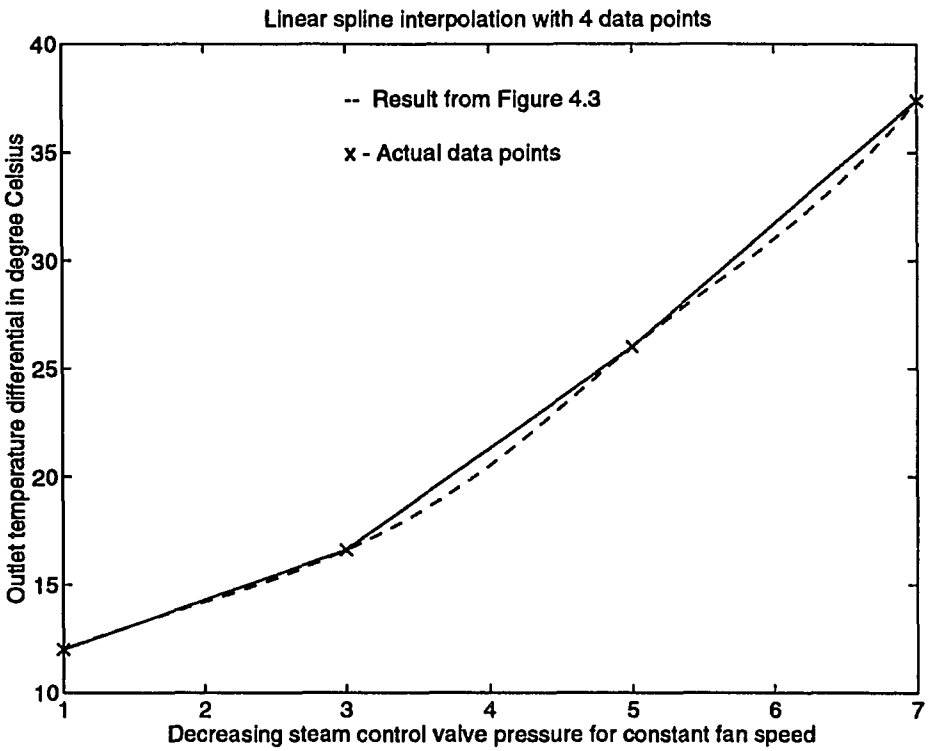

Figure 4.6: Linear spline approximation with fewer points. Fan speed = 350 rpm, steam control valve pressure decreases from 6 psi to 4.5 psi in steps of 0.25 psi.

be

$$S(x,y) = \sum_{j=1}^{N} \sum_{i=1}^{M} w_{ij} B_{ij}(x,y) \tag{4.7}$$

The basis functions $B_{ij}(x,y)$ are obtained using the tensor product method. By this method,

$$B_{ij}(x,y) = B_i(x) B_j(y), \begin{cases} i = 1,2,\ldots,M, \\ j = 1,2,\ldots,N. \end{cases} \tag{4.8}$$

Here $B_i's$ are the one-dimensional basis functions defined on the x-line. B_i and B_j are B-splines of order m and n respectively. Then, the support of B_{ij} is the rectangular lattice defined by $[x_i, y_j]$ x $[x_{i+m}, y_{j+n}]$. Figure 4.7 illustrates the basis function obtained by the tensor product method. This bicubic spline has support [1,1] x [5,5]. It has been proved that there exists a unique set of weights $\{w_{ij}\}$ that satisfy Equation 4.7. The technique used for calculating the weights is as described. One dimensional spline curves are fit to each row of the data space. The set of B-spline basis functions are the same. But there are MxN coefficients or weights. The next step is to fit one-dimensional spline curves in the column direction. The actual data points are used, but the coefficents obtained in the row-wise interpolation are used as ordinates. The MxN coefficients obtained in this manner, constitute the MxN weights. One could have started with column-wise interpolation with little or no effect on the final result. This technique can be extended to any number of dimensions. The interpolation along the rows is denoted by

$$S_1(x,y) = \sum_{i=1}^{M} \sum_{j=1}^{N} a_{i,j} B_j(x) \tag{4.9}$$

The MxN coefficients ($a_{i,j}$'s) are obtained by solving the MxN equations

$$S_1(x_i, y_j) = z_{ij} \begin{cases} i = 1, 2, \ldots, M, \\ j = 1, 2, \ldots, N. \end{cases} \tag{4.10}$$

The interpolation along the columns is denoted by

$$S_2(x_i, y_j) = \sum_{i=1}^{M} \sum_{j=1}^{N} w_{i,j} B_i(x) \tag{4.11}$$

The MxN coefficients ($w_{i,j}$'s) are obtained by solving the MxN equations

$$S_2(x_i, y_j) = a_{ij} \begin{cases} i = 1, 2, \ldots, M, \\ j = 1, 2, \ldots, N. \end{cases} \tag{4.12}$$

The schematic of a two input/two output spline network is given in Figure 4.8. The weights from the input layer neurons to the hidden layer neurons are unity. It is to be noted that the hidden layer is two-dimensional. It would be n-dimensional for a system with n inputs. There are MxN neurons in the hidden layer. The activation functions for the hidden layer neurons are given by Equation 4.8. The connection weights from the hidden neurons to the output neurons are obtained as solutions to Equation 4.12. Thus the entire two input/single output spline network has been designed.

Modeling the HVAC System

The schematic of the HVAC system is given in Figure 1.1. The control variables are fan speed (air flow rate), water flow rate, and steam pressure. The controlled variable is the outlet air temperature. It was observed that the air temperature differential is a function of the control variables and fairly independent of inlet air

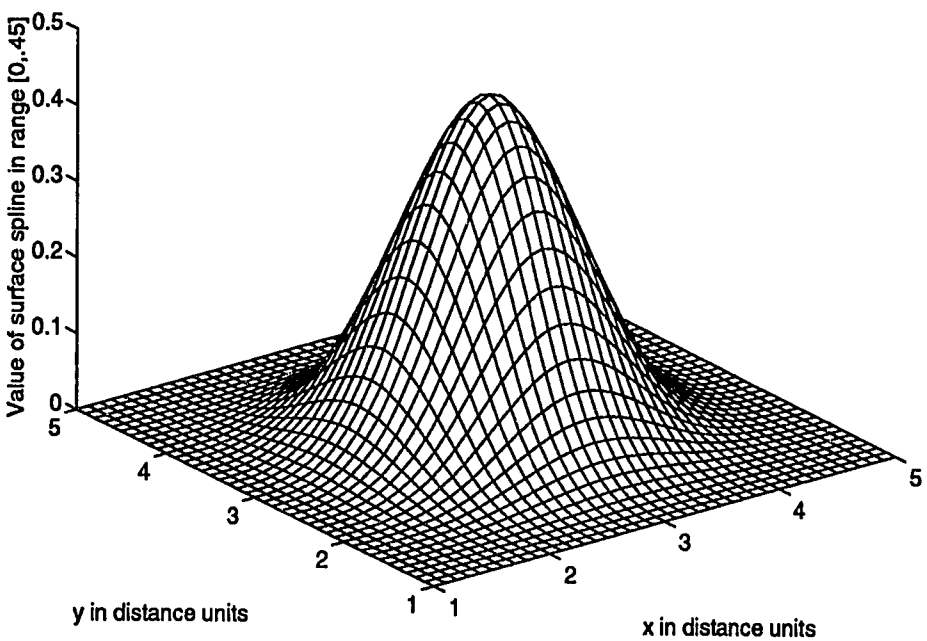

Figure 4.7: Surface B-spline basis function.

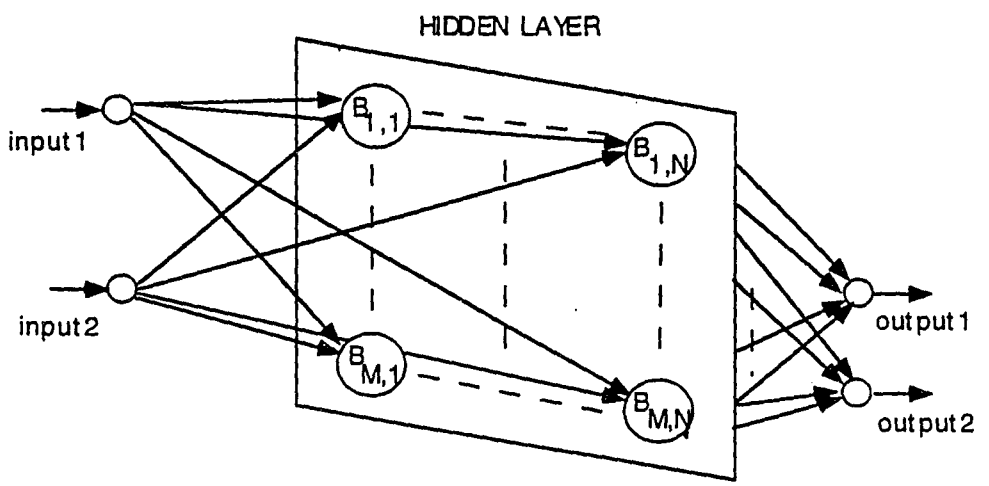

Figure 4.8: Two input - two output spline network.

temperature. The outlet air temperature differerential is, for this reason, taken to be the controlled variable. Thus the schematic of the model for the HVAC can be simplified to the form in Figure 4.9. The objective of the model is to provide the air temperature differential, given the three control variables. The air temperature differential is the difference in temperatures between the incoming and outgoing air streams. Normally the system variables would be picked to facilitate the generation of the required temperature differential. For the HVAC system, the air flow rate, water flow rate, and the steam pressure are controlled by the supply fan, water pump, and the pneumatic control valve respectively. The fan speed varies from 350 rpm to 750 rpm. The pressure on the control valve varies from 4.5 psi to 6 psi. The water pump was set to have three discrete speeds: low, medium and high. A spline network is used to model the system. Since the water pump has three discrete settings, three spline networks were used. Each one of the networks correspond to one of the settings of the water pump. Thus, the networks have fan speed and steam pressure as inputs, and air temperature differential as the output. The schematic of such a network is given in Figure 4.10. The dimension of the hidden layer is 8 x 6. The columns correspond to the steam pressure. The network is designed using the steps described in the previous section. The activation functions for the hidden layer neurons are given by

$$B_{ij}(x,y) = B_i(x)B_j(y), \begin{cases} i = 1,2,\ldots,8, \\ j = 1,2,\ldots,6. \end{cases} \tag{4.13}$$

$B_i(x)$ and $B_j(x)$ are both cubic B-splines. Such an activation function is called a bicubic spline. Figure 4.11 gives the output temperature differential surface to be approximated. Figure 4.12 gives the temperature differential surface approximated

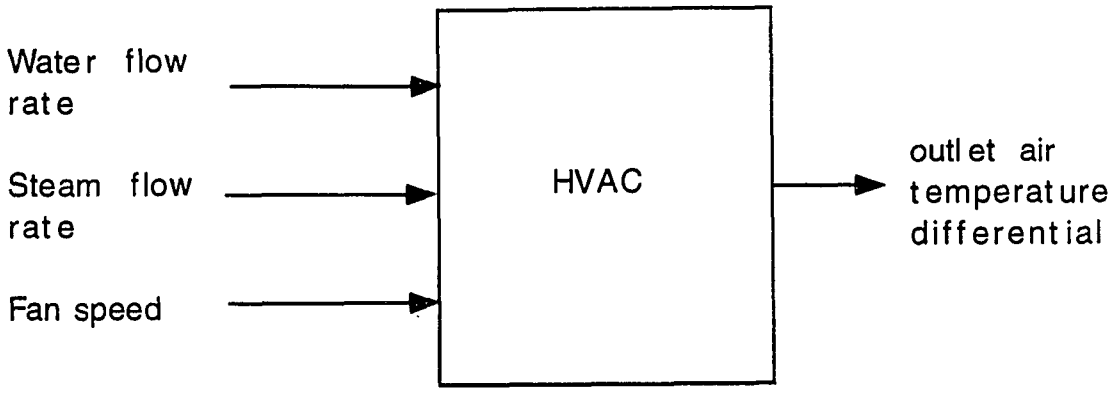

Figure 4.9: Black box model of the HVAC system.

by the network in Figure 4.10 for water pump setting on high. This network had 8 x 6 neurons in the hidden layer. Figures 4.13 gives the approximation error. The error is higher where knots had a larger separation. The approximation error is negligible over a test sample of 56 operating conditions, and is given in Table 4.1. The error is larger when linear splines are used for interpolation along the rows, and cubic splines along the columns. Such activation functions are called linear-cubic. In this case, $B_i(x)$ is cubic, and $B_j(x)$ is linear. The error is larger in this case as illustrated in Table 4.1.

To aid in fault diagnosis, the hot water to air heat exchanger, and the steam to water heat exchanger are also modeled using spline networks. The schematics of the black box models for these are given in Figures 4.14 and 4.15 respectively. The model of the steam to hot water heat exchanger gives the water temperature differential for certain steam pressure, and water flow rate. A two input-single output

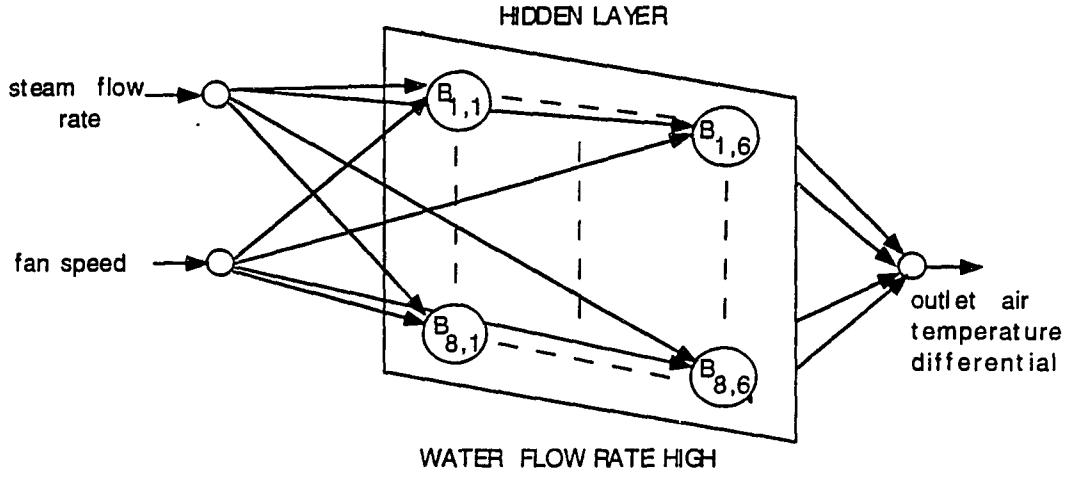

Figure 4.10: Spline network model for the HVAC sytem.

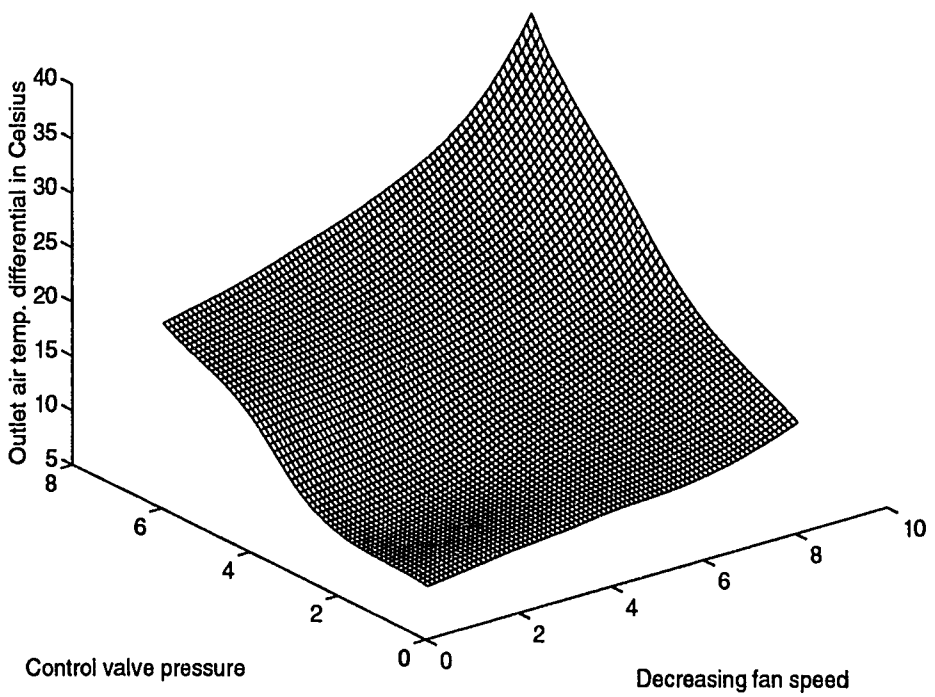

Figure 4.11: Surface to be approximated (water pump speed - high). Steam control valve pressure decreases from 6 psi to 4.5 psi in steps of .25 psi. Fan speed decreases from 750 rpm to 350 rpm in steps of 50 rpm

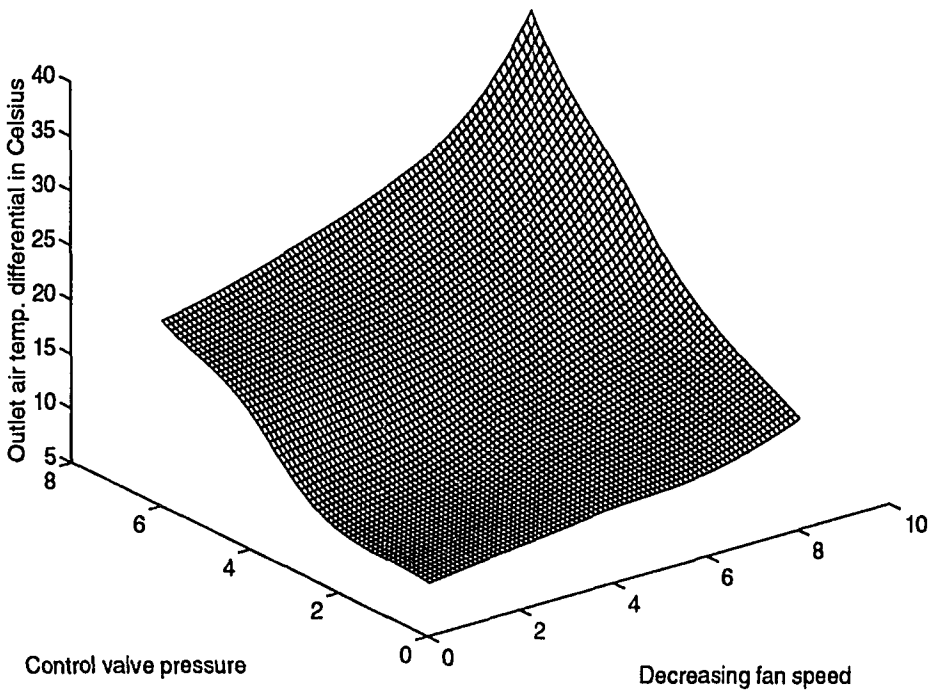

Figure 4.12: Surface approximated by the network for water pump speed - high. Steam control valve pressure decreases from 6 psi to 4.5 psi in steps of .25 psi. Fan speed decreases from 750 rpm to 350 rpm in steps of 50 rpm

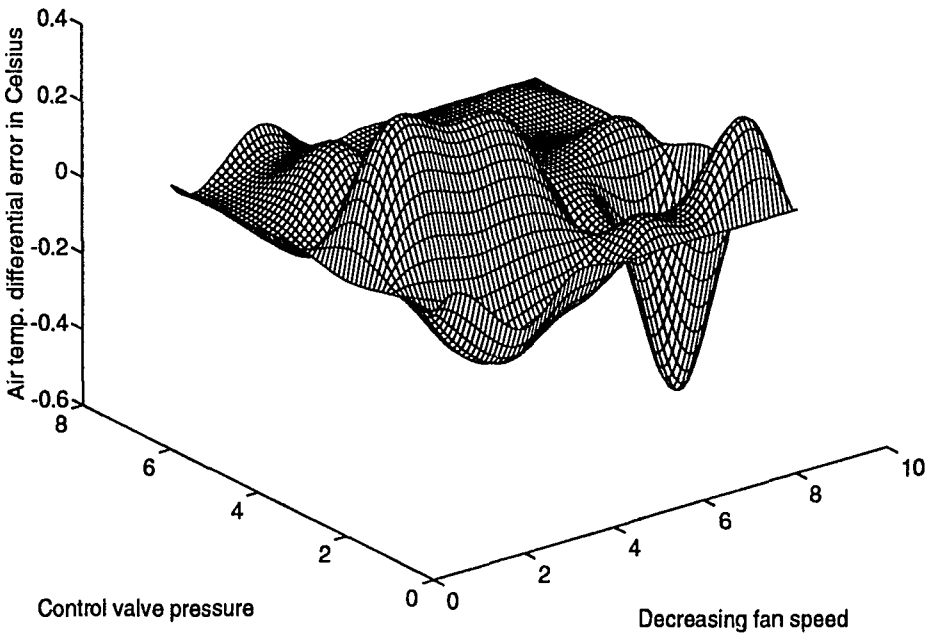

Figure 4.13: Error surface. Steam control valve pressure decreases from 6 psi to 4.5 psi in steps of .25 psi. Fan speed decreases from 750 rpm to 350 rpm in steps of 50 rpm

Table 4.1: Summary of results.

MODELING TECHNIQUE	NUMBER OF TEST POINTS	ROOT MEAN SQUARE ERROR	ACTIVATION FUNCTION
Spline network with 6x8 hidden layer	56	0.1976	bicubic
Least squares method	56	0.1978	bicubic
Spline network with 6x8 hidden layer	56	0.21	linear-cubic
Spline network with 5x7 hidden layer	56	0.4940	bicubic
Spline network with 5x7 hidden layer	56	0.6896	linear-cubic

spline network was used to model this unit. Steam flow and water flow rates are the inputs. The hidden layer had 7 x 5 neurons. The modeled surface is given in Figure 4.16. Three two input-single output networks are used to model the hot water to air heat exchanger. The three networks are for the three different waterpump settings. The hidden layer for each one of the networks had 8 x 6 neurons.

Table 4.1 summarizes the results. The spline networks have done a commendable job in replicating the working of the HVAC system. As the table shows, the error increases as the dimension of the hidden layer decreases. This is understandable. The least squares method of determining the coefficients would work better when the data being approximated is not very smooth. Using basis functions that are of degree less than three could decrease the accuracy. Picking the right dimension

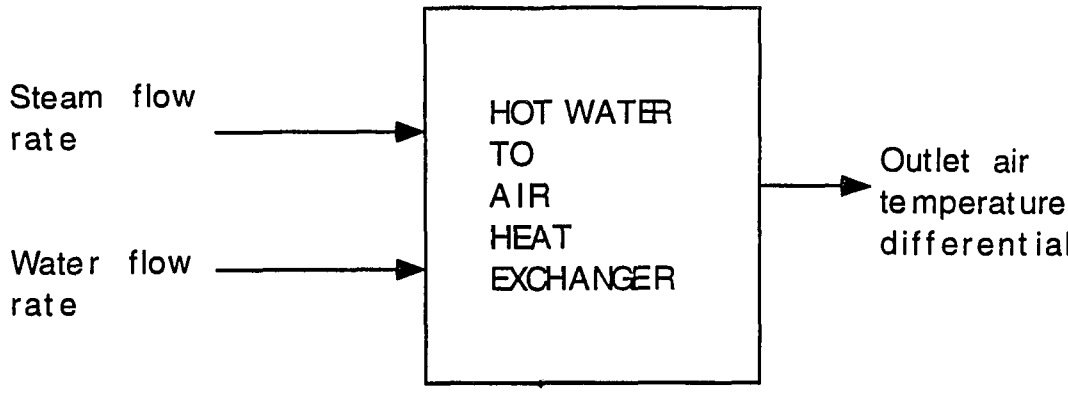

Figure 4.14: Black box model of the hot water to air heat exchanger.

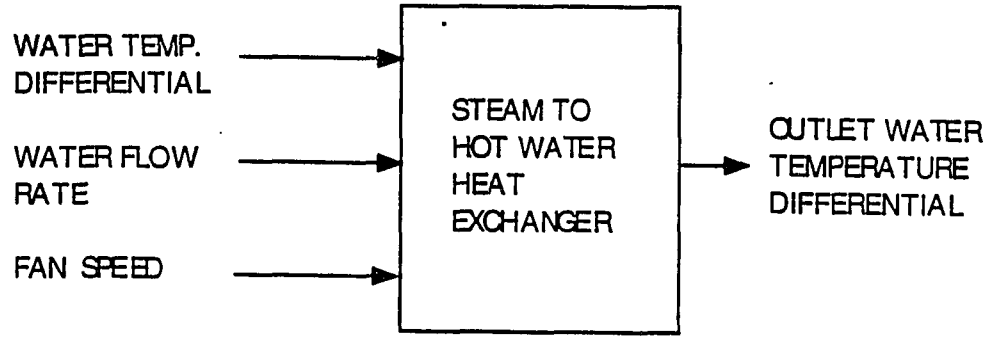

Figure 4.15: Black box model of the steam to hot water to heat exchanger.

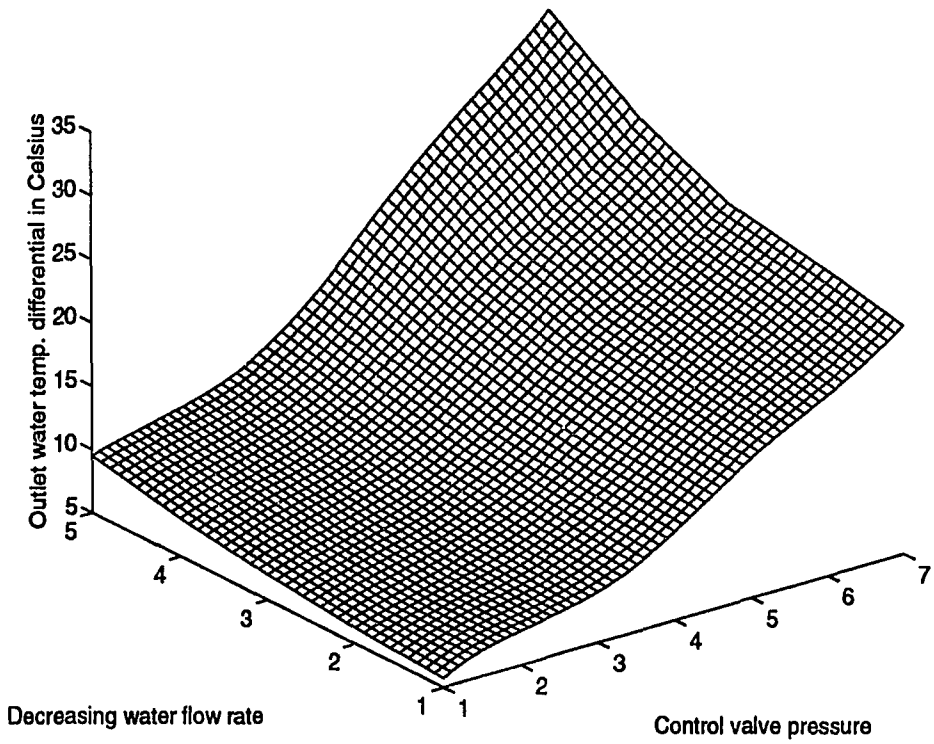

Figure 4.16: Water temperature differential surface for the steam to hot water heat exchanger. Steam control valve pressure decreases from 6 psi to 4.5 psi in steps of .25 psi. Water flow rate is high, medium-high, medium, medium-low, and low for 1, 2, 3, 4, and 5 respectively.

is not critical. One quick and dirty approach is to start with the end point and a middle point in each dimension. Neurons can then be added until the necessary accuracy is obtained. Training the network is extremely straightforward. There are no endless training phases. Convergence is always guaranteed. If functions with numerous transitions are being modeled, one may have to pick more knots around regions of high activity. Gradient methods could be used to detect such regions of rapid transitions. The functions that govern the operation of HVAC system are fairly monotonic. So the knots could be uniformly spaced. Thus, it has been proved that spline networks are an excellent alternative to artificial neural networks for functional approximation applications, that involve fairly smooth functions.

Spline Networks versus Artificial Neural Networks

A comparative study is given below that compares spline networks with neural networks. The objective is to show how the spline networks have improved on the disadvanatges of neural networks, that were discussed in Chapter 2.

1. Size of input/output data: The number of input/output pairs available for modeling the surface in Figure 4.12 were 56. The input vector and the output vectors were two-dimensional and one-dimensional respectively. The neural network failed to converge for the given input/output set. Numerous combinations of layers, and number of neurons per layer were tried. The neural network requires a much larger input/output training set. The size of the input/output set for neural networks is rather ambiguous. Spline networks do not have such convergence problems. If fewer data points are used, the modeling error will be higher. To make a comparative study, one-dimesional data from Figure 4.3 was used. A neural network with six neurons

in the hidden layer was used. The hidden layer neurons had sigmoidal activation functions, and the ouput neuron had linear activation function.

2. Training time: Training spline networks are much faster than training neural networks. Figure 4.17 shows the result of training one-dimensional data using both splines and neural networks. The spline network took 1529 floating point operations (flops) and a cpu time of 0.533 seconds. On the other hand, the neural network took a cpu time of 115 seconds and 3,102,456 flops. The surface spline approximation in Figure 4.12 took a cpu time of 1.65 seconds and 109,074 flops. It took 3,104 training cycles before the neural network converged. The numerous hours spent in identfying the required neural network is not included.

3. Spline networks do not have any convergence problems.

4. Neural networks involve solving complex non-linear equations, whereas spline networks involve soving linear equations.

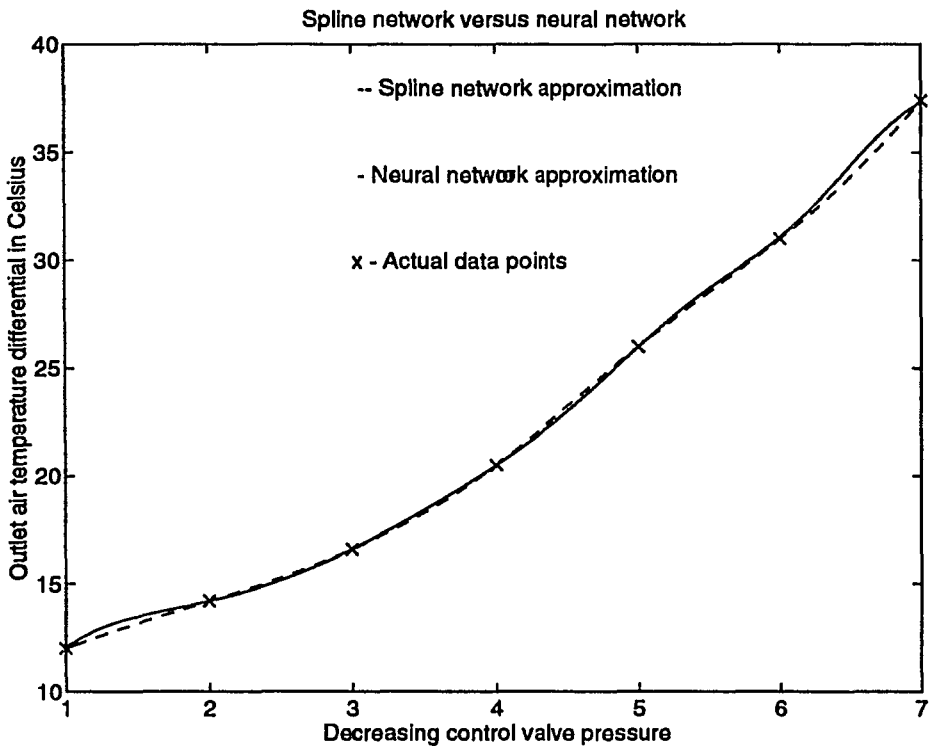

Figure 4.17: One-dimensional spline network approximation versus neural network approximation. Fan speed = 350 rpm, steam control valve pressure decreases from 6 psi to 4.5 psi in steps of 0.25 psi.

CHAPTER 5. FAULT DETECTION USING THE ARTIFICIAL INTELLIGENCE APPROACH

To classify faults in systems, one can take several approaches. There could exist a good working mathematical model of the system, and the parameters of this model could be measurable or obtained through estimation. If this mathematical model encompasses all the behavioral patterns of the system, then using the mathematical model would be the best approach. But most real-life systems cannot be completely represented by mathematical equations. Another technique is to teach a neural network known fault conditions and the factors that are responsible. This is basically a pattern recognition problem. The neural network is trained to classify error signals, as belonging to a certain class of faults. Neural networks are trained on numerical data. The input to the neural network could consist of various error signals. The output layer has as many neurons as there are error classes. Thus each one of the neurons correspond to a certain class. When the output of a neuron is one, it means that the network has identified the error class corresponding to that neuron. These networks assign an error to only one class, though in reality it could belong to more than one class.

These two techniques discussed use mathematical models, and numerical data respectively. But there exists information that cannot be stated mathematically, or

in the form of numerical input/output pairs. This information exists in the minds of technicians and people who run the systems. This is the kind of knowledge that is obtained over period of time from hands on experience. There should be some way of tapping in to the wealth of knowledge stored in these minds. Fuzzy logic techniques help us to exploit what one may call common-sense rules. These are often referred to as linguistic rules in fuzzy logic. For the HVAC system under consideration, only an error in water flow rate could affect the water temperature differential as well as the air temperature differential. Thus, if the technician finds out that both the differentials have departed from expected values, then he would infer that the water pump is at fault. " IF the air temperature differential error is significant AND the water temperature error is significant THEN the water pump is faulty", is an example of a linguistic rule. Fuzzy logic techniques allow us to incorporate all these linguistic rules so as to provide mapping from input to output variables.

Fuzzy Logic System

The basic components of a fuzzy logic system are the fuzzifier, inference engine, and the defuzzifier (Lee 1990). This is illustrated in Figure 5.1. The fuzzifier assigns input variables membership values to different fuzzy sets. The universe of discourse of each variable is divided into linguistic overlapping sets. The number of these sets is decided by the user. Figure 5.2 gives the universe of discourse of both the input variables, A and B, to be {0,4}. The universe of discourse of the output variable, C, is {0,40}. The universe of discourse is divided into low(L), medium(M), and high(H) regions. This is facilitated using membership functions. In Figure 5.2, the membership functions are triangular. The membership functions could take other

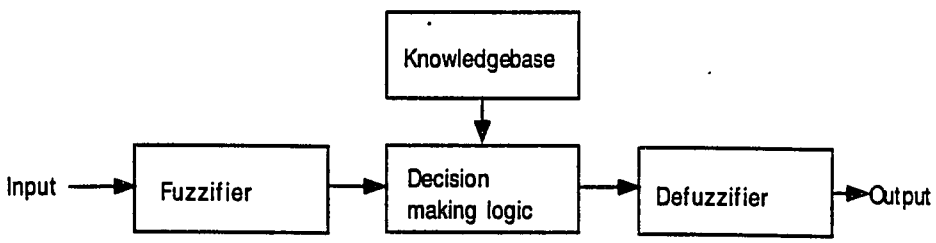

Figure 5.1: Fuzzy logic system.

shapes. The overlapping nature of the membership functions makes it possible to assign the input variables to more than one fuzzy set. In Figure 5.2, if the value of the input variable A is 1.2, it belongs to the fuzzy sets low and medium. The membership values to the different fuzzy sets are given by

$$\begin{aligned} \mu_L(A = 1.2) &= 0.8 \\ \mu_M(A = 1.2) &= 0.2 \\ \mu_H(A = 1.2) &= 0. \end{aligned} \tag{5.1}$$

The membership values give the degree to which input belongs to each one of the fuzzy sets. Fuzziness measures the degree to which an event occurs. The underlying philosophy in fuzzy logic is that membership to any class is fuzzy. This implies that one can only tell the degree to which an element belongs to a certain class.

The inference engine consists of the knowledge base and the fuzzy reasoning

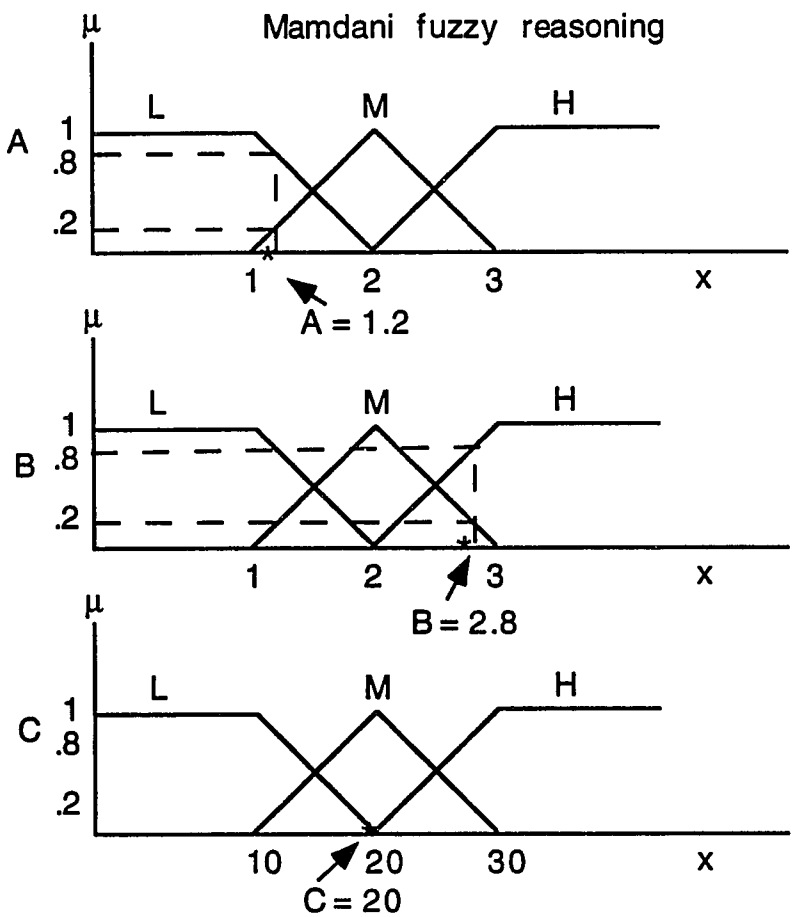

Figure 5.2: Universe of discourse for the input variables.

scheme. The knowledge base consists of all the "IF - THEN" rules. These rules linguistically relate the input variables to the output variables. All of these rules are summarized into a fuzzy association matrix (FAM). These rules are based on common-sense rules, and other available information on the nature of the system. The FAM that relates the input variables, A and B, to C are given in Figure 5.3. The first row of this matrix reads:

IF A is low AND B is low THEN C is low,

IF A is low AND B is medium THEN C is low, and

IF A is low AND B is high THEN C is medium.

Usually only the elements that correspond to significant FAM rules are filled. The element in the second row and second column of Figure 5.3 is not filled, indicating that this particular situation does not arise. The fuzzy reasoning scheme could use the Mamdani fuzzy reasoning scheme or the Mizumoto's fuzzy reasoning scheme. In either scheme, the objective is to assign membership values to the different output fuzzy sets. The output fuzzy sets are low, medium, and high. The first step is to determine all the rules from the FAM that are applicable for the input: A=1.2, B=0.8. A has membership in the low and medium fuzzy sets, whereas B has membership in the medium and high fuzzy sets. Thus the fuzzy rules that apply are,

IF A is low AND B is high THEN C is medium, and

IF A is medium AND B is high THEN C is high.

IF A is low and AND B is medium THEN C is low.

The fourth rule that correpsonds to A and B belonging to medium fuzzy sets is not significant. Then according to the Mamdani scheme (Lee 1990), the membership

		B		
		L	M	H
A	L	L	L	M
	M	L	-	H
	H	H	H	H

Figure 5.3: Fuzzy association matrix(FAM).

values to the three sets are,

$$\begin{aligned} \mu_L(C) &= min(\mu_L(A), \mu_M(B)) \\ &= min(0.2, 0.2) \\ &= 0.2 \\ \mu_M(C) &= min(\mu_L(A), \mu_H(B)) \\ &= min(0.8, 0.8) \\ &= 0.8 \\ \mu_H(C) &= min(\mu_M(A), \mu_H(B)) \\ &= min(0.2, 0.8) \\ &= 0.2 \end{aligned} \tag{5.2}$$

The Mizumoto reasoning scheme (Lee 1990) uses the product of membership functions instead of the minimum.

The final unit in the fuzzy logic system is the defuzzifier. This unit converts

the fuzzy membership values of the output to a crisp number. There are several defuzzification techniues. One of the more popular techniques is the center of gravity method. In this technique the output is given by

$$\begin{aligned} C &= \frac{\sum c_i \mu_i}{\sum \mu_i} \\ &= \frac{c_L \mu_L(C) + c_M \mu_M(C) + c_H \mu_H(C)}{\mu_L(C) + \mu_M(C) + \mu_H(C)} \\ &= 20. \end{aligned} \tag{5.3}$$

Here, c_is are the centers of the fuzzy sets, and μ_is are the respective membership values. From Figure 5.2, $c_L = 10$, $c_M = 20$, and $c_H = 30$. In effect, numerical data has been mapped to numerical data. A neural network learns this mapping based on numerical input/output pairs. There has been lot of effort to combine the powers of neural networks and fuzzy logic. This has resulted in fuzzy-neural, neuro-fuzzy and many other innovative networks. A neural network that uses fuzzy concepts is discussed in the following section.

Fuzzy Neural Network

Neural networks are widely used in pattern classification problems. In such networks there are as many output neurons as there are pattern classes. When an input feature vector is presented to the network, only one of the output neurons is activated. The network is trained to assume that the pattern classes are distinct. But in reality, this is hardly the case. Two or more pattern classes could overlap. The neuron in the output layer that is activated the most is turned on. All the others are simultaneously turned off. It could happen that the output of two neurons are 0.5 and 0.495. The neuron reading 0.5 is turned on, thus denoting belongingness to a certain

pattern. On being presented with an almost identical feature, the output of the same neurons could be 0.495 and 0.5 respectively. Thus nearly identical features may be grouped into different classes. This kind of ambiguity arises due to the belonginess of features to different classes. This reminds one of fuzzy logic and the degree of belonginess to different sets. Thus, fuzzifying the input feature space and the output pattern class space could be the solution to deal with belongingness to more than one class. This is the underlying principle of the fuzzy neural networks discussed in this section (Pal and Mitra 1992).

In this fuzzy neural network based pattern classification, a neural network is the heart of the system. Let the number of variables in the feature vector (F) be m, the number of pattern classes be C, and the number of training samples be N. The first step is to fuzzify the input feature vector. Each one of the features in F is fuzzified. The universe of discourse of the features are fuzzified into various sets. Let these sets be low, medium, and high. Thus if one starts with the m-dimensional feature vector $\mathbf{F}_i = [F_{i1}, F_{i2}, \ldots, F_{im}]$, on fuzzification we obtain the 3m-dimensional vector

$$\mathbf{F}_i = [\mu_L(F_{i1}), \mu_M(F_{i1}), \mu_H(F_{i1}), \ldots, \mu_L(F_{im}), \mu_M(F_{im}), \mu_H(F_{im})]. \quad (5.4)$$

To fuzzify the input features, overlapping membership functions are used. The membership functions could be triangular, gaussian, or any other function. The universe of discourse for the jth feature (F_j) is determined by finding the range $[F_{jmin}, F_{jmax}]$. This range is then divided into three overlapping sets. Once the membership functions have been decided for all m features, the fuzzified feature vector given by Equation 5.4 is easily obtained. This fuzzification phase is illustratred in Figure 5.4.

The next step is assigning member functions to the output of the neural network. The output is of dimension C. For each pattern class, all the training samples that

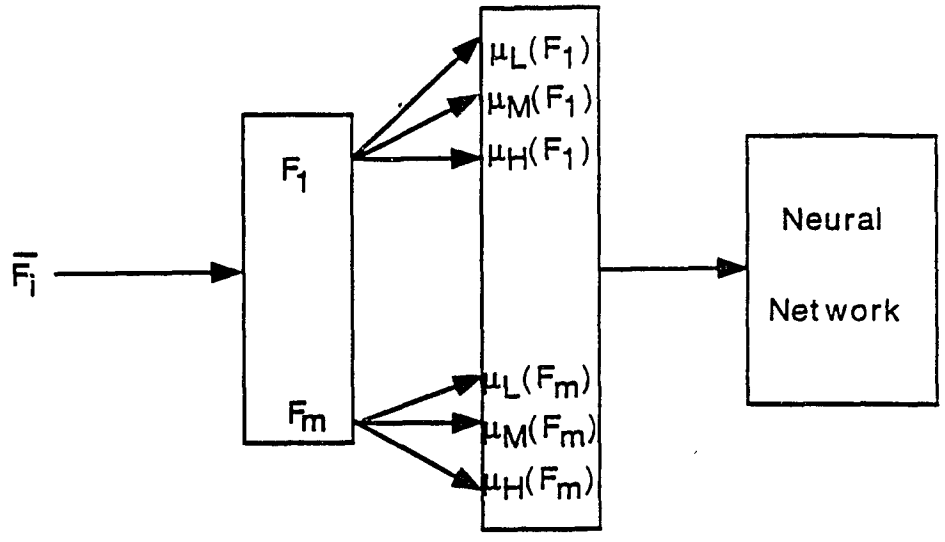

Figure 5.4: Input fuzzification phase.

have some degree of membership to that class are determined. The mean and variance vectors for that class are determined from these training samples. Consider the nth class. Let there be Q training vectors that have some degree of membership to this class. $\mathbf{O}_n$ and $\mathbf{V}_n$ are the mean and standard deviation vectors respectively for the nth class.

$$\mathbf{O}_n = [O_{n1}, O_{n2}, \ldots, O_{nm}] \ \ where, \qquad (5.5)$$
$$O_{k1} = \sum_{i=1}^{Q} \frac{F_{i1}}{Q}.$$

For each input feature vector, its distance from all the C classes are determined. The distance of $\mathbf{F}_i$ from the nth class is calculated using (Pal and Mitra1992)

$$d_{in} = \sqrt{\sum_{j=1}^{m} [(F_{ij} - O_{nj})/V_{nj}]^2}, \qquad (5.6)$$

for $n = 1, 2, \ldots, C$. The membership values of each input feature to any one of the C classes can now be determined. The membership of $\mathbf{F}_i$ to the nth class is given by

$$\mu_n(\mathbf{F}_i) = \frac{1}{1 + \left(\frac{d_{in}}{f_d}\right)^{fe}}, \tag{5.7}$$

for $n = 1, 2, \ldots, C$. The parameters f_d and f_e are constants that control the fuzziness. When the feature vector, $\mathbf{F}_i$, belongs just to the nth class,

$$\begin{aligned} d_{in} &= 0, \\ \mu_k(\mathbf{F}_i) &= 1. \end{aligned} \tag{5.8}$$

In the fuzziest case, the input feature vector will have some membership value to all the classes.

Once the output membership functions have been decided, the neural network can be trained using the backpropagation algorithm discussed in Chapter 2. The network is trained with input/output vector pairs. The input vector is $3m$-dimensional, and the output vector is C-dimensional. The output vector gives the membership values of the input vector to each one of the classes. This technique is used to aid in the classification of HVAC faults.

Fault Detection Scheme and Results

The objective of the fault detection scheme is to detect and locate fault utilizing the minimum hardware. The proposed detection scheme has four temperature sensors. These are the input and output air temperature sensors, and the input and output water temperature sensors. At the outset, the intent was to classify all kinds of faults. Though every possible effort was made to this extent, it was realized to

be a very steap goal. This was owing to factors such as lack of necessary hardware, and the inability to simulate many of the fault scenarios. The goals of the fault detection scheme were narrowed down to detection of faults in the control variables: steam pressure, water flow rate, and fan speed (air flow rate). This still would be a powerful fault diagnostic system.

The HVAC model described in Chapter 4 sounds the alarm when there is discrepancy between the system output air temperature differential and that of the model. The system has to deviate by more than one degree Celsius, before the alarm is sounded. This takes into account modeling error and measurement noise. The two measurements that are available are the air temperature differential and the water temperature differential. The objective is to determine the control variable that is responsible for the error. It needs to be mentioned that different levels of deviations in the control variables can produce almost identical errors in the two measurements. Using the spline-neural model for the steam to hot water exchanger, the error in the water temperature differential (E(δW)) can be calculated. The error in the output air temperature differential (E(δT)) is obtained using the model for the entire HVAC system. These error measurements were normalized by dividing these values by the respective expected temperature differentials. The normalized error variables ranged from zero to one. These normalized differentials seem to correlate better with the control variables, and are used to classify the error patterns.

Numerous error scenarios in the control varibles were simulated using the different models. The error in the output air temperature differential had to be greater than one degree Celsius. Further, it was assumed that the maximum error in the control variables could be half the step change. Thus, if the fan settings were in steps of

50 rpm, the maximum error would be 25 rpm. The error measurements belonged to three different classes. Overlap of classes was obvious. Thus, a fuzzy neural approach was justified.

The input feature vector to the fuzzy neural network consisted of the normalized air and water temperature differential errors. Each error is fuzzified into three sets: low, medium and high. Triangular membership functions given by,

$$\begin{aligned} \mu_L(x) &= \frac{x - x_1}{x_0 - x_1} \qquad x_0 \leq x \leq x_1 \qquad (5.9) \\ \mu_M(x) &= \frac{x - x_0}{x_1 - x_0} \qquad x_0 \leq x \leq x_1 \\ &= \frac{x - x_2}{x_1 - x_2} \qquad x_1 \leq x \leq x_2 \\ \mu_H(x) &= \frac{x - x_1}{x_2 - x_1} \qquad x_1 \leq x \leq x_2 \end{aligned}$$

(5.10)

are used to fuzzify the input feature vector. Here $x_0 = 1,\ x_1 = .5\ and x_2 = 1$. The membership functions are given in Figure 5.5.

For every input feature vector, the corresponding output fuzzy class vector is obtained using the technique in the section on fuzzy neural network. In equation 5.7, f_d and f_e were chosen to be one and three respectively. The classes for any input feature vector could take on values between zero and one. The overall schematic of the fault classification network is given in Figure 5.6. The neural network has two hidden layers with eight neurons each. This was arrived at after numerous trial and error runs. The input layer has six neurons corresponding to the fuzzified input feature vector. The output layer has three neurons corresponding to the three classes. It took 15,000 training cycles to bring the sum squared error below .01 as depicted in Figure 5.7. The learning rate was chosen to be 0.5. The network had sigmoidal

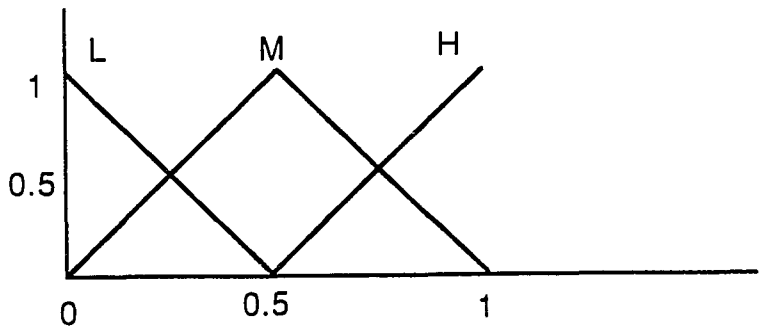

Figure 5.5: Triangular membership functions.

activation functions and was trained using back-propagation.

Fifty different simulated error samples were tested on the network. The error definitely belonged to a certain class if the membership function to that class was greater than 0.5. Otherwise, the class with the greater membership value was picked. Of the fifty samples, forty one were diagnosed correctly. Of these, thirty three had membership values greater than 0.5 to one of the classes. Figure 5.8 shows the feature space that was classified. The fault diagnostics system was successful in detecting the faults, and pinpointing the responsible control variable. Degradation in the heat exchangers may produce similar fault signatures. But those parts are less likely to fail.

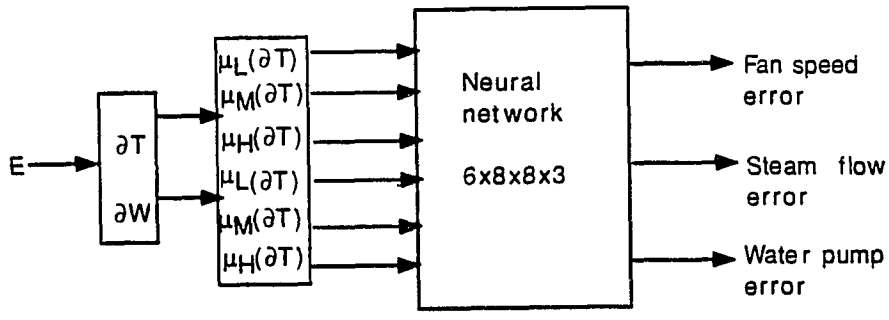

Figure 5.6: Fault classification network.

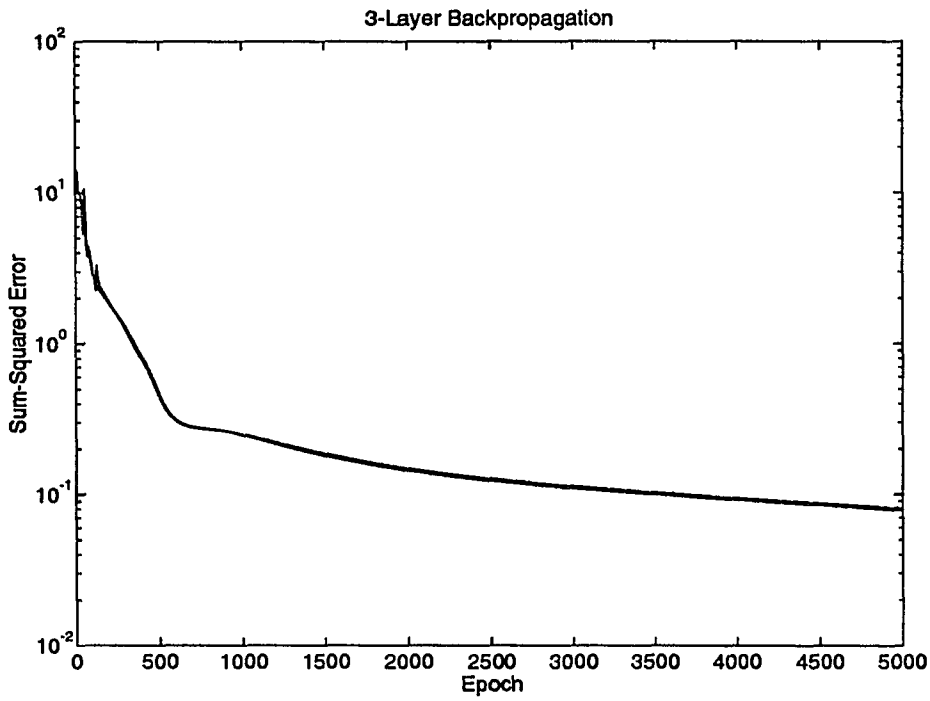

Figure 5.7: Training error.

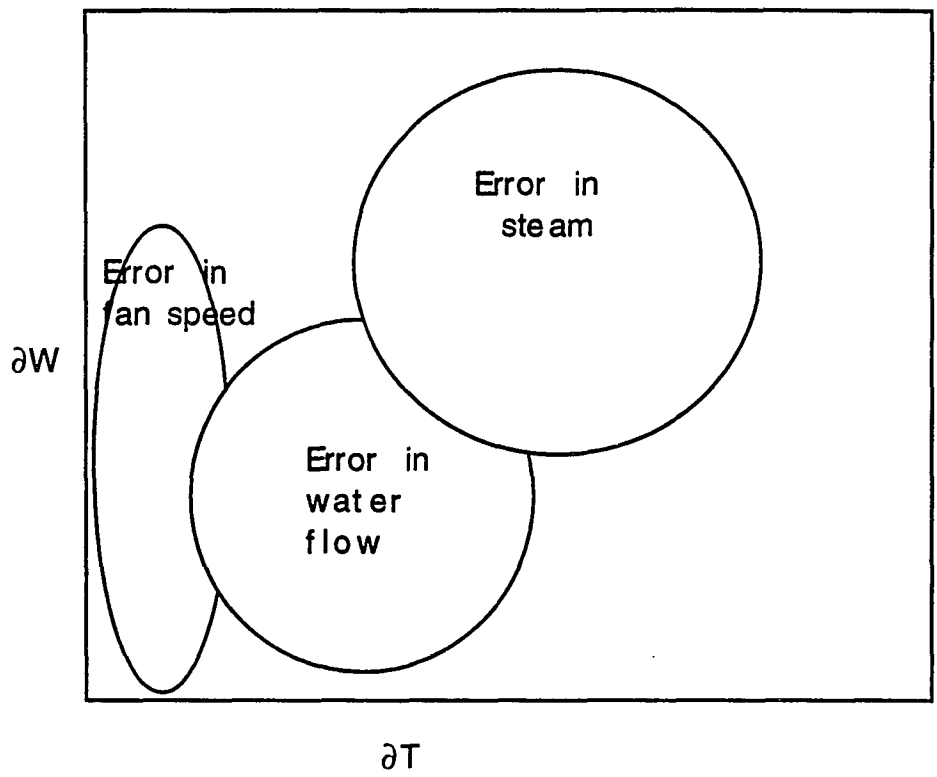

Figure 5.8: Feature space.

CHAPTER 6. CONCLUSION

The main goal of the research was to do a dissertation with practical applications. Working with a real-life HVAC system, has helped the author to understand the differences between working with real situations and simulated ones. The amount of time it takes to collect data is almost Hereculean. A dozen hours of collecting data would leave me with a handful of readings. The HVAC system is very sluggish. Like most practical systems, it is almost impossible for this system to replicate previous results. The number of times that readings had to be retaken was innumerable. The steam pressure could be fluctuating, or the fan speed was nowhere near where it should have been. After nine months of collecting data, it appeared that one could be doing this for ever. This leaves one with the impression that, it would be so much easier to simulate data. If the data generated from a model is used to generate another model, what do we stand to gain? Further, if a mathematical model exists, why spend endless time on a problem that has been solved?

The industry seems to be so far behind what people in the academia consider to be state of the art technology. Most manufactures of control systems rely on the old, but almost reliable PID controllers. Adaptive control, and robust control are buzz words that are left on the book shelves. A lot of effort has to be made to bridge the gap between the researcher sitting at the computer, and the engineer on the

plant floor. Industries are not solely to be blamed for their inertia towards change. Whereas a researcher may be excited with the degree of complexity of his work, the industry is more concerned with the practicality of his work.

The research objective was to do more than modeling with neural networks. Neural networks can definitely model HVAC systems. But the level of uncertainty when working with neural networks, was a major concern. To develop a network that is easier to train was the main thrust of this research. Chapter 2 does an excellent analysis of behind the scenes operation of a neural network. For a functional approximation problem, the neural network performs piecewise sigmoidal approximation. The network shifts the sigmoidal functions around in a manner, that can not be controlled by the user. The network does not have any apriori information regarding the nature of the function. Thus, it does not know how many hidden layers, or neurons per layer are required. Though interpolation with sigmoidal functions can result in highly smooth functions, they have large supports and are not very flexible. It was shown in Chapter 2 that some activation functions can perform better than others in certain situations. The neural networks are being severely handicapped when they are being limited to a single activation function. This led to the proposed network, where each neuron in the network was responsible for a certain region of the functional domain. Ideally, the neurons can take on any activation function. But this would make the modeling task computationally expensive, and extremely complex. Thus, the activation functions were picked to be lower order polynomials. The support of these activation functions are picked to be small enough to keep the error within tolerable limits.

The proposed spline network is a piecewise functional approximation. To make

it realizable, the network was reduced to a piecewise lower order polynomial approximation. Initially, the functional approximation was going to be continuous but not smooth. Here smoothness is used in the context of possessing higher order derivatives. At that time, the only splines the author was aware of were mechanical splines used in drafting. It was coincidence that broadened his horizon to spline interpolation. Using spline techniques made it possible to generate smoother approximations. This dissertation introduces Spline Network for functional approximations. The activation functions for this network are lower order polynomials called B-splines. B-splines could be linear, quadratic, or cubic. The support of these activation functions are picked by the user. This is definitely one of the greatest advantages that spline-neural networks have over conventional neural networks. In conventional neural networks, the support is picked during the long training phase. If the functions being approximated are monotonous, the activation functions for the spline-neural networks could be uniformly spaced. If there are lot of variations, one could cluster a lot of activation functions with smaller supports in the region of high activity. Unlike neural networks, this network does not have any convergence problems. The weights of this network are picked by solving linear equations. Spline-neural networks are extremely easy to train.

Chapter 4 illustrates the ability of spline networks to model the highly complex HVAC system. The HVAC system has been modeled with great accuracy. Three different models were generated. The overall system model is used to sound the alarm. It informs the user of discrepancy in the system operation. The model of the steam to hot water heat exchanger gives the difference in temperatures between the outgoing and incoming water streams. If there is error in water temperature

differential, it indicates a fault in steam flow, water flow, or the heat exchanger. The model of the hot water to air heat exchanger gives the air temperature differential for the actual system water temperature differential. This would indicate whether there actually is an error in this part of the system. A lot of refinements can be made to this model. The HVAC system did not have the facility to measure so many parameters. Further, it was not possible to model the different fault scenarios. Thus, due to hardware limitations, this model is not complete. The model can be made to include transient characteristics of the system. Spline networks can also be used to model non-linear dynamical sytems. This is definitely an area for future research. The working of neural networks and spline networks were compared in Chapter 4. For one-dimensional data, the spline neural network training was more than two hundred times faster than training the artificial neural network. For higher dimensional data, the time taken increases almost exponentially. The time taken to get the required neural network structure could be in the order of weeks. It was also found that there was not enough input/output data for the neural network to learn the system model. Spline networks have definitely proven their ability to model complex systems, that exhibit smooth functional characteristics.

The working model sounds the alarm when there is an intolerable error in the outlet air temperature. The next step is to detect the guilty system component. A powerful fault detection scheme should harness all the the information available, be it mathematical, numerical or linguistic. Since there are no mathematical models to work with, fuzzy neural networks are an ideal choice. Neural networks have established their identity as pattern classifiers. But traditional neural network pattern classifiers assign features to a single class. In reality, they may belong to more.

For this reason, the philosophy of fuzzy logic is exploited in fuzzy neural networks. The input features are trained to belong to more than one output class. Chapter 5 illustrates fuzzy-neural techniques in fault classification. Due to lack of time and necessary hardware, it was not possible to identify all the faults. Faults in steam pressure, fan speed, and water flow rate were classified with about eighty percent accuracy.

The objective of this research was not to attack conventional neural networks, but is to look at neural networks from a different perspective, in an effort to overcome some of the drawbacks of neural networks. The vision of a piecewise functional neural network led to the spline network. The spline network could be an excellent alternative to neural networks for certain functional approximation problems. The areas of functional approximation problems, where spline networks can do a better job is yet to be determined. But, it has been proved that for noise-free and slowly varying functions, spline networks could do a commendable job. For such functions, spline network training is faster and easier than training neural networks.

BIBLIOGRAPHY

[1] Anderson, D., L. Graves, W. Reinert, J. F. Kreider, J. Dow, and H. Wubenna,"A Quasi-Realtime Expert System for Commercial Building HVAC Diagnostics," *ASHRAE Transactions*, Vol. 95, No. 2, pp. 205-212, 1989.

[2] Basart, J. P., and M. S. Chackalackal, " *Advances in Image Processing*," Academic Press, New York, 1992.

[3] Bhat, N. V., P. A. Minderman, Jr., T. McAvoy, and N. S. Wang, "Modeling Chemical Process Systems via Neural Computation," *IEEE Control Systems Magazine*, pp. 24-29, April 1990.

[4] Cybenko, G., "Approximations by Superpositions of a Sigmoidal Function," *Mathmatics of Control, Signals, and Systems*, Vol. 2, No. 4, pp. 303-314, 1989.

[5] Haberl, J. S., and D. E. Claridge, "An Expert System for Building Energy Consumption Analysis: Prototype Results," *ASHRAE Transactions*, Vol. 93, No. 1, pp. 83-92, 1987.

[6] Hajnal, A., W. Maas, P. Pudlak, M. Szegedy, and G. Turan, "Threshold Circuits of Bounded Depth," *Proceedings of the 1987 IEEE Symposium on the Foundations of Computer Science*, pp. 99-110, 1987.

[7] Hornik, K., M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, pp. 359-366, 1989.

[8] Hush, D. R., and B. G. Horne, "Progress in Supervised Neural Networks," *IEEE Signal Processing Magazine*, pp. 8-37, January 1993.

[9] Isermann, R., "Process Fault Detection Based on Dynamic Models and Parametric Estimation Methods- A Survey," *Automatica*, Vol. 20, pp. 387-400, 1984.

[10] Isermann, R., " Process Fault Diagnosis Based on Dynamic Models and Parametric Esitmation Methods," *Fault Diagnosis in Dynamic Systems*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[11] Isubichi, H., R. Fujioka, and H. Tanaka, "Neural Networks that Learn from Fuzzy If-Then Rules," *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 2, pp. 850, 1993.

[12] Lancaster, P., and K. Salkauskas, " *Curve and Surface Fitting: an Introduction*," Academic Press, London, 1986.

[13] Lee, C. C., "Fuzzy Logic in Control Systems: Fuzzy Logic Controller- Parts I and II," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20, No. 2, pp. 404-435, 1990.

[14] Liu, S. T., and G. E. Kelly, "Rule-Based Diagnostic Method for HVAC Fault Detection," *Proceedings of Building Simulation*, pp. 405-413, 1989.

[15] Narendara, K. S., and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, pp. 4-27, 1990.

[16] Nguyen, D. H., and B. Widrow, "Neural Networks for Self-Learning Control Systems," *IEEE Control Systems Magazine*, pp. 18-23, April 1990.

[17] Norford, L. K., and R. D. Little, "Fault Detection and Load Monitoring In Ventilation Systems," *ASHRAE Transactions*, Vol. 92, pp. 590-602, 1992.

[18] Pal, S. K., and S. Mitra, "Multilayer Perceptrom, Fuzzy Sets, and Classification," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, 683-697, 1992.

[19] Pape, F. L. F., J. W. Mitchell, and W. A. Beckman, "Optimal Control and Fault Detection in Heating, Ventilating and Air-Conditioning Systems," *ASHRAE Transactions*, Vol. 91, No. 1, pp. 729-735, 1991.

[20] Patton, R., R. Clark, and P. Frank, "*Fault Diagnosis in Dynamic Systems*," Prentice Hall, Englewood Cliffs, NJ, 1989.

[21] Samad, T., " Neurocontrol: Concepts and Applications," *Advances in Instrumentation and Control*, Vol. 46, pp. 93-99, 1991.

[22] Takagi, H., N. Suzuki, T. Koda, and Y. Kojima, "Neural Networks Designed on Approximate Reasoning Architecture and Their Applications," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 752-760, 1992.

[23] Usoro, P. B., and I. C. Schick, "A Hierarchical Approach to HVAC System Fault Detection and Identification," *Proceedings of the Winter Annual Meeting of the*

Dynamic Systems and Control Division of the American Society of Mechanical Engineers, pp. 156-164, 1985.

[24] Wang, L. X., and Mendel, J. M., "Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least-Squares Learning," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 807-813, 1992.

[25] Willsky, A. S., "A Survey of Design Methods for Failure Detection in Dynamic Systems," *Automatica*, Vol. 12, pp. 601-611, 1976.