ASSESSMENT OF THE MODELING ABILITIES OF NEURAL NETWORKS

by

Alvin Ramsey

Submitted to the Department of
Mechanical Engineering in Partial Fulfillment of
the Requirements for the
Degree of

Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

January 1994

© Massachusetts Institute of Technology 1994
All rights reserved

Signature of Author ______________________
Department of Mechanical Engineering
January 1994

Certified by [illegible]
Professor George Chryssolouris
Thesis Supervisor

Accepted by ______________________
Professor Ain A. Sonin
Graduate Committee

Eng
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
MAR 08 1994
LIBRARIES

DEDICATION

To my mother (어 머 님)

and

to my brother (Bro, on my word,
we *shall* go fishing someday!).

ASSESSMENT OF THE MODELING ABILITIES OF NEURAL NETWORKS

by

Alvin Ramsey

Submitted to the Department of Mechanical Engineering
on January 1994 in partial fulfillment of the
requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

The treatment of manufacturing problems, whether in process control, process optimization, or system design and planning, can be helped by input-output models, namely, relationships between input and output variables. Artificial neural networks present an opportunity to "learn" empirically established relationships and apply them subsequently in order to solve a particular problem. In light of the increasing amount of applications of neural networks, the objective of this thesis is to evaluate the ability of neural networks to generate accurate models for manufacturing applications. Various neural network models has been tested on a number of "test bed" problems which represent the problems typically encountered in manufacturing processes and systems to assess the reliability of neural network models and to determine the efficacy of their modeling capabilities.

The first type of problem tested on neural networks is the presence of noise in experimental data. A method to estimate the confidence intervals of neural network models has been developed to assess their reliability, and the proposed method has succeeded for a number of the models of the test problems in estimating the reliability of the neural network models, and greater accuracy may be achieved with higher-order calculations of confidence intervals which would entail increased computational burden and a higher requirement of precision for the parametric values of the neural network model.

The second type of problem tested on neural networks is the high level of nonlinearity typically present in an input-output relationship due to the complex phenomena associated within the process or system. The relative efficacy of neural net modeling is evaluated by comparing results from the neural network models of the test bed problems with results from models generated by other common modeling methods: linear regression, the Group Method of Data Handling (GMDH), and the Multivariate Adaptive Regression Splines (MARS) method. The relative efficacy of neural networks has been concluded to be relatively equal to the empirical modeling methods of GMDH and MARS, but all these modeling methods are likely to give a more accurate model than linear regression.

Thesis Supervisor: Professor George Chryssolouris

Title: Associate Professor of Mechanical Engineering

ACKNOWLEDGMENTS

Many deserves thanks and recognition for helping me develop the work performed for the Laboratory for Manufacturing and Productivity at MIT. I would like to thank my advisor, Professor George Chryssolouris, for his guidance, support, and opportunity to work in his research group. I must also thank Dr. Moshin Lee, Milan Nedeljkovic, Chukwuemeka (Chux) Amobi, and German Soto for their contributions to the work presented in this thesis; they are among the "buddies" of the lab. I like to thank the rest of the buddies: Subramaniam (the original Buddy), Andrew Yablon, Andrew Heitner, Sean Arnold, Eric Hopkins, Wolfram Weber, Johannes Weis, Giovanni Perrone, Dirk Loeberman, Ulrich Hermann, Helen Azrin, Tina Chen, and Steve Marnock.

And I finally would like to thank my mother, Kun Cha Ramsey, and my brother, Dan Ramsey, for their continuous support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

A large variety of manufacturing processes are used in industrial practice for transforming material's shape, form, and properties. In the metalworking industry, there are more than 200 well established processes used for such purposes [Chryssolouris, 1992]. Modeling of manufacturing processes refer to the creation of the set of relationships that relate input to the process and its outputs. Such descriptions of manufacturing processes are very useful in terms of optimizing the process as well as in terms of controlling it. Very often in industry, the lack of an adequate process model leads to extensive trial and error experimentation, suboptimal processes, and waste of material, labor, and energy.

Manufacturing problems, whether in process control, process optimization, or system design and planning, can be solved only with the help of appropriate models, namely, some sort of a relationship between input and output variables. Such a relationship can be constructed either analytically, on the basis of analysis of the interactions between the input variables, or empirically, on the basis of experimental/historical data. For manufacturing problems, the analytical route is often difficult to take because the relevant input-output relationship may be a product of many complex and interacting phenomena. For this reason, over the past years, researchers active in the field of manufacturing have pursued a variety of empirical approaches to manufacturing modeling. The main drawback of these approaches is that they have relied on regression techniques, which require an *a priori* knowledge of the general algebraic form of the input-output relationship. While such knowledge may be available for particular manufacturing problems, it is very difficult to generalize to other problems.

The analytical route in solving manufacturing problems in process optimization, system design and planning, and process control is often difficult to take because the relevant input-output relationship may be a product of many complex and interacting phenomena. The advent of neural networks presents an opportunity to overcome this difficulty for manufacturing problems.

Modeling of many manufacturing processes often involves an extensive analysis of a physical behavior of the process to derive a mathematical model. Creating a precise and accurate model of the manufacturing process often results in a model which requires a sizeable amount of computation. Simplifying a model to reduce the computational size sacrifices the accuracy of the model. When developing a model for a manufacturing process, accuracy conflicts with workability. A modeling technique for manufacturing processes which can compromise the characteristics of an accurate and a manageable model is needed. A possible approach involves the use of artificial neural networks, which is a computational tool used in artificial intelligence.

There are two specific classes of problems in manufacturing where neural networks can be applied:

- The first class of problems involves process diagnostics, process control, and process optimization. Most control schemes for manufacturing processes use a single sensor to monitor a machine or a process. In many cases, this approach is inadequate primarily due to inaccurate sensor information and the lack of a single process model which can sufficiently reflect the complexity of the process.
- The second class of problems involves manufacturing system design and planning. The design of a manufacturing system can be viewed as

> the mapping of the system's performance requirements onto a description of a system which will achieve the required performance. Presently, there are only few, highly specialized methods which support such a mapping process, and due its the complexity, global optimization cannot be obtained effectively by trial-and-error methods. System designing normally compromise between quality of the solution and design effort.

Current applications include the development of neural network technology for manufacturing applications, particularly related to process diagnostics, control, and optimization, as well as to system planning and design. A new approach to process control and optimization is to synthesize the state variable estimates determined by the different sensors and corresponding process models using neural networks. For system design and planning, neural networks can be used to provide an efficient method of supporting the optimization of a manufacturing system design in a "closed loop" by learning from selected “experimental” values (can be obtained by simulation) which show the interdependencies between decision variables and performance measures. Neural networks can therefore be seen as catalysts for greater CIM capability.

In light of the increasing amount of possible applications of neural networks, the objective of this thesis is to evaluate the ability of neural networks to generate accurate models of physical systems typically encountered in manufacturing. By determining the reliability of neural network models and by comparing the efficacy of neural networks with other methods of empirical modeling as criterions, an assessment of the modeling abilities of neural networks can be formed.

Various neural network models will be tested on a number of "test bed" problems which represent the problems typically encountered in manufacturing processes and systems to determine the efficacy of their modeling capabilities. The problems chosen for the test bed are the following:

- two arbitrary multivariate functions
- laser through-cutting and laser grooving
- the manufacturing system design of an automobile steering column.

The first type of problem to be tested on neural networks is the presence of noise in experimental data. Systems have error associated with it due to the dependence of the output on uncontrollable or unobservable quantities, and the quality of the model developed from data containing such errors will be compromised to a certain degree. A method to estimate the confidence intervals of neural network models will be developed in order to assess their reliability. For a desired degree of confidence (i.e., for a given probability), a confidence region can be calculated for a parameterized model. Treating the neural network as a parameterized model, the confidence intervals can be estimated with this approach.

The second type of problem to be tested on neural networks is the high level of nonlinearity typically present in an input-output relationship due to the complex phenomena associated within the process or system. The relative efficacy of neural network modeling will be determined by comparing results from the neural network models of the test bed problems with results from models generated by other common empirical modeling approaches: linear regression, the Group Method of Data Handling (GMDH), and the Multivariate Adaptive Regression Splines (MARS) method. The GMDH is a modeling technique that groups the input variables in a form of a polynomial regression equation to predict the output of a multi-input single-output (MISO) system, and MARS is a modeling technique which combines recursive partitioning (the disjointing of the solution space for the model into different subregions) and spline fitting.

By exploring these two types of problems applied to the test bed problems mentioned, an assessment of the modeling abilities of neural networks can be formed. Following the introduction, Chapter 2 will provide an overview of neural networks. Chapter 3 will cover current applications of neural networks in manufacturing. Chapter 4 will explain the proposed method of estimating confidence intervals for neural networks. Chapter 5 will provide the results in estimating confidence intervals on neural network models. Chapter 6 will provide the results in comparing the relative efficacy of neural networks. Finally, the conclusion for this thesis will be given in Chapter 7.

CHAPTER 2
NEURAL NETWORKS FOR EMPIRICAL MODELING

Throughout various industries, such as the manufacturing and chemical industries, artificial neural networks have been successfully used as empirical models of processes and systems. This chapter will begin by discussing the advantages and disadvantages of empirical modeling versus analytical modeling. An introduction to artificial neural networks for empirical modeling follows, then the chapter will conclude by overviewing the current uses of artificial neural networks for industrial applications.

2.1. Empirical vs. Analytical Modeling

Analytical modeling of physical systems is the development of a mathematical expression for the system based on knowledge of the physical phenomenons occurring within the system. In general , development of an analytical model of a system entails simplification of the true behavior to make possible the arrival of a manageable and tractable solution, where often the simplification is a gross assumption which can produce enormous errors in the model. After making simplifications and assumptions of the system to arrive to an analytical model representing the physical system, there is still no guarantee that the analytical approach to obtain a model can be used. Although the advent of high speed processors for computers make numerical methods for solving highly complex and nonlinear functions possible, the amount of time required to solve such functions may still be too much for on-line use of models, such as those used for on-line plant monitoring and control. Another reason that analytical models may not be useful is that the simplifications and assumptions required to develop the model may cause gross errors in the predictions, thus making the model useless.

Empirical modeling of physical systems is the development of a model based on experimental observations and either interpolating or extrapolating the behavior of the system outside of the conditions given in the prior observations. Empirical modeling does not necessarily depend on a knowledge of the physical phenomenons that determines the behavior of the system, which can make empirical modeling the better (or sometimes the only) option in modeling. Empirical modeling requires the proper amount of "adequately" distributed observations of the physical behavior of the system, which is not always possible to obtain. If the cost of obtaining each experimental observation is relatively large, then the adequate amount of observations may be too costly to obtain. Also, some *a priori* knowledge of the physical phenomenons which determines the behavior of the system is required in order to determine the parameters necessary to include in the model because the inclusion of parameters which has no bearing on the behavior of the system can produce an invalid model.

When trying to obtain a model to predict a behavior of a system, the analytical approach would be the ideal approach, but if the cost and effort causes this approach to be either impractical or impossible, the empirical approach would be either the better or only choice. The benefits, along with the shortcomings, of the two approaches must be weighed when selecting the modeling approach.

2.2. Neural Network Background

According to Hecht-Nielsen, *neurocomputing* is the technological discipline concerned with parallel distributed information processing systems that develop information processing capabilities in response to exposure to an information environment [Hecht-Nielsen, 1990]. *Neural networks* are the primary information processing structures of interest in neurocomputing. The field of neural networks emerged from the developments of the neurocomputers, and today, development and implementation of neural networks span across the boundaries of neurocomputers and cross into numerous disciplines for many different applications. The evolution of neural networks can be traced back to approximately half a century, which reveals the amount of history behind the relatively new field of neural networks.

2.2.1. The Pursuit of Pattern Recognition

The work which lead to the 1943 paper "A Logical Calculus of the Ideas Immanent in Nervous Activity" by McCulloch and Pitts has been credited by many to be the beginning of neurocomputing [McCulloch and Pitts]. This work inspired the concept of a "brain-like" computer, but the concept was purely academic and there were no direct practical applications suggested. The developments of the network which uses the Adaptive Linear Element (ADALINE) by Widrow and the perceptron network by Rosenblatt in the late 1950's produced some of the first useful networks [Widrow][Rosenblatt].

Rosenblatt's primary interest was pattern recognition. He invented the perceptron, which is a neural network that had the ability to classify *linearly separable* patterns. Figure 2.1 illustrates two linearly separable patterns (classes A and B). The patterns are a function of the two inputs x_1 and x_2, and because a straight line is able to distinguish the class of the outputs of x_1 and x_2, the produced pattern is said to be linearly separable.

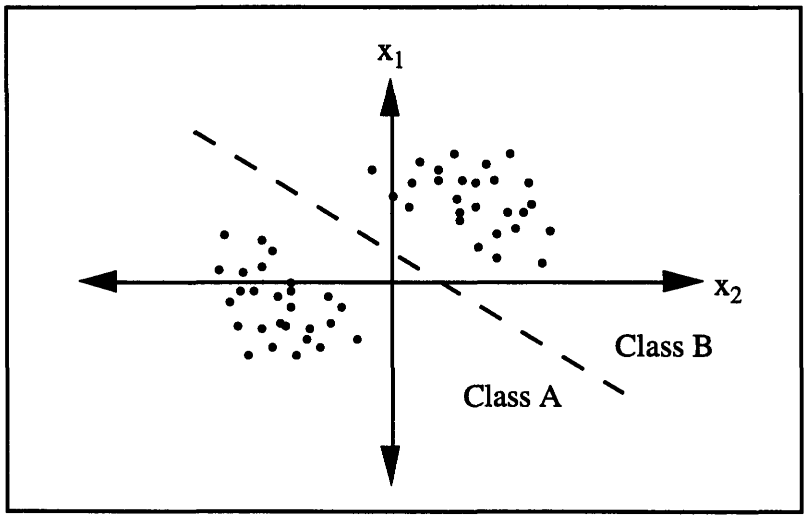


Figure 2.1: Example of a linearly separable pattern.

The perceptron did not have the ability to identify patterns which were not linearly separable, and this weakness was pointed out by Minsky and Papert [Minsky]. The famous example used by Minsky and Papert was the exclusive OR (XOR) problem. The XOR is a two input function with binary inputs which gives a binary output when only one of the inputs are on, either (1, 0) or (0, 1). If both are on (1, 1) or off (0, 0), an output would not be given. Figure 2.2 illustrates how the XOR problem is not linearly separable. No line can separate the O class and the X class.

Work done by Rumelhart, Hinton, and Williams [Rumelhart] in 1986 demonstrated how a network *can* develop a mapping which separates the two classes. The network architecture contained hidden processing units, or nodes, which allowed nonlinear input-output mapping. By solving the XOR problem, the possibilities of neural networks to learn the mapping of an arbitrary input-output relationship were realized.

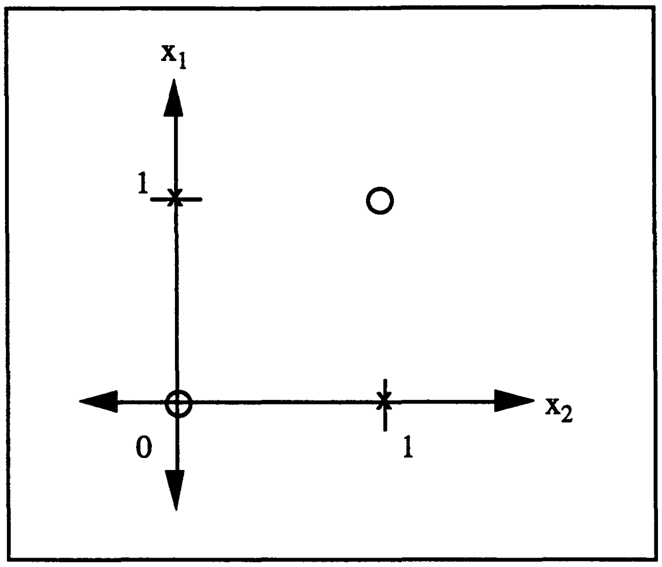


Figure 2.2: The exclusive OR problem.

2.2.2. Backpropagation Neural Networks

According to Hecht-Nielsen [Hecht-Nielsen, 1990], a neural network is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and can carry out localized information processing operations) interconnected via unidirectional signal channels called connections. Each processing element has a single output connection that branches ("fans out") into as many collateral connections as desired; each carries the same signal - the processing element output signal. The processing element output signal can be of any mathematical type desired. The information processing that goes on within each processing element can be defined arbitrarily with the restriction that it must be completely local; that is, it must depend only on the current values of the input signals arriving at the processing element via impinging connections and on values stored in the processing element's local memory.

Although the definition of a neural network given is to some extent restrictive, the possible architectures for a neural network vary widely. A commonly used network architecture is the feedforward neural network. The structure of the feedforward neural network is a connection of nodes arranged in hierarchical layers. The first layer contains the nodes which accept the input signals and send these signals to the nodes in the next layer through the connections (also known as links). The links from the nodes in each layer are connect to the nodes in the following layer up to the final layer, where the input signals are sent in a feedforward manner from the input layer to the output layer. Weight values are associated with each link which can be "corrected" to allow learning for the network. Figure 2.3 gives an illustrative example of a feedforward neural network. The unidirectional flow of the signals from the input to the output nodes classify this network structure as a feedforward neural network, but network structures classified as recurrent neural networks permit signals to flow to nodes within the same layer and to nodes in a previous layer in addition to the forward direction of flow.

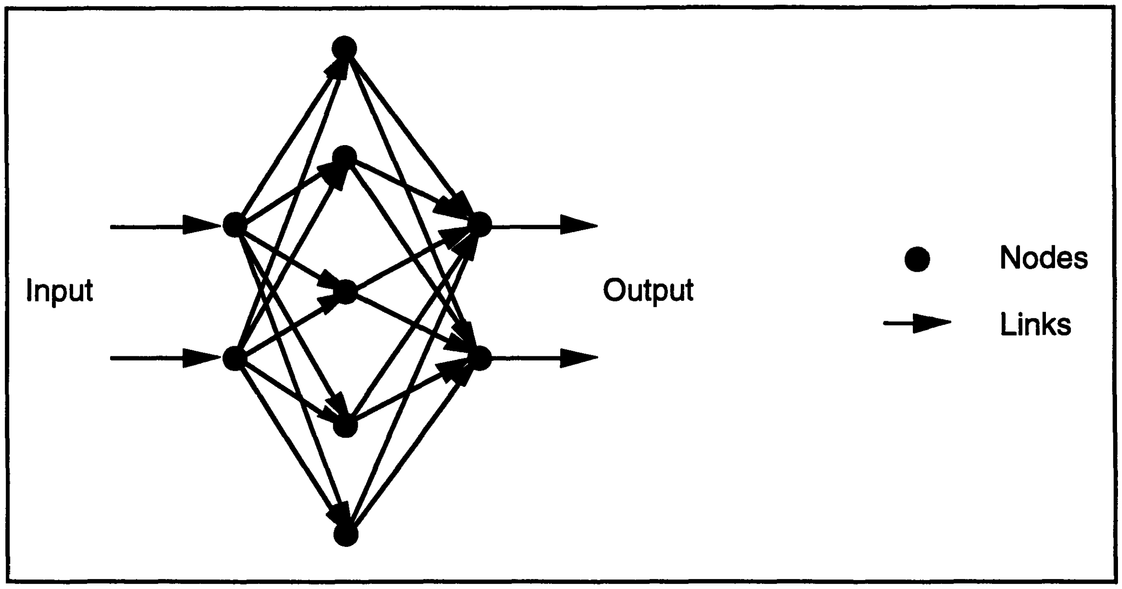


Figure 2.3: An example of a feedforward neural network.

Learning (also called training) is the process of conforming a neural network to map an input-output relation, where for a feedforward network, the conformation is the correction to the weight values to generate the correct input-output relationship. A common learning algorithm for feedforward networks is the error-backpropagation algorithm, which iteratively corrects the values of the weights by propagating an error signal backwards from the output layer to the input layer. Feedforward networks trained by the error-backpropagation algorithm are also known as backpropagation networks.

The outputs from a neural network are a conglomeration of nested nodal functions, and the sigmoidal function is a common choice for the nodal function. The mapping accuracy can be increased by introducing bias values to each nodes, which can also be viewed as a constant input to a node in addition to the inputs from the nodes in the previous layer. Equation (2.1) gives the sigmoidal function for a typical neural network with biases, and Equation (2.2) gives the neural network function for the simple 1-1-1 structure shown in Figure 2.4, where the input and output nodes implement linear functions and the middle node implements the sigmoidal function. The node in the second layer illustrates the sigmoidal function.

$$o = \left(\frac{1}{1 + \exp(-w \cdot x + b)}\right) \tag{2.1}$$

$$y = w_2\left(\frac{1}{1 + \exp(-w_1 x + b_1)}\right) + b_2 \tag{2.2}$$

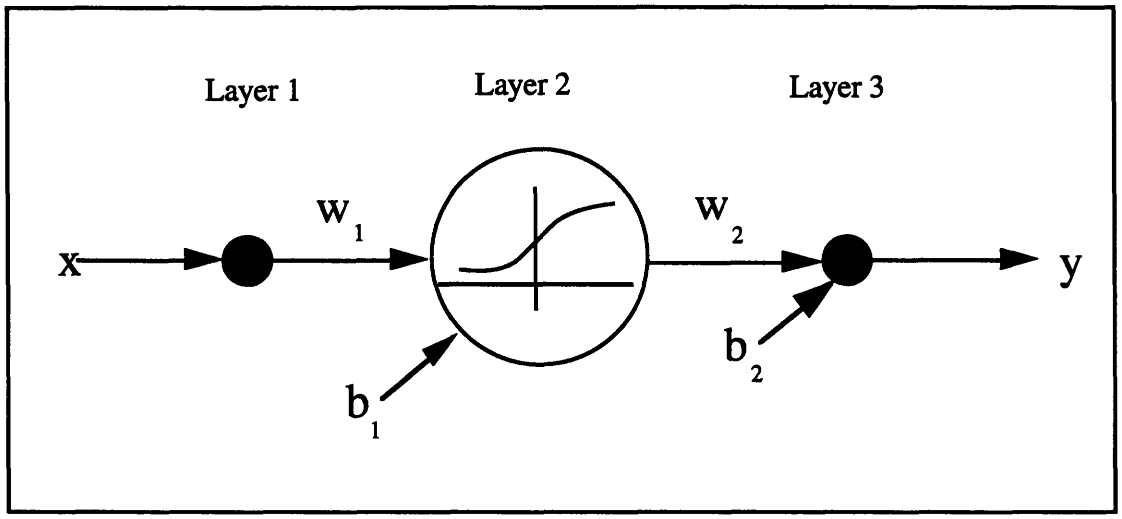


Figure 2.4: A 1-1-1 neural network with linear input and output nodes.

2.2.3. Other Neural Network Architectures

As opposed to supervised learning for multilayer perceptrons, the self-organizing network corrects itself by reading in only the input. This network has several links connecting the input and the output layer, but unlike the perceptron, there are recurrent links which connect the output nodes, or units, which composes the final layer. The input data is read, and the network determines the link with the least distance (the difference between the weight of the link and of the input value) and assigns the the region of the output node with the least distance to the input node. The trained network consists of an input-output layer, where the nodes of the output layer have been properly associated with its respective input nodes. In essence, the self-organized network is a heuristic lookup table, which is optimal for pattern identification used in speech and vision recognition, but is not as efficient for function mapping as the multilayer perceptron [Hecht-Nielsen, 1990][Lippmann].

The counterpropagation network, devised by Robert Hecht-Nielsen, is a synthesis of the self-organizing network and the Grossberg learning network, which is a network consisting of multiple input-output nodes with weighted links that are adjusted by the Grossberg Learning Law [Hecht-Nielsen, 1990]. An assessment of this network given by its creator reveals that the network is optimal for real-time analysis and implements an algorithm similarly used for pattern classification for mapping functions, but a direct comparison to a multilayer perceptron reveals its weakness in generalization, due to its architectural characteristic as a table lookup function [Hecht-Nielsen, 1987].

CHAPTER 3
NEURAL NETWORKS IN MANUFACTURING

In this chapter, a discussion of the industrial applications for neural networks is given. The chapter will begin by addressing the areas of applicability in manufacturing for neural networks and will conclude by overviewing the current applications of neural networks in manufacturing.

3.1. Areas of Applicability

There are two specific classes of problems in manufacturing where neural networks can be applied. The first class of problems involves process diagnostics, process control, and process optimization, and the second class of problems involves manufacturing system design and planning. This section will discuss the applicability of neural networks in these two classes.

3.1.1. Process Control, Diagnostics and Optimization

Most control schemes for manufacturing processes use a single sensor to monitor the process. In many cases, this approach is inadequate primarily due to inaccurate sensor information and the lack of a single process model which can sufficiently reflect the complexity of the process. As an alternative to the typical approach to process monitoring, a multiple sensor approach which is similar to the method a human uses to monitor a manufacturing process can be implemented with neural networks. In such an approach, the measurement of process variables is performed by several sensing devices which in turn feed their signals into different process models which contain mathematical expressions based on the physics of the process.

Synthesis of sensor information can provide a number of benefits for process monitoring as opposed to the current state of the art approach of using a single sensor. This is especially so for processes and systems that are difficult to model and monitor.

A considerable amount of work regarding the control of the manufacturing equipment and processes has been done over the past two to three decades. A major effort has been focussed on applying the scientific principles of control theory to manufacturing processes. Despite the high quality of work in this area, it appears that in reality the vast majority of manufacturing processes remain uncontrolled or empirically controlled due to the lack of two necessary elements for applying control theory principles to the control of manufacturing processes. The first element corresponds to sensing devices that will be robust and reliable enough to provide the necessary signals from the manufacturing process and/or equipment. The second element corresponds to comprehensive process models that will reflect the complexity of the manufacturing process.

Numerically controlled machine tools generally use predetermined parameters such as feed and speed as well as an open-loop control system, which has as inputs the machining parameters normally determined off-line during the process planning stage. These parameters are often determined with the aid of tables, machineability standards, and experience. Due to the fact that these parameters are determined well before the actual manufacturing process occurs, there is no way that automatic adaptation to the actual process can be achieved. Since factors disrupting the process are, for the most part, unpredictable, the choice of machining parameters must be made in such a manner that production is executed without breakdowns, even under the worst possible circumstances. Consequently, the capabilities of the machine are not fully utilized, and the manufacturing process is not efficient. Whenever adaptation to the actual situation or requirements of the process is needed, human intervention is required, often arriving too late or inefficiently.

Adaptive control schemes for the machining process have been researched over the past twenty years. In Adaptive Control Constraint (ACC), the goal of the controller is generally to adjust the machine tool's setting parameters in order to maintain a measured quantity, such as the cutting force, at a specified value. ACC controllers are primarily based on closed-loop control theory. In its most basic form, this scheme has a single input and single output and employs fixed proportional and integral control gains. For the machining process, the input may be the feed rate or cutting speed and the output may be the cutting force. More complex schemes adjust the controller gains on-line in order to improve the control performance. Regardless of the complexity of the scheme, the goal is usually to drive a measured or estimated state or set of states to a desired set point. The general drawback of any ACC system is that maintaining a variable such as the cutting force at a given set point will generally not optimize the process. Indeed, the significant optimization objectives for machining are generally a function of the tool wear and wear rate as well as the process input parameters, rather than direct functions of measured process variables such as the cutting force or tool temperature. Functions relating these objectives with a measured quantity, such as the cutting force, would undoubtedly be very complex and nonlinear, since the cutting process is highly nonlinear. The optimization objectives may not even be a monotonically increasing or decreasing function of the process input parameters. This characteristic is in sharp contrast with standard transfer functions used in linear control design, where provided the system is stable, the quasi-static output of the plant is assumed to be a linear function of the input. In addition, linear control design is primarily effective for time-invariant systems, while the machining process is a complex time-varying system. Furthermore, linear control theory was primarily developed to maintain an objective at a specified set point; however, the goal of a process optimization scheme is not to maintain an objective at a preset value, but rather to obtain the maximum (or minimum) of the objective over the feasible range of process input parameters. For these reasons, classical and modern control theory can be rendered inadequate for optimizing not only the machining process but manufacturing processes in general.

Adaptive Control Optimization (ACO) schemes have been researched in an effort to optimize the machining process. The major obstacles to successful in-process optimization of machining operations have been the lack of sensing devices that can reliably monitor the process and the lack of a single process model which can comprehensively reflect the complexity of the machining process. With a few exceptions, most machining control schemes use a single sensor to monitor the process and therefore consider a single process model. Past research has shown that accurate process models are difficult to build and are generally unreliable for machining control over a wide variety of operating conditions. In contrast, if information from a variety of sensors and sensor-based models is integrated, the maximum amount of information would be used for making control decisions and thus the quality of the decisions would likely be better than decisions based on information from a single sensor. Ideally, different sensor-based models should provide the same estimates for the machining parameters. However, these estimates would generally include a significant amount of random noise. Utilizing several simultaneous sensor-based estimates can be considered analogous to taking several samples from a random distribution. Statistically, as more samples are taken, the confidence interval for the mean becomes narrower. In the same way, as more sensor-based model estimates are considered, the estimates for the machining parameters become more certain; the uncertainty due to randomness in the estimates is reduced. In addition, if one sensor fails during the process, a controller utilizing multiple sensors could probably continue to operate, while a controller based on a single sensor would be forced to stop the process.

3.1.2. System Design and Planning

Particularly in the metal working industry, parts usually spend 5 percent to 15 percent of their time on the machinery and the rest of the time they move around on the factory floor waiting for machines, transportation, etc. This leads one to believe that decisions that are made regarding the design and operation of a production system can be vastly improved if one can establish a framework for decisions in this environment and optimize it from a total performance point of view.

A number of approaches have been proposed in the literature for the design of manufacturing systems. Usually, the overall manufacturing system design problem is decomposed into sub-problems of manageable complexity, meaning that only a single type of decision variable and a single type of performance measure is considered for each sub-problem.

- One sub-problem is the *resource requirements problem.* For this problem, the task is to determine the appropriate quantity of each type of production resource (for example, machines or pallets) in a manufacturing system.
- The *resource layout problem* is the problem of locating a set of resources in a constrained floor space.
- In *material flow problems*, the objective is to determine the configuration of a material handling system.
- The *buffer capacity problem* is concerned with the allocation of work in process or storage capacity in a manufacturing system.

These sub-problems are usually treated separately. This neglects the inter-relationships that exist between the different sub-problems (e.g., the required buffer capacity at a work center in a manufacturing system depends on the number of machines in that work center).

Solution of this problem requires knowledge of the relationship between the performance measures and the decision variables. This relationship is highly nonlinear and difficult to establish for a number of reasons:

- Manufacturing systems are large-scale systems with many interacting components.
- The parameters which are responsible for the behavior a manufacturing system (e.g., processing times), are often uncertain and must be characterized by distributions rather than constant values.

Existing methods address the difficulty of manufacturing system design by simplifying the problem definition. Common strategies for doing so are:

- Restrict the structure of the material handling system. Many approaches, for example, apply only to transfer lines, which have purely serial material flow.
- Restrict the scheduling policies of the manufacturing system to simple rules which are easier to characterize mathematically (e.g., first come, first served).
- Consider only one fixed type of decision variable (e.g., buffer capacity) and one fixed performance measure (e.g., production rate) that can be easily expressed in terms of the decision variables.

Most control schemes for manufacturing processes use a single sensor to monitor the process. In many cases, this approach is inadequate primarily due to inaccurate sensor information and the lack of a single process model which can sufficiently reflect the complexity of the process. As an alternative to the typical approach to process monitoring, a multiple sensor approach which is similar to the method a human uses to monitor a manufacturing process, can be implemented with neural networks. In such a approach, the measurement of process variables is performed by several sensing devices which in turn feed their signals into different process models which contain mathematical expressions based on the physics of the process.

In manufacturing, the primary decision-making tools for system design remain simulation and analytical modeling. Each tool has a major weakness when it comes to the design of complex manufacturing systems: simulation suffers from excessive computational expense, while analytical modeling can be applied only to a very restricted subset of systems (e.g., systems such as transfer lines with purely serial material flow or systems with constant work in process). The approach using neural networks can accommodate manufacturing system design problems in which the performance requirements involve multiple types of performance measures (e.g., production rate and average work in process), and in which design solutions involve multiple types of decision variables (e.g., machine quantities and machine layout and buffer capacities).

3.2. Current Applications of Neural Networks in Manufacturing

Neural networks have been used for many types of manufacturing application. In general, input variables which affect the output of a manufacturing process or system are known, but the input-output relationship is either difficult to obtain or simply not known. Neural networks have served as a heuristic mapping function for various input-output relationships. In this section, an overview of industrial applications of neural networks is given.

Neural networks have modelled the Electric Discharge Machining (EDM) process [Indurkhya, Rajurkar]. EDM is a process by which high strength temperature resistant alloys are machined by a very hot spark emanating from the tool to the workpiece across a dielectric fluid medium. Presently, the relationships between the controllable inputs and output parameters of the EDM process have not been accurately modelled due to their complex and random behavior. A 9-9-2 backpropagation neural network has been developed as the structure, where the nine inputs correspond to the machining depth, tool radius, orbital radius, radial step, vertical step, offset depth, pulse on time, pulse off time, and discharge current, which determines the two output parameters material removal rate and surface roughness. The neural network model proved to be closer to the actual experimental results when compared to multiple regression.

Neural networks have also been implemented to model the grinding operation. Standard automated control was not possible due to the fact that there are so many factors which affect the grinding process. Thus, the standard approach to control the grinding process was the employment of a skilled operator who relies on considerable experience to dynamically adjust the conditions to achieve the desired quality. A hybrid neural network have been implemented for the decision-making model of the process [Sakakura, Inasaki]. The hybrid model consists of a feedforward neural network and a Brain-State in a Box (BSB) network. The feedforward net serves as the input-output model of the grinding process, and the BSB net, which functions as a classifier similar to the associative memory of humans, serves to recall the most suitable combination of the input parameters for the desired surface roughness.

Neural networks have been used to model arc welding. The arc welding process is a nonlinear problem, which makes this process difficult to model. Full automation of this process has not yet been achieved due to the lack of knowledge of some of the physics in arc welding. A backpropagation neural network has been used to model the arc welding process as a multivariable system [Anderson, *et al*]. The outputs of the arc weld model were the bead width and bead penetration, which help to define the characteristics of the finished weld. The inputs of the model were the workpiece thickness, travel speed, arc current, and arc length, and the model gave prediction errors on the order of 5% or less. Some of the current works in this topic includes the implementation of a neural network model for on-line control of the arc weld process.

Quality control of the injection molding operation has been implemented using a backpropagation neural network [Smith]. The input parameters of the model based on the equipment set-up were the temperatures of the four barrel zones along with the gate, head, and die zones. The input parameters based on the runs were the screw speed, DC power, the line speed, the quench temperature, and the head pressure. Other independent variables used for the model were the die and tip diameters, the air pressure, and the time of each sample since the line went up. A total of 16 input parameters were used to determine the two output parameters, which were the mean and variance of the injection molded piece. Results found were that neural networks perform comparably with statistical techniques in goodness of output for process and quality control. The neural network performed comparatively better when modeling quality/process control data which exhibits a nonlinear behavior.

The predicting of wire bond quality for microcircuits has been aided with neural networks [Wang, *et al*]. Wire bonding is the process by which a gold wire, where the thickness is in the order of 0.001 in., is "thermosonically" welded to to a gold or metal oxide pad in the microcircuit. The costly testing of wire bond quality is traditionally implemented by the wire pulling test, and neural networks would serve as a low-cost approach to process control. The results show that the neural network can be used as an accurate and inexpensive alternative for predicting wire bond quality.

Wave soldering has been another process which neural networks attempted to model [Malave, *et al*]. The preheat temperatures and the conveyor speed are two of the machine parameters determine the bond quality, where a total of 26 input parameters affect the machine parameters. The neural network was not able to converge to a working model of the system, but the inability of the neural network to converge has been traced to faulty data. The data were collected randomly; no design of experiments were implemented to ensure a proper representation of the physical system.

CMAC (Cerebella Model Articulation Controller) has been integrated with neural networks for fault predictions in machinery during the manufacturing process [Lee, Tsai]. Through statistical process control (SPC), future values and estimations of machine performance are calculated from the derived performance model, but these are only superficial models of the system. Through the aide of a CMAC, a network can be created with the ability to detect faults by monitoring the output patterns from sensors and actuators. Thus by analyzing the timing sequence, abnormal conditions in the machinery become detectable. The CMAC model is a table-look up technique that provides an algorithm for random mapping of a large memory space to a smaller, practical space. From the mapping, nearby states tend to occupy overlapping memory locations and distance states tend to occupy independent memory locations. This scheme provides a form of linearization between nearby input vectors. The CMAC is not a method to process inputs into accurate outputs, but rather a model for real-time control that a biological organism appears to follow, i.e., in problem solving, a living organism first generates an approximately correct estimate that is sufficient for a response with the conditions at hand; after that, the situation is reassessed for the next action. The important point is that the organism has the capability to generate outputs continuously and that each new output moves the organism in the correct direction closer to the goals. CMAC offers the advantage of rapid convergence, and integrated with neural networks, allows the classification of failure based on pattern recognition by comparing the input with that of a healthy subsystem to provide an indication of distress conditions.

Neural networks have been used as predictive models, but neural networks are also appropriate for the control and optimization of plant dynamics. In supervisory optimization, the optimizer (an expert) would make suggestions to the operator on how to change the operating parameters of the plant so as to maximize efficiency and yet maintain a smoothly running plant. When asked, it sometimes might be difficult for the expert to explain his reasons for altering certain parameters on the plant. This kind of expertise comes from experience and is quite difficult to incorporate into classical models or rule-based systems, but is readily learned from historical data by a neural network [Keeler]. The neural networks can provide several useful; types of information for the plant engineer, including:

a) Sensitivity analysis
b) Setpoint recommendations for process improvement
c) Process understanding
d) Real-time supervisory control
e) Sensor validation - the neural networks can learn the normal behavior of sensors in a system and can be used to validate sensor readings or alarm for sensors that are not functioning properly.

f) Predictive maintenance - the neural networks can be used to predict component failure so as to schedule maintenance before the failure occurs.

As an adjunct technology to neural networks, fuzzy control is useful for implementing default rules for data poor regions. In addition, fuzzy rules are useful at the data analysis stage for screening data, at the modeling stage for describing behavior outside the data region, for incorporation of constraints, and for generation of fuzzy rules describing plant behavior and optimization procedures. Thus, fuzzy rule systems are ideal for bridging the gap between the adaptive neural network systems and hard rule-based expert systems.

The steps taken for the application of neural network technology to plant dynamics are:

1) Extract data from the historical database and store files,
2) Examine the data graphically and screen out any bad data points or outliers.
3) Estimate the time-delays of the plant dynamics (by, e.g. asking the plant engineers)
4) Train a neural network model to predict the future behavior of interesting variables such as yield, impurities, etc..
5) Test accuracy of model versus new data.

Neural networks have also been implemented for identifying on-line tool breakage in the metal cutting process [Guillot, El Ouafi]. A 20-5-1 structured perceptron neural network was used for tool condition identification using time-domain force signal during milling operations, where the dynamometer (or accelerometer or acoustic emission sensor) is acquired and preprocessed according to a desired technique. In this case, the preprocessing technique of choice is a "signal first derivative" technique in which the signal derivative is calculated and thus allowing us to view more distinctly the tool breakage signal from the magnitude of the signal peaks. During testing, the network managed to easily identify the tool breakage patterns learned from training as well as other breakage patterns exhibited considerable differences. The network proved efficient in correctly assessing a broad range of tool conditions using a small set of patterns.

CHAPTER 4
CONFIDENCE INTERVAL PREDICTION FOR NEURAL NETWORKS

The purpose of this chapter is to derive an estimate of a neural network's accuracy as an empirical modeling tool. In general, a model of a physical system has error associated with its predictions due to the dependence of the physical system's output on uncontrollable or unobservable quantities. Neural network models have been used as a predictor for different physical systems in control and optimization applications [Chryssolouris, 1990], and a method to quantify the confidence intervals of the predictions from neural network models is desired.

For a desired degree of confidence (namely, for a given probability), a confidence interval is a prediction of the range of the output of a model where the actual value exists. With an assumption of a normal distribution of the errors, confidence intervals can be calculated for neural networks.

The chapter begins by giving the background of a method of calculating confidence intervals for arbitrary parameterized models. The analysis is extended to include the calculation of confidence intervals for models obtained from corrupted or noisy data. The analysis continues with the derivation of confidence intervals for neural networks.

4.1. Confidence Intervals for Parameterized Models

For a given system with output y, the model for the system is given to be $f(\mathbf{x}; \theta^*)$, where $\mathbf{x}$ is the set of inputs and θ^* represents the true values of the set of parameters θ for the function which models the system [Seber]. The error ε associated with the function in modeling the system is assumed to be independently and identically distributed with variance σ^2, where the distribution has the form $N(0, \sigma^2)$. With n observations, where $i = 1, 2, ..., n$, the system is represented by Equation (4.1).

$$y_i = f(\mathbf{x}_i; \theta^*) + \varepsilon_i \qquad , \quad i = 1, 2, ..., n \tag{4.1}$$

The least-squares estimate of θ^* is $\hat{\theta}$, which is obtained by minimizing the error function given in Equation (4.2) (for neural networks, the error backpropagation algorithm is a common method for minimizing the error function). The predicted output from the model is $\hat{y}_i$, as shown in Equation (4.3).

$$S(\theta) = \sum_{i=1}^{n} \left[y_i - f(\mathbf{x}_i;\theta)\right]^2 \tag{4.2}$$

$$\hat{y}_i = f(\mathbf{x}_i; \hat{\theta}) \tag{4.3}$$

If the model gives a good prediction of the actual system behavior, then $\hat{\theta}$ is close to the true value of the set of parameters θ^* and a Taylor expansion to the first order can be used to approximate $f(\mathbf{x}_i; \hat{\theta})$ in terms of $f(\mathbf{x}_i; \theta^*)$ (Equation (4.4)).

$$f\left(\mathbf{x}_i ;\hat{\theta}\right) \approx f\left(\mathbf{x}_i ;\theta^*\right) + \mathbf{f}_0^{\mathrm{T}} \cdot \left(\hat{\theta} - \theta^*\right) \tag{4.4}$$

$$\text{where } \mathbf{f}_0^{\mathrm{T}} = \left(\frac{\partial f\left(\mathbf{x}_i;\theta^*\right)}{\partial \theta_1^*}, \frac{\partial f\left(\mathbf{x}_i;\theta^*\right)}{\partial \theta_2^*}, \ldots, \frac{\partial f\left(\mathbf{x}_i;\theta^*\right)}{\partial \theta_p^*}\right)$$

By using Equation (4.1) and (4.4), Equation (4.5) gives the difference between the true value y of the system and the predicted value $\hat{y}$, and Equation (4.6) gives the expected value of the difference. The subscript value of 0 is given to denote the set of points other than that used for the least-squares estimation of θ*.

$$y_0 - \hat{y}_0 \approx y_0 - f(\mathbf{x}_0;\theta^*) - \mathbf{f}_0^T \cdot \left(\hat{\theta} - \theta^*\right) = \varepsilon_0 - \mathbf{f}_0^T \cdot \left(\hat{\theta} - \theta^*\right) \tag{4.5}$$

$$E[y_0 - \hat{y}_0] \approx E[\varepsilon_0] - \mathbf{f}_0^T E\left[\left(\hat{\theta} - \theta^*\right)\right] \approx 0 \tag{4.6}$$

Because of the statistical independence between $\hat{\theta}$ and ε_0, the variance can be expressed as Equation (4.7).

$$\text{var}[y_0 - \hat{y}_0] \approx \text{var}[\varepsilon_0] + \text{var}\left[\mathbf{f}_0^T \cdot \left(\hat{\theta} - \theta^*\right)\right] \tag{4.7}$$

For an error ε_0 with a normal distribution with a mean of 0 and a variance of σ^2 ($N(0, \sigma^2 \mathbf{I}_n)$), the distribution of $\hat{\theta} - \theta^*$ can be approximated to have the distribution $N_p(0, \sigma^2 \mathbf{F}.(\hat{\theta})^T \mathbf{F}.(\hat{\theta}))$. The Jacobian matrix $\mathbf{F}.(\hat{\theta})$ has the form shown in Equation (4.8), where the single period has been placed to keep in accord with the notations used from the reference [Seber] which denotes that the matrix has first order differential terms.

$$\mathbf{F}.\left(\hat{\theta}\right) = \frac{\partial \mathbf{f}\left(\mathbf{x},\hat{\theta}\right)}{\partial \hat{\theta}'} = \begin{bmatrix} \left(\frac{\partial f_1(\mathbf{x}_1, \hat{\theta})}{\partial \hat{\theta}_1}\right) & \left(\frac{\partial f_1(\mathbf{x}_1, \hat{\theta})}{\partial \hat{\theta}_2}\right) & \cdots & \left(\frac{\partial f_1(\mathbf{x}_1, \hat{\theta})}{\partial \hat{\theta}_p}\right) \\ \left(\frac{\partial f_2(\mathbf{x}_2, \hat{\theta})}{\partial \hat{\theta}_1}\right) & \cdots & \cdots & \left(\frac{\partial f_2(\mathbf{x}_2, \hat{\theta})}{\partial \hat{\theta}_p}\right) \\ \vdots & \vdots & \vdots & \vdots \\ \left(\frac{\partial f_n(\mathbf{x}_n, \hat{\theta})}{\partial \hat{\theta}_1}\right) & \left(\frac{\partial f_n(\mathbf{x}_n, \hat{\theta})}{\partial \hat{\theta}_2}\right) & \cdots & \left(\frac{\partial f_n(\mathbf{x}_n, \hat{\theta})}{\partial \hat{\theta}_p}\right) \end{bmatrix} \tag{4.8}$$

$$\mathrm{var}\left[y_0 - \hat{y}_0\right] \approx \sigma^2 + \sigma^2 \mathbf{f}_0^T \left(\mathbf{F}_\bullet^T \mathbf{F}_\bullet\right)^{-1} \mathbf{f}_0 \tag{4.9}$$

The matrix has the dimensions n by p, where n is the number of samples used to obtain $\hat{\theta}$, and p is the number of parameters θ_i which composes $\hat{\theta}$. Hence, Equation (4.10) gives the unbiased estimator of σ^2, and using this equation, the Student t-distribution is given in Equation (4.11). Equation (4.12) gives the confidence interval $100*(1 - \alpha)$ for the predicted value $\hat{y}$.

$$s^2 = \frac{\left\| \mathbf{y} - \mathbf{f}\left(\mathbf{x}, \hat{\theta}\right) \right\|^2}{n - p} \tag{4.10}$$

$$t_{n-p} \sim \frac{y_0 - \hat{y}_0}{\sqrt{\mathrm{var}\left[y_0 - \hat{y}_0\right]}} \approx \frac{y_0 - \hat{y}_0}{\sqrt{s^2 + s^2 \mathbf{f}_0^T \left(\mathbf{F}_\bullet^T \mathbf{F}_\bullet\right)^{-1} \mathbf{f}_0}} \approx \frac{y_0 - \hat{y}_0}{s\left(1 + \mathbf{f}_0^T \left(\mathbf{F}_\bullet^T \mathbf{F}_\bullet\right)^{-1} \mathbf{f}_0\right)^{\frac{1}{2}}} \tag{4.11}$$

$$\hat{y}_0 \pm t_{n-p}^{\alpha/2}\, s \left(1 + \mathbf{f}_0^T \left(\mathbf{F}_\bullet^T \mathbf{F}_\bullet\right)^{-1} \mathbf{f}_0\right)^{\frac{1}{2}} \tag{4.12}$$

Other methods for estimating confidence intervals are available which uses a different approach to determine the variance-covariance matrix used in Equation (4-12). The proposed method differs from the existing methods for neural network confidence interval derivation because it does not require information about the second derivatives of the neural network output, and because it accounts for the accuracy of the data with which the neural network model is trained.

4.1.1. Selection of the Variance-Covariance Matrix

From a Monte Carlo study of constructing confidence intervals for parameterized models estimated by nonlinear least squares [Donaldson, Schnabel], three variants of determining the variance-covariance matrix $\hat{\mathbf{V}}$ of the estimated parameters were studied. The three approaches were given to be the following, where s^2 is the estimated residual variance, $\mathbf{J}(\hat{\theta})$ represents the Jacobian matrix or the function with respect to the parameters $\hat{\theta}$,and $\mathbf{H}(\hat{\theta})$ represents the Hessian matrix:

$$\hat{\mathbf{V}}_\alpha = s^2(\mathbf{J}(\hat{\theta})^T\mathbf{J}(\hat{\theta}))^{-1}$$

$$\hat{\mathbf{V}}_\beta = s^2\mathbf{H}(\hat{\theta})^{-1}$$

$$\hat{\mathbf{V}}_\chi = s^2\mathbf{H}(\hat{\theta})^{-1}(\mathbf{J}(\hat{\theta})^T\mathbf{J}(\hat{\theta}))\mathbf{H}(\hat{\theta})^{-1}$$

Results from the Monte Carlo study revealed that the $\hat{\mathbf{V}}_\alpha$ estimation of the variance-covariance matrix gave the best results with minimal efforts, since determining the Jacobian matrix $\mathbf{J}(\hat{\theta})$ required ony first-order differentiation, while determining the Hessian matrix $\mathbf{H}(\hat{\theta})$ required additional differentiation up to the second order, which is a less stable matrix to invert compared to the Jacobian. Thus, the $\hat{\mathbf{V}}_\alpha$ estimation was employed, but with a more detailed and with a higher degree of effort, $\hat{\mathbf{V}}_\beta$ and $\hat{\mathbf{V}}_\chi$ may be employed.

4.2. Derivation of Confidence Intervals for Models Derived from Noisy Data

The purpose of this section is to derive an estimate of a neural network's accuracy based on the accuracy of the data that it is trained with. Discrepancy between the true output of the system and the *observed* output of the system may exist, due to inaccuracies in measurement of the output. Such an estimate may be used to predict the domain of the input over which a neural network model will adequately model the output of the system.

Equation (4.13) gives the equation which represents the system, and Equation (4.14) gives the equation which represents the observed output of the system. The error value ε_1

($\sim N(0, \sigma_1^2)$) is the difference between the true output value and the neural network output, arising from the limitations from the model to capture the unobservable and uncontrollable error. The error value ε_2 ($\sim N(0, \sigma_2^2)$) is the difference between *observed* values and the true values, arising from inaccuracy of the observation. The error value ε ($\sim N(0, \sigma_1^2 + \sigma_2^2) = N(0, \sigma)$) given in Equation (4.15) is the sum of the errors due to the modeling inaccuracy and the observation inaccuracy.

$$y = f(\mathbf{x};\, \theta^*) + \varepsilon_1 \tag{4.13}$$

$$y_{obs} = y + \varepsilon_2 \tag{4.14}$$

$$y_{obs} = f(\mathbf{x};\, \theta^*) + \varepsilon_1 + \varepsilon_2 = f(\mathbf{x};\, \theta^*) + \varepsilon \tag{4.15}$$

Following the analysis in the previous section, the variance of $\left[y_0 - \hat{y}_0\right]$ is given in Equation (4.16). Thus, by using s_1^2, which is the unbiased estimator for σ_1^2 given in Equation (4.17), the confidence interval for a parametric model derived from noisy data is given in Equation (4.18). When the noise level is zero, Equation (4.18) collapses into Equation (4.12).

$$\begin{aligned} \mathrm{var}[y - \hat{y}_0] &\approx \mathrm{var}[y] + \mathrm{var}[\hat{y}_0] \\ &\approx \mathrm{var}[\varepsilon_1] + \mathrm{var}[\mathbf{f}_0^{\mathrm{T}} \cdot (\hat{\theta} - \theta^*)] \\ &\approx \sigma_1^2 + \sigma^2 \mathbf{f}_0^{\mathrm{T}} (\mathbf{F}._{}^{\mathrm{T}} \mathbf{F}.)^{-1} \mathbf{f}_0 \\ &= \sigma_1^2 + (\sigma_1^2 + \sigma_2^2) \mathbf{f}_0^{\mathrm{T}} (\mathbf{F}.^{\mathrm{T}} \mathbf{F}.)^{-1} \mathbf{f}_0 \end{aligned} \tag{4.16}$$

$$s_1^2 = \frac{\left\| \mathbf{y}_{obs} - \mathbf{f}\left(\mathbf{x}, \hat{\theta}\right) \right\|^2}{n - p} \tag{4.17}$$

$$\hat{y}_0 \pm t_{n-p}^{\alpha/2} \left(s_1^2 + \left(s_1^2 + \sigma_2^2\right) \mathbf{f}_0^{\mathrm{T}} \left(\mathbf{F}.^{\mathrm{T}} \mathbf{F}.\right)^{-1} \mathbf{f}_0\right)^{\frac{1}{2}} \tag{4.18}$$

4.3. Derivation of Confidence Intervals for Neural Networks

Equation (4.12) states the confidence interval for a model used as a predictor for y, which was derived from noiseless data. This form of the confidence interval will be applied to a feedforward artificial neural network model being used as a predictor for y. The term $t_{n-p}^{\alpha/2}$ can be found for a given α and the degrees of freedom $n - p$ (where p is the number of weights and bias terms employed by the neural network), and s can be computed from Equation (4.10). The task is now to find the derivative terms in the **F.** and the $\mathbf{f}_0^T$ matrices.

The outputs from a neural network are a conglomeration of nested sigmoidal functions. Equation (4.19) gives the sigmoidal function for a typical neural network, and for demonstrative purposes, Equation (4.20) gives the neural network function for the simple 1-1-1 structure shown in Figure 4-1, where the input and output nodes implement linear functions and the middle node implements the sigmoidal function.

$$o = \left(\frac{1}{1 + \exp(-w \cdot x + b)}\right) \tag{4.19}$$

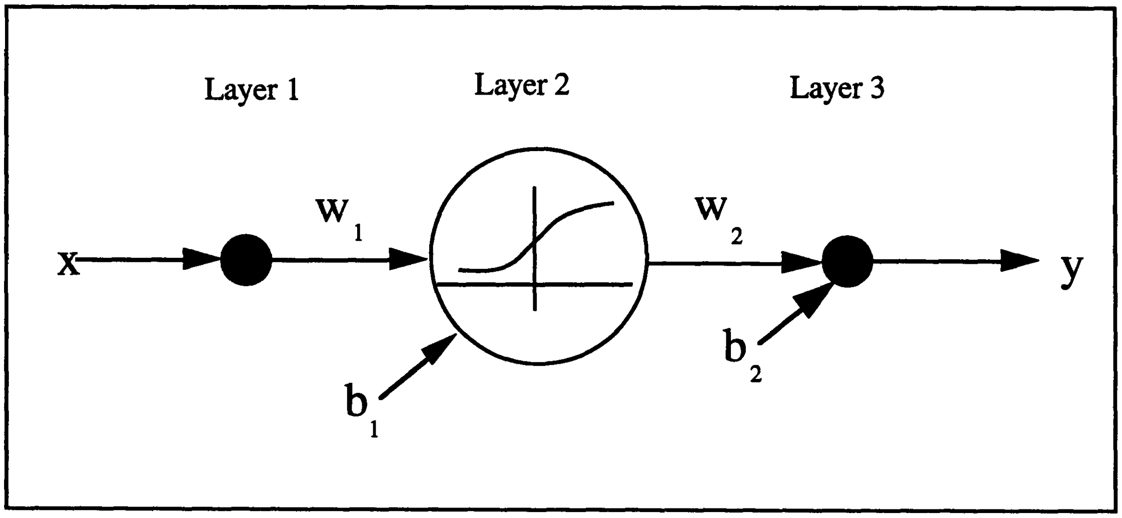


Figure 4-1: A 1-1-1 neural network with linear input and output nodes.

$$y = w_2\left(\frac{1}{1 + \exp(-w_1 x + b_1)}\right) + b_2 \tag{4.20}$$

The **F.** and $\mathbf{f}_0^T$ matrices are the matrices containing the derivatives of the output y and y_0 with respect to the parameters θ_i. Equation (4.20) gives the full sigmoidal equation of the 1-1-1 neural network shown in Figure 1. To obtain the values of the elements in the **F.** and $\mathbf{f}_0^T$ matrices, the changes in the output y with respect to the weights $(\frac{\partial y}{\partial w_i})$ and the bias terms $(\frac{\partial y}{\partial b_j})$ must be found.

Two terms must be defined before the analysis of computing the partial derivatives of the output y. The first term to be defined is the $net_j^{[\beta]}$ quantity. This term is summation of the outputs from the nodes of layer β-1 entering node j in layer β. The expression for $net_j^{[\beta]}$ is given in Equation (4.21), where n is the number of nodes in layer β-1, and b_j is the bias term for node j.

$$net_j^{[\beta]} = \left(\sum_{i=1}^{n} w_i^{[\beta-1]} o_i^{[\beta-1]}\right) - b_j \tag{4.21}$$

The second term to be defined is the $layer^{[\beta]}$ quantity. This term characterizes the response of layer β for a given set of inputs, and this term is defined to be the summation of the $net_j^{[\beta]}$ functions for layer β. Equation (4.22) gives the expression for $layer^{[\beta]}$, where m is the number of nodes in layer β.

$$layer^{[\beta]} = \sum_{j=1}^{m} net_j^{[\beta]} \tag{4.22}$$

With the terms $net_j^{[\beta]}$ and $layer^{[\beta]}$ defined, the derivatives of the output y with respect to the weights and the biases can be analyzed. Equation (4.23) expresses the derivative $\frac{\partial y}{\partial w_\gamma^{[\alpha]}}$, where γ is the specific weight value of layer α, and m denotes the total number of layers in the neural network.

$$\frac{\partial y}{\partial w_\gamma^{[\alpha]}} = \frac{\partial y}{\partial net^{[m]}} \frac{\partial net^{[m]}}{\partial layer^{[m-1]}} \frac{\partial layer^{[m-1]}}{\partial layer^{[m-2]}} \cdots \frac{\partial layer^{[\alpha+3]}}{\partial layer^{[\alpha+2]}} \frac{\partial layer^{[\alpha+2]}}{\partial net_\gamma^{[\alpha+1]}} \frac{\partial net_\gamma^{[\alpha+1]}}{\partial w_\gamma^{[\alpha]}} \tag{4.23}$$

For a neural network which has an output node with a linear linear function, $\frac{\partial y}{\partial net^{[m]}}$ is simply 1, and for an output node with a sigmoidal function is $y(1 - y)$. The term $\frac{\partial net^{[m]}}{\partial layer^{[m-1]}}$ can be broken down into the form given in Equation (4.24). The derivative $\frac{\partial layer^{[m-1]}}{\partial layer^{[m-2]}}$ can be expressed in the form given in Equation (4.25), where p is the number of nodes in layer m-1. Equation (4.26) gives the expression for $\frac{\partial layer^{[\alpha+2]}}{\partial net_\gamma^{[\alpha+1]}}$, and Equation (4.27) gives the expression for $\frac{\partial net_\gamma^{[\alpha+1]}}{\partial w_\gamma^{[\alpha]}}$. Equation (4.28) gives the general form of $\frac{\partial net_\lambda^{[\phi+1]}}{\partial net_\zeta^{[\phi]}}$, where ϕ is an arbitrary layer, λ is an arbitrary node in layer $\phi + 1$, ζ is an arbitrary node in layer ϕ, and A is the number of nodes in layer ϕ.

$$\frac{\partial net^{[m]}}{\partial layer^{[m-1]}} = \frac{\partial net^{[m]}}{\partial net_1^{[m-1]}} + \frac{\partial net^{[m]}}{\partial net_2^{[m-1]}} + \cdots + \frac{\partial net^{[m]}}{\partial net_n^{[m-1]}} = \sum_{i=1}^{n} \frac{\partial net^{[m]}}{\partial net_i^{[m-1]}} \tag{4.24}$$

$$\frac{\partial layer^{[m-1]}}{\partial layer^{[m-2]}} = \sum_{j=1}^{p} \frac{\partial net_j^{[m-1]}}{\partial layer^{[m-2]}} = \sum_{j=1}^{p} \left(\sum_{i=1}^{n} \frac{\partial net_j^{[m-1]}}{\partial net_i^{[m-2]}} \right) \tag{4.25}$$

$$\frac{\partial layer^{[\alpha+2]}}{\partial net_\gamma^{[\alpha+1]}} = \sum_{j=1}^{p} \frac{\partial net_j^{[\alpha+2]}}{\partial net_\gamma^{[\alpha+1]}} \tag{4.26}$$

$$\frac{\partial net_\gamma^{[\alpha+1]}}{\partial w_\gamma^{[\alpha]}} = o_\gamma^{[\alpha]} \tag{4.27}$$

$$\frac{\partial net_\lambda^{[\phi+1]}}{\partial net_\zeta^{[\phi]}} = \frac{\partial}{\partial net_\zeta^{[\phi]}}\left(\sum_{a=1}^{A}\left(w_a^{[\phi]}o_a^{\phi}\right) - b_\lambda\right) = \sum_{a=1}^{A}\left\{w_a^{[\phi]}o_a^{[\phi]}\left(1 - o_a^{[\phi]}\right)\right\} \tag{4.28}$$

For the neural net shown in Figure 4.2, the Equations (4.29) through (4.32) give the values for the derivatives of the output with respect to each of the parameters based on Equation (4.23).

$$\frac{\partial y}{\partial w_1} = \frac{w_2 x\, e^{-w_1 + b_1}}{\left(1 + e^{-w_1 + b_1}\right)^2} \tag{4.29}$$

$$\frac{\partial y}{\partial b_1} = \frac{-w_2\, e^{-w_1 + b_1}}{\left(1 + e^{-w_1 + b_1}\right)^2} \tag{4.30}$$

$$\frac{\partial y}{\partial w_2} = \frac{1}{\left(1 + e^{-w_1 + b_1}\right)} \tag{4.31}$$

$$\frac{\partial y}{\partial b_2} = -1 \tag{4.32}$$

CHAPTER 5
APPLICATION OF CONFIDENCE INTERVAL PREDICTION FOR NEURAL NETWORKS

To determine the efficacy of the modeling capabilities of neural networks, various neural network models will be tested on a number of test bed problems which represent the problems typically encountered in manufacturing processes and systems. In this chapter, the first type of problem to be tested on neural networks is the presence of noise in experimental data. Systems have error associated with it due to the dependence of the output on uncontrollable or unobservable quantities, and the quality of the model developed from data containing such errors will be compromised to a certain degree. The method developed to estimate the confidence intervals of neural network models in section 4.3 will be used to assess the reliability of neural network models.

The first test problems for study are two arbitrary multivariate functions, the second test problems for study are the laser machining processes of laser through-cutting and laser grooving, and the final test problem for study is the manufacturing system design of an automobile steering column. Only the test problems for the multivariate functions and the manufacturing of a steering column will contain the two types of test problems, namely data without noise and data with noise, since the sparseness of available data for laser machining would not give a good statistical indication of the effects of noise on the generated models.

5.1. Test Problem: Multivariate Test Functions

Two arbitrary, multivariate functions were developed to gain some basic understanding of the nature of the confidence intervals for neural network models. The first equation, given in Equation (5.1), is a three-input (w, x, y) and one-output (z) function containing linear and nonlinear terms. The second equation, given in Equation (5.2), was used by Friedman as a test problem for his Multivariate Adaptive Regression Splines (MARS) algorithm, which will be covered in Chapter 6 [Friedman, 1991]. This equation is a five-input (x_1, x_2, x_3, x_4, x_5) and one output (f) function also with linear and nonlinear terms.

$$z(w, x, y) = \frac{\sin(2w)}{1 + 0.1w} + \exp(0.1y)\cos(y)\{-\log[0.2(x + 1)]\} + 0.2x \tag{5.1}$$

$$f(x_1, x_2, x_3, x_4, x_5) = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5 \tag{5.2}$$

5.1.2. Neural Network Modeling of the Multivariate Test Functions

The training data for the learning of the neural network for the test function given in Equation (5.1) consisted of two sets of generated datapoints, where variable w ranged between 0.1 and 1, variable x ranged between 1 and 10, variable y ranged between 0.01 and 0.1, and the output z varied between 1.0 and 2.5. Each set of data contained 125 datapoints, where the first set contained uncorrupted data, and the second set contained a normally distributed noise with a standard deviation of 0.1611 added to the output. An additional 27 datapoints were set aside for the testing of the models. A 3-7-1 structure for the neural networks were used for modeling Equation (5.1). The predictions from the neural network trained with noiseless data with 80% estimated confidence intervals are shown in Figure 5-1, and the predictions from the neural network trained with noisy data with 80% estimated confidence intervals are shown in Figure 5-2. For the neural network modeling the noiseless data, only one of the 27 true values corresponding to the prediction cases (case 6) lies outside of intervals, which translates to 96% of the points lie within the confidence intervals. None of the true values corresponding to the prediction cases from the model trained with the noisy data lied outside the confidence intervals, but it can be seen that these confidence intervals span a much wider range than the intervals for the neural network trained with noiseless data.

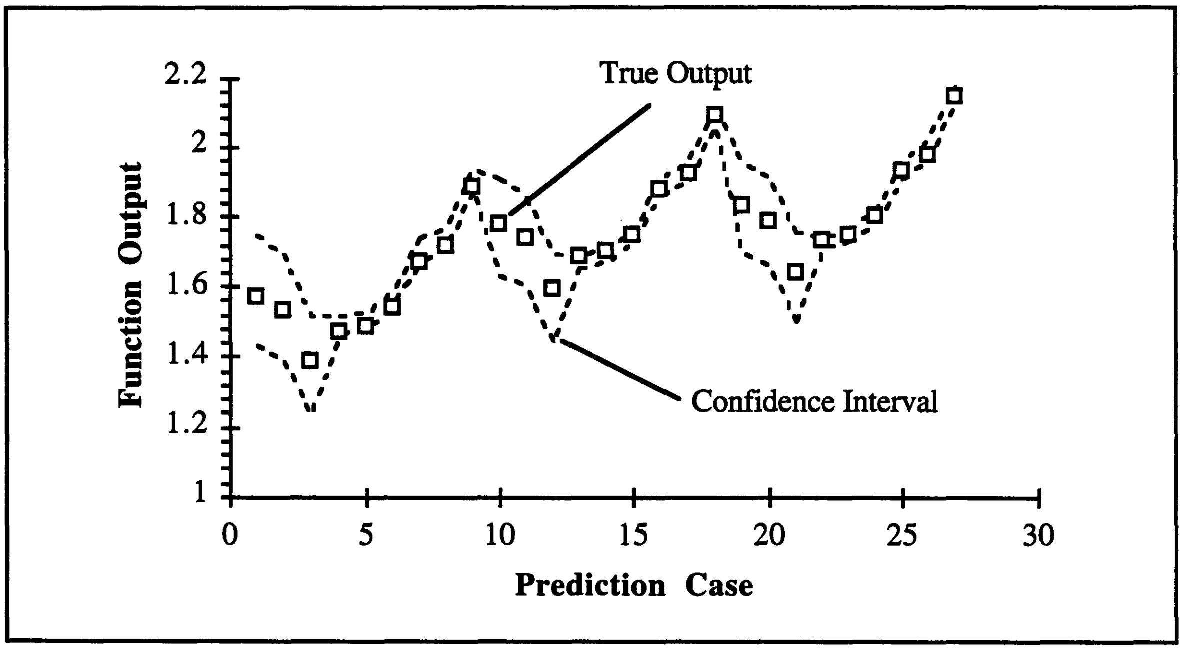


Figure 5-1: 80% confidence intervals for the neural network model of function 1.

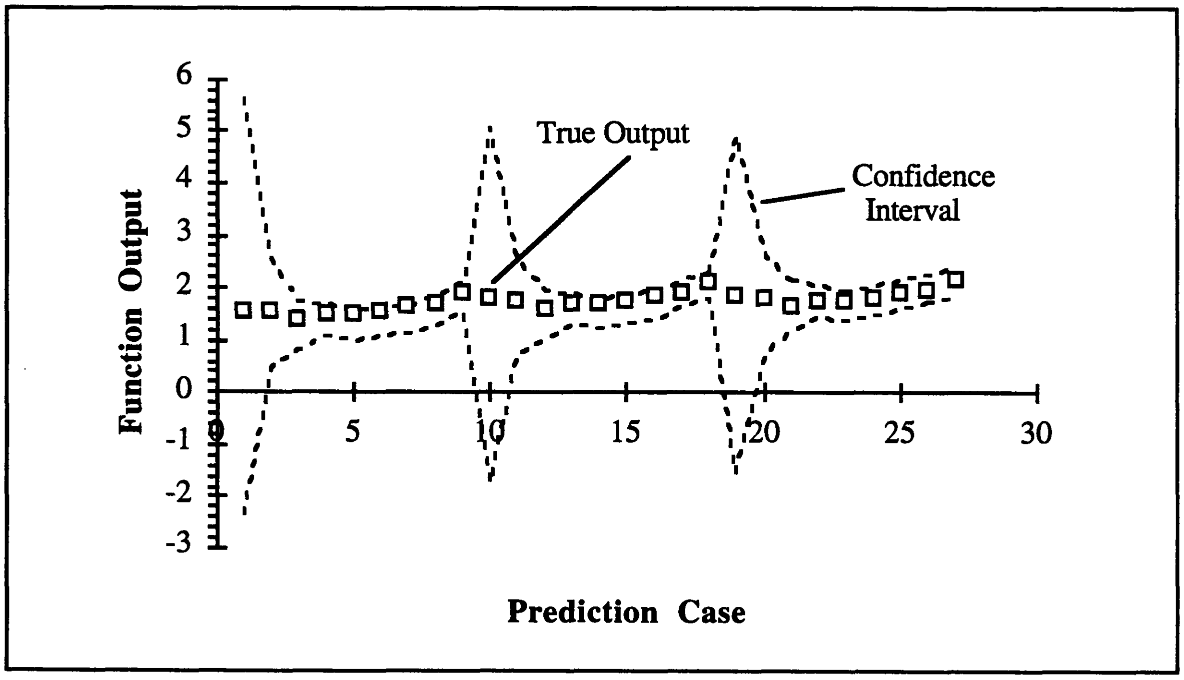


Figure 5-2: 80% confidence intervals for the neural network model of function 1 containing noise.

The training data for the learning of the neural network for the test function given in Equation (5.2) also consisted of two sets of generated datapoints, where variable x_1, x_2, x_3, x_4, and x_5 ranged between 0.1 and 1, and the output f varied approximately between 5.0 and 25.0. Both the uncorrupted and noisy data contained 243 datapoints, where the noisy data set contained a normally-distributed noise with a standard deviation of 0.2756. 32 additional datapoints were set aside for the testing of the models. A 5-8-1 structure for the neural networks were used for modeling Equation (5.2). The predictions from the neural network trained with noiseless data with 80% estimated confidence intervals are shown in Figure 5-3, and the predictions from the neural network trained with noisy data with 80% estimated confidence intervals are shown in Figure 5-4. For the neural network modeling the noiseless data, 17 of the 32 true values corresponding to the prediction cases lie outside the confidence intervals, or 46.9% of the predictions lie within the confidence intervals. For the neural network modeling the noiseless data, 20 of the 32 true values corresponding to the prediction cases lie outside the confidence intervals, or 37.5% of the predictions lie within the confidence intervals.

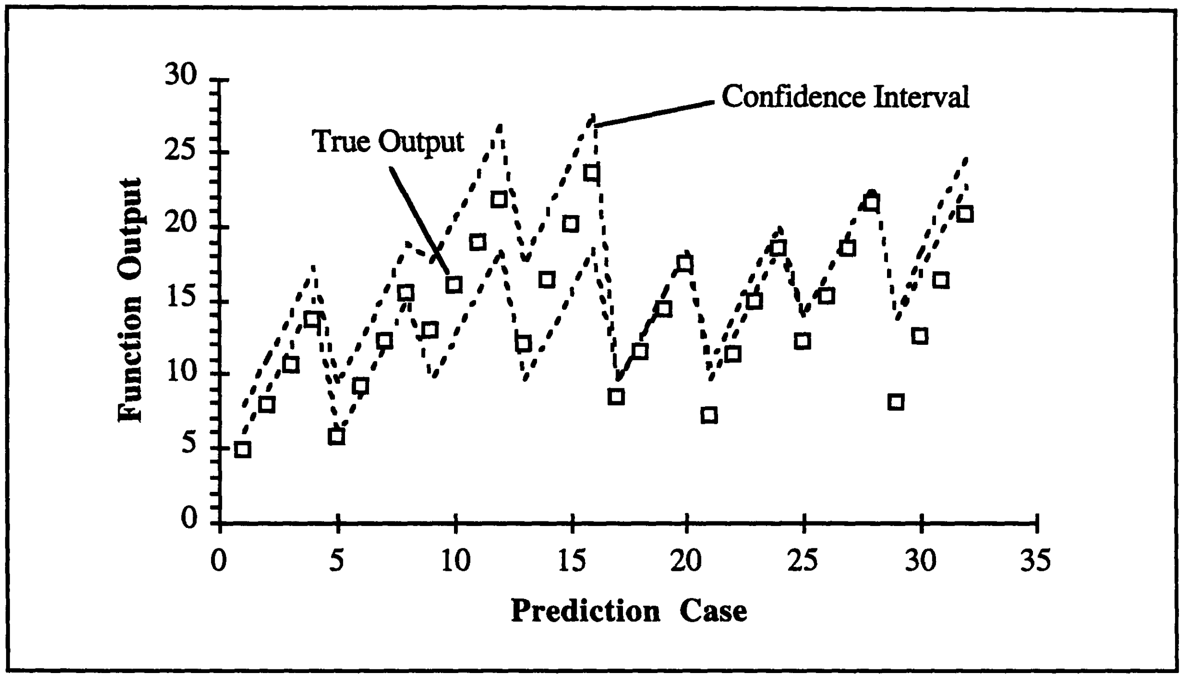


Figure 5-3: 80% confidence intervals for the neural network model of function 2.

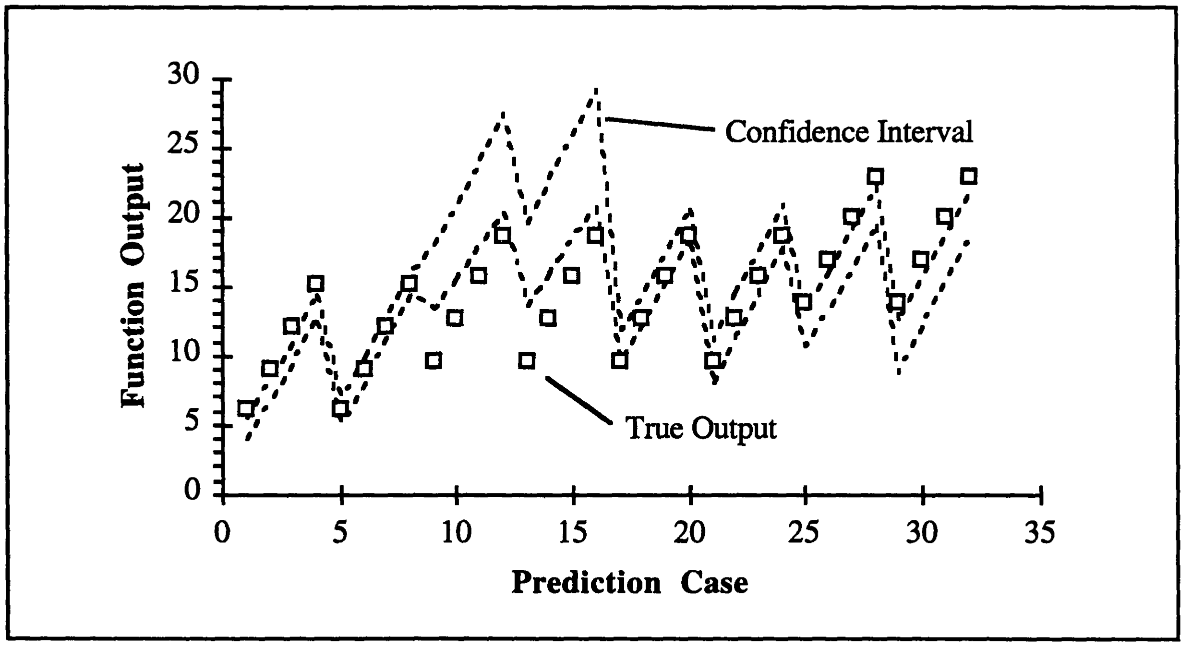


Figure 5-4: 80% confidence intervals for the neural network model of function 2 containing noise.

5.1.3. Neural Network Reliability

For the neural network model for Equation (5-1), since significantly more than 80% of the predictions lie within the computed confidence intervals, conservative confidence intervals were given. Nonconservative confidence intervals were computed for the neural network model for Equation (5-2), since less than 80% of the predictions lie within the computed confidence intervals. Table 5-1 summarizes this information.

Neural Network Model	Percentage of predictions which lie within confidence intervals	RMS error of the model with training data
Equation (5-1)	96%	0.01
Equation (5-1) with noise of 0.1611 standard dev.	100%	0.14
Equation (5-2)	46.9%	0.19
Equation (5-2) with noise of 0.2756 standard dev.	37.5%	0.29

Table 5-1: Summary of the predictions which lie within the 80% confidence intervals and accuracy of fit for the models to the training data.

From Figures 5-1 to 5-4 and from Table 5-1, two observations can be made. First, the confidence intervals for functions with noise are much wider than the intervals for functions without noise. This behavior is due to the accuracy of the neural network model in mapping the training data. Table 5-1 gives the RMS error of the neural network models in fitting with the training data, and the trend of a more accurate mapping can be seen for the models trained without noise when compared with the fit for the models trained with noise.

The second observation from the results is that for the first function, the confidence intervals do give consistently either a conservative or a nonconservative confidence intervals. For the first function in both cases of with and without noisy data, the confidence intervals were conservative estimates, while the second function in both cases of with and without noisy data, the confidence intervals were nonconservative estimates.

5.2. Test Problem: Laser Machining

Another test problem for study entails the modeling of laser machining. Two specific laser machining processes will be studied: laser cutting and laser grooving. For laser cutting, the problem involves the prediction of the maximum cut velocity. For laser grooving, the problem involves the prediction of the groove depth. This section will begin by covering the background of these processes. Following will be the results of modeling experimental data with neural networks will be given. Finally, results of determining the confidence intervals of the neural network models will be given.

5.2.1. Background

Laser machining belongs to the large family of material removing or machining processes. It can replace mechanical material removal methods in many industrial applications, particularly in the processing of difficult-to-machine materials such as hardened metals, ceramics, and composites. Furthermore, laser beams themselves make new material removal methods possible, due to their unique characteristics. The following sections provide background information on two laser machining processes studied for modeling. A further reference on laser machining can be obtained from [Chryssolouris, 1991].

Cutting is the two-dimensional machining of a workpiece using laser. In the laser through-cutting process, a kerf is created through relative motion between the laser beam and the workpiece surface. This process allows intricate two-dimensional shapes to be cut on a flat workpiece. The physical mechanisms for material removal and energy losses occur from the balance of energy from the incoming laser beam with the energy consumed by the conduction heat, energy for melting or vaporization of material, and heat losses to the environment (Figure 5-5). When material is removed through melting, a molten layer forms at the erosion front. The accumulated molten material can be expelled out from the bottom of the kerf with the aid of a coaxial gas jet.

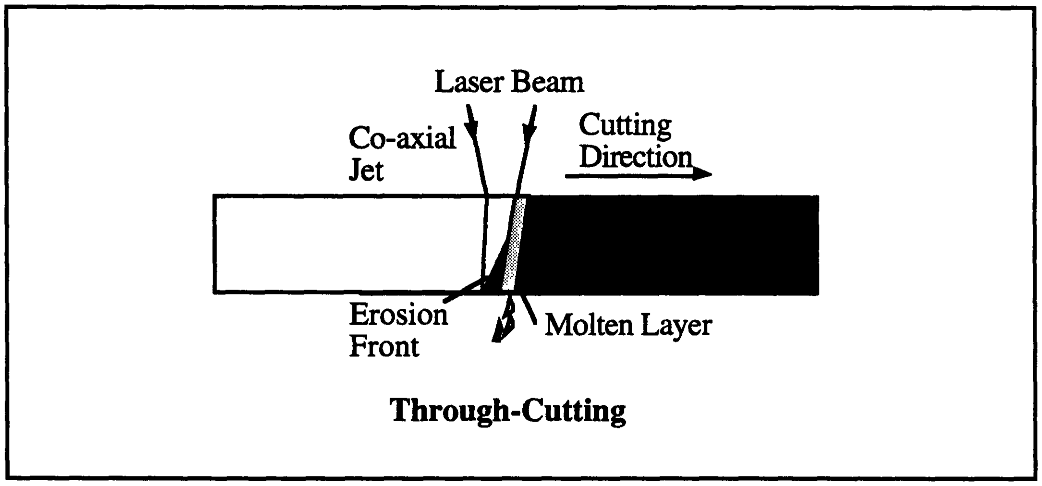


Figure 5-5: Laser through cutting.

Laser grooving entails the material removal of a workpiece to a certain depth less than the thickness of the workpiece. In the laser grooving process, a groove is produced by scanning a laser beam over the workpiece surface, but the laser beam does not penetrate through the entire workpiece thickness. The physical mechanisms of laser grooving are similar to those of cutting. Using the process of laser grooving, the concept for three-dimensional material removal has been developed to make laser machining more applicable to bulk material removal. The three-dimensional material removal process uses two intersecting laser beams to remove a volume of material. Unlike laser through-cutting techniques, each beam creates a groove in the workpiece through either single or multiple passes. A volume of material is removed when the two grooves intersect.

Two such applications employing the laser grooving process are laser turning and laser milling. Turning operations can be accomplished by ring removal or helix removal (Figure 5-6). The ring removal method uses two perpendicular beams to remove concentric rings from a workpiece, and the helical removal method uses two angular beams to create a continuous thread. For the case of laser milling, two laser beams are positioned at oblique angles from the workpiece surface to create converging grooves in a workpiece (Figure 5-7). The volume of material removed is prismatic in shape with a triangular cross sections. Dimensional accuracy is related particularly to the taper angle for each of the two grooves.

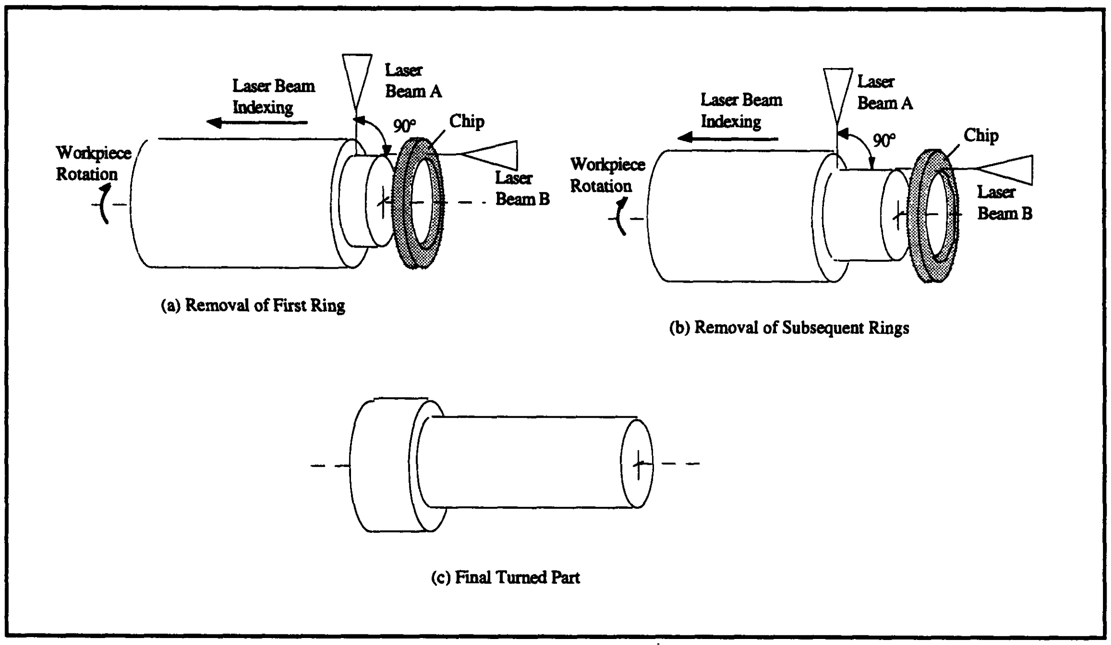


Figure 5-6: Using three-dimensional laser grooving for ring removal.

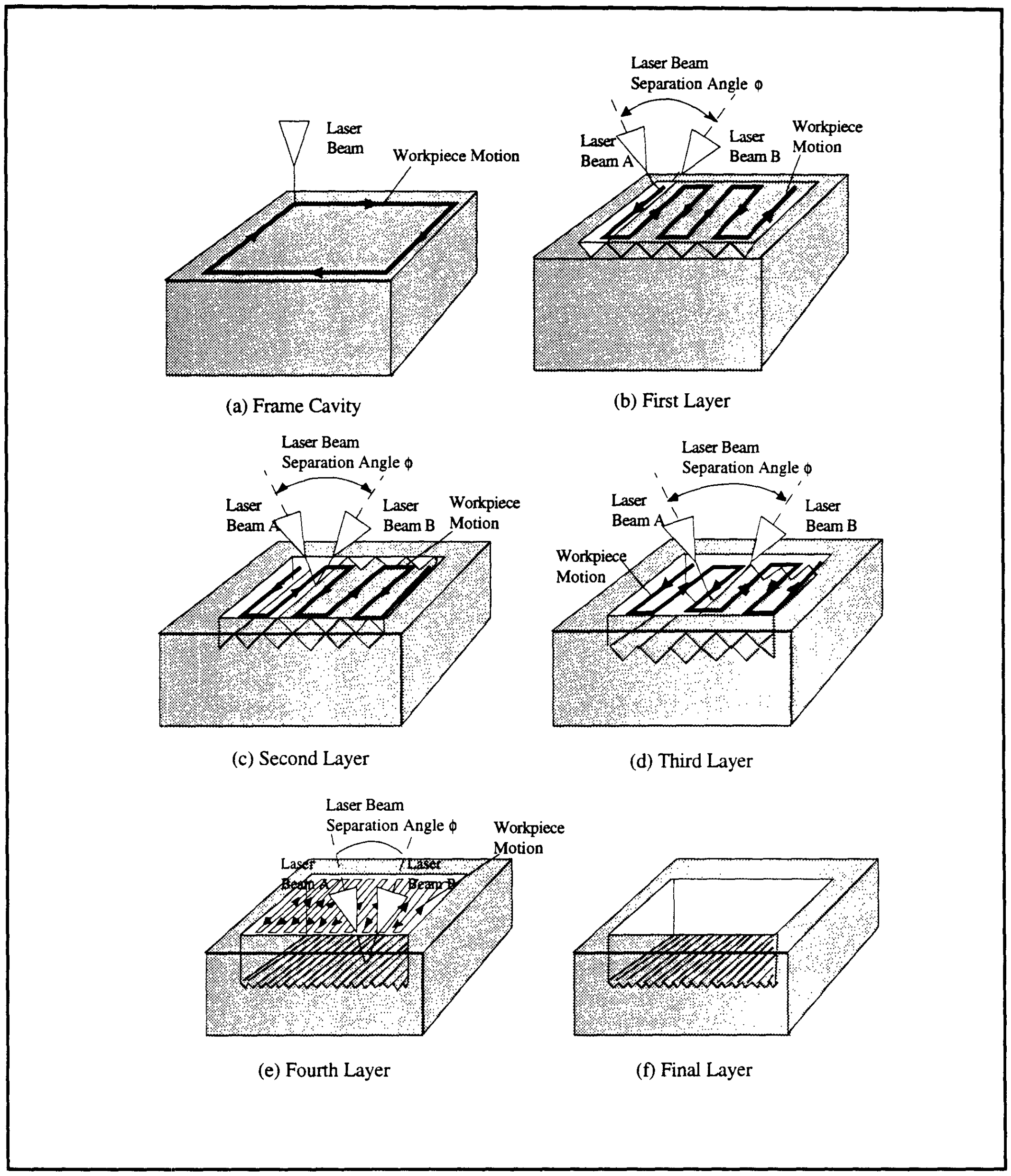


Figure 5-7: Using three-dimensional laser grooving for milling.

5.2.2. Neural Network Modeling of Laser Through-Cutting

For laser cutting, nine cutting experiments were performed to measure the cut velocity. Two parameters were varied for each experimental setup. The laser power varied between the values from 300 to 500 Watts, and the steel workpiece thickness varied between 0.76 and 1.91 mm. The gas jet pressures were set to 28 psi. and the nozzle distance was 1.3

mm. Each experimental setup was repeated 10 times and the final data used for developing the neural network models incorporated the mean cut velocity. The standard deviation from the 10 experimental measures of the cutting velocity was treated as the standard deviation of the noise to the output.

A 2-1-1 neural network model has been created to map the experimental data from the laser cutting experiments. From the nine cutting experiments, seven were used to train the data, and two were used for testing the accuracy of the neural network model. The size of the neural network model was restricted to be small to allow the computation of the confidence intervals of the neural network outputs. The value of the degree of freedom, which equals the number of training datapoints minus the number of parameters (links) in the model (from chapter 4, the value n - p), must be at least 1 to obtain the student's t-distribution value used for Equation (4.12). Confidence intervals of 80% confidence level were constructed, and the results are illustrated in Figure 5-8. From the figure, the estimated confidence levels succeeded to contain the true values of the cut speed for both predictions.

5.2.2. Neural Network Modeling of Laser Grooving

For laser grooving, sixteen grooving experiments were performed to measure the groove depth. Three parameters were varied for each experimental setup. The laser power varied between the values from 400 to 600 Watts, the translational velocity of the laser varied between 1 and 9 mm/s, and the off axial gas pressure varied between 0 and 70 psi. The nozzle distance was 2.4 mm. Each experimental setup was repeated 20 times and the final data used for developing the neural network models incorporated the mean groove depth. The standard deviation from the 20 experimental measures of the cutting velocity was treated as the standard deviation of the noise to the output.

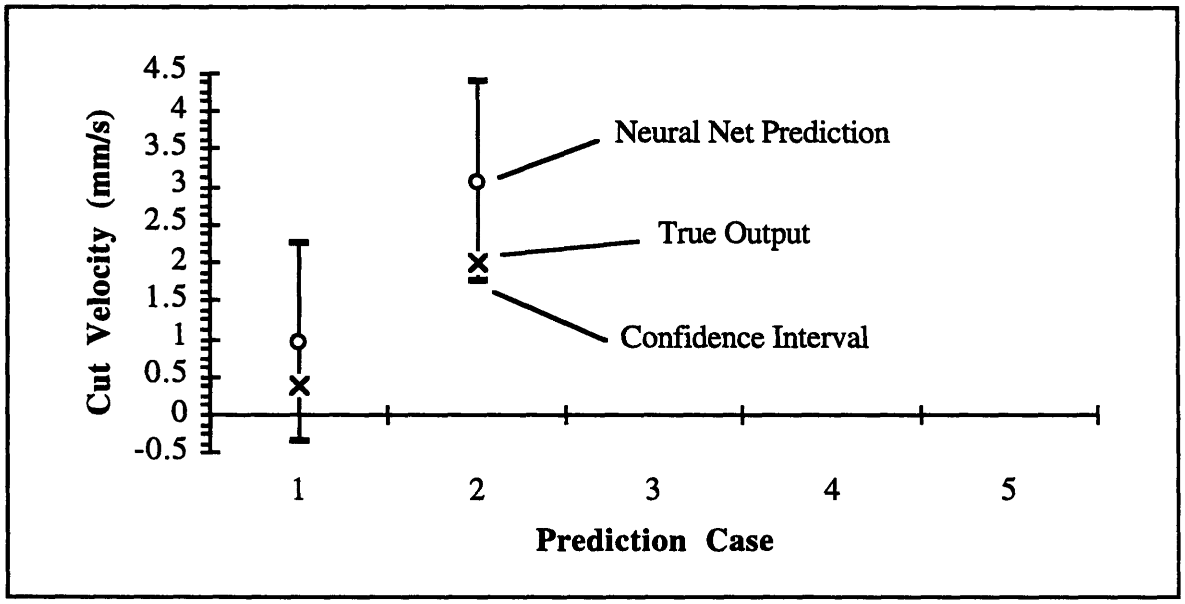


Figure 5-8: 80% Confidence intervals for the neural network model of laser cutting.

A 3-2-1 neural network model has been created to map the experimental data from the laser cutting experiments. From the sixteen grooving experiments, fourteen were used to train the data, and two were used for testing the accuracy of the neural network model. For the same reason given for modeling the laser cutting process, the size of the neural network model was restricted to be small. Confidence intervals of 80% confidence level were constructed, and the results are illustrated in Figure 5-9. From the figure, the estimated confidence levels again succeeded to contain the true values of the cut speed for both predictions.

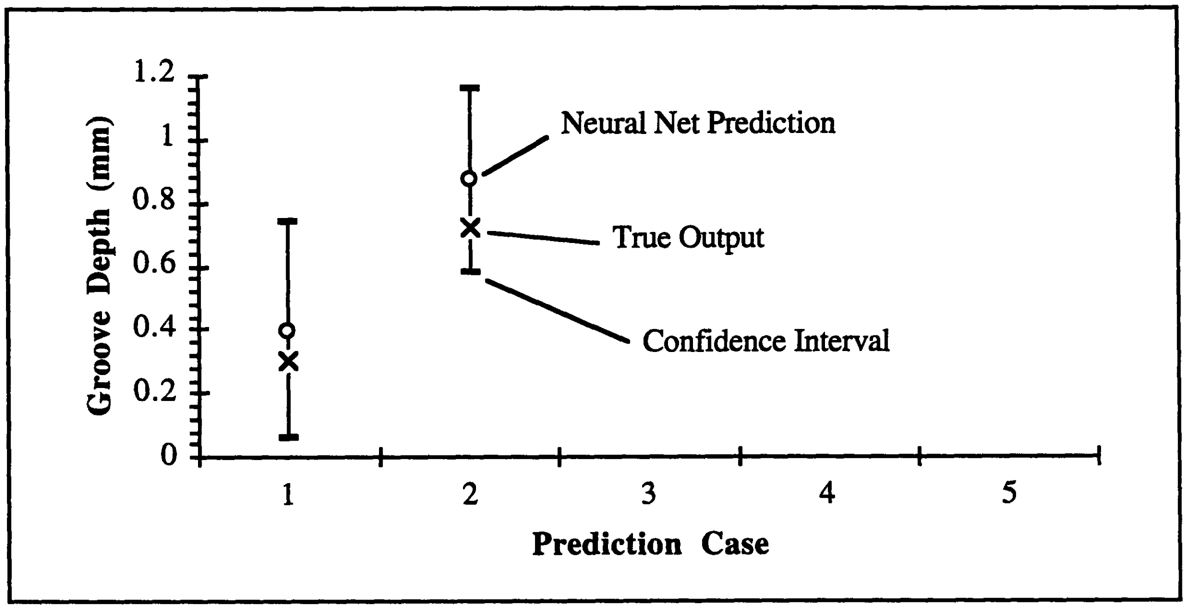


Figure 5-9: 80% Confidence intervals for the neural network model of laser grooving.

5.2.3. Neural Network Reliability

For the test problems of laser machining, the confidence intervals estimated for laser through cutting and laser grooving succeeded to contain the true values for the test cases. Thus, the proposed method of estimating confidence intervals for neural networks gave appropriate intervals for the neural network models of laser machining.

5.3. Test Problem: The Manufacturing of an Automobile Steering Column

An example taken from the automotive industry used for this thesis deals with the assembly of three different models of automobile steering columns belonging to the same general product. It is a hypothetical example based on the assembly specifications of a real product which also has been investigated in the research of Graves and Redfield [Graves]. The goal is to estimate the performance of the manufacturing system for a given system design. For this problem, a framework to determine the efficiency is required. In the following sections, an overview of the basis for such a framework will be given, followed by an explanation of some the significant assembly specifications and assumed information required for the example problem. Simulation was the approach used for generating experimental data, and the final section gives the results of the neural network modeling and prediction of the efficiencies for the manufacturing systems. The simulation package used was WITNESS 5.0 by ISTEL Inc.

5.3.1. An Evaluation Framework for Manufacturing Systems

For this thesis, an evaluation framework encompasses the definition of manufacturing system decision variables and the definition of a performance index [Chryssolouris *et al.* 1990], which differs from existing frameworks [Kaplan (1983), Suresh and Meredith (1985), Wabalickis (1988), Swamidass and Waller (1990), Son (1991)] primarily in the combination of specifically defined decision variables and the consideration of costs that occur over the anticipated life of the system, particularly those due to part design changes.

A manufacturing system can be defined as a combination of humans with machinery and equipment that are bound by a common material and information flow. The configuration of a such a system can be viewed as the process of mapping the system's performance requirements (specified via numerical *performance measures*) onto a description of a physical system (specified via numerical *decision variables*) which will achieve the required performance (Fig. 5-10).

Given performance requirements, the task of manufacturing system configuration requires the description of a suitable physical system by specifying a body of *information* [Suh]. This body of information can be represented either symbolically, in the form of a drawing, or numerically, in the form of values of a collection of decision variables. Symbolic representations are advantageous because each symbol can represent a large collection of information and because they are easily interpreted by humans. Numeric (decision variable-based) representations are advantageous because they can be easily stored and manipulated by computers. For this thesis we will restrict our attention to numeric decision variables, because they are easier to manipulate automatically and thus would easier to incorporate into a procedure which automates the configuration process.

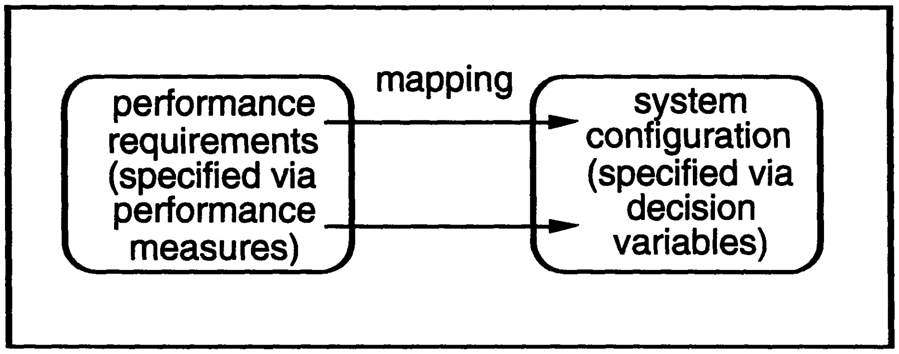

performance requirements (specified via performance measures)
mapping
system configuration (specified via decision variables)

Figure 5-10: Configuration of a manufacturing system.

Once the decision variables are defined, an evaluation framework requires the definition of a performance index. The performance index used in this thesis accounts for the inputs to a manufacturing system in terms of the costs incurred over the life of the system – particularly those related to part design change. Thus the evaluation framework considers the flexibility of a manufacturing system. The performance index also accounts for the output that is achieved by a manufacturing system, in terms of the number of parts that it produces. This index is called efficiency, and has the general form *output/input.*

$$\text{efficiency} = y = \frac{\text{number of good parts produced during system life cycle}}{\text{total life cycle cost}} \qquad (5\text{-}3)$$

The units of this index are [parts/\$]. If a figure for the revenue per part is available, then e can be converted to a unitless efficiency. The denominator, the total life cycle cost, consists of acquisition costs, operation costs and system modification costs due to part design change. These costs are incurred over the life T of the system, which consists of n_t periods of duration t ($T = n_t t$). The costs, described Figure 5-11, are broken down by period and are incurred at the beginning of each period.

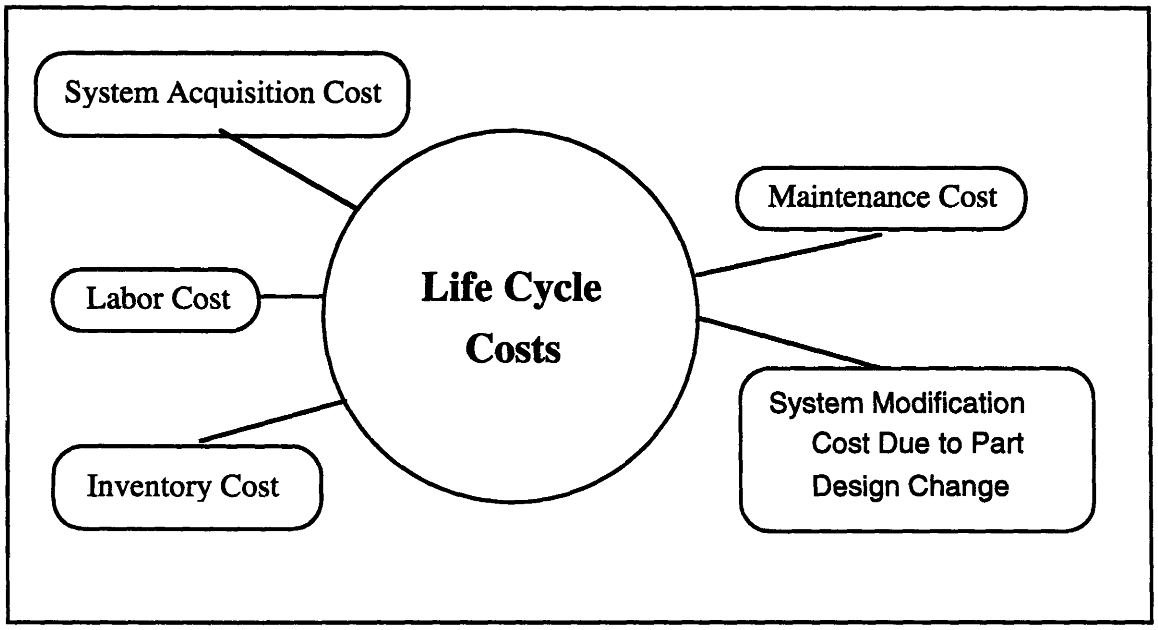


Figure 5-11: Summary of total life cycle costs

System acquisition cost is incurred not only when the system is first implemented, but also when increased demand requires an expansion of system capacity. The system modification cost due to part design change is an important and new contribution of the efficiency definition, because it accounts for the flexibility of the system. This cost is a function of the probability that at least one of the features worked on by a machine in the system will be modified whenever the part design changes. This probability is a function of the probabilities of individual features requiring change when the part design changes and also of the number of features processed by the machine.

5.3.2. The Manufacturing of an Automobile Steering Column

This section will cover the significant assembly system specifications and assumed information made for generating the simulations. Three different models of steering columns have to be assembled on one assembly line. Model 1 has most options which includes a turn/cruise lever, a tilt lever and a hazard switch. Model 2 is the basic model, which has no options. Model 3 is an alternative option, which has no tilt lever and hazard switch. Because of the models belonging to one general product, they have most parts in common. The general product consists of eleven different parts: steering column, bracket, bolts for bracket, turn/cruise lever respectively turn lever (for model 2), steering wheel, nut, retainer, damper, horn pad, hazard switch, and tilt lever. Except of the hazard switch and the tilt lever, all parts are schematically shown in Figure 5-12.

For assembling the three models, 28 different tasks have to be processed. Three candidate resource types are assumed to be available for the steering column assembly: an operator, a robot and a paint machine. Each is able to process a particular set of tasks and requires certain tools for processing these tasks. Table 5-2 gives an overview of the available resource types, the task times, the tools required for processing the tasks and the cost of each tool.

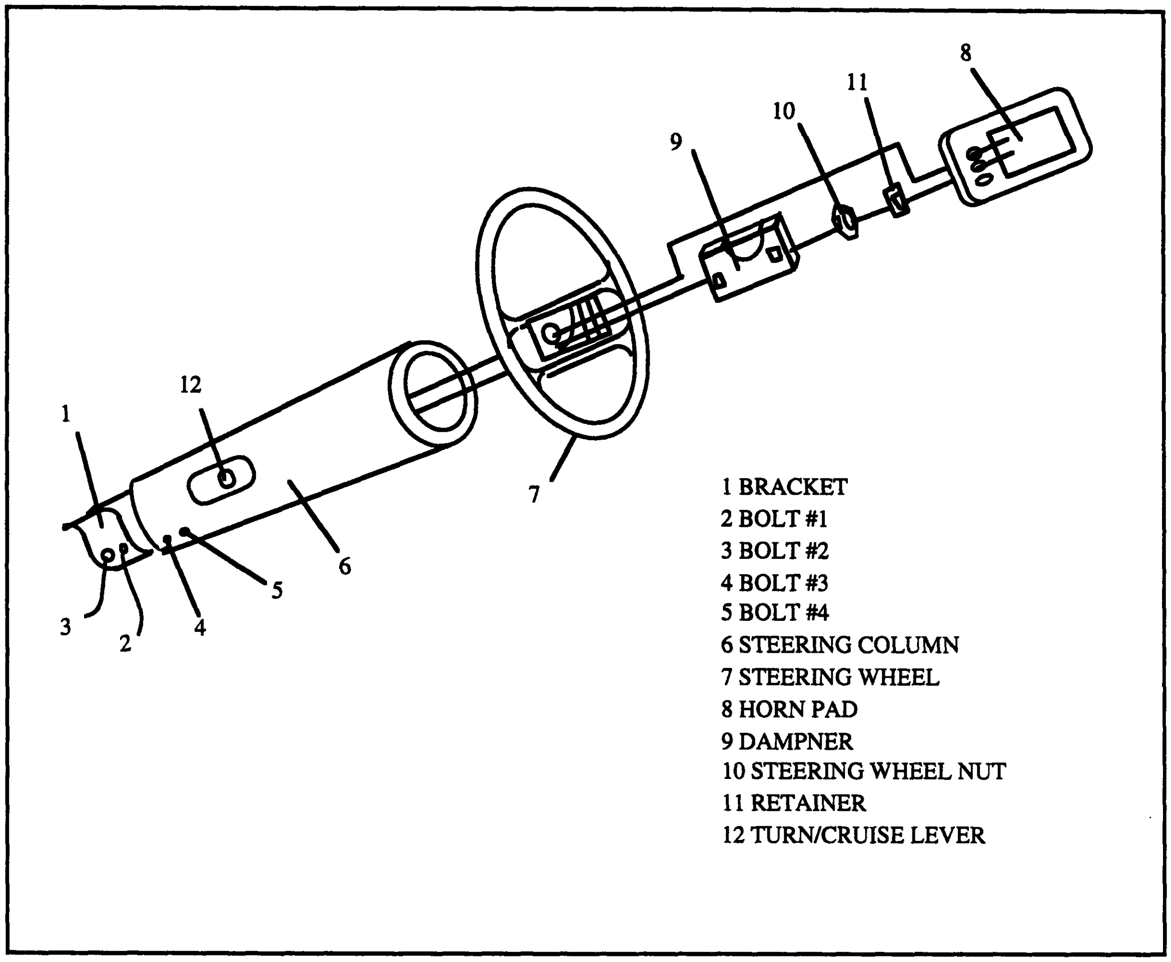


Figure 5-12: Schematic of the automobile steering column (hazard switch and tilt lever are not shown).

Except for task 6 (which has to be done by a paint machine), an operator can to process any of the tasks given in Table 5-2. The alternative resource of the operator for processing tasks 7 to 22 can be a robot . The processing times are given in the first of the three columns belonging to each resource type. In the second and third column, the tools required by each resource for processing each task and the acquisition costs of the tools are listed.

	Manual labor			Robot			Paint Machine		
	Task time	Tool no.	Tool cost	Task time	Tool no.	Tool cost	Task time	Tool no.	Tool cost
Task no. and description	(sec)		($)	(sec)		($)	(sec)		($)
1 Schedule steering column	23	100	0	-			-		
2 Bracket & bolt to column	8	100	0	-			-		
3 Finger-start bolt #2	3	100	0	-			-		
4 Finger-start bolt #3	3	100	0	-			-		
5 Finger-start bolt #4	3	100	0	-			-		
6 Paint steering column	-			-			43	401	7 875
7 Schedule steering wheel	17	100	0	20	201	5 675	-		
8 Schedule horn pad	9	100	0	11	201	5 675	-		
9 Steering wheel to column	7	100	0	6	202	8 925	-		
10 Place damper to column	7	102	9 975	6	203	12 600	-		
11 Drive steering wheel nut	6	103	9 975	3	204	12 600	-		
12 Inspect nut torque	2	103	9 975	2	204	12 600	-		
13 Install nut retainer	2	104	1 313	1	205	5 775	-		
14 Inspect retainer	2	100	0	5	206	12 500	-		
15 Install turn/cruise lever	14	105	525	12	207	9 650	-		
16 Install turn lever	8	100	0	4	207	7 350	-		
17 Horn pad to wheel	8	100	0	5	208	8 925	-		
18 Install tilt lever	7	100	0	3	209	6 825	-		
19 Secure bracket bolt #1	3	108	6 825	1	210	14 700	-		
20 Secure bracket bolt #2	3	108	6 825	1	210	14 700	-		
21 Secure bracket bolt #3	3	108	6 825	1	210	14 700	-		
22 Secure bracket bolt #4	3	108	6 825	1	210	14 700	-		
23 Test bracket secureness	2	100	0	-			-		
24 Test horn pad secureness	2	100	0	-			-		
25 Install hazard switch	9	109	1 313	-			-		
26 Test turn/cruise lever	21	110	15 750	-			-		
27 Test hazard switch	7	110	15 750	-			-		
28 Electrical-test horn pad	12	110	15 750	-			-		

Table 5-2: Available resource types and required tools for processing the tasks.

Table 5-3 gives an overview of the costs for the resources. The costs considered are acquisition costs, fixed maintenance costs, variable maintenance costs which occur during break down times of resources, and the design change costs of the robot and the paint machine. Furthermore, the labor cost rate for the operators is given. These costs are estimated based on data provided by industry for similar equipment.

	robot	paint machine	operator
acquitsition cost ($)	527,900	141,300	-
fixed maintenance cost rate ($/year)	16,700	5,200	-
variable maintenance cost rate ($/hour)	50	50	-
design change costs ($)	1,300	21,000	-
labor cost rate ($/hour)	no labor required	no labor required	30

Table 5-3: Costs of resources.

Additional information for calculating the efficiency is given in Table 5-4. The number of working days per year is assumed to be 235, the number of shifts per day should be 2, and the time duration of one shift is set to 8 hours. Thus, the total available working time is 3760 hours per year. With an assumed total annual volume of 500,000 units per year, the system cycle time has to be less than 27 seconds per part. It is distributed in 63% for model 1, 29% for model 2, and 8% for model 3. The tool change times of an operator are given with 2.5 seconds, whereas a robot has a tool change time of 2.0 seconds.

number of working days per year	235
number of shifts per day	2
time duration of one shift (hours)	8
total available working time per year (hours)	3,760
total annual volume (parts)	500,000
fraction of the total annual demand for model 1	0.63
fraction of the total annual demand for model 2	0.29
fraction of the total annual demand for model 3	0.08
required system cycle time (seconds)	27
tool change time of an operator (seconds)	2.5
tool change time of a robot (seconds)	2.0

Table 5-4: Additional information for calculating the efficiency.

Table 5-5 gives the tasks required for each model. From the assumed total annual demand of 500,000 units and the fractions of 63% for model 1, 29% for model 2, and 8% for model 3, the annual demand of the model is 315,000 units of model 1 per year, 145,000 units of model 2 per year, and 40,000 units of model 3 per year.

Model 1	0.63 * (total annual demand) = 315,000 units/year														
	Tasks:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		15		17	18	19	20	21	22	23	24	25	26	27	28
Model 2	0.29 * (total annual demand) = 145,000 units/year														
	Tasks:	1	2	3	4	5	6	7	8	9		11	12	13	14
			16	17		19	20	21	22	23	24				28
Model 3	0.08 * (total annual demand) = 40,000 units/year														
	Tasks:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		15		17		19	20	21	22	23	24		26		28

Table 5-5: Task data per model.

The tardiness cost rate and the inventory carrying cost per part per period for each model have to be set. Table 5-6 gives the assumed values of these rates. The design change costs of tools are assumed to be equal to their acquisition costs. Furthermore, it is assumed that a product design change occurs once every 3 years. The probability of individual feature changing, when a product design change occurs, is set to 0.9 and the interest rate is assumed to be 10%.

	Model 1	Model 2	Model 3
Tardiness Cost Rate ($/(parts*sec))	12	9	11
Inventory Carrying Cost per part ($/year)	300	240	280

Table 5-6. Tardiness cost rates and inventory carrying costs.

Assembly systems used for the given steering column assembly problem are assumed to consist of two main elements: work stations and buffers. For simplification, variations in material handling systems will not be considered in the case study of the steering column.

It is also assumed that each work station can hold one or more resources and that no two different resource types can be assigned to one work station. The fact that task 6 and only this task has to be performed by a pain machine, the minimum number of work stations is three. Tasks 1 to 5 can only be performed manually. Therefore, it is reasonable to assign these tasks to one work station. Thus, all assembly systems generated in the given example of the steering column are assumed to have the first two work stations in common, at which tasks 1 to 6 are processed. The maximum number of work stations is set to eight.

Buffers are characterized by the buffer capacity and the buffer frequency. The capacities of all buffers in one assembly system are assumed to be equal. In general, the capacity is assumed to vary in the interval between 10 and 100 parts per buffer. The buffer frequency depends on the number of work stations used in one assembly system. It represents the quotient of the number of installed buffers in the assembly system to the potential number of buffer locations which is x-1 in case of an assembly system with x work stations.

The convention for setting buffers used for this problem is illustrated in Table 5-13. In the first tow column the number of work stations and the number of installed buffers are used for identifying the different cases. The corresponding buffer frequency is given in the third column. The next seven columns build up a matrix which is characterizing the assignment of buffers to buffer locations. Buffer locations marked by a black circle signalize that a buffer is installed. If the circle is white the corresponding location is not occupied by a buffer. In case of an assembly system with less than eight work stations the infeasible buffer locations are marked by a stroke.

number of work stations	number of installed buffers	buffer frequency	buffer between WS1 and WS2	buffer between WS2 and WS3	buffer between WS3 and WS4	buffer between WS4 and WS5	buffer between WS5 and WS6	buffer between WS6 and WS7	buffer between WS7 and WS8
3	2	1.000	●	●	-	-	-	-	-
3	1	0.500	●	○	-	-	-	-	-
4	3	1.000	●	●	●	-	-	-	-
4	2	0.667	●	●	○	-	-	-	-
4	1	0.333	○	●	○	-	-	-	-
5	4	1.000	●	●	●	●	-	-	-
5	3	0.750	●	●	●	○	-	-	-
5	2	0.500	●	○	●	○	-	-	-
5	1	0.250	○	●	○	○	-	-	-
6	5	1.000	●	●	●	●	●	-	-
6	4	0.800	●	●	●	●	○	-	-
6	3	0.600	●	○	●	○	●	-	-
6	2	0.400	○	●	○	●	○	-	-
6	1	0.200	○	○	●	○	○	-	-
7	6	1.000	●	●	●	●	●	●	-
7	5	0.833	●	●	●	●	●	○	-
7	4	0.667	●	●	○	●	●	○	-
7	3	0.500	●	○	●	○	●	○	-
7	2	0.333	○	●	○	●	○	○	-
7	1	0.167	○	○	●	○	○	○	-
8	7	1.000	●	●	●	●	●	●	●
8	6	0.857	●	●	●	●	●	●	○
8	5	0.714	○	●	●	●	●	●	○
8	4	0.571	●	●	○	●	●	○	○
8	3	0.429	●	○	●	○	●	○	○
8	2	0.286	○	●	○	●	○	○	○
8	1	0.143	○	○	●	○	○	○	○

WS: work station
● buffer installed ○ buffer not installed - no feasable buffer location

Figure 5-13: Convention for setting buffers.

A total of 279 simulations were generated for this problem, where 225 points were designated for training and 54 for testing the neural network model. The problem has been broken down into a five variable problem for determining the efficiency:

1) Task allocation (15 possible allocations for a system to select)
2) Resource allocation (11 possible allocations for a system to select)
3) Buffer capacity
4) Buffer frequency
5) Steering wheel model

5.3.3. Neural Network Modeling of the Manufacturing of the Steering Column

Figure 5-14 shows the plot of the predictions from a 5-8-1 neural network. The figure also shows the estimated envelope calculated for 80% confidence intervals. 8 of the 54 predictions (85%) lie outside the confidence interval. Figure 5-15 shows the plot of the predictions from a 5-8-1 neural network trained with a normally distributed noise added to the output of the training data with a 0.0183 standard deviation. 78% of the predictions lie outside the confidence interval.

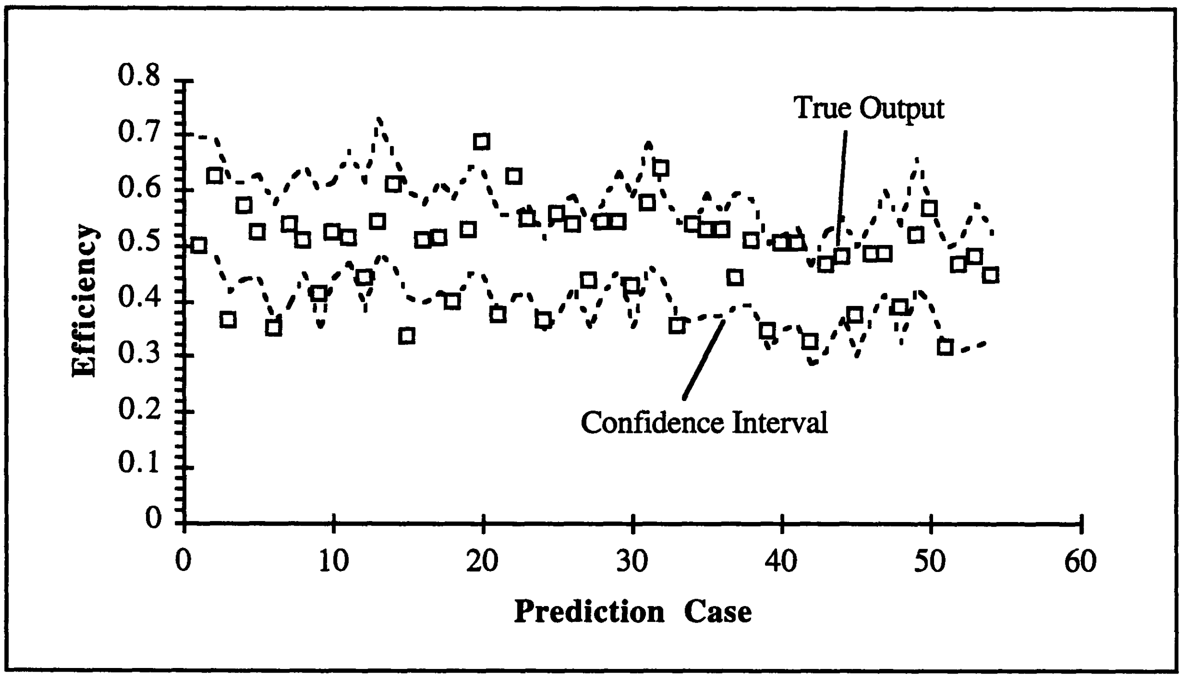


Figure 5-14: 80% confidence intervals for the neural network model predictions.

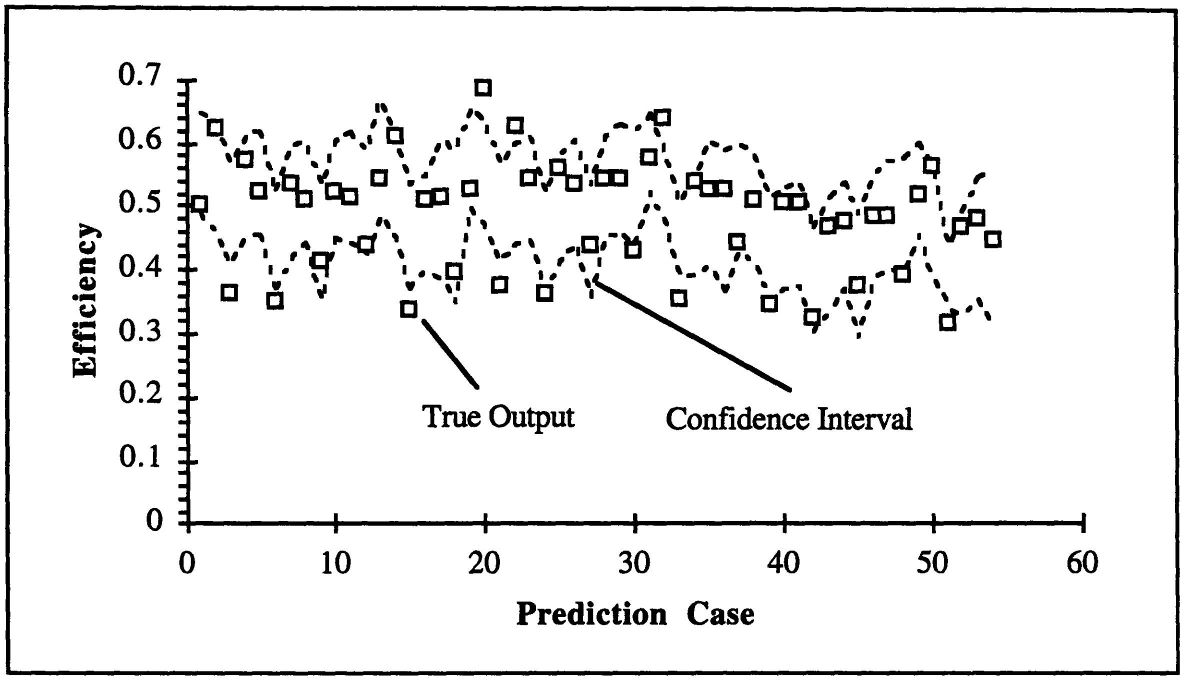


Figure 5-15: 80% confidence intervals for the neural network model predictions trained from noisy data.

For the neural network model trained with and without noisy data, the 80% confidence intervals gave expected results, since the estimated intervals did not include a significant amount of predictions above or below 80%. Table 5-7 summarizes this information.

Neural Network Model	Percentage of predictions which lie within confidence intervals
Trained without noise	85%
Trained with noise of 0.0485 standard dev.	76%

Table 5-7: Summary of the predictions which lie within the 80% confidence intervals.

5.3.4. Neural Network Reliability

For both neural network models trained with data containing noise and not containing noise, the confidence intervals estimated for the predicted values of efficiency contained the expected level of true values for the test cases. Approximately 80% of the computed efficiencies lie within the 80% confidence intervals. Thus, the proposed method of estimating confidence intervals for neural networks gave the appropriate levels of confidence for the neural network models for the networks trained with uncorrupted data and the network trained with data containing noise. The confidence intervals can then be used for assessing the reliability of the neural network predictions.

CHAPTER 6
RELATIVE PERFORMANCE OF NEURAL NETWORKS

In this chapter, the second type of problem tested on neural networks is the high level of nonlinearity present in an input-output relationship within manufacturing processes and systems. The relative efficacy of neural network modeling of nonlinear systems is determined by comparing results of the neural network models with results from models generated by other common empirical modeling approaches on the test bed problems. A direct comparison of results from three different modeling methods will be conducted for the assessment of the relative capability of neural networks as a modeling tool. The modeling methods used for comparison are linear regression, the Group Method of Data Handling (GMDH), and the Multivariate Adaptive Regression Splines (MARS) algorithms. This chapter begins by providing an overview of the three modeling methods. Subsequently, after presenting the results of modeling the test bed problems presented in Chapter 5, the assessment is given of the relative efficacy for each algorithm.

6.1. Linear Regression

For the case of one input variable *x*, and assuming that the statistical relationship between the response variable *y* and the input variable *x* is linear, a model of the relationship can be written as

$$y_i = \beta_0 + \beta_0 x_i + \varepsilon_i, \quad i = 1, 2, ..., \text{n}. \tag{6-1}$$

The following are the usual assumptions made for the parameters and variables in this model:

1) x_i is the ith observation on the input variable, where $x_1, x_2, ..., x_n$ correspond to particular settings of the input variable chosen *a priori* to the observation.
2) y_i is the output response that corresponds to the settings x_i of the input variable.
3) β_0 and β_1 are the coefficients, or parameters, in the linear relationship; β_0 is the intercept and β_1 is the slope.
4) The random variables $\varepsilon_0, \varepsilon_1, ..., \varepsilon_n$ are errors that create the scatter around the linear relationship $\beta_0 + \beta_0 x_i$, $i = 1, 2, ..., n$, respectively. We assume that these errors are identically and independently distributed with mean of zero and a variance σ^{2}.

To obtain an estimate for the parameters in the model such that $S(\beta_0, \beta_1)$, which is the sum of squares of the deviations of the model with respect to the observations and is given by Equation 6-2, the method of least squares can bed used. A multivariate linear regression model is obtained in the same method and the relationship is in the form similar to that given in Equation (1), but the variables and coefficients are replaced by vectors and matrices.

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} \left[y_i - (\beta_0 + \beta_1 x_i)\right]^2 \tag{6-2}$$

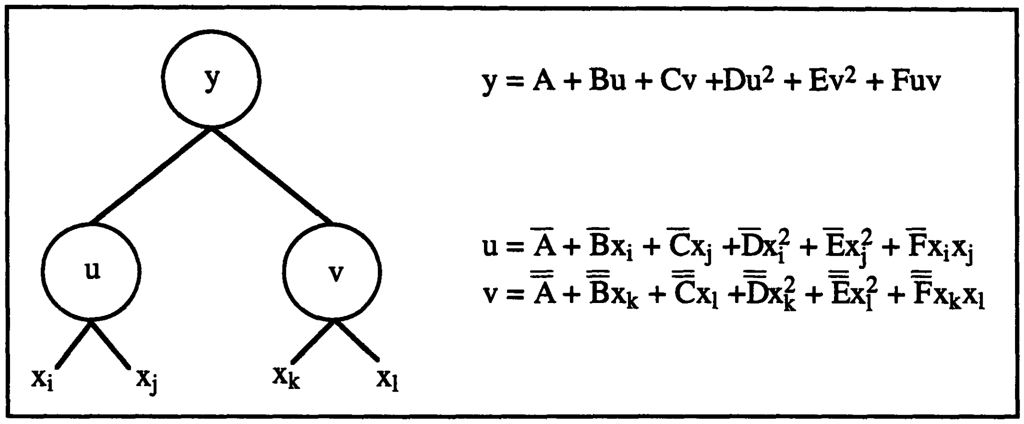
6.2. Group Method of Data Handling

The Group Method of Data Handling (GMDH) is a modeling technique which groups the input variables in a form of a polynomial regression equation to predict the output of a MISO system [Farlow]. Developed by Alexey Ivakhnenko, the transfer function for a GMDH model consists of a network of polynomial functions arranged in layers (or generations). In essence, this network consists of units which performs a series of second-order regression, where the form of the second-order equation is shown in Equation (6-3). The polynomial equation of the form given in Equation (6-3) has been often referred to as the Ivakhnenko polynomial The complete network is in essence a combination of Ivakhnenko polynomials, creating an empirical regression model of the function it is mapping.

$$u = A + Bx_i + Cx_j + Dx_i^2 + Ex_j^2 + Fx_ix_j \qquad (6\text{-}3)$$

In the first generation, two of the input variables are used to obtain an Ivakhnenko polynomial equation based on the training data collected to develop the model, and a root-mean square error is computed for that particular polynomial. Another pair of input variables are used to obtain an Ivakhnenko polynomial with a computed RMS error associated with that equation, and this process is repeated until all paired combinations of the input variables have been used. For m input variables, $\left(\frac{m(m-1)}{2}\right)$ polynomial equations can be developed. This equation is often presented in the form $\binom{m}{2}$.

After the algorithm computes an Ivakhnenko polynomial for all possible pairs of input parameters, the fist generation of equations have been developed. The output from the first generation of polynomials will be used as the input variables of the second generation of polynomials, where the RMS error is computed for each polynomial. Figure 6-1 illustrates how the output of the second generation Ivakhnenko polynomial can be determined from the input variables. The output from the second generation of polynomials will be used as the input for the third generation, and generations will continue to be developed until the GMDH obtains the optimal model. Figure 6-2 gives an illustration of how a GMDH model is structured.



y
u
v
x_i
x_j
x_k
x_l
$y = A + Bu + Cv + Du^2 + Ev^2 + Fuv$
$u = \overline{A} + \overline{B}x_i + \overline{C}x_j + \overline{D}x_i^2 + \overline{E}x_j^2 + \overline{F}x_ix_j$
$v = \overline{\overline{A}} + \overline{\overline{B}}x_k + \overline{\overline{C}}x_l + \overline{\overline{D}}x_k^2 + \overline{\overline{E}}x_l^2 + \overline{\overline{F}}x_kx_l$

Figure 6-1: The second generation output *y* as a function of the input parameters x_i, x_j, x_k, and x_l.

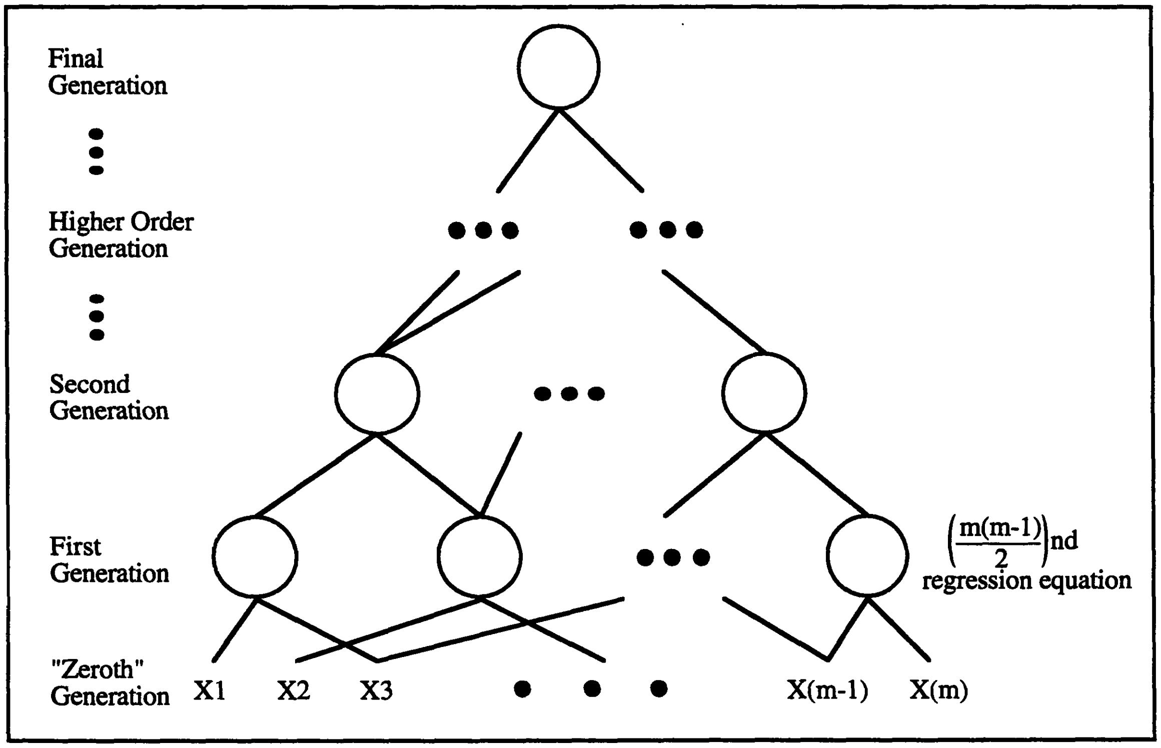


Figure 6-2: A complete GMDH model, showing the relationship between the input variables and the output *y*.

The method relies on having a large number of input variables, where the minimum number of input variables for the GMDH algorithm to work effectively is three. The method creates a number of polynomial combination of two of the input variables and combine these equations further until a final grand equations has been derived. It has been demonstrated that the GMDH algorithm can be used for modeling functions [Chen, McAulay] [Madala].

The GMDH algorithm can be viewed as a polynomial neural network, where the processing function of the nodes is a polynomial rather than a sigmoidal function. The algorithm also has a self-organizing nature; the GMDH automatically develops the structure of the regression model. One advantage of the GMDH is that optimization is based on a series of least-squares fitting rather than an iterative method of minimizing the errors like the error backpropagation. Training is a matter of performing linear algebra rather than a numerical method which requires a large amount of time for convergence. Also, since least-squares fitting is implemented for GMDH (as opposed to the iterative approach of learning algorithms for neural networks), less effort is required in developing the model. Another advantage of the GMDH is that it cannot overtrain. Overtraining a neural network can be a problem when not enough points are given and too much iteration is used to train the network, and this is one point to consider when comparing the two methods. Due to the training algorithm for the GMDH model, training stops when it reaches the best possible configuration. Figure 6-3 summarizes this algorithm.

STEPS TO DEVELOP THE GMDH ALGORITHM

1. Construct the new variables $z_1, z_2, ..., z_{\binom{m}{2}}$ with the standard regression polynomial form:

$$y = A + Bu + Cv + Du^2 + Ev^2 + Fuv$$

There will be $\binom{m}{2} = \frac{m(m-1)}{2}$ equations, where m is the number of original input variables used for the model.

2. Screen out the least effective variables using the criteria $_j < R$, where R is the predefined RMS tolerance, and r_j is the RMS value of the particular regression equation.

3. Test the optimality (is the lowest RMS value in this iteration smaller or larger than the lowest RMS value of the last iteration?):

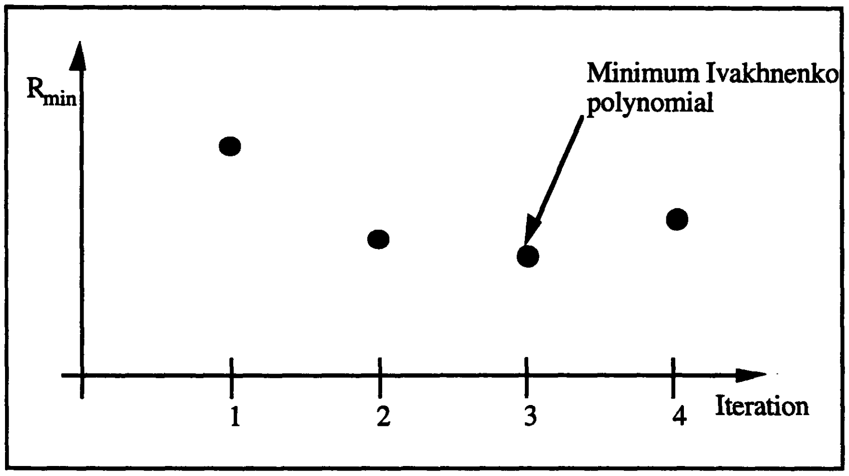


Figure 6-3: Steps to develop the GMDH algorithm.

6.3. Multivariate Adaptive Regression Splines

Multivariate Adaptive Regression Splines (MARS) is a new methodology for nonlinear regression modeling. The model takes the form of an expansion in product spline basis functions, where the number of basis functions as well as the parameters associated with each one (product degree and knot locations) are automatically determined by the data. The recursive partitioning approach to regression motivates this procedure, but unlike recursive partitioning, this method produces continuous models with continuous derivatives.

6.3.1. Recursive Partitioning Regression

In statistics, adaptive algorithms have been in long use in for function approximation, where one of the paradigms seen is recursive partitioning. Adaptive computation is one that dynamically adjusts its strategy to take into account the behavior of the particular problem to be solved, and recursive partitioning is the recursive and optimal splitting of the domain of interest into a good set of subregions. The recursive partitioning regression model takes the form

$$\text{if } \mathbf{x} \in R_m, \text{ then } f(\mathbf{x}) = g_m\left(\mathbf{x} \mid \{a_j\}_1^p\right)$$

$\mathbf{x}$ is the set of input variables and $\{R_m\}$ are disjointed subregions representing a partition of the domain of interest, where the $g_m(\mathbf{x} \mid \{a_j\})$ are generally simple parametric functions. The partitioning is accomplished through the recursive splitting of previous subregions, where the starting region is the entire domain D. At each partitioning stage, the subregions are formed into two "daughter" regions R_l and R_r which take the form

```
if $\mathbf{x} \in R$, then
    if $x_v \le t$,  then $\mathbf{x} \in R_l$
                     else $\mathbf{x} \in R_r$
end if
```

where v labels one of the covariates and t is a value on that variable. A goodness-of-fit criterion is used to optimize the splitting process which determines the appropriate splint points (or knots). The final procedure is the recombination of the subregions in a reverse manner until an optimal set is reached, based on a criterion which penalizes both for "lack-of-fit" (simply the quantification of the residual of the fit to the data) and increasing number of regions.

Recursive partitioning is a very powerful methodology which can be rapidly computed. However, in general, there are several drawbacks to using recursive partitioning as a regression modeling technique:

- Recursive partitioning models have disjoint subregions and are usually discontinuous at subregion boundaries.
- Recursive partitioning has an innate inability to adequately estimate functions that are linear or additive. This behavior is due to the recursive division of established subregions during the forward step procedure that automatically produces predictor variable interactions unless all successive partitions occur on the same predictor variable.
- The form of the recursive partition model, which is an additive combination of functions of predictor variables in disjoint regions, makes estimation of the true form of the unknown function difficult for large number of variables.

6.3.2. Adaptive Regression Splines

The basis function $B_m(\mathbf{x})$, can allow a continuous model to be developed from the partitioned regions with continuous derivatives (namely, functions with the absence singularity regions) by incorporating a set of two-sided truncated power basis functions of the form

$$B_m^{(q)}(\mathbf{x}) = \prod_{k=1}^{K_m} \left[\pm 1 \; (x_{v(t)} - t_k)\right]_+^q$$

where q represents the order of the spline, and the quantity K_m represents the number of splits that gave rise to B_m. Figure 6-4 gives the forward stepwise portion of the MARS algorithm, which determines the optimal knot locations for the partitioning of the domain. Another algorithm called the backward stepwise MARS algorithm must also be implemented, which prunes the basis functions obtained from the forward algorithm to improve the lack-of-fit criterion.

```
$B_1(\mathbf{x}) \leftarrow 1; M \leftarrow 2$
Loop until $M > M_{\max}$: lof* $\leftarrow \infty$
    For $m = 1$ to $M - 1$ do:
        For $v \notin \{v(k, m) \mid 1 \leq k \leq K_m\}$
            For $t \in \{x_{vj} \mid B_m(\mathbf{x}_j) > 0\}$
                $g \leftarrow \sum_{i=1}^{M-1} a_i B_i(\mathbf{x}) + a_M B_m(\mathbf{x}) [+(x_v - t)]_+ + a_{M+1} B_m(\mathbf{x}) [-(x_v - t)]_+$
                lof $\leftarrow \min_{a_1,\ldots,a_{M+1}} \mathrm{LOF}(g)$
                if lof < lof*, then lof* $\leftarrow$ lof; $m^* \leftarrow m; v^* \leftarrow v; t^* \leftarrow t$ end if
            end for
        end for
    end for
    $B_M(\mathbf{x}) \leftarrow B_{m^*}(\mathbf{x}) [+(x_v - t^*)]_+$
    $B_{M+1}(\mathbf{x}) \leftarrow B_{m^*}(\mathbf{x}) [-(x_v - t^*)]_+$
end loop
end algorithm
```

Figure 6-4: The MARS forward algorithm.

6.4. Results of the Various Modeling Approach to the Test Bed Problems

In this section, a direct comparison with neural networks of the results from modeling using linear regression, the Group Method of Data Handling (GMDH), and the Multivariate Adaptive Regression Splines (MARS) algorithms will be applied for the assessment of the relative capability of neural networks as a modeling tool. The test problems used for the comparisons are identical to the set of test bed problems presented in Chapter 5, namely the multivariate test functions, laser machining, and the manufacturing of the automobile steering column.

6.4.1. Modeling of the Multivariate Test Functions

The criterion for the comparisons of the different modeling approaches is the root-mean-square (RMS) error of the predicted outputs from each model. Figures 6-5 and 6-6 show the comparisons of the results of the different modeling methods applied to the test functions given in Equation (5-1). The neural network gave the most accurate predictions of the models developed from the uncorrupted data, and GMDH gave the most accurate predictions of the models developed from data containing noise.

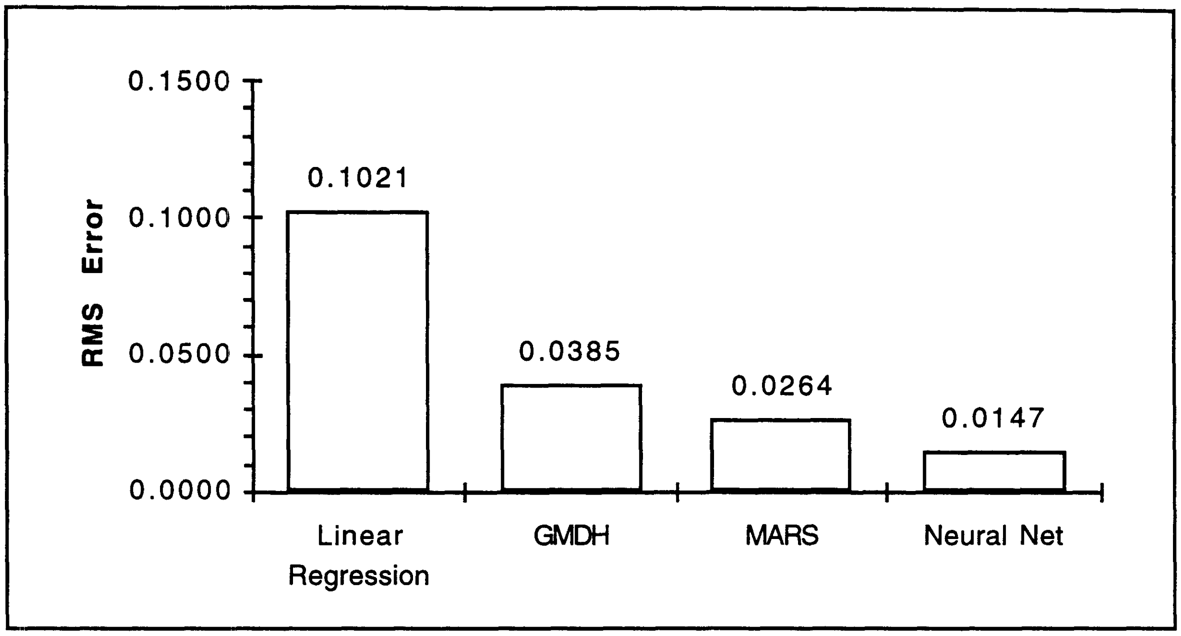


Figure 6-5: Comparison of results of the different models of Equation (5-1).

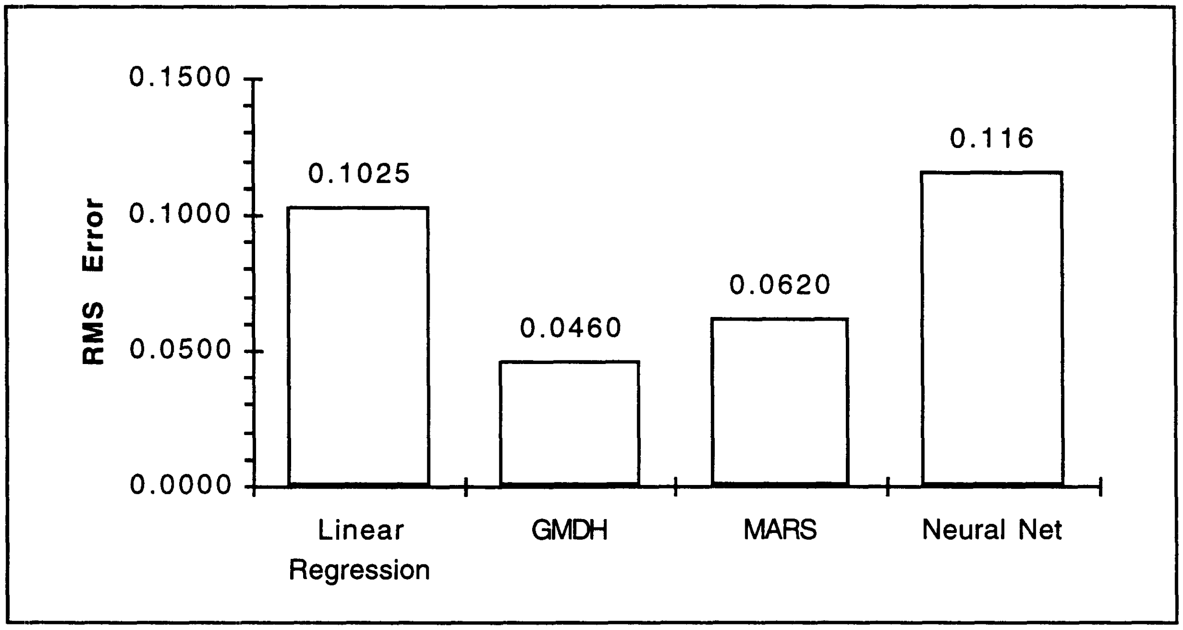


Figure 6-6: Comparison of results of the different models of Equation (5-1) containing noise.

Figures 6-6 and 6-7 show the comparisons of the results of the different modeling methods applied to the test functions given in Equation (5-2). Linear regression gave the most accurate predictions of the models developed from both the uncorrupted data and data containing noise. Except for neural networks, all the models demonstrated approximately the same level of accuracy.

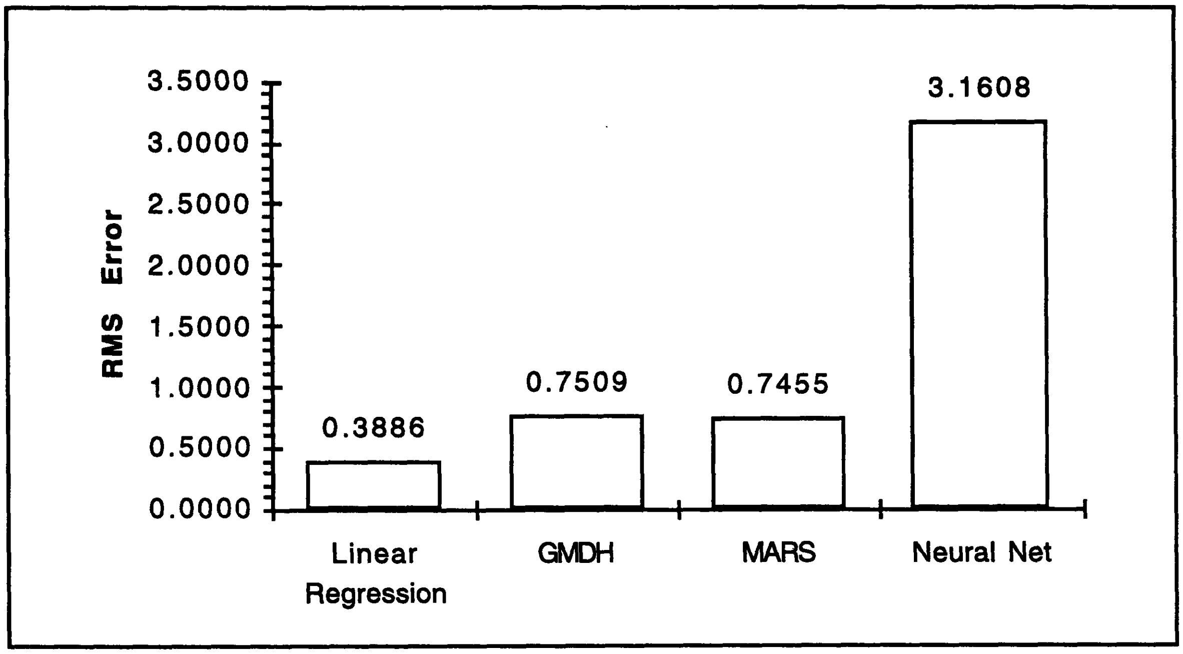


Figure 6-7: Comparison of results of the different models of Equation (5-2).

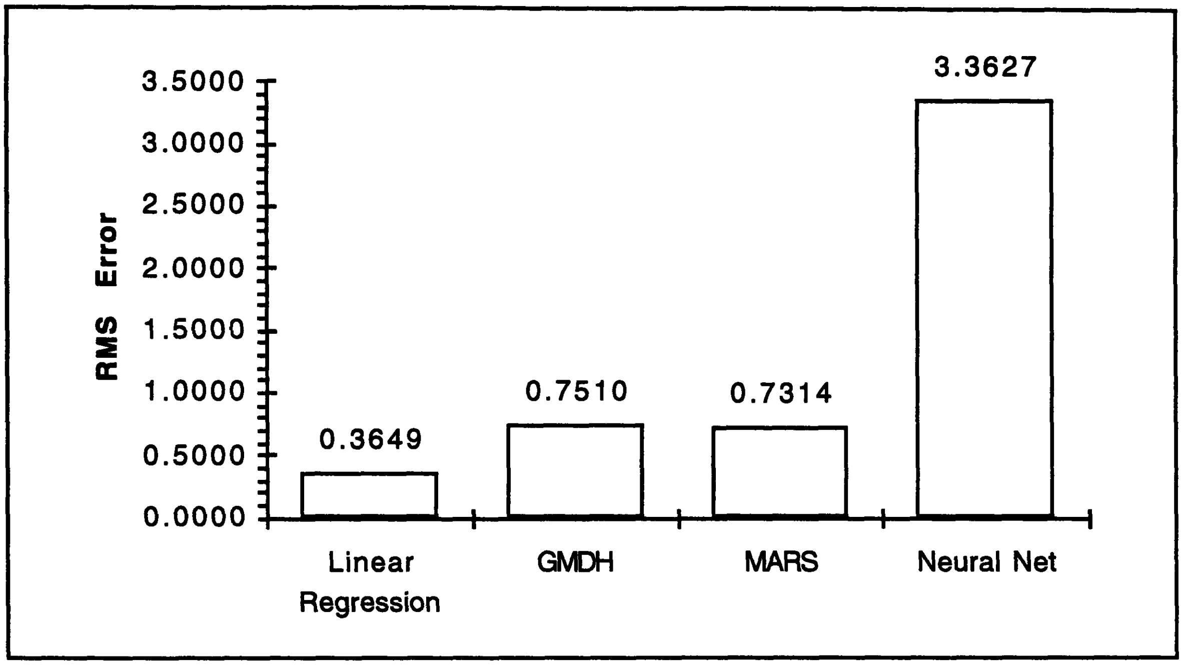


Figure 6-8: Comparison of results of the different models of Equation (5-2) containing noise.

6.4.2. Modeling of Laser Machining

In this section, the results from the neural network models generated in Chapter 5 are compared with linear regression, MARS, and GMDH models. For some cases, working models could not be generated due to the lack of sufficient amount of data or, for the case of GMDH, there were not sufficient amount of input variables.

6.4.2.1. Laser Through-Cutting

A GMDH model was not made for the laser through-cutting problem, because minimum number of input variables required for GMDH is three and only the power and workpiece thickness were the only two variables. Also, the MARS model gave highly inaccurate predictions which were approximately an order of magnitude greater than the true values; thus left out of the comparison. The source of the large inaccuracy is due to the lack of a sufficient amount of training data.

Figure 6-9 shows the comparisons of the results of linear regression and neural network models applied to laser cutting. Instead of using the RMS error as the criterion for comparison, the figure includes the true values with the predicted cut velocities for both test cases.

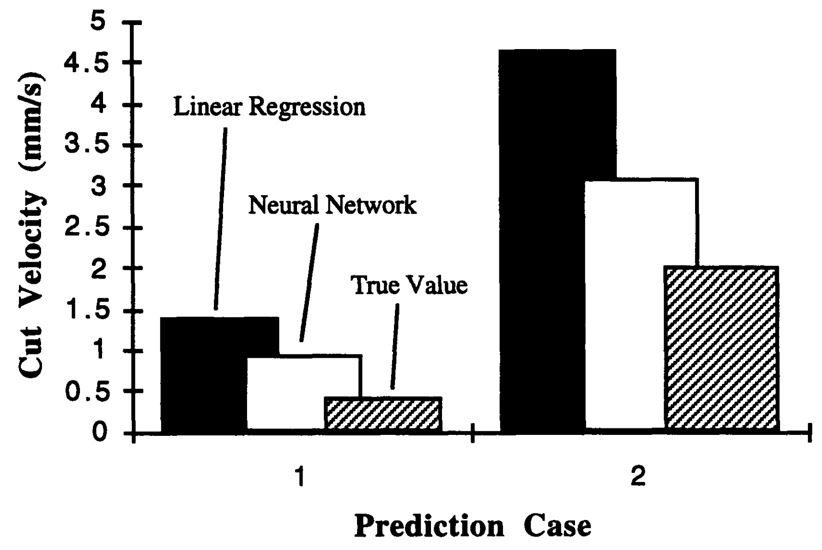


Figure 6-9: Comparison of results of modeling laser through-cutting.

6.4.2.2. Laser Grooving

The GMDH model gave highly inaccurate predictions which were on the order of 10^{12}, and similarly for the MARS model of laser through cutting, the source of the large inaccuracy is due to the lack of a sufficient amount of training data. Figure 6-10 shows the comparisons of the results of linear regression, MARS, and neural network models applied to laser grooving. The figure includes the true values with the predicted groove depth for both test cases the criterion for comparison. The comparison does not give a conclusive result, but it does show that the neural network model was more consistent in predicting a close value of the groove depth in comparison with the other models.

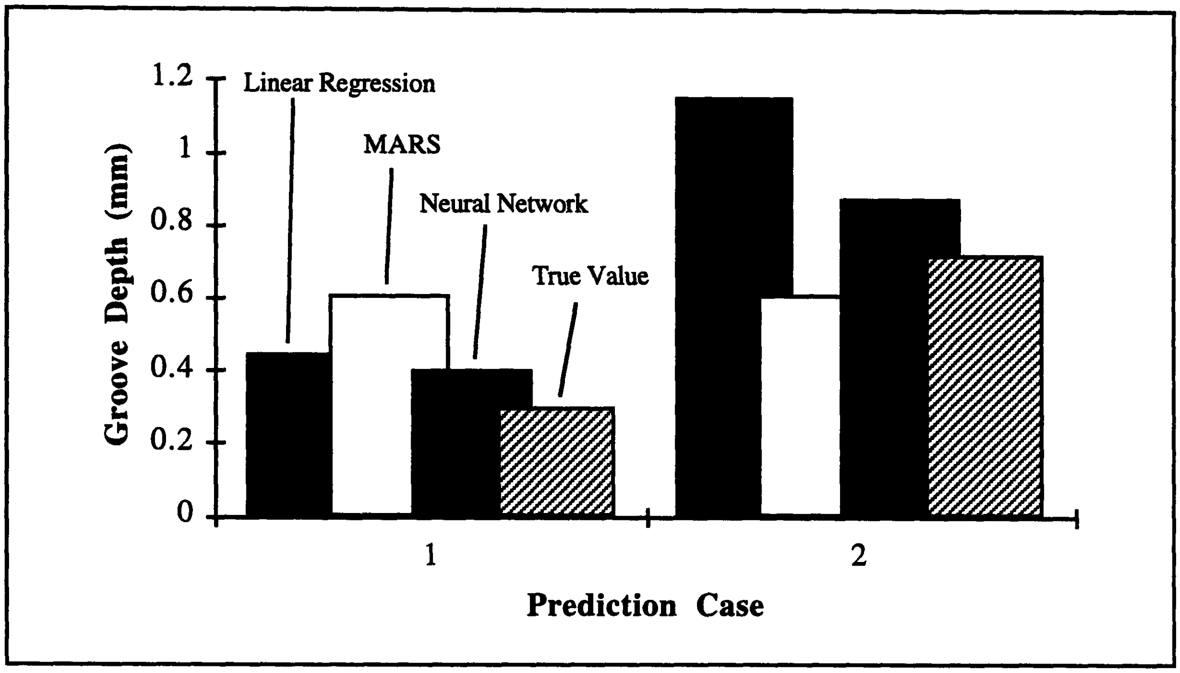


Figure 6-10: Comparison of results of modeling laser grooving.

6.4.3. Modeling of the Manufacturing of an Automobile Steering Column

Figure 6-11 and 6-12 show the comparisons of the results of the different modeling methods applied to the problem of the manufacturing of an automobile steering column. The MARS model gave the most accurate predictions of the models developed from the uncorrupted data and from data containing noise, but comparable accuracy was practically achieved by all the models.

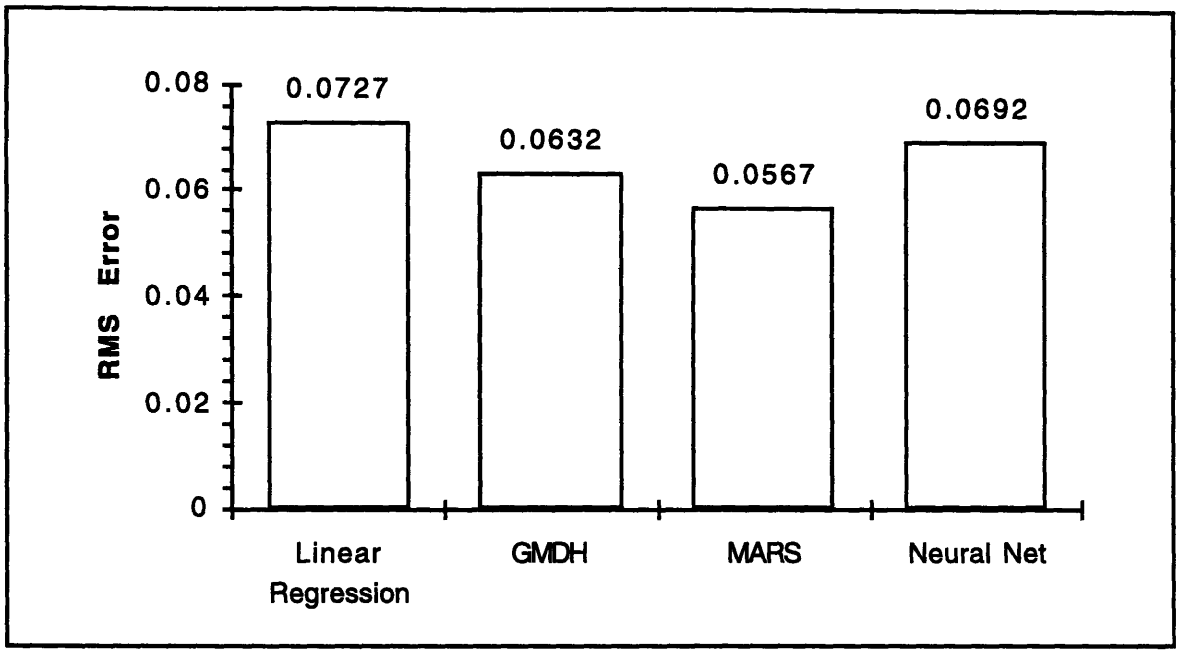


Figure 6-11: Comparison of results of modeling the manufacturing systems.

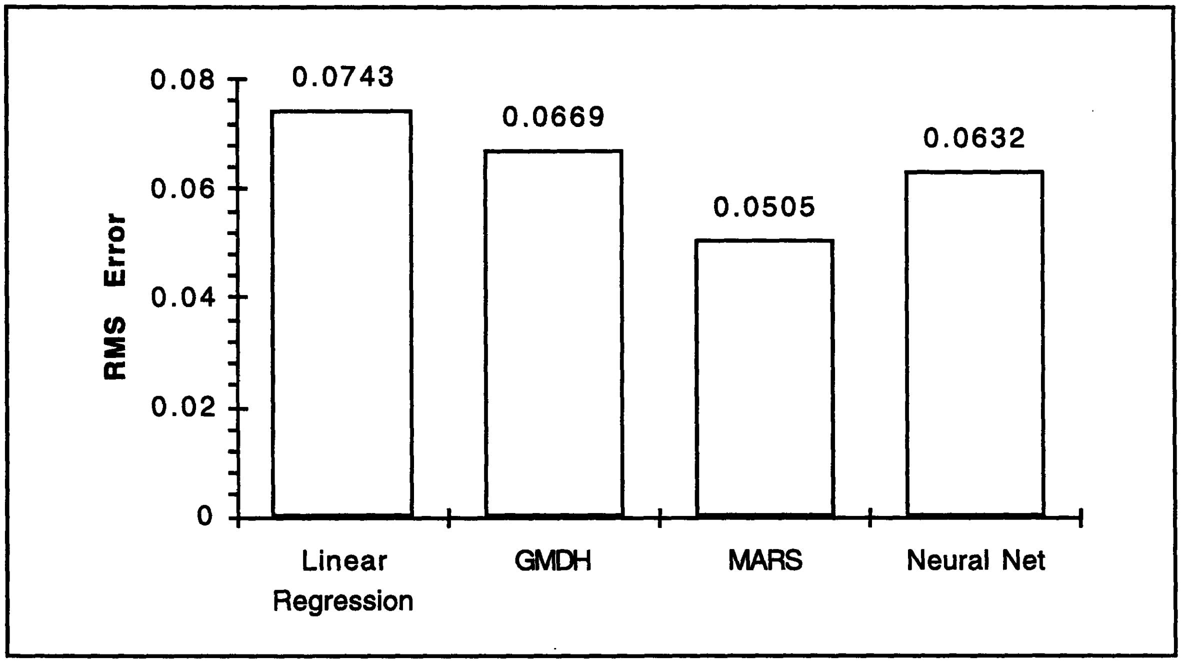


Figure 6-12: Comparison of results of modeling the manufacturing systems with noisy data.

6.4.4. Overall Assessment of the Relative Modeling Efficacy of Neural Networks

A general assessment of the relative efficacy of neural networks cannot be made from the results presented in this chapter, because the results are not conclusive. For the multivariate test functions, neural network models gave the most accurate predictions for the first function, yet gave the most inaccurate predictions for the second function. For the laser machining problems, neural network models gave the most accurate predictions, but for the problem regarding the manufacturing of a steering wheel column, neural networks only outperformed linear regression for accuracy of predictions. In general, the results show that all these modeling methods provide approximately the same level of accuracy for a given application.

CHAPTER 7
CONCLUSION

An approach to assess the reliability of neural network models has been introduced, namely the estimation of confidence intervals of neural network models for a specified level of confidence. In addition to the estimation of reliability, the relative efficacy of neural networks as a modeling tool has been explored by comparing the accuracy of generated models from different methods on a set of test bed problems.

7.1. Use of Confidence Intervals to Assess Neural Network Reliability

For a desired level of accuracy, a confidence interval can be computed for a neural network model with the assumption of a normal distribution of error from the neural network, and calculation of the confidence interval can include the effects of developing the model from noisy data. In order to estimate the error in predicting the true output, a first-order approximation of the error of the neural network model is estimated, which involves computing the Jacobian of the neural network outputs with respect to the weights. Other approximations of the variance of the neural network error which includes the computation of the Hessian matrix of the neural network outputs may be used.

The proposed method of computing confidence intervals estimated both conservative and nonconservative intervals for predictions from the neural network models of the multivariate test functions. For the first test function, conservative confidence intervals were estimated; a significantly *larger* than expected number of the predictions lied within the intervals. For the second test function, nonconservative confidence intervals were estimated; a significantly *less* than expected number of predictions lied within the intervals. For the test problems of laser machining and manufacturing of the automobile steering column, the confidence intervals gave expected results. Thus, the proposed method has succeeded for a number of the models of the test problems in estimating the reliability of the neural network models, but were not consistent with all the models. Greater accuracy may be achieved by using a higher-order approximation of the error of the neural network model, but the drawbacks would entail increased computational burden and a higher requirement of precision for the parametric values of the model.

In assessing the reliability of neural networks using the proposed method of estimating the variance for parameterized model, the limits for the confidence intervals are dependent on the closeness of the neural network model in mapping the training data and are also dependent on the relative degree of freedom of the neural network model. If a close fit of the model to the training data is achieved (i.e., if the neural network training *converges*) and if the amount of training data samples substantially exceeds the number of links in the neural network model, then the confidence interval sizes for the predictions should be small and a reliable model should be generated for the neural network.

7.2. Relative Efficacy of Neural Network Models

The approach to determine the relative efficacy of neural networks has been to compare the results of the neural network models from test bed problems with results from models generated by other common empirical modeling approaches, specifically linear regression, the group method of data handling (GMDH), and the multivariate adaptive regression splines (MARS). Neural network models generated the most accurate models for some of the test problems, but was not consistent in generating fairly accurate models for all the test problems. In modeling one of the multivariate functions, the neural network gave the least accurate model. Thus, the results are not conclusive to concede whether modeling with neural networks are relatively effective.

From the comparisons, the results are also not conclusive to concede which of the modeling methods is the most effective, because none of the modeling methods consistently outperformed the others. In general, the accuracy of the models generated by each of the modeling methods did not vary significantly with each other, although linear regression models consistently gave poorer accuracy. The efficacy of neural networks is relatively equivalent to the efficacy of GMDH and MARS, but these modeling methods are likely to give a more accurate model than linear regression.

It should be noted that neural networks did not rely on a supple amount of data as much as GMDH and MARS, which was evident in the laser machining problems. Neither GMDH nor MARS were able to generate a model for the laser problems due to the highly inaccurate models generated with the sparse amount of data. Thus, neural networks have a less dependency on the amount of available data for generating a model as compared to GMDH and MARS.

7.3. Summary

It has been noted that neural networks can be seen as a catalyst for greater CIM capablity, due to the development of neural network technology for manufacturing applications, particularly related to process diagnostics, control, and optimization, as well as to system planning and design. The objective of this thesis has been to evaluate the ability of neural network to generate accurate models of physical systems typically encountered in manufacturing. An assessment of the reliability of neural network models can be made by computing estimations of confidence intervals of neural network models, and the relative efficacy of neural networks has been concluded to be relatively equal to the empirical modeling methods of GMDH and MARS.

REFERENCES

Andersen, K., Cook, G. E., Springfield, J. F. and Barnett, R. J. (1991), "Applications of Artificial Neural Networks for Arc Welding," Intelligent Engineering Systems Through Artificial Neural Networks, pp. 717-727.

Bishop, C. (1992), "Exact Calculation of the Hessian Matrix for the Multilayer Perceptron," Neural Computation, **4**, pp. 494-501.

Chen, C. L. Philip and McAulay, A. D. (1991), "Robot Kinematics Learning Computations Using Polynomial Neural Networks," Proceedings for the 1991 IEEE International Conference on Robotics and Automation, pp. 67-82.

Chryssolouris, G. (1991), Laser Machining: Theory and Practice. Springer-Verlag.

Chryssolouris, G. (1992), Manufacturing Systems: Theory and Practice. Springer-Verlag.

Chryssolouris, G., Lee, M., Pierce, J., and Domroese, M. (1990), "Use of Neural Networks for the Design of Manufacturing Systems," Manufacturing Review, vol. 3, no. 3, pp. 187-194.

Conover, W. J. (1980), Practical Nonparametric Statistics. John Wiley & Sons.

Donaldson, J. R. and Schnabel, R. B. (Feb. 1987), "Computational Experience with Confidence Regions and Confidence Intervals for Nonlinear Least Squares," Technometrics, vol. 29, no.1, pp. 67-82.

Farlow, S. J. (1984), Self-Organizing Methods in Modelling: GMDH Type Algorithms. Marcel Dekker, Inc..

Freund, J. E. and Walpole, R. E. (1987), Mathematical Statistics. Prentice Hall Inc..

Friedman, J. H. (1979), "A Tree-Structured Approach to Nonparametric Multiple Regression," Lecture Notes in Mathematics (757): Smoothing Techniques for Curve Estimation, pp. 5-22.

Friedman, J. H. (1991), "Multivariate Adaptive Regression Splines," The Annals of Statistics, vol. 19, no. 1, pp. 1-141.

Friedman, J. H., Grosse, E. and Stuetzle, W. (1983), "Multidimensional Additive Spline Approximation," SIAM J. Sci. Statist. Comput., **4**, pp. 291-301

Graves, S. C. and Redfield, C. H. (1988), "Equipment Selection and Task Assignment for Multiproduct Assembly System Design," The International Journal of Flexible Manufacturing Systems, **1**, pp. 31-50.

Hecht-Nielsen, R. (1990), Neurocomputing. Addison-Wesley.

Hecht-Nielsen, R. (1987), "Counterpropagation Networks," 1987 IEEE International Conference on Neural Networks, pp. 19-32.

Hogg, R. V. and Ledolter, J. (1987), Engineering Statistics. Macmillan Publishing Company.

Indurkhya, G. and Rajurkar, K. P. (1991), "Artificial Neural Network Approach in Modelling of EDM Process," Intelligent Engineering Systems Through Artificial Neural Networks, pp. 845-850.

Kaplan, R.S. (1983), "Measuring Manufacturing Performance: A New Challenge for Managerial Accounting Research," The Accounting Review, Vol. 58, No. 4, pp. 686-705.

Keeler, J. (1992), "Vision of Neural Networks and Fuzzy Logic for Prediction and Optimization of Manufacturing Processes," Applications of Artificial Neural Networks III, p. 447-455.

Kohonen, T. (1988), Self-Organization and Associative Memory. Springer-Verlag.

Lee, J. and Tsai, J. (1992), "On-Line Fault Detection Using Integrated Neural Networks," Applications of Artificial Neural Networks III, p. 436-446.

Lee, M. (1993), "Methods for Configuring Manufacturing Systems," PhD. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.

Lippmann, R (1987), "An Introduction to Computing with Neural Nets," IEEE Acoustics, Speech and Signal Processing Magazine, pp. 4-22.

Madala, H. (1990), "Inductive Learning Networks in Comparison with Deductive Techniques," 1990 24th Asilomar Conference on Signals, Systems, and Computers, Vol. 1.

Malave, C. O., Sastri, T., and Johnson, R. (1991), "Prediction of Wave Solder Machine Parameters Based on Printed Circuit Board Design Characteristics," Intelligent Engineering Systems Through Artificial Neural Networks, pp. 833-838.

McCulloch, W. S. and Pitts, W. (1943), "A Logical Calculus of the Ideas Immanent in Nervous Activity," Bulletin of Math. Bio., **5**, pp. 115-133.

Minsky, M. and Papert, S. (1969), Perceptrons. MIT Press.

Nedeljkovic, M. (1993), "Assembly System Design with the Help of Neural Networks", M.S. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.

Rumelhart, D. E. and McClelland, J. L. (1986) Parallel Distributed Processing, Volume 1: Foundations. MIT Press.

Rosenblatt, F.(1958), "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," Psychol. Rev., **65**, 386-408.

Sakakura, M. and Inasaki, I. (1992), "A Neural Network Approach to the Decision-Making Process for Grinding Operations," Annals of the CIRP, Vol. 41/1/1992, pp. 353-356.

Seber, G. A. and Wild, C. J. (1989), Nonlinear Regression. John Wiley and Sons, Inc.

Smith, A. E. (1991), "Quality Control Using Backpropagation: An Injection Molding Application," Intelligent Engineering Systems Through Artificial Neural Networks, pp. 729-734.

Son, Y.K. (1991), "A Cost Estimation Model for Advanced Manufacturing Systems," International Journal of Production Research, Vol. 29, No. 3, pp. 441-452.

Suh, N.P. (1990), The Principles of Design, Oxford University Press, New York.

Suresh, N.C. and J.R. Meredith (1985), "Justifying Multimachine Systems: An Integrated Strategic Approach," Journal of Manufacturing Systems, Vol. 4, No. 2, pp. 117-134.

Swamidass, P.M. and M.A. Waller (1990), "A Classification of Approaches to Planning and Justifying New Manufacturing Technologies," Journal of Manufacturing Systems, Vol. 9, No. 3, pp. 181-193.

Wabalickis, R.N. (1988), "Justification of FMS with the Analytical Hierarchy Process," Journal of Manufacturing Systems, Vol. 7, No. 3, pp. 175-182.

Wang, Q., Sun, X., Golden, B. L., and DeSilets, L. (1991), "Predicting Wire Bond Quality Using Backpropagation," Intelligent Engineering Systems Through Artificial Neural Networks, pp. 827-832.

Widrow, B.(1962), "Generalization and Information Storage in Networks of ADALINE Neurons," in Yovitts, G.T. [Ed.], Self-Organizing System, Spartan Books.

Winston, P. H. (1992), Artificial Intelligence. Addison-Wesley.