American University in Cairo
AUC Knowledge Fountain

Archived Theses and Dissertations

End milling process modeling using artificial neural networks

Khaled Abdel-Hamid Elsayed

Follow this and additional works at: https://fount.aucegypt.edu/retro_etds

 Part of the Industrial Engineering Commons

2004/28

The American University in Cairo
School of Science and Engineering
Interdisciplinary Engineering Programs

End Milling Process Modeling Using Artificial Neural Networks

by

Khaled Abdel-Hamid Elsayed

A thesis submitted in partial fulfillment of the requirements for the degree of:

Master of Science in Engineering

With specialization in:

Industrial Engineering

Under the supervision of:

Dr. Mohamed Gadallah
Associate Professor, Mechanical Engineering Department, AUC

Dr. Keith Hekman
Associate Professor, Mechanical Engineering Department, AUC

Spring 2004

2004/25

The American University in Cairo
School of Sciences and Engineering

End Milling Process Modeling Using Artificial Neural Networks

A Thesis Submitted by
Khaled Abdel-Hamid Elsayed

to the Interdisciplinary Engineering Programs

May 13, 2004

in partial fulfillment of the requirements for the degree of

Master of Science in Engineering with Specialization in Industrial Engineering

has been approved by

Dr. Mohamed Gadallah, (Advisor)
Associate Professor, Mechanical Engineering Department, AUC

[redacted]

Dr. Keith Hekman, (Advisor)
Associate Professor, Mechanical Engineering Department, AUC

[redacted]

Dr. Nahed Sobhi
Professor, Faculty of Engineering, Ain Shams University

[redacted]

Dr. Adel Shalaby
Professor, Mechanical Engineering Department, AUC

[redacted]

Dr. Maher Y.A. Younan
Professor, Mechanical Engineering Department, AUC

[redacted]

[redacted]
Department Chair/
Program Director

June 1, 2004
Date

[redacted]
Dean

June 3, 2004
Date

ACKNOWLEDGMENTS

Deepest thanks and appreciation are due to my supervisors: **Dr. Mohamed Gadallah** and **Dr. Keith Hekman** for their valuable advice, support, and continuous guidance during the preparation for this thesis. The financial support in the form of a research assistantship from the **American University in Cairo** to **Dr. Mohamed Gadallah** is also gratefully acknowledged.

ABSTRACT

This thesis deals with the development of a process modeling methodology of the end milling using artificial neural networks (ANNs) and the experimental selection is based on the orthogonal arrays (OAs) and design of experiments (DOE). Process variables include depth of cut (a), spindle speed (n), feed rate (f), and tool diameter (d). The interest is to measure the resulting dynamic variations of cutting forces with cutting time. Two supervised neural networks, a radial basis network (RBN) and a feed forward network (FFN) are utilized as predictive tools for the process modeling. A systematic approach for designing and training both types of neural networks using MATLAB functions is given, aiming at reducing both predictive error and number of training patterns. A comparison between these two neural networks based on predictive capability and number of training patterns is presented. Moreover, designing and developing a neural network using insufficient information is studied and presented. Modeling with insufficient information can sometimes be a practical necessity. In this case, a process modeler can compromise the accuracy of the information for the experimental cost.

In this thesis, several experimental models are presented based on (OAs) and DOE including 2-levels, 3-levels, 4-levels, and 5-levels OAs. Results show that each model has a potential for prediction if used by itself. Furthermore, if models are combined in a sequence, the resulting composed models have better capabilities for prediction. Results of different composed models indicate that using a certain sequence leads to a better model with faster convergence and with less predictive error. Using such developed model, experimental milling is not needed to find the cutting force; rather, the model can be used for future prediction.

Without the inclusion of DOE, several hundred milling experiments are needed over several ranges of feasible input parameters to feed the neural model. Neural Network modeling is an expensive modeling technique by itself. Combination of DOE and neural network yields a very cost effective modeling means with low number of experiments. Overall, roughly 160 experiments were conducted for the need of training and testing the used neural models. The resulting neural models are valid for cutting force prediction inside and outside the variables domains. Besides, using the DOE along a certain neural network design can compensate the limitations of experiment pseudo randomness and can approximate the process using insufficient information with reasonable error.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

NOMENCLATURE

a	:	Depth of Cut.
ANN	:	Artificial Neural Network.
ANOVA	:	Analysis of Variance.
CU	:	A Conventional Universal Milling Machine.
d	:	Tool Diameter.
DOE	:	Design of Experiments.
EBP	:	Error Back-Propagation.
f	:	Feed Rate.
FFN	:	Feed Forward Neural Network.
F_{Max}	:	Maximum Force.
F_{Mean}	:	Overall Mean Force.
F_{Min}	:	Minimum Force.
$F_{M\text{-}Max}$	:	Mean of Maximum Force per Revolution.
$F_{M\text{-}Min}$	:	Mean Minimum Force per Revolution.
F_{Stdev}	:	Overall Standard Deviation.
LOGSIG	:	Log Sigmoid Transfer Function.
MSE	:	Performance Function Relays on the Mean Sum of Squares of The Network Errors.
MSEREG	:	A Performance Function Relays on the Mean Squared Error and the Mean Squared Weights and Biases.
NL8-1	:	Experimental Model uses L8 OA and Conducted on a CNC Machine
NL8-2	:	Another Experimental Model uses L8 OA and Conducted on a CNC Machine.
NL-add	:	Additional Experiments Conducted on a CNC Machine.
OA	:	Orthogonal Arrays.
PURELIN	:	Linear Transfer Function.
RBN	:	Radial Basis Neural Network.
RPM	:	Revolution per Minute.
TANSIG	:	Hyperbolic Tangent Sigmoid Transfer Function.
TRAINGDX	:	Network Training Function that Updates Weight and Bias Values According to Gradient Descent Momentum and Adaptive Learning Rate.

TRAINLM	:	Network Training Function that Updates Weight and Bias Values According to Levenberg-Marquardt Optimization.
UL25	:	Experimental Model uses L25 OA.
UL27-1	:	Experimental Model uses L27 OA.
UL27-2	:	Another Experimental Model uses L27 OA.
UL32	:	Experimental Model uses L32 OA.
UL8-1	:	Experimental Model uses L8 OA.
UL8-2	:	Another Experimental Model uses L8 OA.
UL9	:	Experimental Model uses L9 OA.
UL-add	:	Additional Experiments.

CHAPTER 1

INTRODUCTION

1.1 Introduction

The final processing of most mechanical parts after the primary operations such as casting, rolling, and forging is one of the metal cutting operations. The aim of metal cutting processes is to efficiently and safely produce the final parts with desired shapes, dimensions, and surface quality. Metal cutting can shape products with different sizes. Large industries like automotive and aerospace have large machine shops with numerous metal cutting activities. Metal cutting can be applied to metal parts that have different material properties such as steels and aluminum alloys. The same cutting processes can also be used in other applications dealing with nonmetal materials such as polymers and ceramics. One classification divides the metal cutting operations into two main groups. The first group is the operation that contains single-point tool such as turning. In this group, the cutting tool shapes the parts using a single cutting edge. The second group is milling which includes all operations conducted with multi-point tool.

As an example of the significance of the size of industrial applications using metal cutting processes, in United States alone, the annual value of the machining operations is about $168 billion [22]. This emphasizes the weight and importance of metal cutting in industry and points to the significance of working on achieving development and improvement to such operations. Consequently, the science of metal cutting has received a tremendous attention from both academic researchers and industry practitioners. Developing an accurate model that can predict the performance of metal cutting system

and consequently, lessen the requirement of carrying out experiments is one of the most important topics that received that attention. Satisfactorily achieving that task is very difficult because the features of metal cutting system are complex and contain many interacting factors. In particular, modeling the process of milling is more complicated because the dynamics are more difficult and the resulting cutting forces are periodic [23]. Therefore, recently, there has been a strong need for utilizing powerful modeling tools. One of these modeling tools is artificial neural network (ANN). It can accurately model complex systems after using some experiments for training phase and other experiments for validation phase.

1.2 The Process of Milling

In the milling operation, tools with multiple cutting edges are rotated and fed toward the machining parts to form the desired shape by removing the unwanted material. Milling is a very versatile process capable of producing simple two-dimensional flat shapes to complex three-dimensional interlaced surface configurations. Milling has several operations and ways of milling. Milling operations can be categorized into two main groups, face milling (or end milling) and peripheral milling (or slab milling). The end milling has several utilizations for facing, profiling, and slotting operations. The obtained machining surfaces in the peripheral milling operations are parallel to the axis of rotation and they are generated by the teeth located on the periphery of the cutter body. Besides, ways of milling can be grouped based on feed direction which has two basic types. Specifically, they are conventional (up) milling that feed works against the rotation of the cutter and climb (down) milling that requires about 20% less HP than conventional milling

since the feed motion is in the same direction of the milling. Figures 1.1~1.4 illustrate examples of different operations and ways of milling.

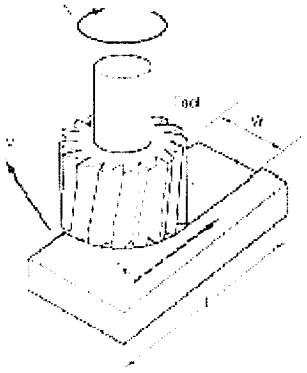

Figure 1.1 A Face Milling operation [25]

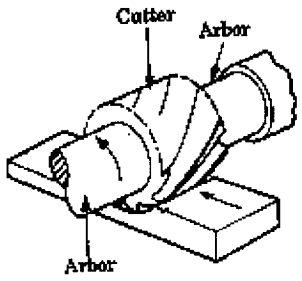

Figure 1.2 A Peripheral Milling Operation [25]

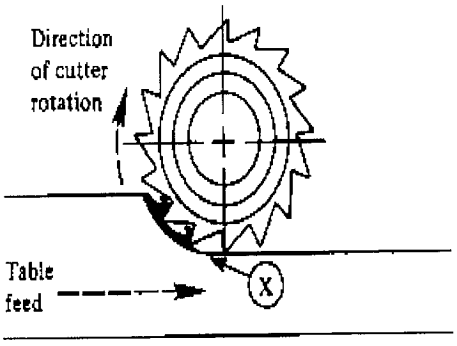

Figure 1.3 Up Milling [25]

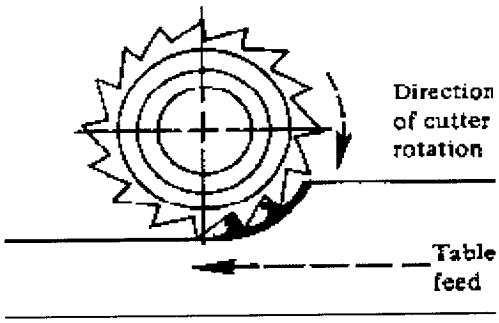

Figure 1.4 Down Milling [25]

The system of milling can be viewed as an intricate system with several input and output parameters. Figure 1.5 gives an illustration for a milling system. The system can be divided into four elements, they are: input parameters, a machine, internal parameters, and output parameters. The system has several

input machining parameters have to be considered and planned before machining to get desirable output parameters. Tool material, cutting tool features, depth of cut, spindle speed, feed rate, and cooling fluids are some significant examples of these decision variables. The optimum value of one input variable for one cutting situation can be unwanted in other situations. For instance, cooling fluids, in many milling applications, have vital benefits such as reducing the cutting temperature and lubricating the tool and the work-piece. In other applications, however, using cooling fluids will not be necessary, and in some cases will not be recommended to avoid the thermal shock that can result from suddenly cooling. Besides, searching for the most suitable values of some input parameters has to consider the values of other parameters. For example choosing a cutting tool for a particular milling operation will depend on many other factors such as the geometry feature to be machined and the work-piece material to be cut. Moreover, the selection of the input parameters can rely on the type and condition of the machine that will be used. The type of machine will place a limit on the input parameters and their assigned values. However, knowing the actual performance of the machine condition usually requires deep experience along some estimation from the machining planners for determining the most suitable input values that will optimize the output parameters, and realize the maximum benefit from the used machine.

These machining parameters, in the range of acceptable conditions of the machine, produce some internal parameters like cutting forces, and machine vibrations. These internal parameters have strong relation with the milling

system's output parameters such as quality of products and productivity. As the stability of internal parameters is obtained, the output parameters can be improved. Moreover, instability of these internal parameters can badly affect the machine condition, and lead to additional loss. Therefore, measuring these parameters helps in monitoring the processes and can give an image of the quality of machined parts.

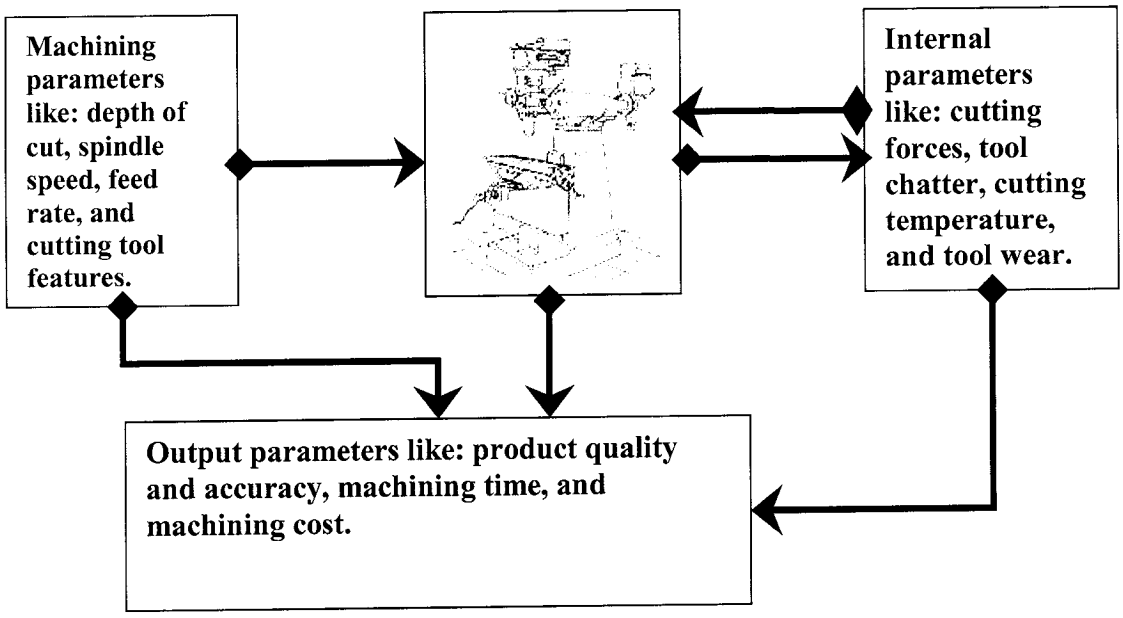

Figure 1.5 An Illustration for a Milling System

One of the most important internal parameters is the resultant cutting forces. The contact between the cutting tool and the machining parts generates significant and irregular forces during the cutting pass. These cutting forces can determine machine power requirements because they create the torques on spindle and drive motors which generate power drawn from motors. Industry practitioners and researchers tend to control these forces to have constant average force as possible and to reduce the maximum force, since

excessive forces can lead to some unwanted machining performance such as tool failure, tool deflections, geometric work-piece errors, poor surface finish, and machine structure deflections. Consequently, controlling the cutting forces has great economical benefits and monitoring resultant cutting forces can be used as a performance measurement for machining operations. Cutting forces have strong influence on tool breakage, tool wear, and work-piece deflection. Therefore, there has been much effort and research work in the area of cutting force modeling over the past decades. A number of models for predicting cutting force as a function of machining parameters such as depth of cut, cutting speed, feed rate, and tool geometry have been presented. The development of these models varies from empirical models to the use of artificial intelligence such as neural network as a modeling tool. Section 2.3 will give a literature review for these models.

Another important internal parameter is the tool chatter. Chatter is an undesirable and often unavoidable self-excited vibration between the cutting tool and work-piece. Without careful planning of the input parameters, controlling the internal parameters and considering the machine condition, the milling system can produce parts with inferior quality and decreased efficiency.

1.3 Design of Experiments

An essential objective of a designer of a product or a process is to get a reasonable conclusion about the effect of design parameters on response variables under different conditions. DOE allows for a systematic approach to quantify the effects of these parameters using a technique called the analysis of variance (ANOVA).

1.4 Artificial Neural Network (ANN)

ANN is an information handling means that is inspired by the way biological nervous systems deal with information. ANN is basically composed of processing elements connected in parallel called neurons [8, 24]. Every connection contains an adjustable parameter called weight. The output of the ANN comes from combination of each single neuron's output by these connections. The ANN has several types and applications such as the self-organizing (unsupervised) ANN used for classification problems, and the supervised ANN used for nonlinear multivariate function mapping. Since this work uses the supervised ANN for modeling process, a description for that type will be presented. The supervised ANN can be trained to acquire knowledge by presenting some different input values of usually, sophisticated nonlinear function with their true output values. For training pattern, the ANN adjusts its weight parameters based on the magnitude of the error between the true output and the ANN output. The role of the ANN algorithms is to minimize the error function with every new training pattern until the error is gradually reduced to become an acceptable small value. At this time, the ANN can be used to predict new output values for any input values. A block diagram shows this training process in Figure 1.6.

Figure 1.7 illustrates the basic features of simple neuron model, the basic processing unit of ANN. It shows a simple model of neuron that has n inputs symbolized as, $[X_0 \ldots X_n]$. Every input connection encompasses synaptic weight. The weights are represented by symbols $[W_0 \ldots W_n]$. As a simple example, each of these input values is multiplied by its connection weight then the products fed through an activation function to generate an output result.

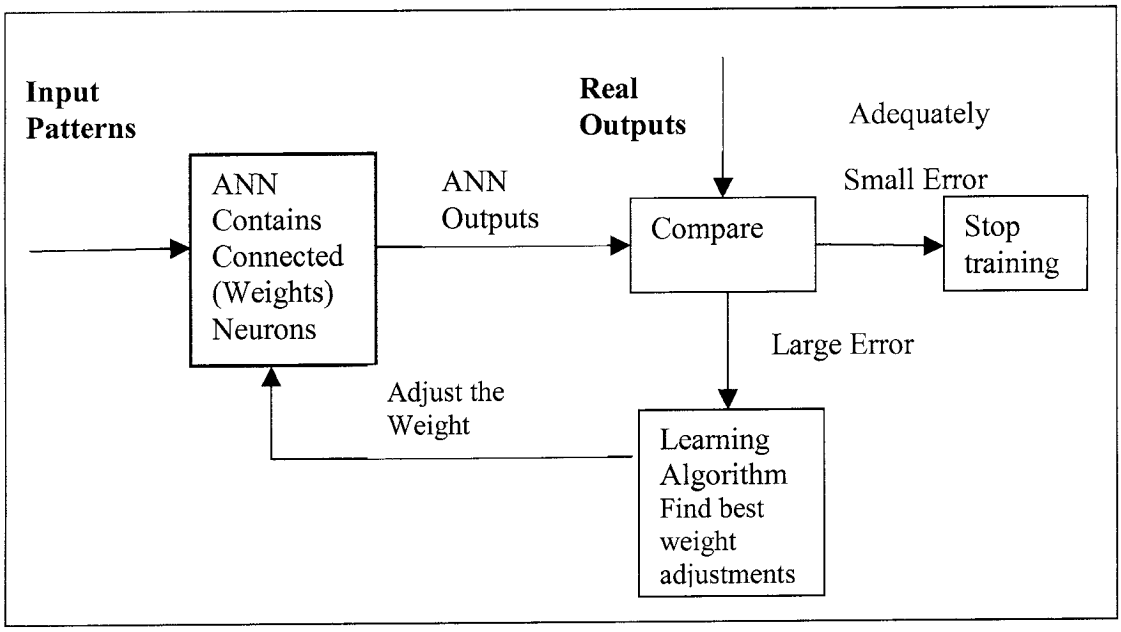

Figure 1.6 Block diagram representation of training process [24]

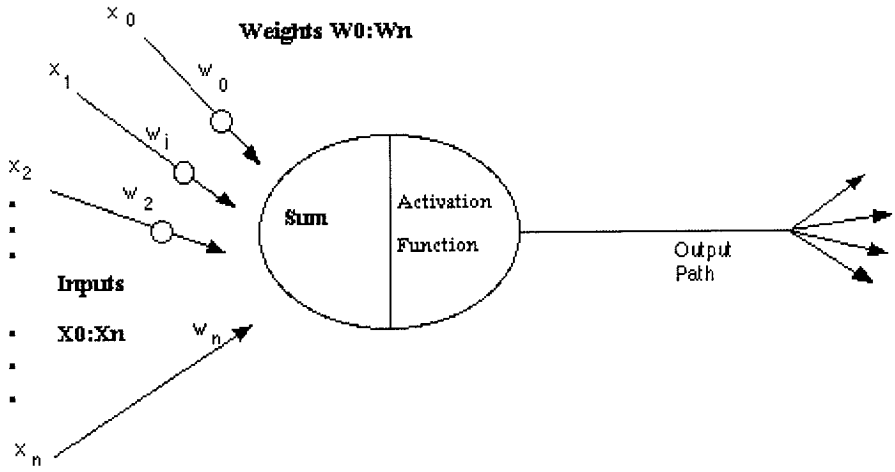

Figure 1.7 A Representation of a Neuron Model [8]

1.5 Objectives

Generally, milling process is complex as it utilizes multiple cutting edges, involves many interacting parameters, and it has dynamic effects like structural vibrations. Developing an accurate predictive model for the milling process is desired for better

planning and process control in such a way that would enhance the milling output parameters. Designing a model that predicts cutting forces as a function of machining parameters can be used to effectively plan the milling process before implementing the actual machining. The ANN is an effective modeling and approximation tool. It can map a relationship between input and output parameters by receiving some experiments which represents the actual system performance. The number of experiments can be minimized by using DOE and OAs as a selection tool that chooses the effective experiments. Therefore, combining the ANN as a modeling tool and DOE to select training experiments can be promising methodology that efficiently models the milling process.

This thesis basically deals with the modeling process of the end milling using artificial neural networks and DOE. The thesis consists of five chapters. This chapter presents the subject of this thesis. In chapter 2, a literature review in the area of metal cutting and milling operation modeling is presented. Chapter 3 provides the various equipments and tools used in the experimental work. This is followed by presenting the ANOVA for the several models. In chapter 4, the predictive models for predicting the cutting forces of face end milling operation using several OAs and two ANNs are presented. A summary of all conducted experiments used in the training and validation phases and several predictive results are given. The details of these experiments as well as results are given in appendices. This also includes the results and the process models of using insufficient information. Finally, chapter 5 presents conclusions of obtained results and recommendations for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Before the foundation of metal cutting science and until now, most machining techniques used in industry are extensively based on the past experience of the operators and even founded on trial-and-error, especially when planning for new machining situations. Such an approach is time consuming, expensive, and lacks a thorough scientific basis. At the start of metal cutting science, the researchers gave great attentions to understanding the physics of metal cutting. After that, and up till now, the development of accurate and reliable machining process model is much needed by both academic researchers and industry practitioners. The output parameters viewed as significant and needed for prediction can be: cutting forces, surface finish, tool life or tool failure, machining and work-piece accuracy. The discussion of several modeling processes of metal cutting is presented in two categories: analytical and numerical modeling and modeling using artificial intelligence. These will be discussed in sections 2.3 and 2.4 respectively.

2.2 The Requirements of Metal Cutting Modeling

The machining researchers and practitioners usually aim at a model that has the following features. The model should accurately estimate the result of using a combination of input parameters and suitably determine if the tool cuts the work-pieces in a safe, efficient, and inexpensive manner. The model should be computationally efficient, uncomplicated and suitable for use by a skilled machinist without knowledge of model

intricacies. The model should find out if the resulting stress and peak force magnitude exceed the allowable values. Besides, it should determine if the expected power does not exceed the available spindle motor and conclude if the tool meets chatter conditions.

2.3 Analytical and Numerical Modeling

By definition, analytical modeling is a set of equations based on physical significance describing the performance of the metal cutting system. These equations are usually easy to implement and they can give excellent understanding of the physics of the metal cutting. However, such models have the following assumptions:

First, cutting process is orthogonal. Usually, it uses the simple two-dimensional orthogonal cutting to explain the general features of metal cutting. In orthogonal cutting, the material is removed by a cutting edge that is perpendicular to the direction of relative tool-part motion. In orthogonal cutting analysis, several assumptions are made. The cutting force is uniform along the cutting edge, and the resulting chip is uniform flat. This simplifies the system of metal cutting. For example, when using two-dimensional orthogonal cutting concept, the cutting forces will only apply in the direction of velocity (tangential force Ft) and uncut chip thickness (feed force Ff). Second, these models are simplified by assuming that the resulting stress and shear distributions along the stress and shear planes are uniform. Furthermore, dynamic features such as the spindle run-out and the thermal effect are neglected.

Since the developed equations contain a number of coefficients, it is not applicable to use the analytical model alone. It is more sensible to carry out few experiments used to identify these coefficients. Then, these coefficients can be used to model the processes within the range of the conducted experiments. In this case, the model will be called

mechanistic model. There has been a great effort directed to produce a numbers of coefficient tables within various ranges and some common alloys. For example, a handbook containing several cutting coefficients for many alloys was presented by Cincinnati [6].

This approach was used to predict the cutting force, torque and power very quickly for a set of process variable such as depth of cut, spindle speed, tool geometry, feed rate and work-piece material. However, the mechanistic model has some problems and limitations. First, the model can't predict out the conducted experiments' range which lead to the need for performing many experiments. Second, it is time consuming to search for suitable tables to choose suitable coefficients. Third, the ignorance of many dynamic features of the cutting process and the simplification applied to the cutting configuration and the developed models are still far from being considerably complete to model the actual cutting process. Therefore, there has been a great effort to lessen the disadvantages of this approach and to advance the cutting analysis and to consider cutting dynamic characteristics.

Among these efforts, a simulation model based on the analytical models for predicting cutting forces of end-milling was presented by Milfelner and Balic in [17]. For constructing this model, a number of experiments have been conducted and the results have been saved in a database, then the analytical equations of cutting forces of end-milling were entered in the model. These equations contain a number of parameter coefficients related to input parameters such as the features of cutting tool, work-piece material, and cutting conditions. When such input parameters, within the conducted experiments range, used in the model, the program can calculate the parameter coefficients

quickly from data in the database. Still, the operators are restricted with the stored experimental ranges.

Abrari and Elbastawi [1] presented a set of force functions that implement an analytical integration of the cutting forces along the cutter edge rather than the numerical integration. The developed functions are based on the projection ship of the load area in any tool passion onto the reference coordinate planes. These equations were applied separately to some milling operations of flat and ball end mills and the obtained results were compared to the actual experimental data. The results showed that these equations can calculate the cutting force with values close to the cutting force obtained from the actual experiments.

A dynamic model of the process of turning machining based on mathematical models was presented by Acosta, Switek, and Garcia [3]. It started with giving a review for some relationships and mathematical models of independent and dependent variables used to obtain some machining parameters such as cutting force, power requirements and surface finish. The model was implemented by developing software using these mathematical models to output the machining parameters. Some experiments were performed and showed that the model is capable of illustrating many machining parameters within certain ranges and with some deviations due to ignorance of some dynamic features such as vibration. The developed work showed that the model needs many relationships since computing a machining parameter depends on the presence of these relationships.

Li et al [13] presented a theoretical model for cutting forces in face milling based on a predictive machining theory and the mechanics of milling. A windows based simulation system was presented with friendly user-interface. The model outputs milling

force variation against cutter rotation in either numerical or graphical form. The main feature of the model is considering the milling operations as a simultaneous process of cutting with a number of single point cutting tools and carefully considering the mechanics and the dynamics of milling. Oxley's predictive machining theory [20] is used as a foundation in which the cutting forces can be calculated from input data of work-piece properties, tool geometry, cutting conditions, and the type of milling. A number of experiments were performed to validate and test the model. The results confirmed good prediction accuracy and the error range was from 1% ~ 12%.

The orthogonal cutting modeling used for the mechanics of ball end milling, with considering the dynamics of cutting forces was studied by Abrari, Elbestawi, and Spence [2]. This method considers the tool as very thin slices and the cutting force as the summation of cutting forces applied to each slice. The model considers the effects of the surface undulations, instantaneous deflection and the interface of the flank face of the tool with the finished surface. The paper also gives a study for the stability of the ball end milling in the semi-finishing operation of die cavities. The developed model of X-Y cutting forces showed good results with 10% deviation.

Another way for reducing the limitation of analytical models is by simulating the process using numerical and finite element methods (FEM). As an example for this approach, a research work presented a finite element simulation based model to predict chip formation, cutting temperatures, tool stresses and cutting forces resulted in 2-D flat end milling for selected cutting conditions [18]. The experiments conducted on a horizontal high speed milling center were used to compare the resulting values with the

predicted ones. This comparison demonstrated the effectiveness of FEM simulations in predicting process variables in a simple flat end milling operation.

2.4 Modeling Using Artificial Intelligence

Another methodology that uses the great capabilities of artificial intelligence, such as neural networks and fuzzy systems, has been investigated to efficiently predict many dynamic features. The application of this approach has been successfully applied in industry with two styles: offline, and online.

A back-propagation neural network model is used to predict surface roughness and tool flank wear in hard turning using Cubic Boron Nitride (CBN) tools for different cutting conditions by Ozel, and Karpat [19]. The conducted experiments are planned by using the fractional factorial design (FFD). The validation phase demonstrated that the model was capable of accurate surface roughness and tool wear prediction for the range it has been trained. A regression model was developed and was compared to the neural network model. Comparison showed that the neural model gave better prediction accuracy.

Another back-propagation neural network model was used to predict the tool flank wear in machining (17-4PH) stainless steel using turning lathe by Chien and Tsai [5]. The predictive values were used as a constraint to optimize some cutting conditions based on maximum metal rate. The feeding experiments are planned using the Taguchi method and orthogonal arrays [21]. The Genetic Algorithms and Taguchi method were employed as tool for designing the parameters of the predictive model and optimization was used to minimize errors of predication. Tool flank wear prediction accuracy for the model ranged from 6.64% ~ 8.6%.

A new approach using neural network for modeling flat end milling operation was presented by El-Mounayri, Gadallah, and Briceno [7]. Feed rate, spindle speed, and radial depth of cut were used as input parameters to give a representation of the expected corresponding cutting forces. The FFD techniques were used for planning the experiments needed for training the neural network. This includes using four OAs specifically, L9 OA, L27 OA, L27 OA (extended range), and L36 OA. A comparison between these arrays for training the neural model is specified. The comparison proved that L36 OA model results in better predictive model.

As an example for online application, an online predictive model for surface roughness in turning operations is presented by Ho Shinn-Ying et al [9] using an adaptive NEURO-FUZZY inference system (ANFIS) and computer vision. The model aims at precisely predicting the features of surface roughness for certain cutting parameters. Experimental results demonstrated better modeling and prediction accuracy than previous models, which are polynomial network-based.

2.5 Conclusion

There has been some research effort directed to advance the mathematical models based on the physics and the geometry of metal cutting. This includes considering the effect of some dynamic features and giving more comprehensive analysis for the process of metal cutting. Besides, the implementation ways of these models have received much attention. This includes developing faster techniques for solving the equations and relationships of the process variables, and developing computer software that enhances the manipulation of these models. This approach has relatively added more accuracy and has

made the mathematical model easier to use. However, it is completely complex to obtain accurate models that totally consider the dynamic elements of metal cutting process.

Compared with this approach, the literature shows that neural modeling is accurate with generalization nature that makes it possible to train with some experiments and use the model to predict other situations with good result. Most research work using ANN in metal cutting focuses on the significance of applying ANN as modeling tool. This includes comparisons between the neural models and the traditional models. It also includes some important cutting applications in which the ANNs have been successfully applied to predict the processes. Moreover, some research work depends on OAs as a method for minimizing and planning the training experiments. Very few researches compared the effect of using different OAs as training patterns. Little research analyzes the capabilities of applying different ANN structures and different design parameters to model the metal cutting process. Very few researchers tackle the problems associated with ANN training when dealing with metal cutting such as over-fitting. Moreover, no research work has been performed to present the ANN capabilities when employed to model an application using insufficient information. The input parameters included are limited to few parameters especially depth of cut, feed rate, spindle speed. Other input parameters should be included such as the geometry of cutting tool, tool diameter and work-piece material.

Therefore, the motivation of this thesis is to provide an efficient and effective methodology for modeling the end milling process using a combination of ANNs and DOE. This will include studying and comparing the capabilities of using the neural models that can be used for modeling such process. Specifically, they are radial basis network (RBN) and feed forward neural network (FFN). Each neural model contains a number of

design parameters that need to be investigated. This will also include studying and comparing the modeling capabilities when using several levels OAs as training patterns along the used ANNs. The combination of different levels OAs is expected to provide better modeling capabilities since they host more variations of process variables but this leads to wonder which combination sequence should be followed. In this thesis, various OAs, from 2-levels to 5-levels will be used. Furthermore, two combination sequences will be followed and compared. The first one will combine OAs with arbitrary way. The second one will start with comparing the use of each OA separately then, the best OA that provides the minimum predictive error will be selected. After that, the same step will be followed, using all possible combinations of the selected OA in previous step. The best composed model will then be selected. This will be repeated until all OAs are included. Finally, modeling using insufficient data can be a practical necessity due to some limitations of time and/or resources, the process of modeling the end milling process using insufficient data will be studied.

CHAPTER 3

EXPERIMENTAL SETUP AND RESULTS

3.1 Experimental Equipment and Setup

The employed experimental equipment and setup consist basically of six main elements: two milling machines, work-pieces, tools, a dynamometer, a data acquisition system, and a pc. This section will identify and describe these elements.

All conducted experiments are machined by using two milling machines. They are CNC milling machine and conventional universal milling machine (CU). The CNC machine can run at spindle speeds from 100 rpm to 2500 rpm. Its axis travels 290 mm in X-direction, 170 mm in Y-direction, and 235 mm in Z-direction. While the conventional machine has some limitations regarding the alternative values of both spindle speed and feed rate.

Casting aluminum work-pieces used in the experiments are purchased from the local market. Their dimensions are chosen to be 80 x 65 x 60 mm to suit the fixing area at the top late area of the dynamometer. The surfaces of all work-pieces are prepared before milling experiments to avoid any surface defaults. Every work-piece is designed to have two holes for clamping the work-piece to the dynamometer. Every work-piece can be used for four experiments by machining its top and bottom sides as shown in Figure 3.1.

Six IZAR® HSS 2-flute end-mill tools with different diameter sizes were used in all experiments. They are 6 mm, 7 mm, 8 mm, 10 mm, 11 mm, and 12 mm. All tools have the same cutting angles and the same material. Additional information can be found in appendix A.

The three Cartesian force components illustrated in Figure 3.2 are measured by a KISTLER 9257B dynamometer. A work-piece can be fixed in its top late area of 170 x 100 mm. The dynamometer starts the process of measuring the cutting forces by sending analogous signals in the form of voltages which are proportional to the actual cutting forces occurring to three amplifiers by an integrated cable.

For the need of getting and interpreting these signals, a HUMSOFT MF 614 data acquisition card is used with a pc. The used card has some important features. It contains a converter unit that receives analogous signals and converts them to digital signals. Therefore, it used to receive the analogous signals from the amplifiers and output digital signals. Besides, this card can work with a Real Time Toolbox for MATLAB that contains a library of real-time blocks. This enables to create a simulation diagram using SIMULINK and consequently, benefits most of the SIMULINK advantages. Therefore, SIMULINK simulation diagram is designed using some real-time blocks provided in the Real Time Toolbox [11] and MATLAB to understand the coming digital signals and give online diagrams, immediately processing the data and store the data in mat file format. It also gives the ability to specify different sampling periods for each output. The three components of cutting forces were sampled at 500 HZ for 1 second.

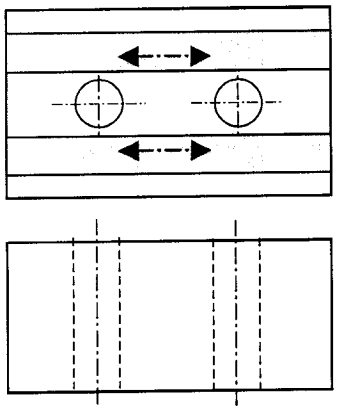

Figure 3.1 An Illustration Representing a Machining Work-piece.

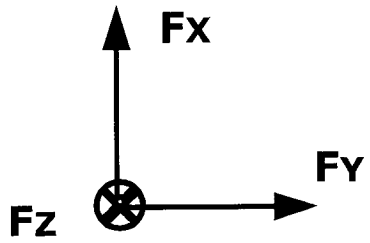

Figure 3.2 The Three Orthogonal Components of The Measured Forces.

An illustration for that model is shown in Figure 3.3. In this figure, the data acquisition board which outputs real-time digital signals as discussed before is represented by an [ADAPTER] block. This block loads the hardware driver and allows modifying its setting. [RT BUF IN] input block was used for representing the real time input channel in the simulation diagram. This block allows specifying the sampling frequency and it is designed to acquire the data at the real time and it can also store them into buffer and processed later. This gives the ability to display the three components of cutting forces using the [SCOPE] blocks that uses MATLAB graph window in real time. The [SCOPE] blocks in the simulation diagram are named as X, Y, Z respectively. At the same time, using [TO FILE] block, the data are stored to a file. Figure 3.4 illustrates the entire elements of the experimental setup and the way they are interacting.

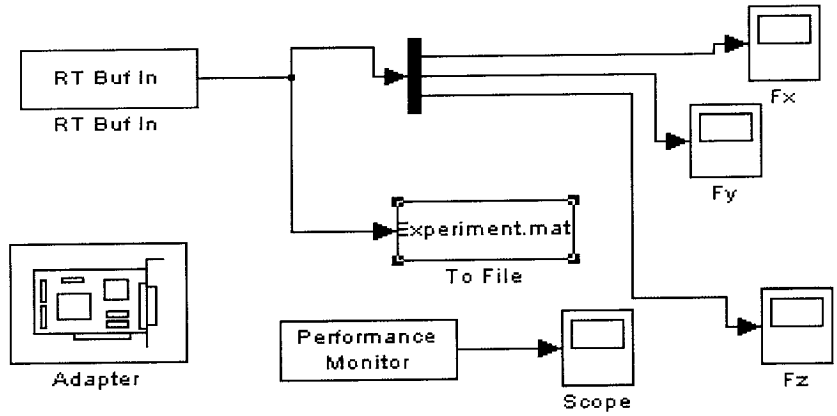

Figure 3.3 The SIMULINK Simulation Model Used to Measure the Cutting Forces

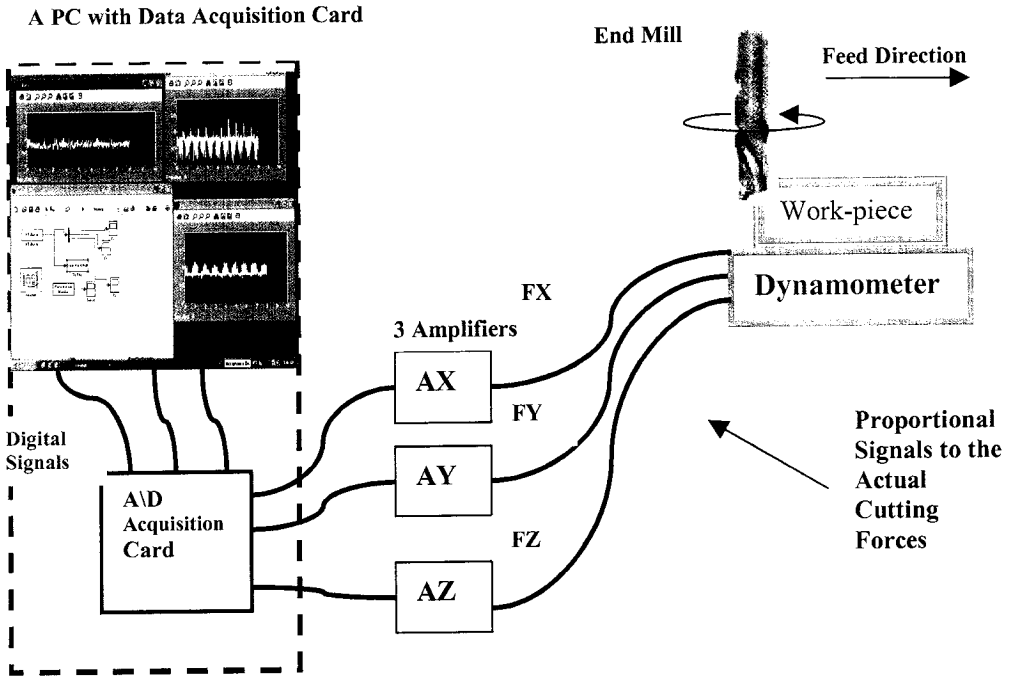

Figure 3.4 The Experimental Setup

3.2 Pre-Experimental Work

The initial and crucial step for constructing a model using ANN is to conduct some experiments that include some important input variables and measuring the desired output

variables for a particular cutting process. These experiments are used later to provide the required knowledge to the ANN model to learn the relationship between the input and output variables. The end milling is selected as cutting process specifically, slot end milling process with one path. Depth of cut (a), spindle speed (n), feed rate (f), and tool diameter (d) are chosen as the process variables. These four variables are assumed to be independent variables. Other process variables are chosen at fixed levels for all experiments. Table 3.1 shows the selected values of the process variables in all experiments.

Table 3.1 Control Variables Used For All Experimental Models.

a mm		n rpm		f mm/min		d mm	
CU	CNN	CU	CNN	CU	CNN	CU	CNN
0.5	0.5	400	500	35.5	50	6	6
0.75	1	560	750	50	100	7	8
0.8	1.5	800	1000	71	150	8	10
1	2.5	1120	1500	100	200	10	12
1.25		1600		140		11	
1.5				200		12	
1.8				280			
2							
2.25							
2.5							
2.8							
3							
3.5							
4							

The resulting dynamic 3-D cutting forces will be measured as they are selected to be the output's response. These three resulting force components F_X, F_Y, and F_Z are combined to one resultant force using Equation 3-1.

$$F = \sqrt{[(F_X)^2 + (F_Y)^2 + (F_Z)^2]} \qquad \textbf{3-1}$$

Figure 3.5 illustrates the resultant force of three cutting revolutions using CU machine as a sample. In this illustration, it is clear that the resultant cutting force is periodic. However, the cutting force behavior of each revolution is not identical. For example, the maximum force and minimum force are different for each revolution. The reasons for this behavior are due to the noise in the measurement, machine vibrations, and the impurity of work-piece material. It is also shown that within a revolution the two maximum peaks and two minimum peaks are not similar because of the spindle run-out of the machine.

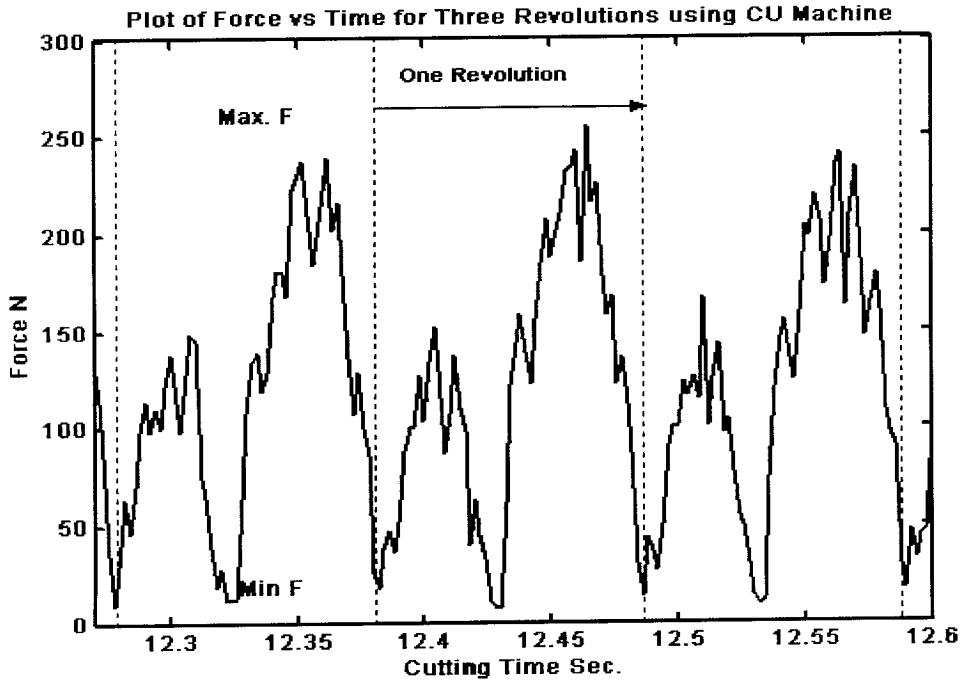

Figure 3.5 Force vs. Time for Three Revolutions Using CU Machine.

Therefore, using the resultant force, the outputs are presented in the form of six values. The first four values are: the maximum, the minimum, the mean, and the standard deviation of the resultant force values. These values are symbolized as F_{Max}, F_{Min}, F_{Mean}, and F_{Stdev}. The last two values are obtained by firstly, calculating the maximum and minimum

peaks by each revolution. Then, the mean of these maximum and minimum values are computed. $F_{M\text{-}Max}$ and $F_{M\text{-}Min}$ are the symbols of these two values. Figure 3.6 represents the form of the inputs and the outputs.

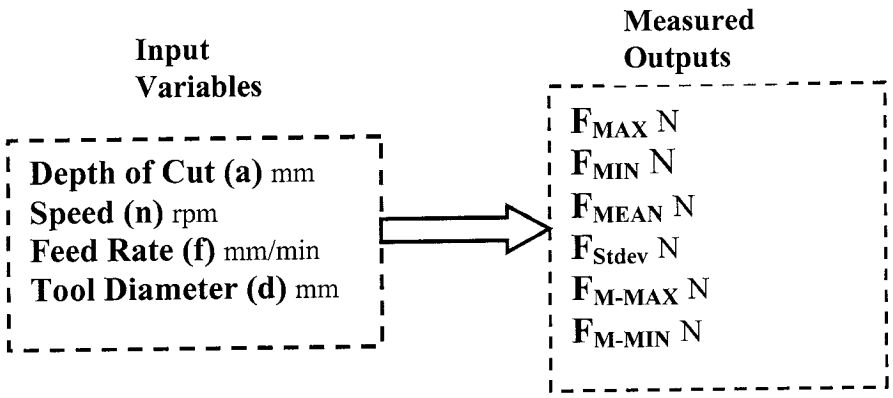

Figure 3.6 Input Variables and the Corresponding Measured Forces

Besides using the equipment and setup elements with their precautions given in their instruction manuals, validation tests should be conducted on them before initiating the work of these experiments. These preliminary validation tests are useful for getting the actual performance and the associate noise of the experimental equipment and setup. If for example, the associate noise is too high, the corresponding experimental results will be untrustworthy and ineffective for building the ANN model. Therefore, a number of initial experiments was performed and repeated on the two milling machines described in section 3.1. Then, the repeatability errors were calculated for the six output values: F_{Max}, F_{Min}, F_{Mean}, F_{Stdev}, $F_{M\text{-}Max}$, and $F_{M\text{-}Min}$, using Equation 3-2.

$$\textbf{Repeatability Error} = \frac{\textbf{Absolute } (V_1 - V_2)}{V_1} \qquad \textbf{3-2}$$

Where, V_1 is the measured force value of the first experimental run and V_2 is the measured force value of the repeated experimental run. The repeatability error measures the possibility of each machine to respond the identical performance when using the same input machining parameters. The results are listed in Table 3.2 and Table 3.3. The results show that the mean repeatability error is 5.95% for CNC machine and is 8.26% for CU. The results also show that the resultant errors from F_{Min} and $F_{M\text{-}Min}$ are too high and once excluded the mean repeatability error for CNC and CU become 2.29% and 3.51% respectively. Accordingly, the two machines can be acceptably trustworthy for the F_{Max}, F_{Mean}, F_{Stdev}, and $F_{M\text{-}Max}$ values, while they are not trustworthy for F_{Min} and $F_{M\text{-}Min}$ values. One possible reason for the increase of repeatability error for F_{Min} and $F_{M\text{-}Min}$ is that their values are very small and consequently will be strongly affected by the noise variations resulting from many sources such as machine noise, vibrations and force measuring equipment.

Table 3.2 Repeatability Testing Experiments for the CNC machine.

a mm	n rpm	f mm/min	d mm	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	$F_{M\text{-}Max}$	$F_{M\text{-}Min}$
1.5	1000	50	10	126.328	0.211	35.888	31.478	95.496	1.607
				125.893	0.248	36.539	31.579	95.562	1.694
The Repeatability Error				**0.34%**	**17.42%**	**1.81%**	**0.32%**	**0.07%**	**5.39%**
1	750	100	8	124.808	0.247	43.494	34.223	104.865	2.592
				129.444	0.297	45.604	33.207	106.357	3.355
The Repeatability Error				**3.71%**	**20.31%**	**4.85%**	**2.97%**	**1.42%**	**29.43%**
0.5	500	200	12	170.909	0.433	53.085	38.369	122.287	5.046
				173.491	0.415	54.724	39.796	126.653	5.193
The Repeatability Error				**1.51%**	**4.31%**	**3.09%**	**3.72%**	**3.57%**	**2.93%**
The Mean Errors = 5.95%				**1.86%**	**14.01%**	**3.25%**	**2.34%**	**1.69%**	**12.58%**

Table 3.3 Repeatability Testing Experiments for the CU machine.

a mm	n rpm	f mm/min	d mm	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	1600	230	12	193.296	0.189	51.349	51.647	156.462	3.591
				199.995	0.174	52.057	51.948	158.120	3.449
The Repeatability Error				**3.47%**	**8.04%**	**1.38%**	**0.58%**	**1.06%**	**3.93%**
2.5	800	230	8	415.081	0.399	222.959	92.558	346.261	23.316
				442.991	0.469	228.989	96.286	353.982	21.453
The Repeatability Error				**6.72%**	**17.60%**	**2.70%**	**4.03%**	**2.23%**	**7.99%**
0.5	560	100	10	130.995	0.170	31.734	30.935	89.473	2.256
				124.052	0.264	32.986	32.166	92.486	2.201
The Repeatability Error				**5.30%**	**55.48%**	**3.95%**	**3.98%**	**3.37%**	**2.44%**
1.5	560	100	10	264.828	0.205	70.154	76.550	198.989	1.590
				255.130	0.210	69.240	74.596	194.386	1.922
The Repeatability Error				**3.66%**	**2.14%**	**1.30%**	**2.55%**	**2.31%**	**20.86%**
The Mean Errors = 8.26%				**5.23%**	**25.07%**	**2.65%**	**3.52%**	**2.64%**	**10.43%**

3.3 Experimental models

In order to vigorously model the milling force, an enormous number of experiments are needed over several ranges of feasible input parameters, so that the non-linearity and the interactions of the input and output parameters can be modeled. However, approximately similar modeling results can be achieved with lower number of experiments and consequently lower experimental cost, when the experiments are effectively planned along using powerful modeling tool.

As discussed before, four variables namely, Depth of cut a, spindle speed n, feed rate f, and tool diameter d are selected, while other variables such as tool material, number of tool flutes, and work piece material are chosen at certain level. Therefore, every experiment represents one point in 4-D space. The X-Y-Z measured forces components are acquired then the resultant force is determined and presented as discussed before. The resultant cutting force variations are determined only at the cutting time while the force variations before and after cutting time are not considered in the calculation. The uncut

force variations represent the machine vibration and other noise variations. Trying to reduce the effect of noise variations, the mean of uncut force variations was calculated and then it was subtracted from the cut force variations. This is illustrated as an example in Figure 3.7.

Using fractional factorial designs, a number of orthogonal Arrays (OAs) are presented considering several levels covering the chosen space of the given input variables. At the beginning, 5 experimental models called UL8-1, UL9, UL27-1, UL32, and UL25 considering two, three, four, and five levels for each variable parameter are selected. Then additional two models are presented called UL8-2, and UL27-2 using two and three variable levels. Besides these experimental models, a set of 21 experiments covering different input values is presented. This model has some input values within the selected range and other points outside that range. The role of that model is to provide unseen experiments for the ANN for the need of validation and generalization test with the range and outside the range.

All these experimental models are conducted using the CU machine while another two models called NL8-1 and NL8-2 are utilizing only two variable values, are performed on the CNC machine. Another set of 17 experiments chosen to have different input positions in the space is conducted on the CNC machine called N-ad and used for the validation process. Table 3.4 and Table 3.5 summarize the chosen input parameters and the corresponding outputs. Figures 3.8~3.18 illustrate the input points in the space corresponding to the tools used for all experimental models. The inclusive details can be found in the appendix B.

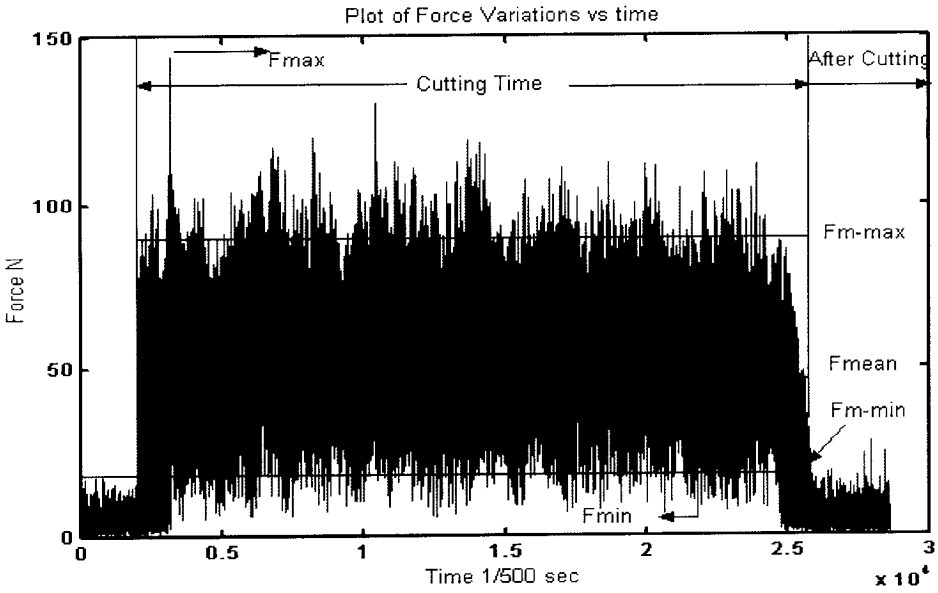

Figure 3.7 Force Variations vs. Time.

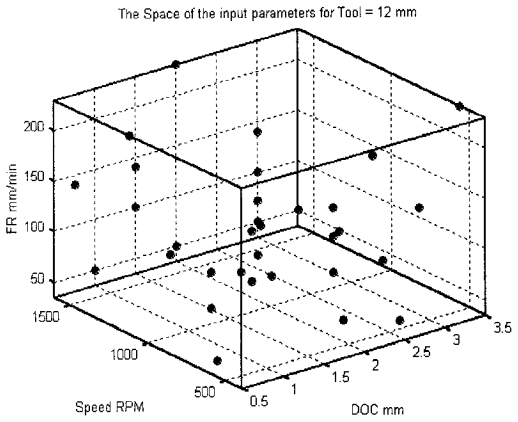

Figure 3.8 A Space Representation of CU Input Parameters for Tool = 12 mm.

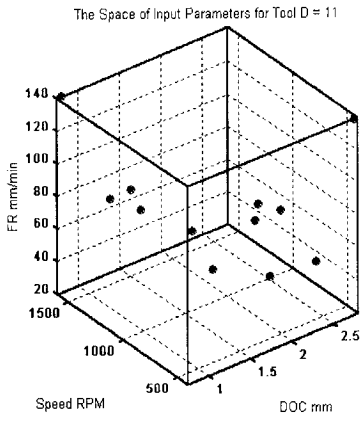

Figure 3.9 A Space Representation of CU Input Parameters for d = 11 mm.

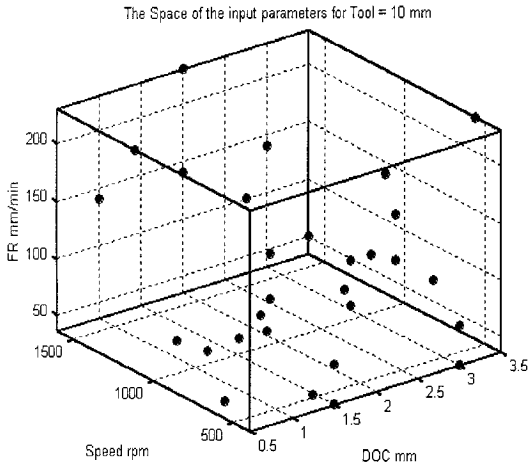

Figure 3.10 A Space Representation of CU Input Parameters for d = 10 mm.

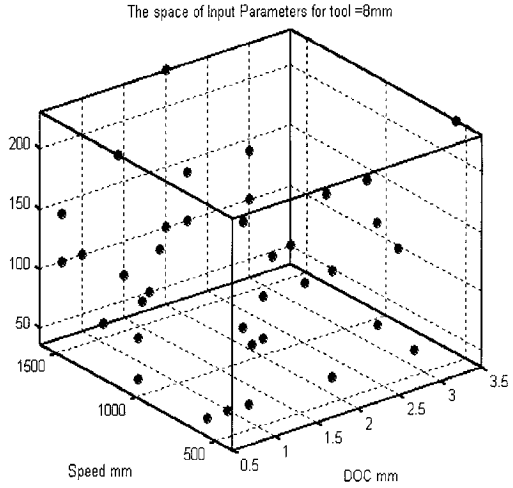

Figure 3.11 A Space Representation of CU Input Parameters for d = 8 mm.

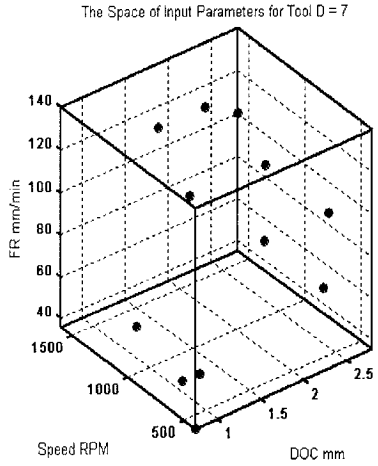

Figure 3.12 A Space Representation of CU Input Parameters for d = 7 mm.

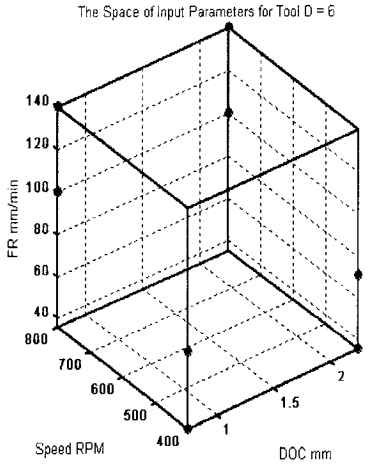

Figure 3.13 A Space Representation of CU Input Parameters for d = 6 mm.

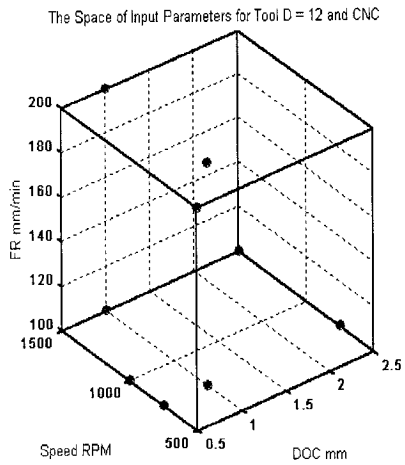

Figure 3.14 A Space Representation of CNC Input Parameters for d = 12 mm.

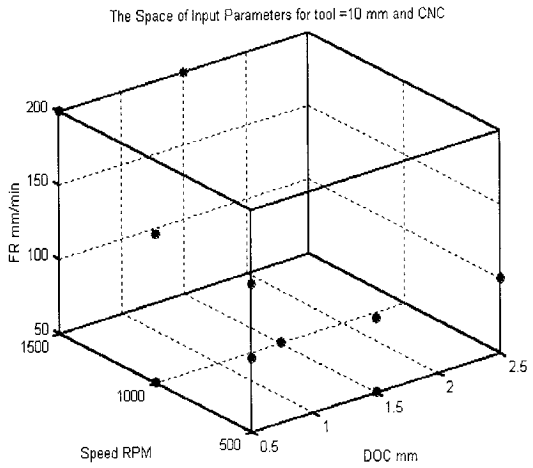

Figure 3.15 A Space Representation of CNC Input Parameters for d = 10 mm.

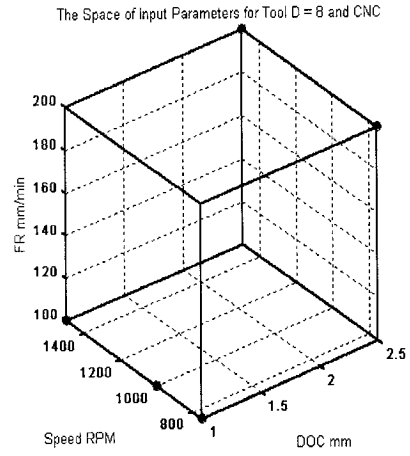

Figure 3.16 A space Representation of CNC input parameters for d = 8 mm.

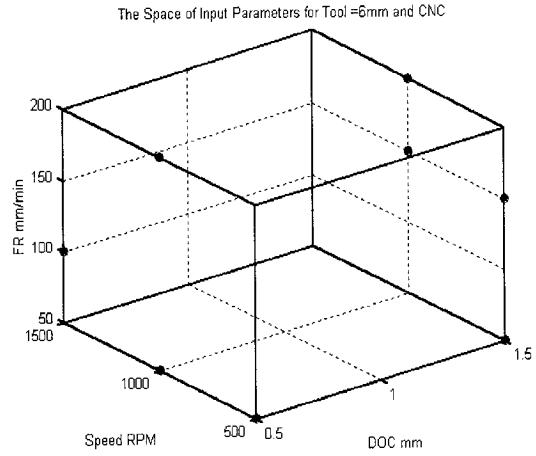

Figure 3.17 A space Representation of CNC input parameters for d = 6 mm.

Table 3.4 Input Variables for each Experimental Model.

Model Name	Model Levels			
	a mm	n rpm	f mm/min	d mm
UL8_1	1.5	800	71	8
	3	1600	140	12
UL9	0.8	560	50	7
	1.8	1120	100	8
	2.8	1600	140	11
UL27_1	0.5	560	50	8
	2	1120	140	10
	3.5	1600	280	12
UL32	0.75	400	35.5	6
	1.5	800	71	8
	2.25	1120	100	10
	3	1600	140	12
UL25	0.75	400	35.5	7
	1.25	560	50	8
	1.75	800	71	10
	2.25	1120	100	11
	2.75	1600	140	12
UL8_2	1	560	50	7
	2.5	1120	100	11
UL27_2	1	560	50	8
	2	1120	100	10
	3	1600	140	12
NL8_1	0.5	500	50	6
	1.5	1000	150	10
NL8_2	1	750	100	8
	2.5	1500	200	12

Table 3.5 A Summary of Force Components for Experimental Models.

Model Name	Value Definition	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
UL8-1	max	367.49	0.56	138.58	108.10	292.81	4.30
	mean	253.18	0.34	75.44	62.97	190.97	2.90
	min	106.04	0.11	32.37	20.77	69.18	1.79
	std	105.11	0.15	33.22	29.58	84.39	0.87
UL9	max	494.76	0.79	251.94	107.66	378.66	31.57
	mean	239.83	0.55	88.79	54.42	182.52	8.54
	min	112.11	0.36	30.63	21.61	80.93	1.95
	std	120.29	0.12	66.67	32.04	98.15	9.54
UL27-1	max	1283.47	9.15	643.58	302.60	1057.47	88.18
	mean	388.24	2.02	159.41	88.71	305.89	15.92
	min	81.22	0.27	27.52	13.62	61.12	2.46
	std	307.58	2.23	165.84	75.94	263.32	22.51
UL32	max	410.42	2.28	204.13	122.35	344.82	33.92
	mean	251.36	0.58	89.24	60.58	187.08	7.95
	min	109.10	0.11	24.98	19.80	69.55	1.43
	std	97.74	0.54	53.55	25.26	80.48	8.92
UL25	max	666.67	2.15	284.02	169.84	518.62	27.51
	mean	244.43	0.67	87.98	58.27	185.17	7.54
	min	90.74	0.20	24.94	14.52	58.33	2.10
	std	129.66	0.53	62.33	35.08	103.57	6.43
UL27-2	max	591.64	3.52	215.99	174.13	460.20	23.06
	mean	285.55	0.89	89.24	69.57	218.09	6.79
	min	111.80	0.25	33.02	22.50	86.00	1.95
	std	125.12	0.77	47.02	38.86	99.67	4.95
UL8-2	max	355.80	0.93	164.76	77.08	280.99	24.43
	mean	231.69	0.45	86.74	43.77	166.78	8.85
	min	126.18	0.15	34.40	25.11	89.45	1.45
	std	89.02	0.29	45.31	18.28	67.03	7.48
NL8-1	max	366.19	0.79	126.57	95.12	284.87	10.09
	mean	159.80	0.37	50.20	38.76	122.43	3.85
	min	62.57	0.11	19.82	12.50	47.26	1.65
	std	98.24	0.23	35.69	26.67	77.72	2.98
NL8-2	max	416.04	0.82	198.09	110.26	341.06	30.43
	mean	281.17	0.35	93.38	68.02	221.55	7.80
	min	127.13	0.06	39.65	29.99	105.61	2.23
	std	100.81	0.26	53.25	27.32	84.41	9.65

3.4 Analysis of Variance (ANOVA)

In order to determine which variables affect the process, modeling and optimization researcher and practitioner need methods to identify and discriminate among the involved variables. One of the most important methods is analysis of variance (ANOVA). In the following discussion, ANOVA study will be presented using UL27-1 model. The results are summarized in a table including six columns. The first shows the source of the variability. The second shows the sum of Squares (Sum Sq) due to each source. The third shows the degrees of freedom (DF) associated with each source. The fourth shows the Mean Squares (Mean Sq), which is the ratio (Sum Sq)/ (D F). The fifth shows the statistical significance test (F_0). The sixth shows the p-values for the F_0 statistics. This analysis is implemented for the six yields F_{Max}, F_{Min}, F_{Mean}, F_{Stdev}, $F_{M\text{-}Max}$, and $F_{M\text{-}Min}$. This is summarized in Tables 3.6~3.11.

Table 3.6 The ANOVA Summary for FMax of UL27-1 Model.

Source	Sum Sq.	DF	Mean Sq.	F_0	Prob > F_0
a	1177835.5	2	588917.8	44.93	1.0039E-07
n	295304.9	2	147652.4	11.265	0.00067221
f	610165.8	2	305082.9	23.276	0.000010193
d	140487.3	2	70243.64	5.3591	0.01493
Error	235934.2	18	13107.45		
Total	2459727.6	26			

Table 3.7 The ANOVA Summary for FMin of UL27-1 Model.

Source	Sum Sq.	DF	Mean Sq.	F_0	Prob> F_0
a	57.7	2	28.8586	12.6863	0.000365
n	12.8	2	6.3991	2.8131	0.08648
f	3.7	2	1.8412	0.80938	0.46069
d	13.9	2	6.9604	3.0598	0.071799
Error	40.9	18	2.2748		
Total	129.1	26			

Table 3.8 The ANOVA Summary for FMean of UL27-1 Model.

Source	Sum Sq.	DF	Mean Sq.	F_0	Prob> F_0
a	323666.0	2	161833	38.1213	3.38E-07
n	123178.1	2	61589.04	14.5079	0.000177
f	181757.0	2	90878.49	21.4073	1.74E-05
d	10061.0	2	5030.488	1.185	0.3285
Error	76413.9	18	4245.217		
Total	715075.9	26			

Table 3.9 The ANOVA Summary for F_{Stdev} of UL27-1 Model.

Source	Sum Sq.	DF	Mean Sq.	F_0	Prob> F_0
a	73932.3	2	36966.15	50.0226	4.46E-08
n	22397.5	2	11198.73	15.1541	0.000138
f	29518.7	2	14759.34	19.9723	2.69E-05
d	10777.4	2	5388.714	7.292	0.004791
Error	13301.8	18	738.9893		
Total	149927.7	26			

Table 3.10 The ANOVA Summary for FM-Max of UL27-1 Model.

Source	Sum Sq.	DF	Mean Sq.	F_0	Prob> F_0
a	856042.7	2	428021.3	49.32	4.96E-08
n	272096.3	2	136048.1	15.6765	0.000114
f	453211.3	2	226605.7	26.1113	4.78E-06
d	65188.2	2	32594.08	3.7557	0.043334
Error	156212.3	18	8678.46		
Total	1802750.7	26			

Table 3.11 The ANOVA Summary for FM-Min of UL27-1 Model.

Source	Sum Sq.	DF	Mean Sq.	F_0	Prob> F_0
a	4494.3	2	2247.174	12.5593	0.000385
n	2906.6	2	1453.323	8.1225	0.003062
f	2228.5	2	1114.238	6.2274	0.008801
d	329.6	2	164.8201	0.92117	0.41602
Error	3220.6	18	178.9246		
Total	13179.8	26			

Using confidence level = 90%, upper-tail percentage point of the $F_{2;\ 18}$ distribution = 2.62. This concludes that all variables are significant for F_{max} and $F_{M\text{-}Max}$, F_{Stdev}, but (a) and (f) respectively, are the most significant. For F_{Min} and $_{FM\text{-}Min,}$ (a) is also the most significant variable. For other variables, (d) and (n) have little significance, while (f) is insignificant. Regarding F_{Mean}, (a) and (f) are the most significant variables respectively, (n) has some significance, and (d) is not significant. Generally, all process variables are significant and (a) and (f) are the most significant variables. This concludes that the selection of a, n, f, and d as the process variables to model the end milling process is proper selection since, all of them are significant. Furthermore, the modeling capability of UL27-1 can be enhanced with adding experiments that have more (a) and (f) levels. In particular, additional experiments can be conducted containing new (a) and (f) values with the same (d) and (n) values. This can enhance the modeling capabilities, since the larger model host more variations of the process variables that are significant.

CHAPTER 4

MODELING PROCESS USING NEURAL NETWORKS

4.1 Predictive Systems Using Neural Network

The artificial neural network (ANN) is used here as a tool for mapping input vector received (P) by input neurons to output vector (Y) resulting from output neurons. The mapping process tries to approximate the input function F (P) to another unknown function F (P, W), where W is a parameter vector called weight function. This mapping process is achieved by designing an ANN and training it with a number of input vectors and their corresponding output vectors. The task of training is to adjust W at every training pattern to find the best achievable mapping between input and output, by finding the best possible approximation of F (P). Finally, the obtained weight values can be used to predict possible output values when the neural network is responded to the input values. Therefore, the weight values represent knowledge learned through the training patterns. In the training phase, the fractional factorial design (FFD) is used as a systematic way to reduce the number of training experiments, for every experimental model by which ANN is trained. This study started by comparing every model alone then, comparing every possible combination of models. The comparison here is based on the overall mean error of F_{Max}, F_{Mean}, F_{Stdev}, and $F_{m\text{-}max}$ while, the F_{Min} and $F_{m\text{-}min}$ are excluded from the comparison and the error E is computed by using Equation 4-1. The input vector and output vectors for the ANN are respectively, P = [a, n, f, d] and Y = [F_{Max}, F_{Min}, F_{Mean}, F_{Stdev}, $F_{m\text{-}max}$, $F_{m\text{-}min}$].

$$E = \frac{\text{Absolute (F Real-F Predicted)}}{\text{F Real}} \qquad 4\text{-}1$$

In this work, two supervised neural networks are employed to model the end milling. Specifically, they are radial basis network (RBN) and feed forward neural network (FFN). A flexible MATLAB program was developed to model the process of these two neural models. The modeling process used in this program is summarized in Figure 4.1. It starts with determining which training and validation patterns will be used. These patterns are scaled according to some normalization factors. The input patterns are chosen to be normalized between 1 and -1 while the output patterns are normalized at 1 and 0. The normalization process is performed using a linear mapping given in Equation 4-2.

$$F_n = (R - R_{min}) \frac{(S_{max} - S_{min})}{(R_{max} - R_{min})} + (S_{min}) \qquad 4\text{-}2$$

Where, F_n is the normalized variable, R is the real value before normalization, R_{max} and R_{min} are the maximum and the minimum values before normalization, and S_{max} and S_{min} are the maximum and the minimum values after normalization.

After that, the neural model can be initiated after choosing its type and specifying its design parameters. Then, the obtained network will be trained with the chosen training patterns. After that process, the neural model will be tested with the patterns it was actually trained. The purpose of that test is to show the learning capability of the neural model. The obtained result will be rescaled and compared to the actual training patterns and the

corresponding error called training error is stored. The trained neural model will be tested by unseen patterns called validation patterns. The aim of that phase is to test the generalization capability of the neural model. The validation patterns are selected to include some unseen patterns within the range of training patterns and others outside that range. That way, testing the neural model as a general model will be more comprehensive.

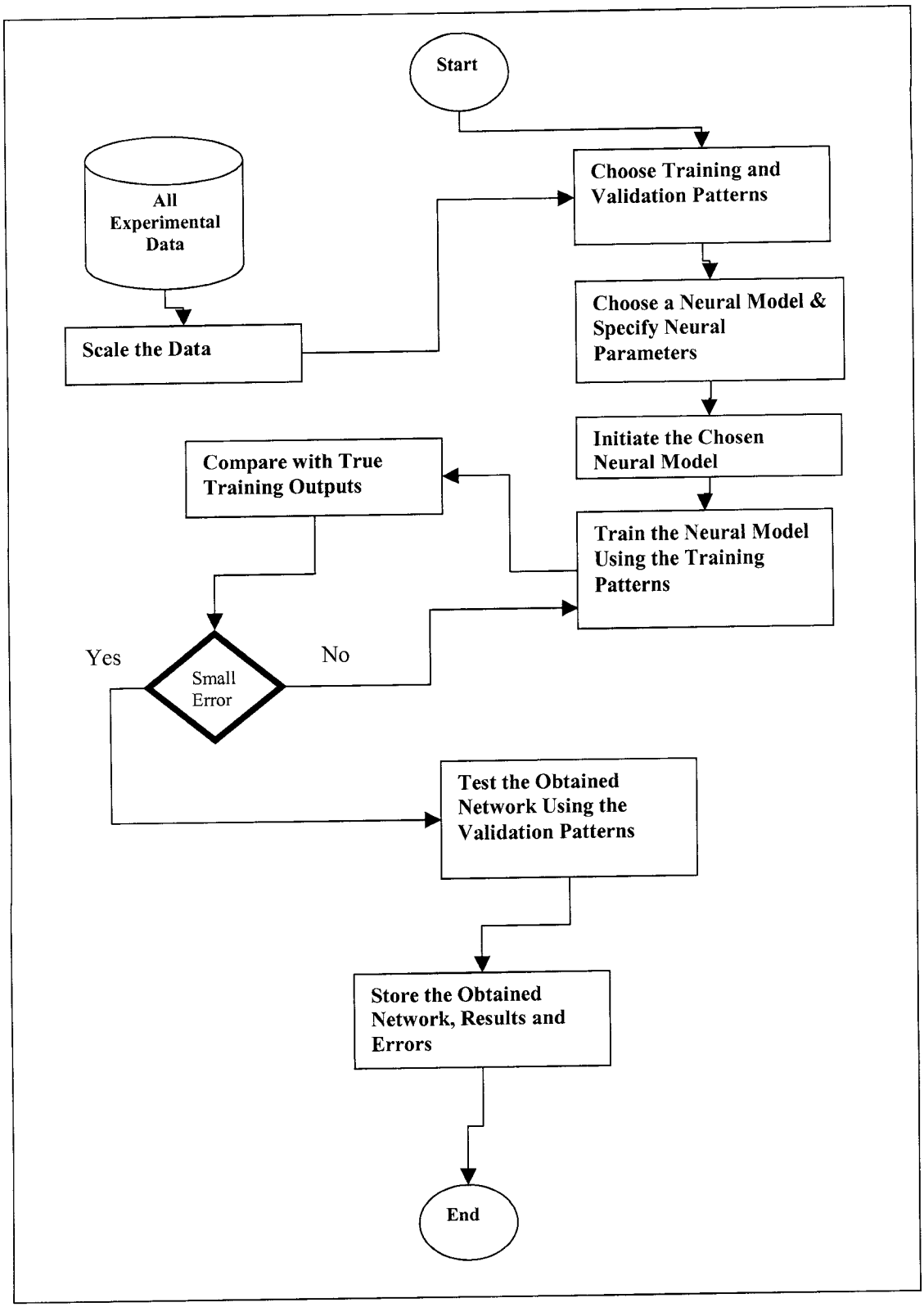

Figure 4.1 The General Procedure Used for Neural Modeling Process.

4.2 Radial Basis Network (RBN)

Radial basis networks consist of two layers. The first layer is a hidden radial basis layer of S^1 neurons, uses the Gaussian function for mapping, and the second layer is an output linear layer called PURLIN of S^2 neurons as shown in Figure 4.2. The profile of the Radial basis transfer function and PURLIN function can be found in Figure 4.3 and Figure 4.4.

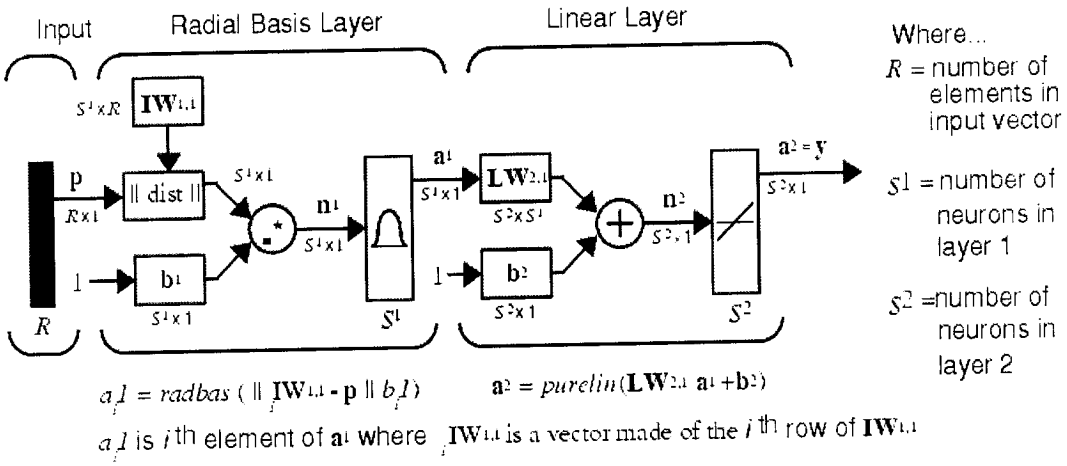

Figure 4.2 The Basic Elements of Radial Basis Neural Network [15].

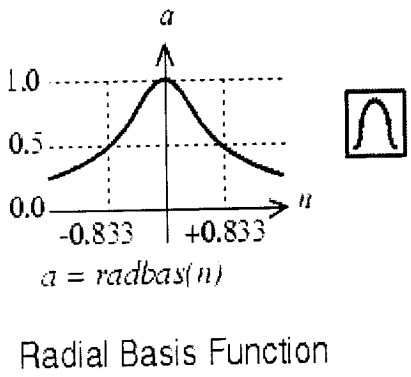

Figure 4.3 Radial Basis Transfer Function [15].

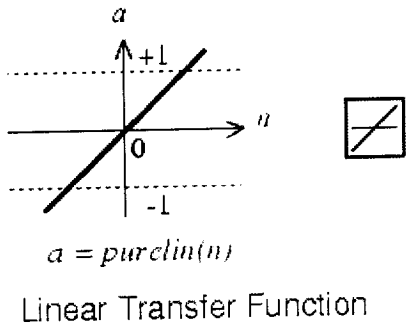

Figure 4.4 Linear Transfer Function [15].

The algorithm of RBN depends on adding neurons to the hidden layer until it meets the specified mean squared error goal or until the size of neurons reach the maximum

allowable number. The design of radial basis NN depends on two main parameters. The first parameter is called spread of radial basis functions. It is recommended to choose a spread value such that it should be greater than the lowest of the input values and smaller than the highest values. The second parameter of radial basis network deals with the stopping criteria of training. This parameter is very important for avoiding over-training, since adding too much training experiments can cause the network to learn the noise in the data. The stopping criteria can be controlled by two methods. The first method is by specifying a certain number of neurons by which the training process will be stopped once it completes that number. The second way is by presenting a training error goal, which will force the model to stop when the training performance reaches that goal. In other words, the training can stop when either the number of neurons reaches the allowable number or when the training error reaches the training goal.

4.2.1 RBN Modeling Process

In this part, the modeling process for end milling operation using sufficient number of experiments was presented. The initial aim was to reach minimum predictive error and to study the possibility of using certain experimental models as training experiments instead of using just sufficient training experiments. This can accelerate the convergence of the model.

4.2.1.1 The Selected Parameters of RBN

The first studied parameter was the spread value. As an initial search interval, different values, from 0 - 1 are used and studied using sufficient number of training experiments, since both input and output values were scaled between -1 and 1 for input vectors and between 0 and 1 for output vectors. Results are available in appendix C1. The

result shows that the best spread values are within the range 0.8 to 0.95. This is illustrated in Figure 4.5 and Figure 4.6.

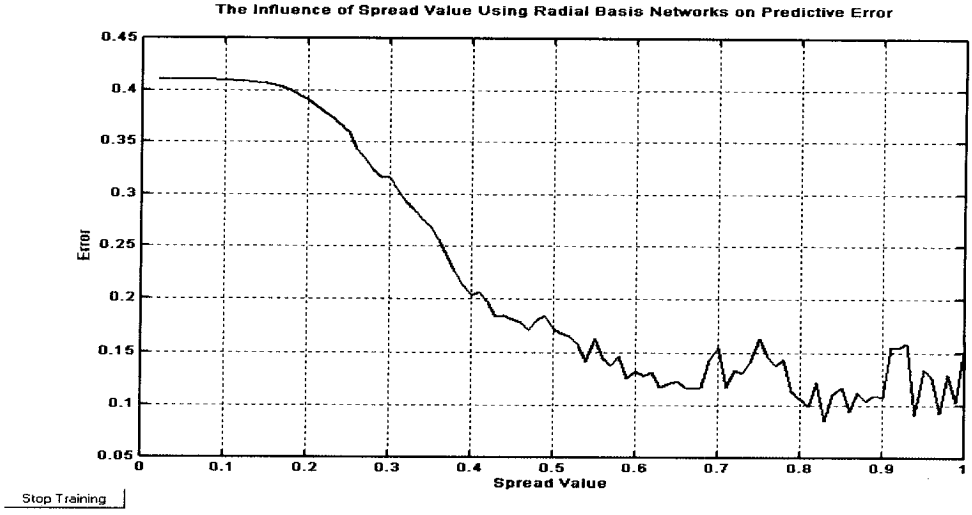

Figure 4.5 The Influence of Spread Value Using Radial Basis Networks on Predictive Error.

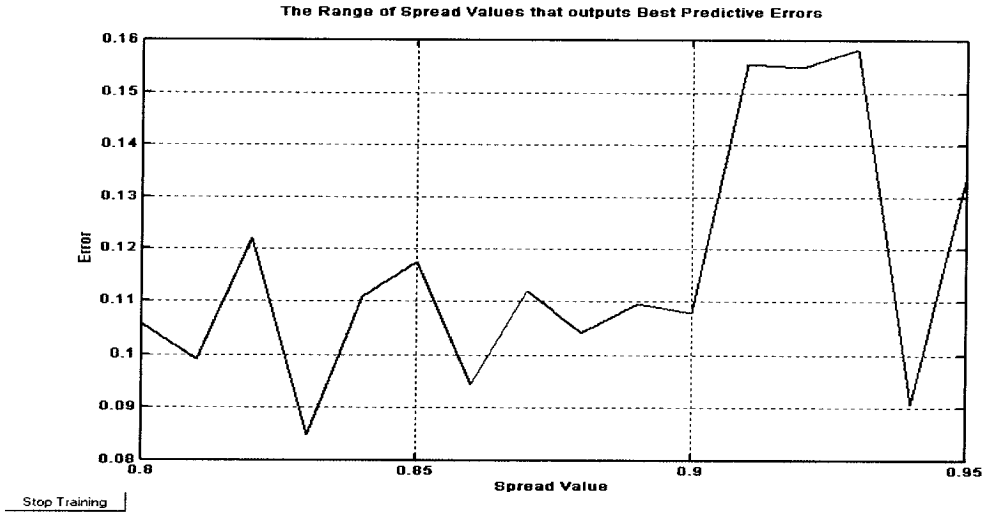

Figure 4.6 The Range of Spread Value That Outputs The Best Predictive Errors.

Therefore, the spread value was initially, chosen at 0.83, then at the end of training, an investigation was carried out to attempt another spread value that can reduce the errors. The second RBN parameter is to decide the value of training goal. As shown in Figure 4.7 the training of RBN should be stopped when training performance reaches a value in the

interval 0.9~1.5 based on the sum square errors. Therefore, the training error goal is chosen to be 1. The third RBN parameter is the number of allowable neurons that can be used. After trying different values for the neuron number, the allowable neurons are selected as 30. Figure 4.8 shows the best neuron number that leads to best predictive error in the interval 23 - 37 neurons.

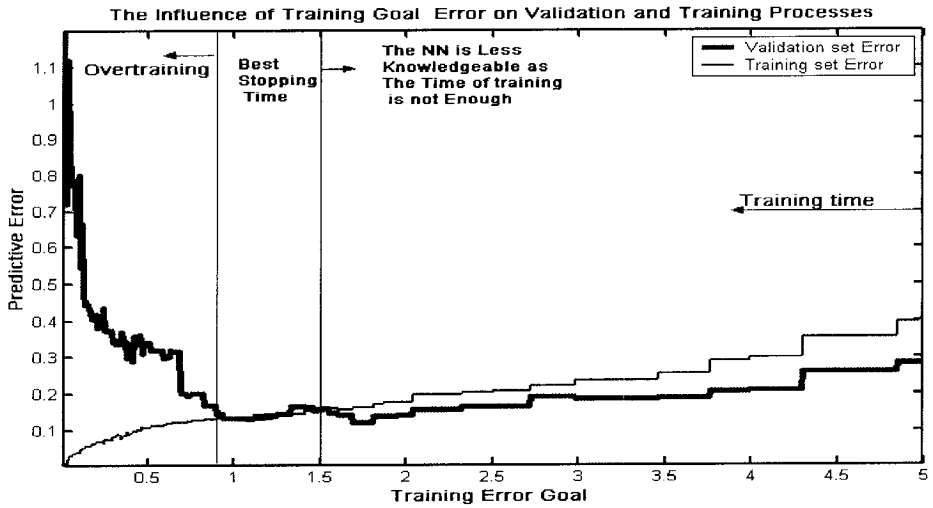

Figure 4.7 The Influence of Training Goal Error on Validation and Training Processes.

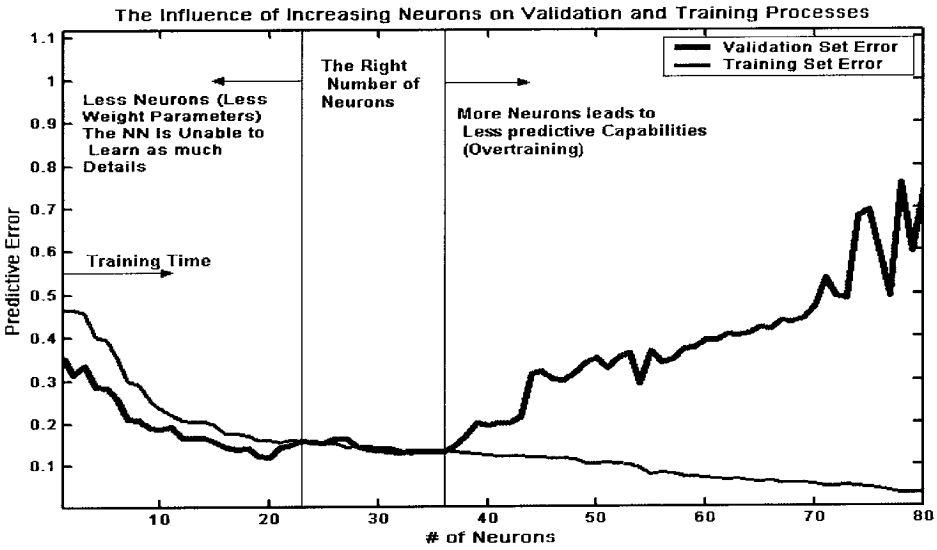

Figure 4.8 Influence of Increasing Neurons on Validation and Training Processes.

4.2.1.2 Training and Validation phases for RBN

After selecting the features of RBN, every experimental model is used alone as training patterns. This outputs different predictive force errors as a result of different training experiments. This means that every individual model has a different potential for approximation. In specific, the results show that the three levels models have better capabilities to train the RBN especially, UL27-1 which has higher input ranges. The results are listed in Table 4.1 and a comparison for E_{mean} and E_{Stdev} for basic models is illustrated in Figure 4.9. The results show also that the predictive force error is too high and more training experiments are needed. Therefore, the strategy is to use model UL27-1 as a basic model when using a composed model consisting of two models or more. This will start with studying the possible composed model resulting from two basic models. The resultant model will be called the 2-composed model. The results of training 2-composed model can be found in Table 4.2 and a simple illustration shows their mean and standard deviation errors in Figure 4.10.

Table 4.1 The Predictive Force Errors Result From Training the Basic Models.

Model Name	Predictive Force Errors						Excluding FMin., FM-Min.	
	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
UL8-1	0.43	2.48	0.46	0.46	0.46	0.47	0.45	0.09
UL8-2	0.44	2.60	0.55	0.46	0.45	0.92	0.47	0.08
UL9	0.46	3.48	0.53	0.49	0.49	0.81	0.49	0.08
UL27-1	0.28	8.72	0.28	0.34	0.31	0.74	0.30	0.08
UL27-2	0.38	4.11	0.45	0.40	0.40	0.71	0.41	0.07
UL32	0.41	3.86	0.51	0.44	0.43	0.93	0.45	0.11
UL25	0.43	4.25	0.60	0.44	0.47	0.89	0.48	0.10

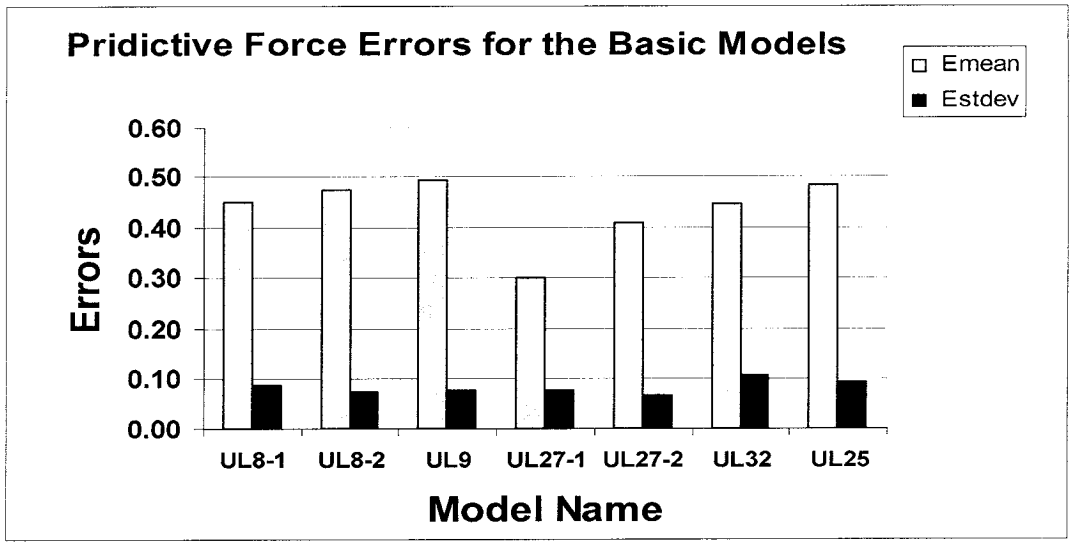

Figure 4.9 RBN Predictive Force Errors for the Basic Models.

Table 4.2 Predictive Force Errors for 2-composed model.

Basic models with UL27-1	Model Name	Predictive Force Errors						Excluding F_{Min}, $F_{M\text{-}Min}$	
		F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	$F_{M\text{-}Max}$	$F_{M\text{-}Min}$	E_{Mean}	E_{Stdev}
UL8-1	UL35-1	0.21	6.26	0.18	0.28	0.23	0.43	0.23	0.09
UL8-2	UL35-2	0.27	7.53	0.30	0.32	0.29	0.85	0.29	0.08
UL9	UL36	0.24	4.21	0.24	0.28	0.26	0.53	0.25	0.08
UL27-2	UL54	0.17	4.52	0.13	0.22	0.17	0.58	0.17	0.07
UL32	UL59	0.18	3.64	0.13	0.17	0.16	0.64	0.16	0.07
UL25	UL52	0.19	4.18	0.23	0.17	0.17	0.65	0.19	0.06

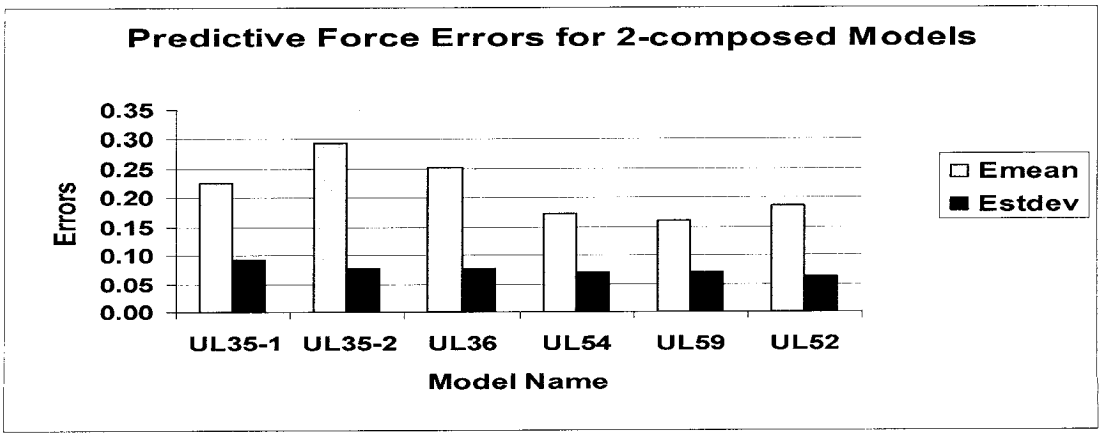

Figure 4.10 Predictive Force Errors for the 2-composed Models.

The results of constructing 2-composed models show that the mean of predictive force errors was reduced by half. This means that adding more training experiments succeeded to provide useful knowledge and better mapping to the RBN model. The results also show that the best predictive error equals 16% and is achieved when using the 2-composed model UL59 as the training patterns. The mean predictive error curve resulting from using UL59 model is illustrated in Figure 4.11 and Figure 4.12 illustrates the error curve for F_{Max}, F_{Min}, F_{Mean}, and $F_{M\text{-}Max}$ for last 20 training experiments. The same curve is illustrated for F_{Min} and $F_{M\text{-}Min}$ in Figure 4.13 that shows that the corresponding error for $F_{M\text{-}Min}$ is better than F_{Min}. However both results are too high. This means that noise associated with F_{Min} and $F_{M\text{-}Min}$ mislead the neural model. The best predictive error that can be achieved until now using UL59 is 16%. Since the error is still very high, there is a need for adding more experiments. This leads to use 3-composed models starting with UL59 as a basic model

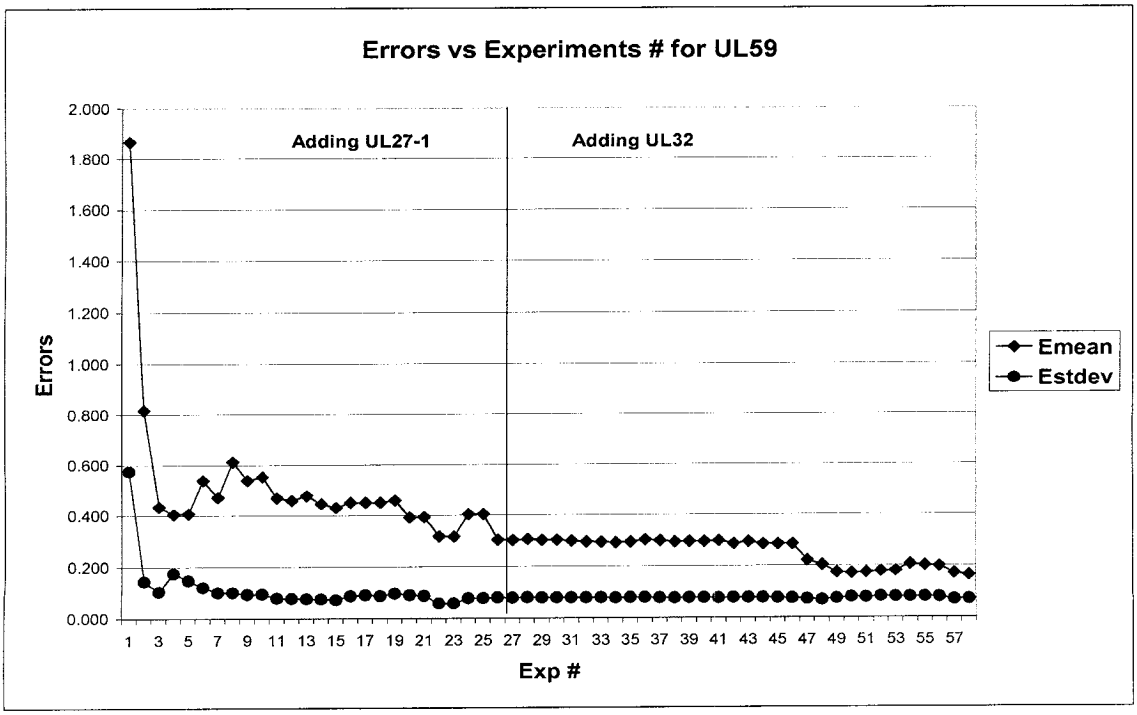

Figure 4.11 Predictive Force Errors vs. Experiment Number for using UL59.

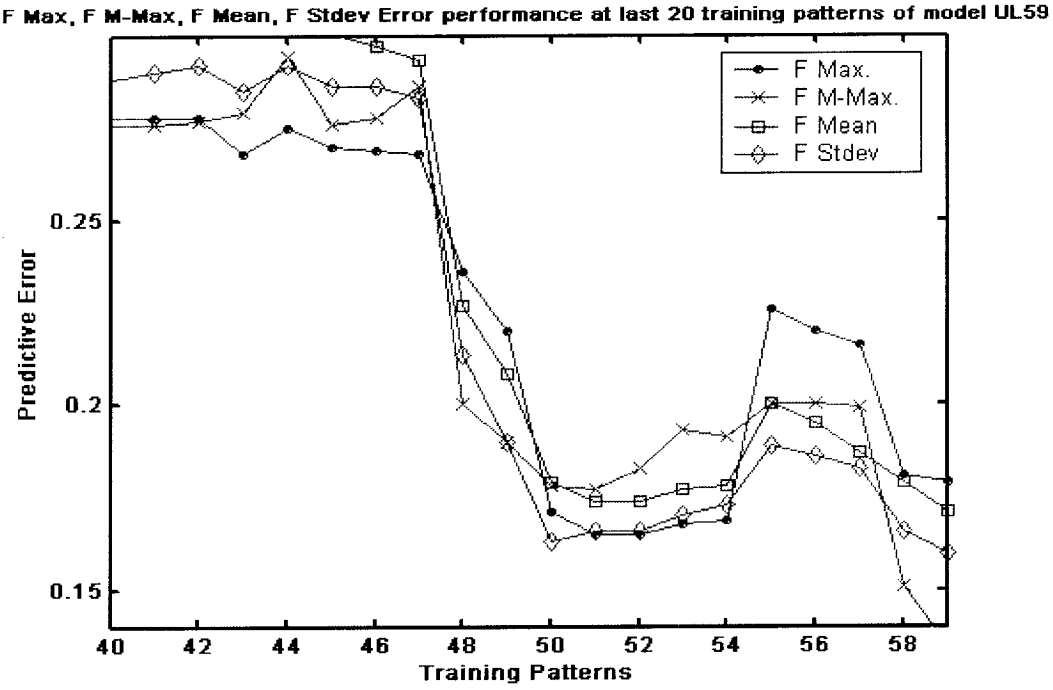

Figure 4.12 F_{max}, $F_{M\text{-}Max}$, F_{Mean}, F_{Stdev} Error performance at last 20 training patterns of model UL59

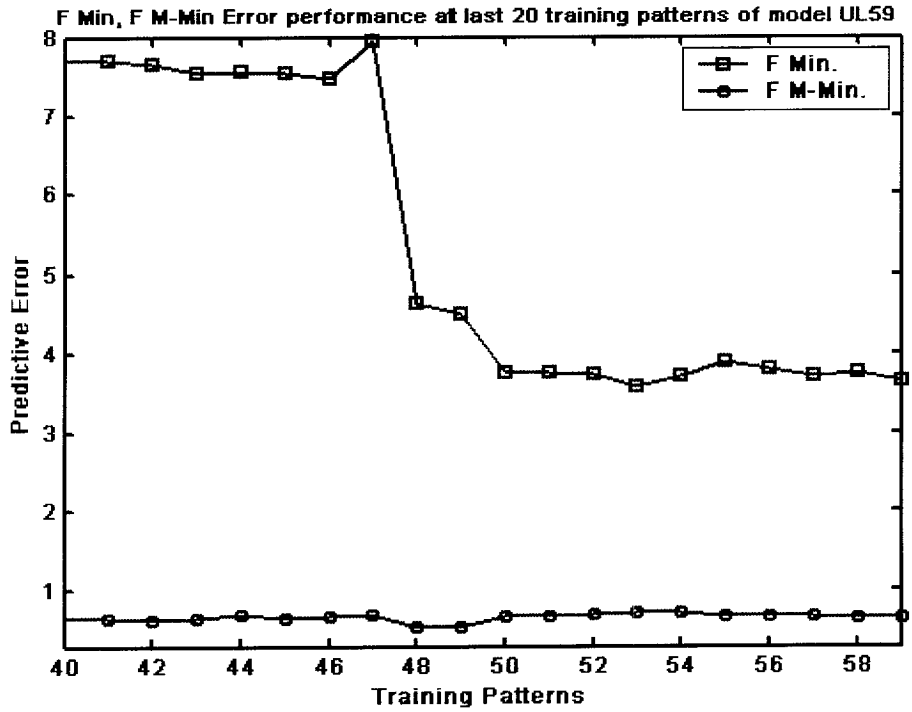

Figure 4.13 F_{Min} & $F_{M\text{-}Min}$ Error performance at last 20 training patterns of model UL59

Table 4.3 gives the possible 3-composed models using UL59. This is also illustrated in Figure 4.14. The result shows that adding new training experiments enhances the prediction capability. However, it also shows that the degree of enhancement doesn't depend on the number of experimental models added. For instance, model UL86 has the largest experimental numbers but it has predictive errors = 14.9% while UL67-1 has only 67 experiments, and has better predictive errors. However, the best experimental model that offers the best predictive error was UL84. One possible reason could be that the training patterns need to be added with the experimental model that has more useful information that cover new input space location.

Table 4.3 Predictive Force Errors for 3-composed model.

Basic models with UL59	Model Name	Predictive Force Errors						Excluding F_{Min}, $F_{M\text{-}Min}$	
		F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	$F_{M\text{-}Max}$	$F_{M\text{-}Min}$	E_{Mean}	E_{Stdev}
UL8-1	UL67-1	0.132	3.054	0.173	0.146	0.138	0.552	0.147	0.068
UL8-2	UL67-2	0.156	4.726	0.177	0.163	0.171	0.614	0.167	0.082
UL9	UL68	0.159	3.007	0.170	0.170	0.147	0.412	0.162	0.064
UL27-2	UL86	0.133	3.568	0.171	0.156	0.135	0.561	0.149	0.060
UL25	UL84	0.127	4.319	0.138	0.133	0.115	0.625	0.128	0.055

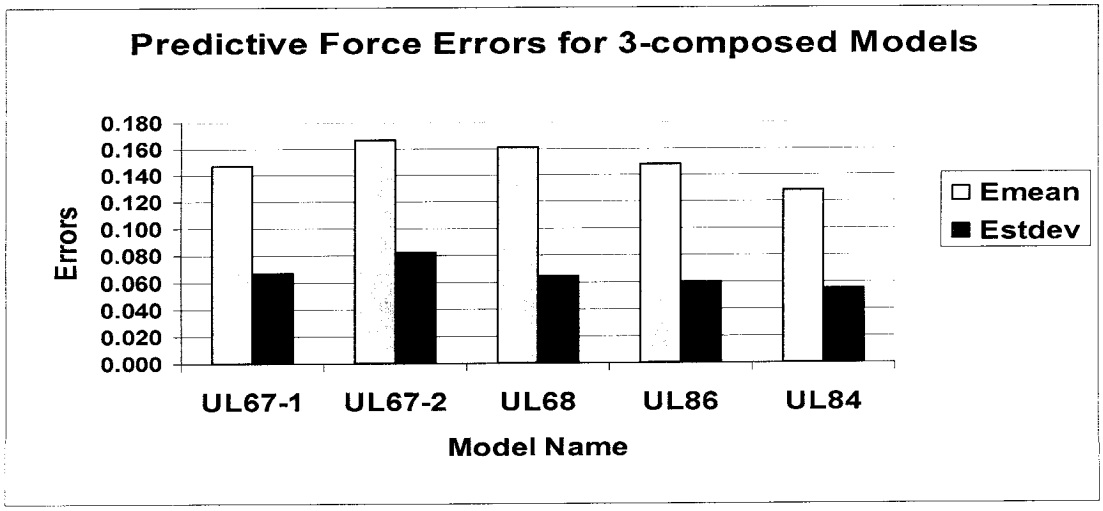

Figure 4.14 Predictive Force Errors for 3-composed models.

UL84 was used for training as it has predictive errors of 12%. Accordingly, it was selected. The models or even points that present new space points should be taken into consideration. Therefore, not only all possible 4-composed models will be studied but also analysis of every added experiment to UL84 will be presented. This will help overcoming the effect of over-training problem and giving the optimum number of training experiments. Figure 4.15 and Table 4.4 present the mean and standard deviation errors for all 4-composited models as well as the mean and standard deviation errors for the effective points for each model that reduce the predictive errors. The results show that adding the effective points results in better predictive error than adding all model experiments.

Table 4.4 Predictive Force Errors for 4-Composed models and Effective Points.

Basic models with UL84	Exp. Added	Model Name	Predictive Force Errors						Excluding F_{Min}, $F_{M\text{-}Min}$	
			F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	$F_{M\text{-}Max}$	$F_{M\text{-}Min}$	E_{Mean}	E_{Stdev}
UL8-1	1 to 8	UL92-1	0.17	3.13	0.17	0.16	0.15	0.6	0.16	0.07
	7	UL86-1	**0.123**	**4.754**	**0.153**	**0.107**	**0.095**	**0.698**	**0.119**	**0.056**
UL8-2	1 to 8	UL92-2	0.127	4.64	0.147	0.145	0.127	0.543	0.136	0.054
	6	UL85	**0.12**	**4.897**	**0.148**	**0.106**	**0.098**	**0.676**	**0.118**	**0.059**
UL9	1 to 9	UL93	0.168	2.731	0.153	0.145	0.129	0.377	0.149	0.053
	2	UL86-2	**0.119**	**4.522**	**0.138**	**0.106**	**0.099**	**0.59**	**0.116**	**0.056**
UL27-2	1 to 27	UL111	0.138	2.275	0.117	0.134	0.101	0.563	0.123	0.056
	10 to12	UL87	**0.145**	**3.254**	**0.105**	**0.13**	**0.105**	**0.531**	**0.121**	**0.06**
Adding all Effective points		UL90	**0.107**	**4.063**	**0.103**	**0.095**	**0.077**	**0.552**	**0.095**	**0.053**

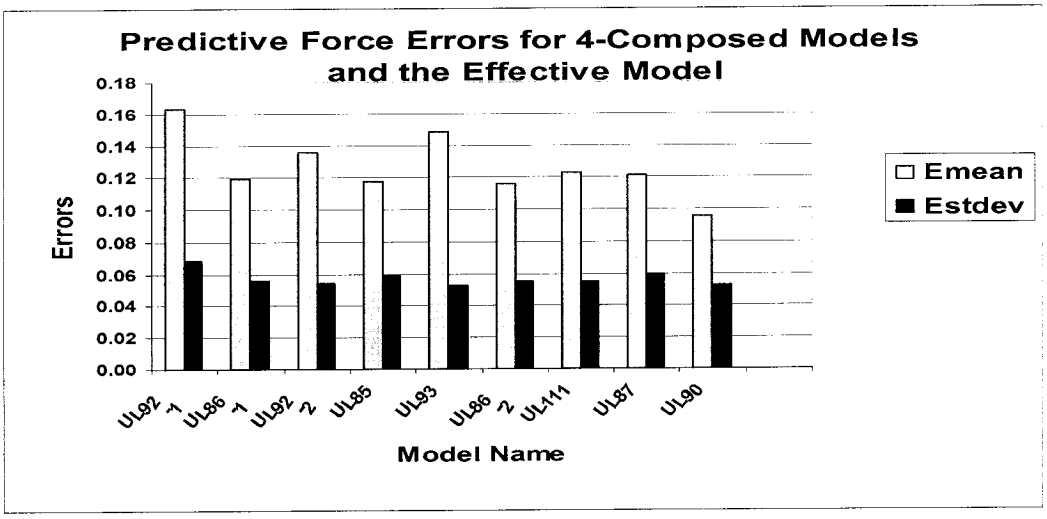

Figure 4.15 Predictive Force Errors for 4-composed models and effective points.

The mean errors gained from training UL90 is 0.095. In order to confirm the result obtained, the excluded experiments were added to the validation experiments and the RBN was tested again. The predictive error for the bigger validation pattern was 0.104 which is very close to the error obtained before. This result gives more confidence for the stability of designed RBN and for the efficiency of using UL90.

All conducted experimental models were combined together in one big model called UL136 then, it was used to train the RBN model. The resultant predictive errors, shown in Table 4.5, explain that away from $F_{Min.}$ and $F_{M\text{-}Min.}$, the mean of the predictive error is 0.1. This demonstrates that using UL136 model is sufficient for modeling the cutting forces of the milling operation. However, using UL90 model resulting from utilizing the previous training procedures was more efficient since it yields almost similar predictive errors from using UL136 with smaller training experiments. This means that UL90 is fast converging better than UL136. This is illustrated in Figure 4.16.

Table 4.5 Predictive Force Errors after using all experimental models.

Model Name	Predictive Force Errors						Excluding F_{Min}, F_{M-Min}	
	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
UL136	0.10	0.87	0.09	0.11	0.10	0.33	0.10	0.05

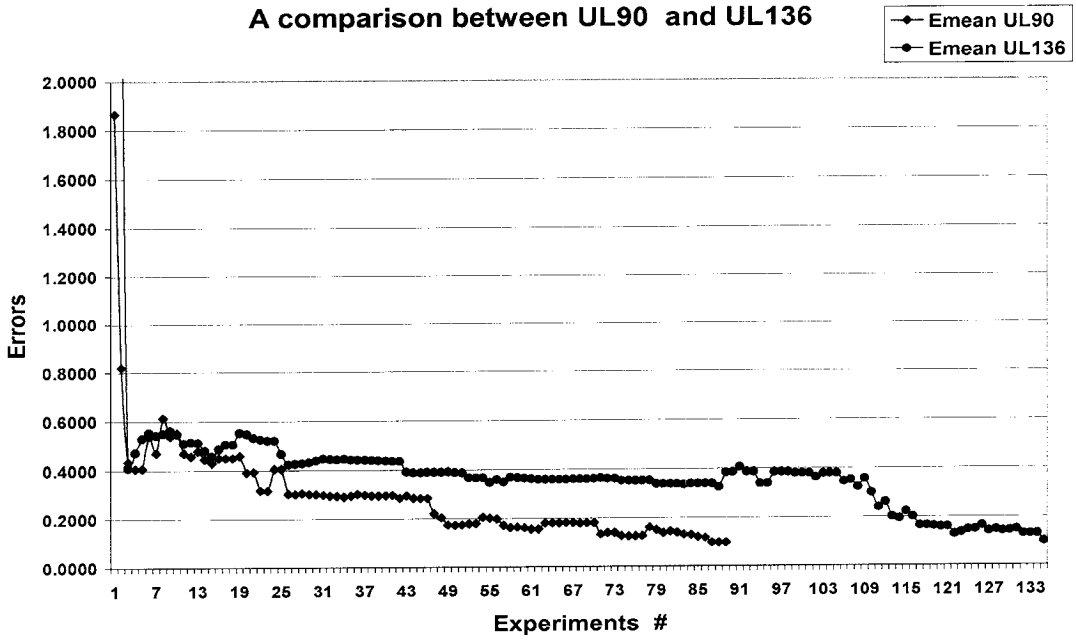

Figure 4.16 A comparison between UL90 and UL136.

At the end of the training phase, an investigation was preformed on both UL90 and UL136 applying another spread values from 0.7 - 1 to search if there is another value that can yield less predictive error. The results of this investigation led to setting the spread value to 0.97 for UL90 and 0.86 for UL136. This adjustment slightly enhanced the error for both the UL90 and UL136. A comparison of the new predictive error is presented in Table 4.6 and in Figure 4.17. The results demonstrate the same conclusion that using UL90 led to an approximately similar predictive error as using UL136 model. Moreover, UL136 requires larger number of experiments to converge to the proper error. This will be referred

to the efficiency of modeling. Finally, Figure 4.18 compares the real values of F_{Max} to the values predicted by the RBN model using UL90. In Figure 4.19 the same comparison is presented using larger validation experiments.

Table 4.6 Predictive errors for UL90 and UL136 after adjusting the spread values.

Model Name	Spread Value	Predictive Force Errors						Excluding F_{Min}, F_{M-Min}	
		F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
UL136	0.86	0.09117	1.55244	0.10493	0.09467	0.08032	0.32462	0.092	0.052
UL90	0.97	0.09286	4.14243	0.12367	0.06927	0.07357	0.58967	0.089	0.055

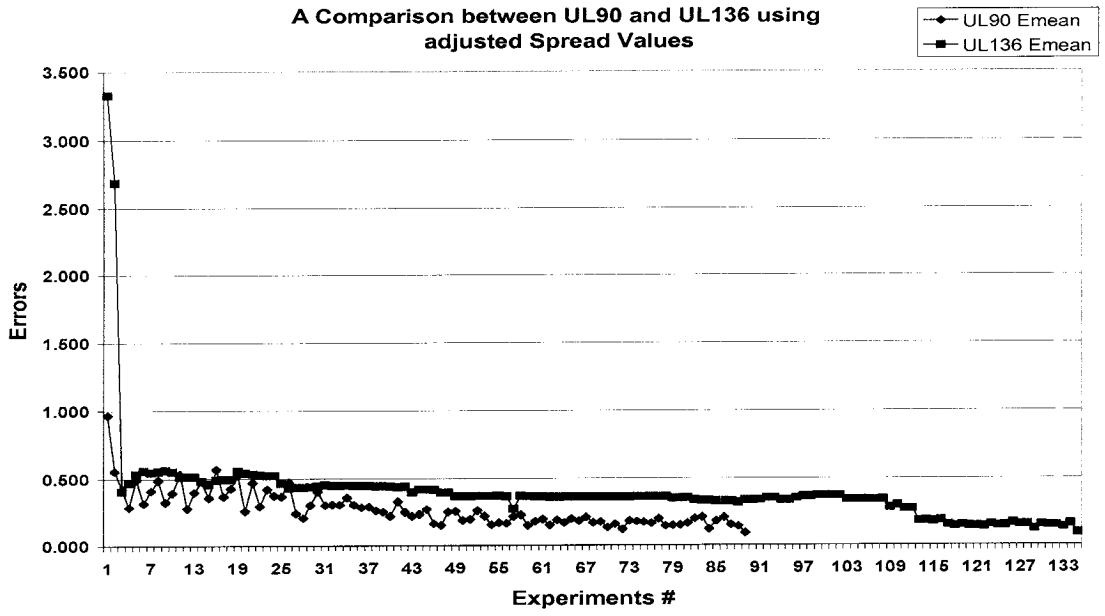

Figure 4.17 A comparison between UL90 and UL136 using the new spread Values

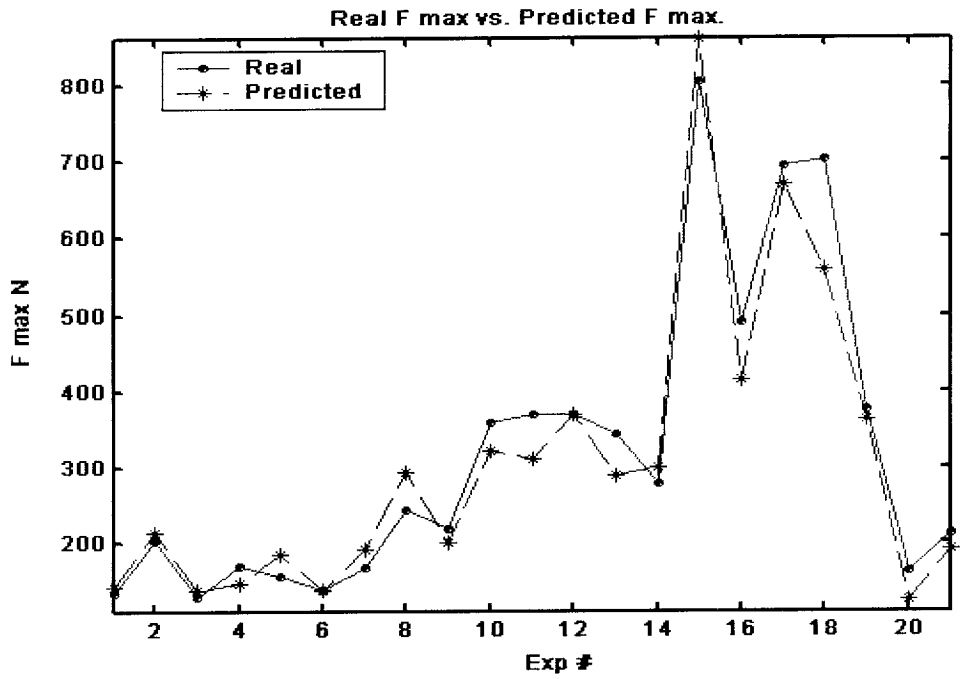

Figure 4.18 A Comparison between Real F max and Predicted F max Using RBN.

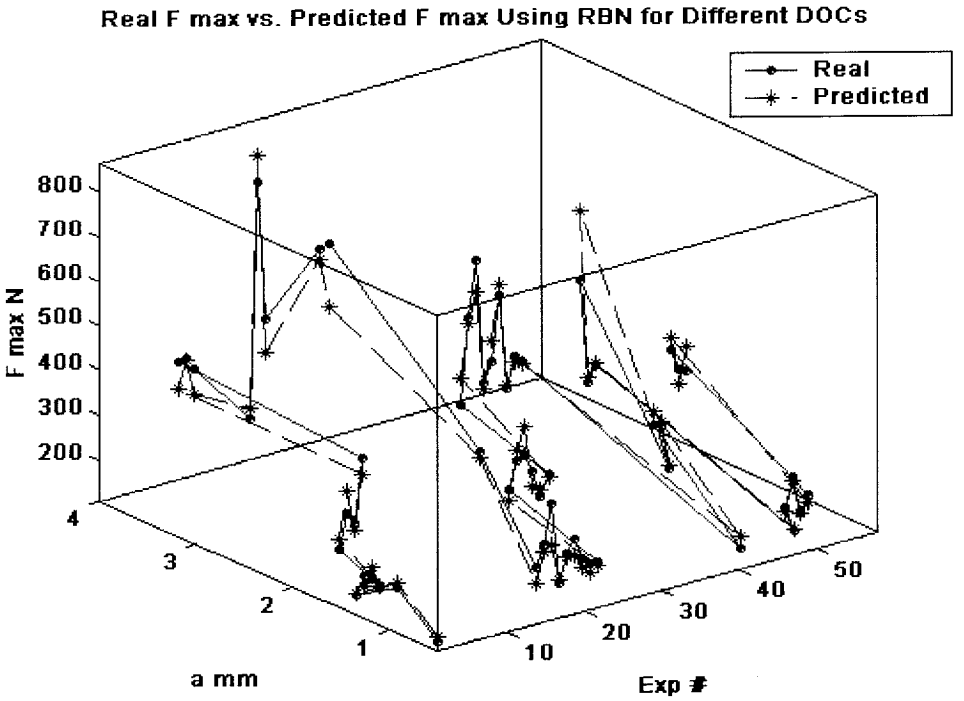

Figure 4.19 Real F_{Max} vs. Predicted F_{Max} for Different depth of cuts Using RBN and Larger Validation Experiments

4.3 Feed-Forward Neural Network (FFN)

In this part, an investigation to utilize FFN, an essential class of neural network as modeling tool for the milling operation. FFN is generally composed of a set of layers that represent the input layer, one or more hidden layers of processing neurons, and an output layer of processing neurons. Figure 4.20 illustrates the basic structure of FFN.

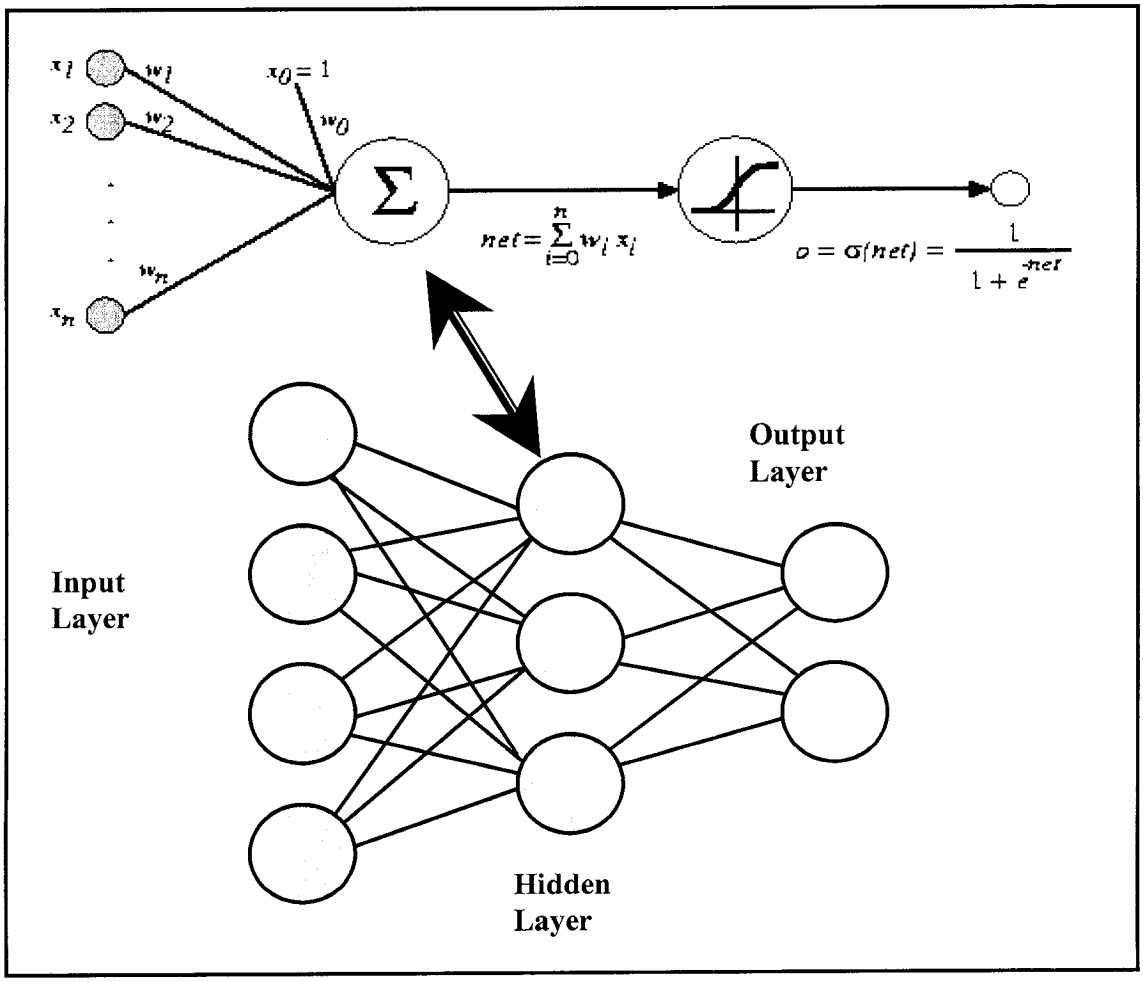

Figure 4.20 The Main Structure of the FFN

This type of neural network has been applied successfully to map complicated nonlinear functions. Error back-propagation algorithm (EBP) is mainly composed of two

passes conducted for each training pattern, namely these two passes are a forward pass and a backward pass. In the forward pass, the signal of input is fed into the network layer by layer from the input layer to the output layer. The resultant output of the network is then used to compute the error between the actual response of the network and the real output. This error is used to adjust the weights of different layers in the backward pass layer by layer from the output layer to the input layer. An illustration of these two passes for one hidden layer FFN can be found in figure 4.21.

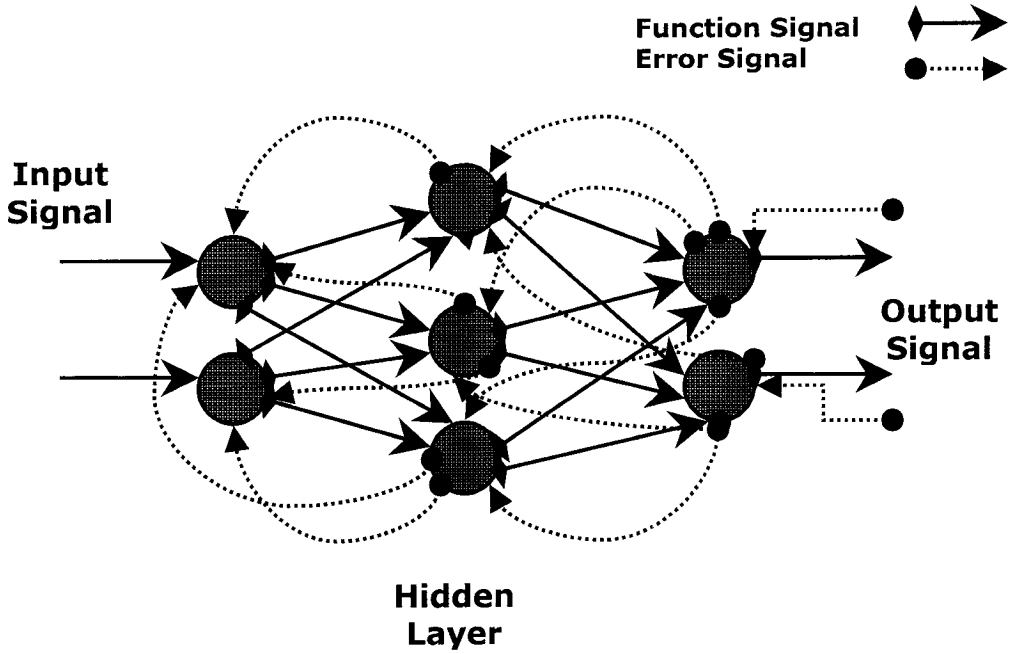

Figure 4.21 Two basic passes of EBP for one hidden layer FFN

4.3.1 FFN Process Modeling

In this part, a study is carried out to design robust FFN using the capability of MATLAB toolbox functions. Designing FFN is critical and complex since it has several elements such as number of hidden layers, number of neurons in the hidden layers, type of

the transfer function in each layer, and type of training algorithms and learning functions. Besides, other general designing elements are needed to be specified such as number of epoch and the value of training goal. A FFN designer usually goes through a period of trial and error in selecting these elements before coming up with an acceptable design. The selected elements for this study are presented in the following section.

4.3.1.1 The Selected Design Elements of FFN

Since the design elements of FFN consist of several elements with several possible values, the investigation will concentrate on the most significant elements considering the instructions provided by MATLAB and recommendation of previous search attempts. There are 7 elements specifically, they are: number of hidden layers, the neuron size of each hidden layer, the training algorithm, the type of transfer function, the number of training epochs, training goal and the performance function. Finally, since the used training algorithms have parameters that are randomly initialized and are adjusted adaptively during the training process, the training process will be repeated 30 times and the mean of the best 10 based on the mean error will be calculated. Again, the mean error will be calculated using Equation 4-1 and the error of both F_{Min} and $F_{M\text{-}Min}$ will be excluded.

The first FFN design element is the number of hidden layers and their neuron sizes. The number of hidden layers and their neurons sizes need to be determined in addition to input and output layers. The instruction help provided in MATLAB specifies that a network of one or two layers is enough to be trained to have reasonable capabilities to approximate any nonlinear function. An investigation for studying the effect of utilizing different FFN design architectures on the training error and validation error considering the processing time associated with each case is presented in Table 4.7. The value of the

performance function is chosen to be as MSEREG [15] while other parameters such as transfer function and training algorithms are fixed to their default values in this study. The result shows that adding more hidden layers reduces the training error and exponentially increases the processing time. The validation error is also enhanced with adding more hidden layers but when the hidden layers reach three layers especially with bigger sizes, the validation error gets bigger with adding more layers. This means that adding more than 2 hidden layers with big sizes badly affect the generalization capabilities of FFN. The best architectures in this investigation are (4 – 20 - 6) and (4 – 15 – 15 - 6) with processing times = 3.01 and 7.25 sec. respectively. Therefore, the architecture containing one hidden layer with size of 20 neurons will be selected.

Table 4.7 The Effect of Hidden Layers and their neuron sizes.

			No. of Hidden Layer				
			1	2	3	4	5
5 neurons	Error	Training	0.21574	0.166138	0.160975	0.150304	0.152711
		Validation	0.274682	0.190254	0.174631	0.147919	0.156392
	Time sec		0.01	0.012	0.21	0.5	0.94
10 neurons	Error	Training	0.160537	0.156403	0.135674	0.134285	0.129316
		Validation	0.192602	0.188083	0.168097	0.168787	0.159681
	Time sec		0.27	0.7	1	2.2	5.5
15 neurons	Error	Training	0.174875	0.129615	0.12616	0.125695	0.116648
		Validation	0.142965	0.134252	0.188059	0.191474	0.188374
	Time sec		2.9	7.25	20.54	75.78	225.25
20 neurons	Error	Training	0.156927	0.119063	0.117327	0.112547	
		Validation	0.135128	0.154403	0.198939	0.239125	
	Time sec		3.01	34.25	125.25	177	

Second, the transfer function of each layer used for the design of FFN must have derivative functions. They can be divided into two basic groups. The first category is nonlinear transfer functions. The most well known nonlinear transfer functions are hyperbolic tangent sigmoid transfer function named in MATLAB as TANSIG, and log sigmoid transfer function named as LOGSIG in the MATLAB. The profiles of these two

functions are illustrated in Figure 4.22, and Figure 4.23. The second category contains the linear transfer functions such as the linear transfer function named PURELIN in MATLAB illustrated in Figure 4.4. As much different combinations of previous functions as possible should be tried and studied aiming at choosing the best suitable transfer functions for the given modeling case. However, there was a consideration to involve at least one nonlinear transfer functions at the hidden layer in these combinations. 8 different transfer function combinations are studied using a signal hidden layer of 20 neurons and using MSEREG as performance function. The result is listed in Table 4.8 and illustrated in Figure 4.24.

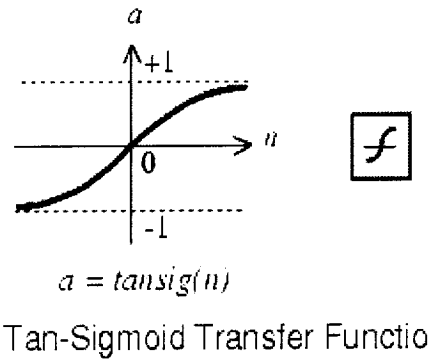

Figure 4.22 hyperbolic tangent sigmoid transfer function [15].

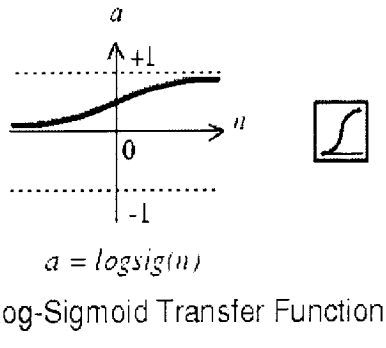

Figure 4.23 hyperbolic Log sigmoid transfer function [15].

Table 4.8 The Influence of Transfer Function on the Training and Validation Errors.

Type of Transfer Function				Mean Error	
				Training	Validation
1	TANSIG	TANSIG	TANSIG	0.11455887	0.14050232
2	LOGSIG	LOGSIG	LOGSIG	0.12130911	0.13938138
3	TANSIG	TANSIG	PURELIN	0.10481236	0.15895717
4	LOGSIG	LOGSIG	PURELIN	0.1554931	0.15379353
5	PURELIN	TANSIG	PURELIN	0.11384984	0.14560919
6	PURELIN	LOGSIG	PURELIN	0.12852037	0.13432894
7	PURELIN	TANSIG	TANSIG	0.13203903	0.15817547
8	PURELIN	LOGSIG	LOGSIG	0.12683928	0.15144036

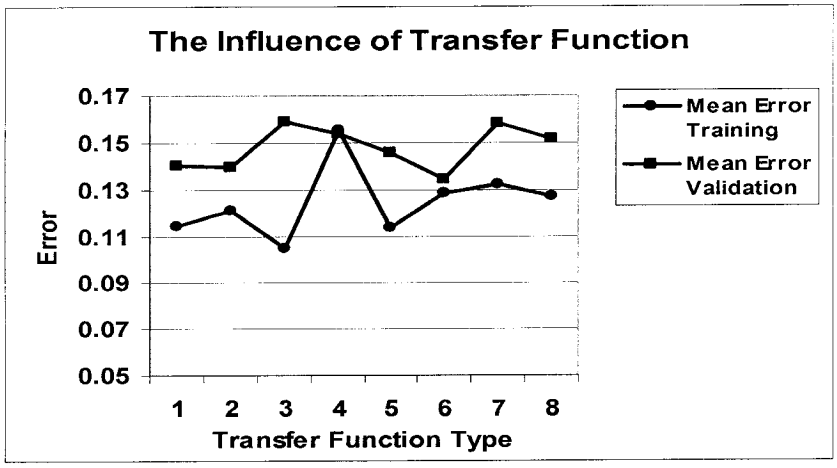

Figure 4.24 Transfer Function Type vs. the Training and Validation Error

The result shows that all combinations especially, the sixth ones result in good training and validation errors with values close to each other. This means involving at least single hidden layer with nonlinear transfer function is enough to model the process. Thus, the sixth combination will be selected.

Third, the role of training algorithms is to iteratively correct the weights and bias of the network for the duration of the training process to minimize the network performance. MATLAB has several different training algorithms using methods such as gradient, steepest descent, and numerical optimization technique. In this study two different training algorithms are used. The first one is called TRAINGDX which utilizes gradient descent with momentum and adaptive learning rate back-propagation. The second training algorithm is TRAINLM which is using LEVENBERG-MARQUARDT algorithm to apply the back-propagation. The result will be presented after discussing epoch number, training goal, and performance function, since they are dependable parameters.

The fourth FNN design element is the epoch number and training goal. The number of training epochs depends on the used training algorithm. At the case of utilizing TRAINGDX as training algorithm, the number of epochs was 2000 epochs since

TRAINGDX needs large number of epochs to provide suitable results. When using TRAINLM, since 50 epochs are enough to get adequate result, the epoch number was 50.

As discussed before in section 4.2.1.1, training goal has important role in improving the generalization capability of the network by stopping the training process early, before getting over-fit to the training patterns. However, MATLAB provided a performance function called MSEREG that can play the same role. Therefore, the effect of training goal will be discussed when using other performance function. This will be discussed in the following section.

Fifth, the performance functions that can be used for training FFN can be any of the differentiable performance functions provided in MATLAB such as MSE or MSEREG. The MATLAB instruction states that the default performance function is MSE; the mean sum of squares of the network errors. However, it also states that the generalization of FFN can be improved if the performance function depends on the mean of the sum of squares of the network weights and biases beside the mean sum of squares of the network errors. This feature is presented in MSEREG functions. By adding this term, the FFN will be forced to less response to the training patterns and to have smaller weights and consequently less over-fit. The equations describing the functions MSE and MSEREG are presented in Equation 4-3 and Equation 4-4, as mentioned in MATLAB instruction [15]. Equation 4-4 contains a performance ratio γ that can control the behavior of MSEREG function. The possible values of this ratio can be between 0 and 1. If γ is too large this will approximately cancel the additional item discussed before and consequently, the chance of getting over-fit will be increased. While if γ is too small this can make the FFN to not fit the training patterns.

$$F = mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2 \qquad \textbf{4-3}$$

$$msereg = \gamma mse + (1-\gamma)msw \qquad \textbf{4-4}$$

Where γ is the performance ratio and

$$msw = \frac{1}{n}\sum_{j=1}^{n} w_j^2$$

An investigation was performed to study the effect of using different values for the performance ratio using MSEREG as performance function and two training algorithms: TRAINLM, and TRAINGDX. A similar investigation with the use of MSE as performance function, was conducted to study the use of different training goal values, from 0.0001 ~ 0.005. The associated results of both investigations are presented in

Table 4.9 and Table 4.10. The results show that like the training goal, MSEREG function can be used for the need of generalization enhancement for the network, especially with the training algorithm TRAINLM. The training algorithm TRAINGDX has also good training capability and good generalization especially with MSE and training error goal = 0.0005 but it is much slower than TRAINLM. Away from the speed of training algorithms, the results demonstrate that the use of TRAINLM outperforms the use of TRAINGDX.

Table 4.9 The Effect of the Performance Function and Training Goal using TRAINLM Algorithm.

The Performance Function using TRAINLM Algorithm					
MSE			MSEREG		
	Error			Error	
Training Goal	Training	Validation	Performance Ratio	Training	Validation
0.0001	0.11456	0.14050	0.9	0.10246	0.14505
0.0005	0.12131	0.13938	0.8	0.14605	0.13490
0.001	0.10481	0.15896	0.7	0.13204	0.15836
0.005	0.15549	0.15379	0.6	0.12684	0.15130

Table 4.10 The Effect of the Performance Function and Training Goal using TRAINGDX Algorithm.

The Performance Function using TRAINGDX Algorithm					
MSE			MSEREG		
	Error			Error	
Training Goal	Training	Validation	Performance Ratio	Training	Validation
0.0001	0.16196	0.1469589	0.9	0.17871	0.1647444
0.0005	0.15721	0.1331796	0.8	0.33851	0.1685281
0.001	0.15364	0.1443579	0.7	0.24035	0.1763143
0.005	0.19165	0.1837558	0.6	0.20879	0.2143644

4.3.1.2 Training and Validation Phases of FFN

From the studies carried on the selection of FFN elements discussed in previous section, a network that has (4, 20, 6) structure, has (PURELIN, LOGSIG, PURELIN) transfer functions, utilizes TRAINLM algorithm with at most 50 epochs, and uses MSEREG as performance function with performance ratio=0.8 can be selected to model the end milling process. The same training sequence discussed in sections 4.2.1.2 and 4.2.2.2 will be used.

Therefore, at the first training step, every basic experimental model will be used individually as training patterns. The corresponding validation errors are listed in Table 4.11 and illustrated in Figure 4.25. The result shows that every model has its own ability of approximation. The result also shows that the best models are UL27-1, UL27-2, and UL32

respectively. The predictive error is too high; of range 26%. Therefore, the model UL32 will be used as a basic model for the 2-composed models to reduce the predictive errors.

Table 4.11 FFN Predictive Force Errors Resulting from Training Basic Models.

Model Name	Predictive Force Errors						Excluding F_{Min}, F_{M-Min}	
	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
UL8-1	0.341	1.855	0.521	1.440	0.680	3.989	0.746	0.931
UL9	0.276	4.061	0.416	0.437	0.265	2.207	0.348	0.191
UL27-1	0.283	0.903	0.267	0.342	0.265	2.069	0.289	0.094
UL27-2	0.274	0.768	0.261	0.366	0.189	0.677	0.272	0.155
UL32	0.172	4.284	0.301	0.262	0.236	1.755	0.243	0.115
UL25	0.275	0.958	0.287	0.473	1.246	1.207	0.570	0.594
UL8-2	0.364	2.823	0.381	0.688	0.832	0.835	0.566	0.380

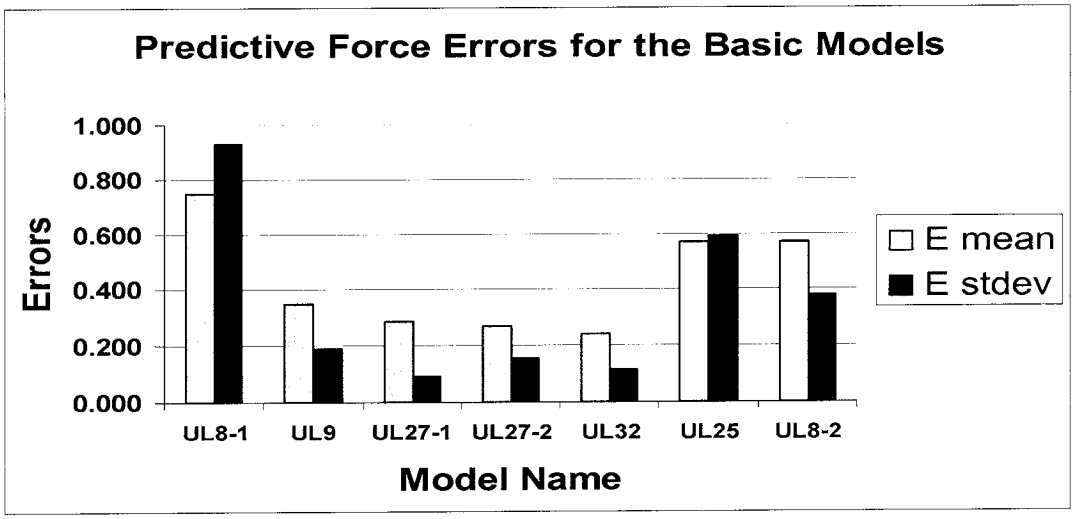

Figure 4.25 FFN Predictive Force Errors for Basic Models.

The 2-composed models using UL32 and the corresponding errors are listed in Table 4.12 and a comparison for mean errors and standard deviations is illustrated in Figure 4.26. The results show that the predictive errors are improved and consequently, adding more training experiments was useful.

Table 4.12 FFN Predictive Force Errors resulting from Training 2-Composed Models.

Basic models with UL32	Model Name	Predictive Force Errors						Excluding F_{Min}, F_{M-Min}	
		F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
UL8-1	UL40-1	0.340	3.708	0.240	0.222	0.237	0.691	0.260	0.114
UL9	UL41	0.249	1.665	0.113	0.252	0.214	0.763	0.207	0.108
UL27-1	UL59-1	0.196	1.042	0.197	0.255	0.172	0.531	0.205	0.081
UL27-2	UL59-2	0.246	1.323	0.189	0.179	0.140	0.587	0.188	0.100
UL25	UL57	0.189	0.599	0.169	0.160	0.161	0.731	0.170	0.076
UL8-2	UL40-2	0.299	0.821	0.278	0.127	0.170	3.755	0.219	0.101

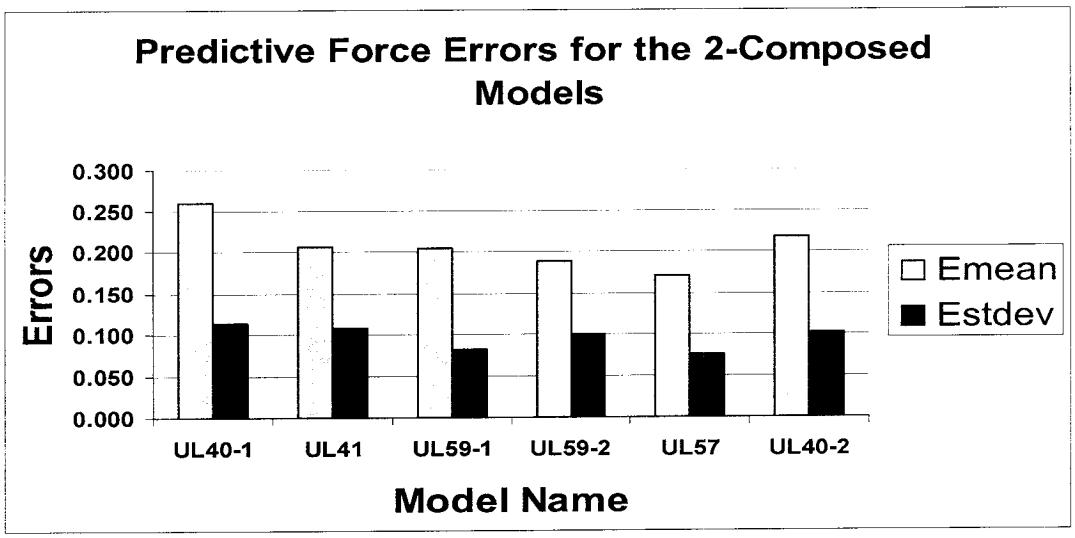

Figure 4.26 Predictive Force Errors for the 2-Composed Models.

Until now, the best predictive error that can be obtained is 17% when using UL57. Thus, to obtain more enhancements, the 3-composed models were used using UL57 as a basic model. This is listed in Table 4.13 and a comparison for mean errors and standard deviations is illustrated in Figure 4.27. The result shows that the predictive errors for all 3-composed models was fairly, improved. It also shows that the best models are UL65-2, and UL84-1 that have roughly the same predictive errors. This confirms that the degree of enhancement doesn't rely on the number of training experiments.

Table 4.13 FFN Predictive Force Errors resulting from Training 3-Composed Models.

Basic models with UL57	Model Name	Predictive Force Errors						Excluding F_{Min}, F_{M-Min}	
		F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
UL8-1	UL65-1	0.169	0.726	0.262	0.180	0.157	0.384	0.192	0.112
UL9	UL66	0.169	0.642	0.160	0.186	0.172	0.821	0.172	0.088
UL27-1	UL84-1	0.173	0.600	0.156	0.166	0.148	0.436	0.161	0.059
UL27-2	UL84-2	0.262	0.800	0.147	0.143	0.111	0.334	0.166	0.128
UL8-2	UL65-2	0.128	0.790	0.175	0.180	0.163	0.307	0.161	0.061

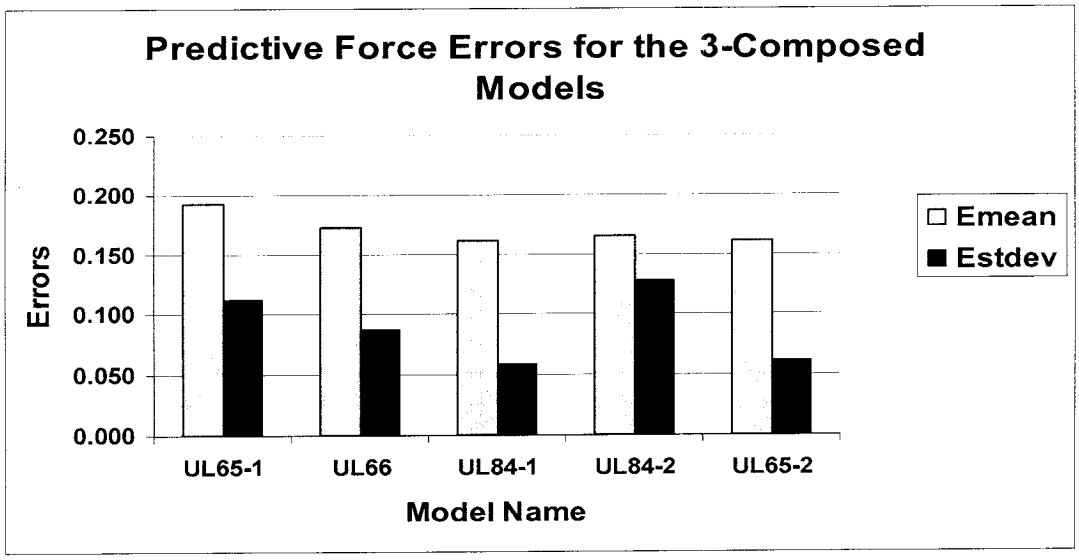

Figure 4.27 Predictive Force Errors for the 3-Composed Models.

UL65-2 will be used as basic model for constructing the possible 4-composed models, as it contains fewer experiments. Table 4.14 and Figure 4.28 show the predictive errors for each model. UL92-2 gave the best predictive error that equals 13. 3% and it will be used as a basic model for 5-composed model. Besides, adding effective model rather than constructing the 5-composed models will be attempted.

Table 4.14 FFN Predictive Force Errors resulting from Training 4-Composed Models.

Basic models with UL65-2	Model Name	Predictive Force Errors						Excluding F_{Min}, $F_{M\text{-}Min}$	
		F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	$F_{M\text{-}Max}$	$F_{M\text{-}Min}$	E_{Mean}	E_{Stdev}
UL8-1	UL73	0.207	1.296	0.179	0.232	0.213	0.420	0.208	0.090
UL9	UL74	0.152	7.200	0.150	0.190	0.169	3.161	0.165	0.080
UL27-1	UL92-1	0.151	1.038	0.156	0.161	0.131	0.539	0.150	0.083
UL27-2	UL92-2	0.132	0.723	0.135	0.152	0.115	0.341	0.133	0.058

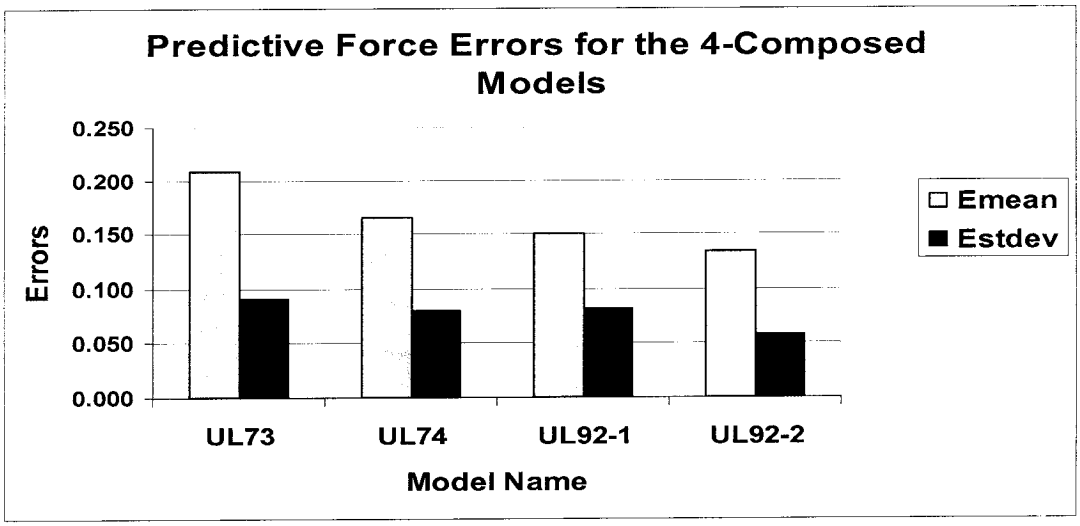

Figure 4.28 Predictive Force Errors for the 4-Composed Models.

The mean and the standard deviation of the predictive errors for the 5-composed models are summarized in both Table 4.15 and Figure 4.29. The result shows that training FFN model with the 5-models enlarged the predictive errors and the model may be over trained. The additional experiments are detrimental to the past knowledge of the neural model. One possible reason is that the neural model learns the actual performance and the associated noise and updated itself with every training pattern. When the training patterns are too large, the knowledge part of noise may work against the part of true performance.

Table 4.15 FFN Predictive Force Errors result from Training the 5-Composed Models.

Basic models with UL92-2	Model Name	Predictive Force Errors						Excluding F_{Min}, F_{M-Min}	
		F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
UL8-1	UL100	0.139	1.671	0.192	0.160	0.149	1.123	0.160	0.082
UL9	UL101	0.129	0.538	0.127	0.175	0.125	0.667	0.139	0.066
UL27-1	UL119	0.137	0.983	0.232	0.137	0.097	1.436	0.151	0.085

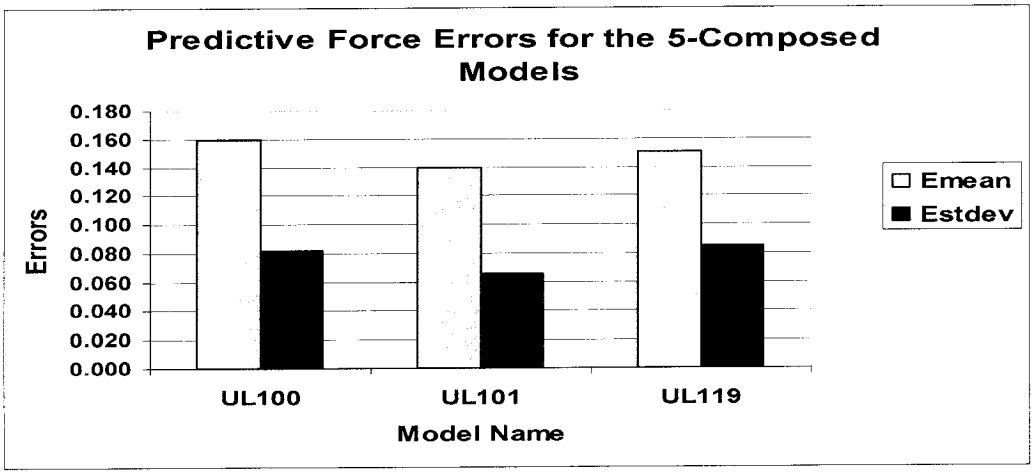

Figure 4.29 Predictive Force Errors for the 5-Composed Models.

Therefore, instead of adding the rest models, the search for effective points that can enhance the approximation will be given. The corresponding results are presented in Table 4.16 and Figure 4.30. The result shows some enhancement. Adding other experiments didn't enhance the predictive error. The error fluctuates between 13% and 17%. This is illustrated in Figure 4.31.

Table 4.16 FFN Predictive Force Errors result from Training the 5-Composed Models Using Effective points.

Basic models with UL92-2	Exp. Added	Model Name	Predictive Force Errors						Excluding F_{Min}, F_{M-Min}	
			F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
UL8-1	3,7	UL94-1	0.11	4.68	0.16	0.15	0.11	2.2	0.133	0.1
UL9	4,9	UL94-2	0.13	0.66	0.14	0.14	0.12	0.38	0.133	0.07
UL27-1	2,14,15, 25	UL96	0.11	0.52	0.12	0.16	0.13	1.31	0.130	0.06
Adding all Effective points		UL100	0.13	0.74	0.119	0.14	0.12	0.48	0.127	0.06

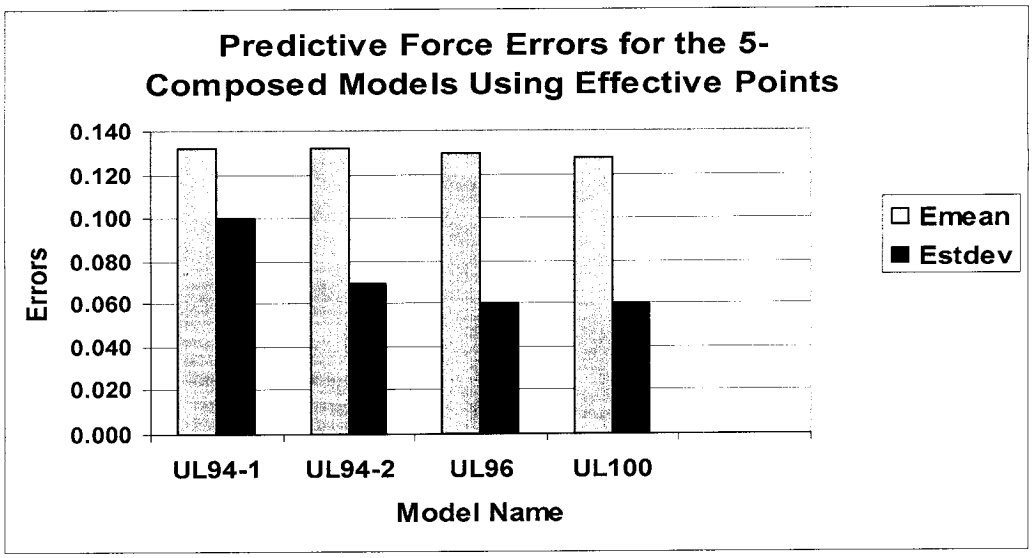

Figure 4.30 Predictive Force Errors for 5-Composed Models Using Effective model.

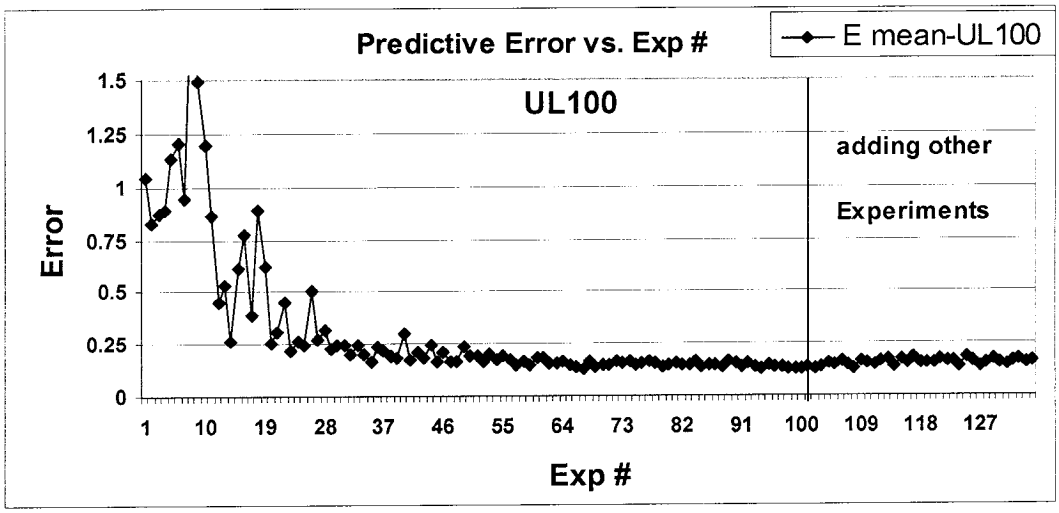

Figure 4.31 Predictive Error vs. Exp # Using UL100.

All experimental models were combined together in one big composed model called UL136. This model is used to train the FNN using random sequence. This is illustrated in Figure 4.32 . The predictive error using UL136 was 14.8% while, the best predictive error occurred using 124 experiments, called UL124 and it equals 13.1%. The results of UL136 and UL124 are listed in Table 4.17. Compared with the results of training

FNN with random sequence, training FFN using the previous sequence results in a better predictive error with faster convergence. A comparison between UL100 and UL124 starting from experiment 50 is given in Figure 4.33. Figure 4.34 illustrates the real F max values vs. the predicted F max values obtained by FFN.

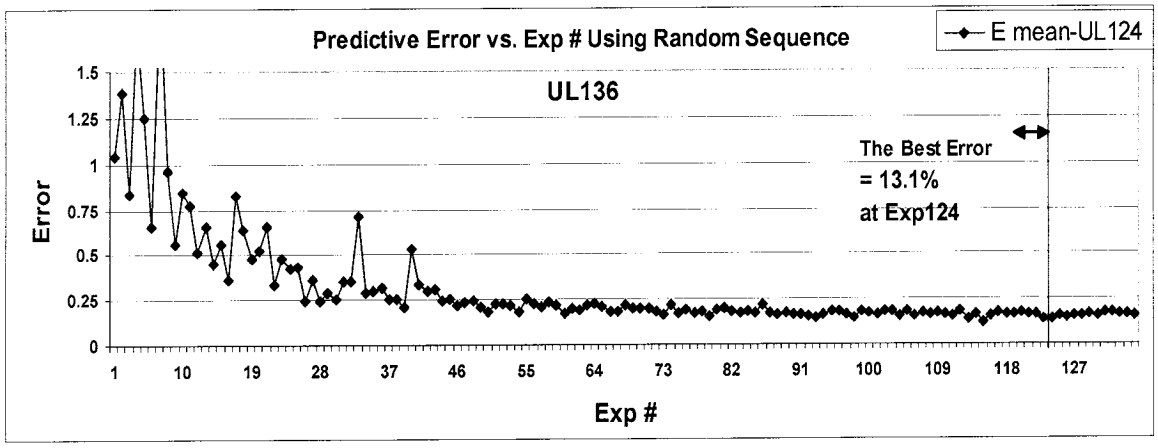

Figure 4.32 Predictive Error vs. Exp # Using Random Training Sequence.

Table 4.17 Predictive Error Using Random Training Sequence.

Model Name	Predictive Force Errors						Excluding F_{Min}, F_{M-Min}	
	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
UL136	0.152625	4.273568	0.133037	0.168685	0.138954	0.29865	0.148325	0.072702
UL124	0.135214	5.099718	0.132139	0.152652	0.105645	0.70894	0.131413	0.058384

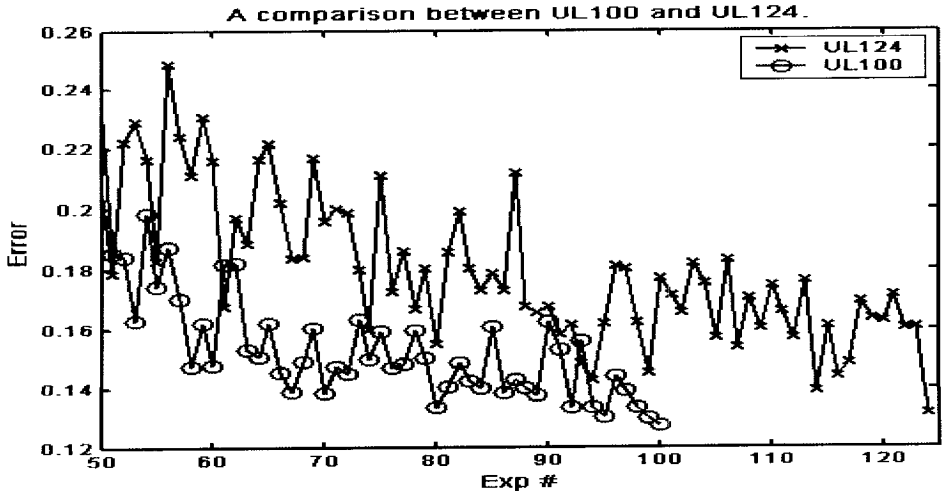

Figure 4.33 A comparison between UL100 and UL124.

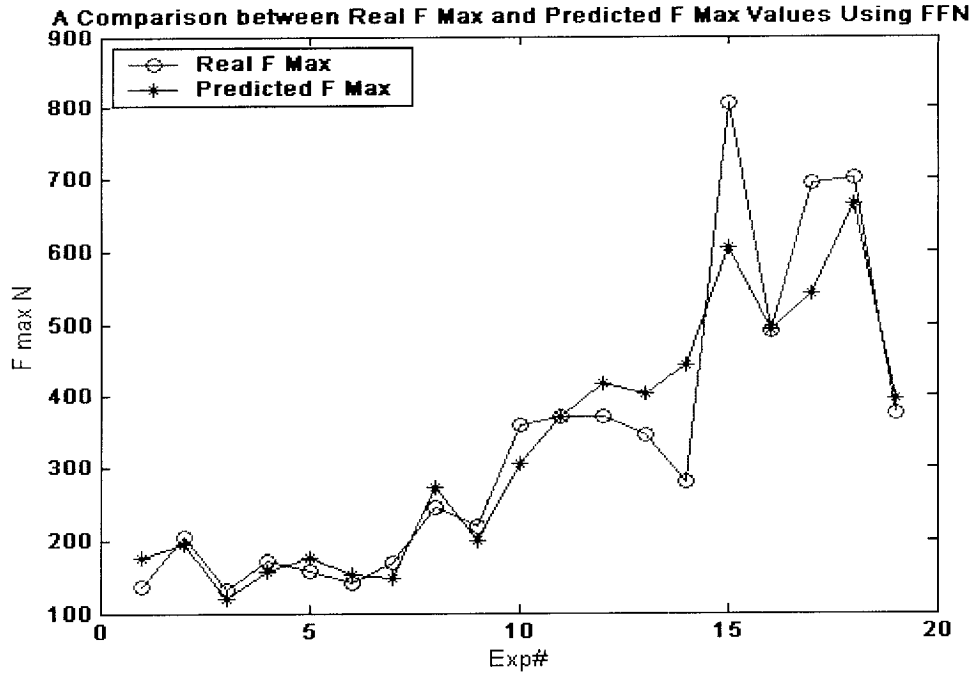

Figure 4.34 A Comparison between Real F max and Predicted F max Values Using FFN.

4.4 Modeling Process Using Insufficient Data

In this part, a study is presented to investigate neural network modeling using insufficient data. RBN was used in this study. The insufficient data includes only 33 experiments conducted on CNC machine. The 33 experiments consist of two parts. First, 20 experiments were chosen for training purpose. These experiments include N8-1 model, NL8-2 model, and additional four experiments selected for the aim of providing new experiments to cover space locations that are not covered by the previous models. The second part includes 13 experiments for validating and testing the predictive capability of the designed RBN. Figure 4.35 gives a space representation for these experiments and it shows the position of the additional 4 experiments selected to cover new space locations.

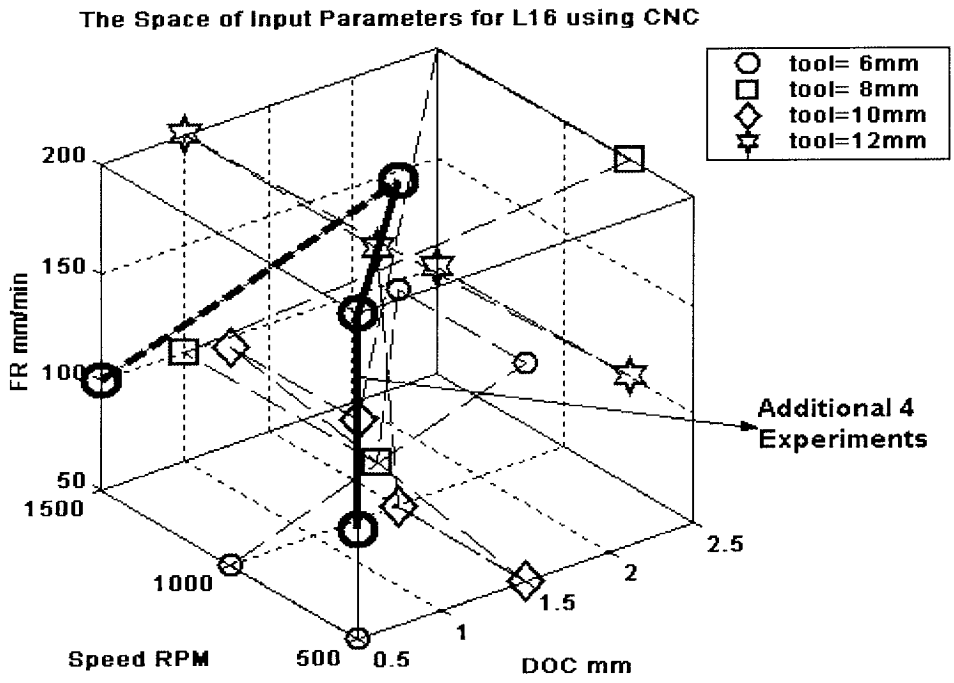

Figure 4.35 An Illustration of Space Points for the Insufficient Data Models

4. 4.1 RBN parameters selections using insufficient data

The same procedure specified in selecting RBN parameters in the section 4.2.1.1 will be implemented in this part. Using sufficient training patterns and validation patterns, the initial network parameters will be chosen. However, before searching for the best parameters and since the available training patterns are very small, some considerations should be taken into account. First, the spread value should be high enough to make the approximation smoother. Second, training error should be as small as training time gets longer to give the opportunity to use as many neurons as possible. Finally, the allowable number of neurons should be large enough. Since the algorithm of RBN does not permit using neurons that are more than the number of input patterns, the maximum number of allowable neurons will not exceed 20 neurons. These considerations are ways to compensate the limitations of training experiments.

The searching results of RBN parameters are illustrated in Figure 4.36, Figure 4.37, and Figure 4.38 respectively. Figure 4.36 shows that adding larger spread value increasingly enhances the predictive error until the best error at spread value equals 1.1. Increasing the spread value more than 1.1 raises the predictive error. Figure 4.37 shows that the best predictive error with reasonable training error is reached when the training error goal is in the range 0.095 - 0.175. Below this range, the training error is better while the predictive error is worst and above this range, both training error and predictive error are worst. Below this range, the error goal is so small that the RBN gets much time to fit the training patterns in such a way that the generalization of RBN becomes worst. Above this range, the error makes the RBN stop the process of training early. This leads to worst training error since the RBN will not adequately fit the training data and consequently worst validation error will be obtained.

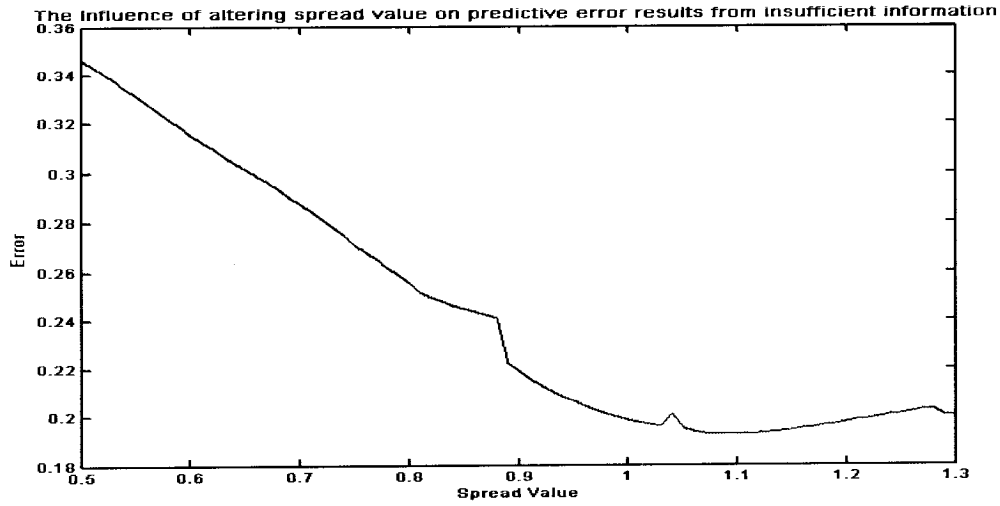

Figure 4.36 Several Spread Values for RBN vs. Predictive Errors using insufficient information.

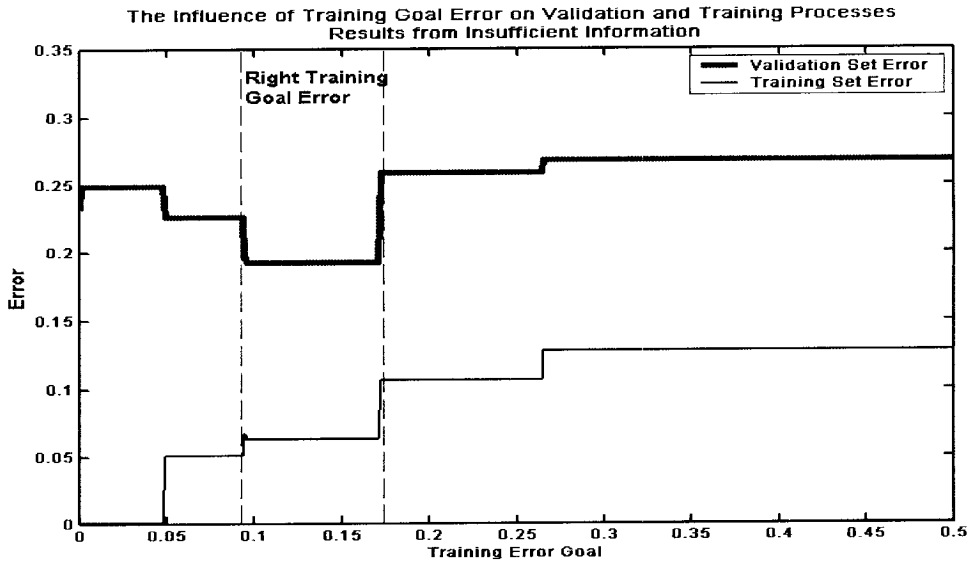

Figure 4.37 Training Error Goal for RBN vs. Predictive Error using insufficient information

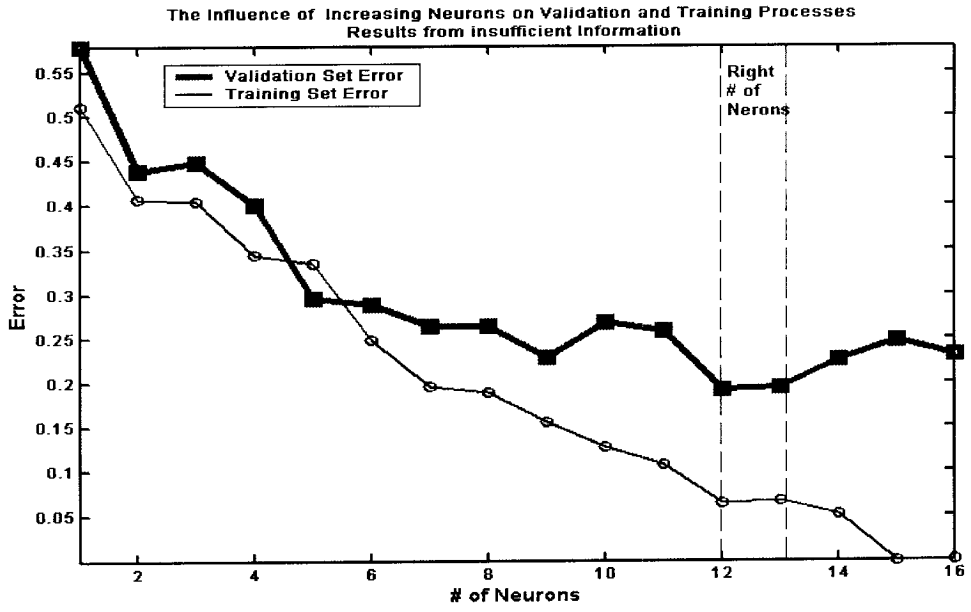

Figure 4.38 Number of Neurons for RBN vs. Predictive Error Using Insufficient Information

Figure 4.38 shows that adding more neurons leads to better training error and better predictive potential. However, adding neurons more than 13 will reduce the predictive capability, and make the RBN over-fit the training patterns.

These results confirm the previous considerations since the suitable spread value is 1.1, training error goal is between 0.1 and 0.175, and the preferable number of neurons should be 13.

4.4.2 RBN Training and Validation phases using insufficient data

Using these parameters, every model was used by itself as training patterns, then the two models were combined together and used for training processes. The results are shown in Table 4.18. Equation 4-1 is also used to compute the predictive error and again, the F_{Min} and $F_{M\text{-}Min}$ are excluded from calculating E_{mean} and E_{Stdev}. A comparison of the predictive error vs. training experiments for these models is illustrated in Figure 4.39 . The result shows that the insufficient model returns 19.2 % as the best predictive error. This means that the combination of using NL16 and using RBN with such design has good capability to model the nonlinearity of this process using insufficient information. This is confirmed in Figure 4.39.

Table 4.18 predictive errors for NL8-1, NL8-2, and NL16.

Model Name	Predictive Force Errors						Excluding F_{Min}, $F_{M\text{-}Min}$	
	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	$F_{M\text{-}Max}$	$F_{M\text{-}Min}$	E_{Mean}	E_{Stdev}
NL8-1	0.509	0.711	0.506	0.575	0.504	0.686	0.524	0.060
NL8-2	0.425	0.400	0.305	0.479	0.426	0.718	0.408	0.117
NL16	0.191	0.520	0.157	0.230	0.192	0.643	0.192	0.072

Trying to reduce the amount of predictive error, additional training patterns that cover new space locations should be fed to the previous RBN to provide new useful information for the model. Therefore, four additional experiments were selected in different locations. The results of adding one, two, three and four experiments are listed in Table 4.19. The results show that the predictive error is reduced to 15.7 %. This means

that selecting additional experiments covering new input space locations can improve the prediction capability further.

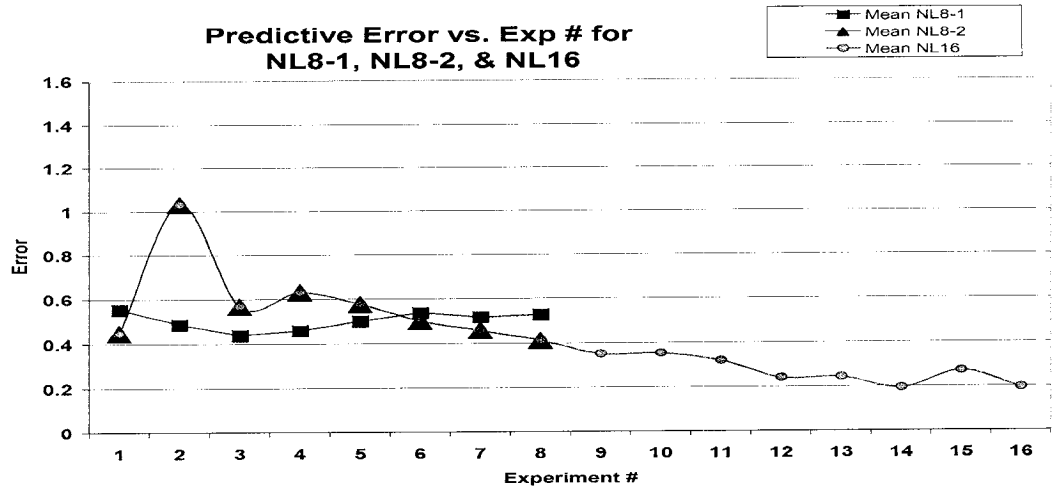

Figure 4.39 Predictive errors vs. experiments # for NL8-1, NL8-2, and NL16

Table 4.19 Sequence of adding additional four experiments to NL16.

Additional Experiments	Predictive Force Errors						Excluding F_{Min}, $F_{M\text{-}Min}$	
	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	$F_{M\text{-}Max}$	$F_{M\text{-}Min}$	E_{Mean}	E_{Stdev}
1	0.147	0.511	0.146	0.205	0.149	0.632	0.162	0.054
2	0.146	0.444	0.140	0.198	0.152	0.752	0.159	0.051
3	0.132	0.512	0.147	0.210	0.153	0.653	0.160	0.059
4	0.129	0.495	0.142	0.202	0.147	0.632	0.158	0.056
4&2	0.146	0.433	0.137	0.195	0.151	0.752	0.157	0.051
4&3	0.180	0.447	0.154	0.249	0.199	0.670	0.196	0.071
4&1	0.141	0.499	0.142	0.200	0.143	0.628	0.157	0.054
4,1&2	0.227	0.541	0.235	0.338	0.262	0.810	0.265	0.094
4,1&3	0.160	0.447	0.154	0.246	0.181	0.674	0.185	0.069
4,1,3&2	0.141	0.444	0.135	0.206	0.145	0.799	0.157	0.057

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

The main contributions of this work can be determined in four points. First, predictive force modeling of the face milling process has been successfully developed using both RBN and FFN neural network models. The result of validation phase using new experimental situations within the range of training patterns and others outside that range is 8.89% for RBN and 12.7% for FFN. Figure 5.1 gives a comparison for real F max values and the corresponding predicted F max values using FFN and RBN. The results demonstrated that RBN outperforms FFN. Specifically, the capability of prediction of RBN is better, the modeling process of RBN is faster and needs less training patterns, and the design steps are much easier. FFN has more design elements and the space of its parameters values is wider. The FFN uses parameters that are randomly initialized and are adjusted adaptively during the training process. Therefore, the result of any study for FFN should depend on the average of the number of runs possibly 10-30 runs, to avoid any noise that may arise from randomization. RBN has a tendency to over-fit the training patterns which may deteriorate the neural model. Therefore, RBN should be carefully designed to avoid the over-fitting problem.

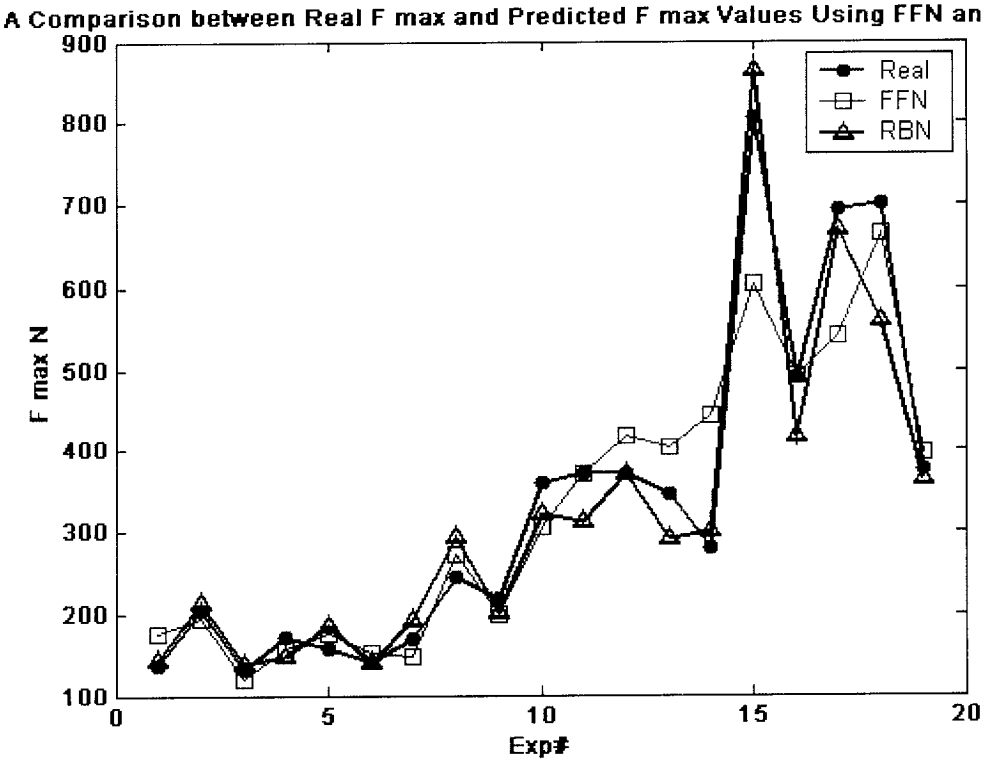

Figure 5.1 A Comparison between Real F_{Max} and Predicted F_{Max} Values Using FFN and RBN.

Second, inclusive study for the design steps and parameters for both RBN and FFN are presented in this work. A systematic way for designing robust neural model for both types is presented. Design problems such as over-fitting are tackled in this study. Especially for RBN, investigation for better neural design can be crucial factor in reducing the predictive error in lesser training patterns. The study has confirmed that involving at least single hidden layer with nonlinear transfer function is enough to model the nonlinear process. It also shows that using TRAINLM training algorithm along MSEREG performance function enhances the generalization strength of FFN [15].

Third, The DOE and OAs are used for the need of selecting efficient models in lesser number of experiments. Several OAs are used specifically, they are L8-1, L8-2, L9, L27-1, L27-2, L32, and L25. Comprehensive study for these experimental models is presented and compared. This includes studying the capability of training each experimental model if used separately. In addition, the study of using combinations of these models in a certain sequence is given. The effect of adding effective experiments that cover new space locations is given. The study shows that using OAs is efficient and most experimental models have good approximation capabilities especially L27OA. Besides, L16 model composed of 2 different L8 OAs performs well relative to other smaller size models. Study using composed models indicates that a certain sequence leads to better model with faster convergence and with better predictive error. This includes that the sequence the neural model will follow is important and can lead to lesser training patterns with better predictive error.

Fourth, the study of modeling milling process using insufficient information is presented. Composed model L16 with 4 effective experiments has been used in the training patterns. Results show that using certain neural design besides using certain training sequence have successfully modeled the milling process with reasonable predictive error. Therefore, insufficient information as necessitated by limitations of time and/or resources should be used cautiously with neural network modeling.

5.2 Future work

Currently, four input control parameters are used and the resulting cutting forces are measured for input to ANN. Other input parameters such as the number of flutes, tool geometry modification, and work-piece material should be included in any future study. Furthermore, surface roughness can be of interest besides optimization of cutting forces and manufacturing cost. These are valid important extensions for any future work. More inclusive study for the sequence of training patterns should be considered in any future work.

REFERENCES

1 Abrari F. and Elbestawi M. A., "Closed Form Formulation of Cutting Forces for Ball and Flat End Mills", Int., J. Mack Manufact. Vol. 37, No. 1, pp. 17-27, 1997.

2 Abrari F., Elbestawi M. A., Spence A. D., "On the Dynamics of Ball End Milling Modeling of Cutting Forces and Stability analysis", Elsevier Science Ltd, Vol. 38, No. 3, 215-237, 1998.

3 Acosta C., Switek W., Garcia E., "Dynamic Modeling In Turning Machining", Elsevier Science Ltd, Vol. 33, No. 1-2, p397-400, 1997.

4 Briceno J.F.; El-Mounayri H.; Mukhopadhyay S., "Selecting an artificial neural network for efficient modeling and accurate simulation of the milling process", International Journal of Machine Tools and Manufacture, May 2002, vol. 42, iss. 6, pp. 663-674(12)

5 Chien Wen-Tung, Tsai Chung-Shay "The Investigation on the Prediction of Tool Wear and Determination of Optimum Cutting Conditions in Machining 17-4PH Stainless Steel", Journal of Materials Processing Technology, 340-345, 140 (2003).

6 Cincinnati, OH. Machining Data Handbook, Metcut Research Associates Inc., 2nd Edition, 1972.

7 El-Mounayri, H., Gadallah, M. H., and Briceno, J., "Enhanced modeling and optimization of milling using a hybrid technique," Transactions of the Canadian Society for Mechanical Engineering, vol. 26, no. 2, pp. 179-199, 2002.

8 Haykin, Simon, Neural Networks-A Comprehensive Foundation, Prentice Hall, New Jersey, 1999.

9 Ho Shinn-Ying, Lee Kuang-Chyi, Chen Shih-Shin, Ho Shinn-Jang, "Accurate Modeling And Prediction Of Surface Roughness By Computer Vision In Turning Operations Using An Adaptive Neurofuzzy Inference System", International Journal of Machine Tools & Manufacture, 42, 441-1446, 2002.

10 Humusoft, User's Manual-MF614 Multifunction I\O Card.

11 Humusoft, User's Manual-Real Time Toolbox for uses with SIMULINK, Version 3.1.

12 IZAR Technical Department, tecnica@izar-tool.com.

13 Li X. P., Nee A.Y.C., Wong Y. S., Zheng H. Q., "Theoretical Modeling and Simulation of Milling Forces", Journal of Materials Processing Technology, 266-272, No 89-90, 1999.

14 Martinez Wendy L., Martinez Angel R., Computational Statistics-Handbook with MATLAB, Chapman & Hall/CRC, 2002.

15 MATLAB, Neural Network Toolbox User's Guide Version 4.0.1.

16 MATLAB Release 12, 2002.

17 Milfelner M., Cus, F., Balic J., "Analytical Simulation of Cutting Forces in End Milling", 16th International Conference on production Research (ICPR16),Quebec City, Canada, 2001.

18 O¨zel Tugrul, Lucchi, Marco, Rodríguez Ciro A., "Production of Chip Formation and Cutting Forces in Flat End Milling: Comparison of Process Simulation With Experiments", Engineering Research Center for Net Shape Manufacturing (ERC/NSM) The Ohio State University, Columbus, Ohio, 2002.

19 O¨zel Tugrul, Karpat Yiğit, "Prediction of Surface Roughness and Tool Wear in Finish Dry Hard Turning Using Back Propagation Neural Networks", Department of Industrial and Systems Engineering Rutgers, The State University of New Jersey, Piscataway, New Jersey, U.S.A.,2003.

20 P.L.B. Oxley, Mechanics of Machining-An Analytical Approach to Assessing Machinability, Ellis Horwood Limited, Chichester, UK, 1989.

21 Ross, P.J., Taguchi Techniques for Quality Engineering, Mc Graw Hill Co., New York, 1988.

22 Tlusty George, Manufacturing process and Equipment, Prentice Hall, New Jersey, 2000.

23 Trent Edward M., Wright Paul K, Metal Cutting 4th ed., Butterworth-Heinemann, 2000.

24 Zurada Jacek M., Introduction to Artificial Systems, West Publishing Company, USA, 1996.

25 http://www.mfg.mtu.edu/cyberman/machining/trad/milling.

Appendix A

EQUIPMENT INFORMATION

A.1 Tool information [12]

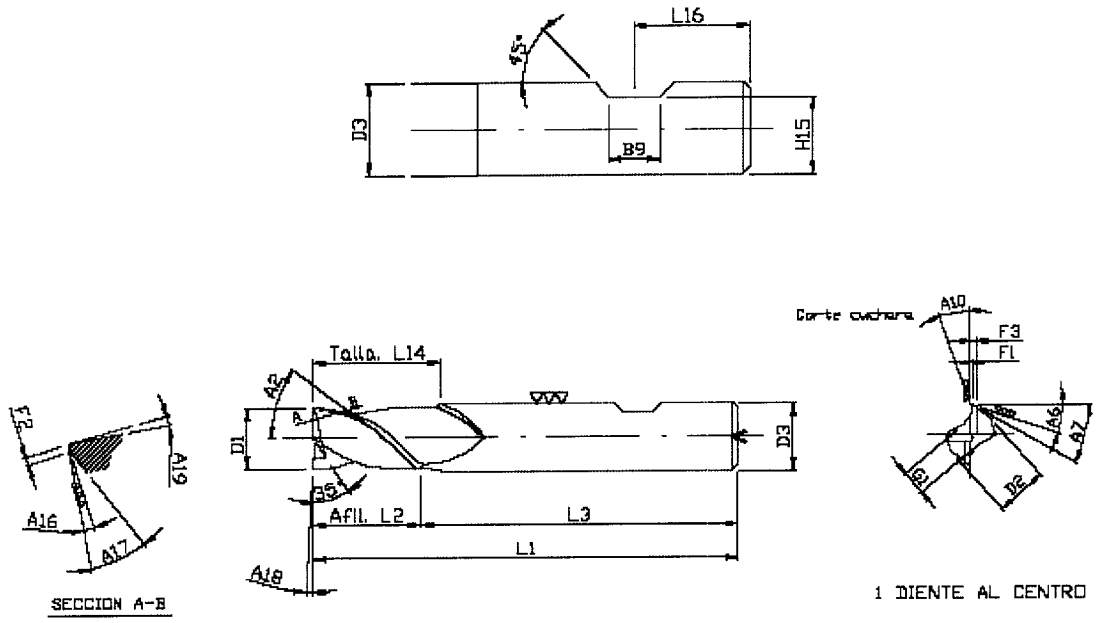

Figure A 1 An Illustration for the Tool Geometry (Two Flute-End Millis REF: 4420).

Where:

A2	⇒ Helix Angle	F1	⇒ Radial Relief Land Width
A6	⇒ Radial Relief Angle	F2	⇒ Axial Relief Land Width
A7	⇒ Radial Clearance	F3	⇒ Radial Clearance Land Width
A10	⇒ Radial Rake Angle	G1	⇒ Core Diameter
A16	⇒ Axial Relief Angle	H15	⇒ Weldon Dimension
A17	⇒ Axial Clearance Angle	L1	⇒ Overall Length
A18	⇒ Concavity Angle	L2	⇒ Cutting Length
A19	⇒ Axial Rake Angle	L3	⇒ Shank Length
B9	⇒ Weldom Width	L14	⇒ Grinding Fluted Length
D1	⇒ Cutting Diameter	L16	⇒ Weldom Dimension (position)

D2	⇒ Clearance Diameter	P1t	⇒ Lead
D3	⇒ Shank Diameter	P1t	⇒ Lead

Appendix B

EXPERIMENTAL RESULTS

B 1 Experimental Model UL8-1.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	1.5	800	71	8	114.30	0.18	44.47	31.60	92.41	4.30
2	1.5	800	140	12	367.49	0.56	100.78	98.15	282.43	3.26
3	1.5	1600	71	8	106.04	0.32	32.37	20.77	69.18	3.17
4	1.5	1600	140	12	226.51	0.31	64.91	55.09	178.70	2.10
5	3	800	71	12	221.80	0.11	70.56	59.45	160.52	1.79
6	3	800	140	8	351.92	0.48	138.58	108.10	292.81	3.18
7	3	1600	71	12	359.34	0.37	67.30	65.09	264.94	1.95
8	3	1600	140	8	278.03	0.41	84.55	65.51	186.75	3.43

B 2 Experimental Model UL8-2.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	1	560	50	7	145.07	0.65	48.77	36.93	107.23	4.61
2	1	560	100	11	206.88	0.57	66.28	48.05	146.05	5.09
3	1	1120	50	7	124.21	0.14	34.40	22.14	89.45	2.83
4	1	1120	100	11	161.02	0.43	38.08	28.08	106.42	3.04
5	2.5	560	50	11	328.40	0.41	120.12	71.94	223.64	10.50
6	2.5	560	100	7	365.70	0.56	164.76	87.20	280.99	24.43
7	2.5	1120	50	11	278.64	0.59	88.83	49.68	177.85	5.76
8	2.5	1120	100	7	272.66	0.18	121.17	73.12	208.29	8.62

B 3 Experimental Model UL9.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	0.8	560	50	7	117.55	0.64	30.63	27.37	80.93	3.24
2	0.8	1120	100	8	144.58	0.79	52.83	21.61	96.44	11.40
3	0.8	1600	140	11	112.11	0.51	35.11	23.49	92.39	3.05
4	1.8	560	100	11	280.04	0.45	109.97	66.42	224.52	5.42
5	1.8	1120	140	7	278.59	0.36	92.11	81.24	236.16	2.05
6	1.8	1600	50	8	177.13	0.55	63.67	24.99	116.57	13.28
7	2.8	560	140	8	494.76	0.57	251.94	107.66	378.66	31.57
8	2.8	1120	50	11	258.84	0.46	71.68	52.93	173.82	4.92
9	2.8	1600	100	7	294.84	0.57	91.12	84.06	243.17	1.95

B 4 Experimental Model UL25.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	0.75	400	35.5	7	103.40	0.47	30.30	26.54	74.07	3.02
2	0.75	560	50	8	122.15	0.71	44.66	25.27	90.42	8.39
3	0.75	800	71	10	90.74	0.57	24.94	14.52	58.33	2.98
4	0.75	1120	100	11	116.40	0.43	26.35	23.46	77.40	2.54
5	0.75	1600	140	12	140.91	0.35	37.22	32.74	113.16	3.11
6	1.25	400	50	10	158.76	0.26	70.74	33.27	119.43	8.04
7	1.25	560	71	11	205.11	0.25	55.28	55.50	158.27	2.68
8	1.25	800	100	12	240.66	0.20	61.51	62.70	182.69	2.85
9	1.25	1120	140	7	210.79	0.25	55.99	47.94	139.19	2.10
10	1.25	1600	35.5	8	128.85	0.74	42.35	23.79	93.26	7.83
11	1.75	400	71	12	372.06	0.26	104.71	103.68	270.40	4.23
12	1.75	560	100	7	375.52	0.75	134.95	103.69	303.72	10.60
13	1.75	800	140	8	300.87	1.80	135.55	70.92	250.56	15.78
14	1.75	1120	35.5	10	141.29	0.45	53.66	24.65	102.01	5.96
15	1.75	1600	50	11	167.84	0.33	37.98	35.95	112.83	2.41
16	2.25	400	100	8	437.20	1.46	205.51	101.39	339.09	27.51
17	2.25	560	140	10	368.03	1.07	182.64	78.74	283.22	17.90
18	2.25	800	35.5	11	194.20	0.26	57.21	53.02	152.92	3.09
19	2.25	1120	50	12	263.04	0.57	67.68	62.62	185.16	3.27
20	2.25	1600	71	7	225.08	0.39	72.85	49.13	163.96	4.10
21	2.75	400	140	11	666.67	2.15	284.02	169.84	518.62	17.40
22	2.75	560	35.5	12	322.50	0.30	88.88	86.85	233.87	4.00
23	2.75	800	50	7	234.85	0.57	102.19	54.39	188.35	7.26
24	2.75	1120	71	8	279.17	0.57	114.93	65.86	222.80	14.31
25	2.75	1600	100	10	244.67	1.47	107.35	50.37	195.46	7.07

B 5 Experimental Model UL27-1.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	0.5	560	50	8	85.16	2.61	27.52	16.58	62.41	8.44
2	2	560	140	8	512.09	1.78	168.41	169.84	450.91	5.71
3	3.5	560	280	8	1283.47	6.92	643.58	302.60	1057.47	88.18
4	2	1120	50	8	340.07	0.27	79.93	72.82	206.25	3.05
5	3.5	1120	140	8	656.29	3.02	303.71	166.13	543.29	29.67
6	0.5	1120	280	8	262.72	0.73	58.26	49.68	190.09	3.10
7	3.5	1600	50	8	400.03	9.15	121.82	101.06	290.78	17.95
8	0.5	1600	140	8	156.95	0.32	30.88	24.52	95.82	2.46
9	2	1600	280	8	698.58	1.88	244.85	143.97	468.79	12.97
10	0.5	560	50	10	89.59	0.51	28.00	20.55	69.71	3.61
11	2	560	140	10	421.81	1.79	172.47	88.41	326.70	12.34

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
12	3.5	560	280	10	969.20	5.47	536.65	226.05	831.99	85.96
13	2	1120	50	10	230.74	0.28	57.53	52.93	160.09	3.45
14	3.5	1120	140	10	426.49	3.85	208.69	90.56	343.99	17.73
15	0.5	1120	280	10	111.18	0.35	47.41	20.27	90.08	5.93
16	3.5	1600	50	10	265.56	3.28	104.80	48.70	196.12	15.13
17	0.5	1600	140	10	81.22	0.74	29.40	13.62	62.69	5.54
18	2	1600	280	10	295.77	0.69	150.14	67.33	257.09	9.11
19	0.5	560	50	12	104.41	2.27	29.36	14.56	61.12	9.45
20	2	560	140	12	347.43	0.38	156.98	78.46	262.96	17.92
21	3.5	560	280	12	1011.85	3.94	529.23	245.77	904.77	43.68
22	2	1120	50	12	209.65	0.35	62.95	48.84	154.37	2.48
23	3.5	1120	140	12	469.86	0.69	192.99	114.31	374.12	5.77
24	0.5	1120	280	12	146.06	0.95	46.91	35.26	124.84	4.42
25	3.5	1600	50	12	281.41	0.49	88.52	74.57	231.53	3.06
26	0.5	1600	140	12	123.70	1.17	44.22	21.27	97.61	9.40
27	2	1600	280	12	501.31	0.77	138.76	86.60	343.41	3.19

B 6 Experimental Model UL27-2.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	1	560	50	8	163.40	1.00	53.62	36.60	117.75	6.87
2	1	560	100	10	211.02	0.66	65.04	54.47	160.15	2.98
3	1	560	140	12	296.67	0.96	79.60	66.24	227.65	6.07
4	1	1120	50	10	114.11	0.16	27.88	18.31	70.80	3.84
5	1	1120	100	12	177.64	0.45	42.90	40.36	128.25	2.97
6	1	1120	140	8	202.95	1.10	49.49	48.68	150.75	3.71
7	1	1600	50	12	155.00	0.52	39.78	31.17	110.05	3.86
8	1	1600	100	8	140.17	0.36	37.90	26.84	98.66	4.97
9	1	1600	140	10	138.65	1.06	38.01	26.42	93.03	4.44
10	2	560	50	8	274.80	0.20	91.02	63.90	203.08	6.94
11	2	560	100	10	399.43	0.66	126.80	93.64	292.06	6.90
12	2	560	140	12	376.36	0.64	169.02	101.91	325.55	13.13
13	2	1120	50	10	197.57	0.24	57.53	39.16	136.70	4.33
14	2	1120	100	12	257.02	0.03	89.52	57.31	190.49	5.20
15	2	1120	140	8	264.01	0.90	84.36	66.12	194.13	5.11
16	2	1600	50	12	222.55	0.57	65.09	44.41	152.32	5.68
17	2	1600	100	8	163.29	0.72	44.88	34.59	119.01	5.75
18	2	1600	140	10	206.97	1.15	65.15	44.02	145.71	4.97
19	3	560	50	8	256.75	0.06	131.80	58.71	215.20	10.42
20	3	560	100	10	449.02	0.83	171.07	106.87	326.45	13.52
21	3	560	140	12	572.19	1.81	235.99	125.11	419.80	19.27
22	3	1120	50	10	291.49	1.49	109.13	70.08	236.28	8.28

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
23	3	1120	100	12	333.93	0.33	145.98	81.81	265.17	7.25
24	3	1120	140	8	482.05	1.91	178.51	136.10	413.49	13.49
25	3	1600	50	12	265.50	0.60	88.01	61.82	205.80	6.51
26	3	1600	100	8	331.97	2.13	105.88	86.62	253.54	7.22
27	3	1600	140	10	315.51	1.87	112.03	74.35	248.92	6.61

B 7 Experimental Model UL32.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	0.75	400	35.5	6	116.47	0.25	35.74	29.85	82.53	3.44
2	0.75	400	71	6	178.39	0.11	56.71	53.39	136.02	1.67
3	0.75	800	100	6	141.75	0.91	43.59	36.52	106.33	2.84
4	0.75	800	140	6	181.48	0.26	60.93	54.36	151.41	1.43
5	0.75	1120	35.5	8	129.38	0.34	24.98	19.80	69.55	2.33
6	0.75	1120	71	8	110.94	0.29	30.82	25.88	80.13	1.66
7	0.75	1600	100	8	109.10	0.32	31.84	24.60	83.39	2.54
8	0.75	1600	140	8	121.57	0.35	33.64	29.32	93.97	1.51
9	1.5	400	35.5	10	197.84	0.19	62.03	50.06	140.40	5.13
10	1.5	400	71	10	297.90	0.23	97.34	73.78	208.21	5.54
11	1.5	800	100	10	279.27	0.15	79.41	64.18	198.44	2.05
12	1.5	800	140	10	320.36	0.44	98.77	67.59	213.77	3.52
13	1.5	1120	35.5	12	175.84	0.22	42.84	37.67	117.77	2.91
14	1.5	1120	71	12	195.93	0.33	52.70	50.08	149.11	2.30
15	1.5	1600	100	12	200.73	0.34	52.69	49.80	155.92	2.06
16	1.5	1600	140	12	226.39	0.32	65.59	56.46	179.64	2.18
17	2.25	400	35.5	6	288.23	0.67	117.47	66.98	209.53	15.29
18	2.25	400	71	6	386.66	0.85	170.69	84.43	285.83	33.92
19	2.25	800	100	6	324.55	0.15	145.19	71.19	254.60	20.50
20	2.25	800	140	6	375.92	1.81	178.05	84.38	297.50	29.96
21	2.25	1120	35.5	8	143.07	0.47	39.50	31.44	102.50	3.43
22	2.25	1120	71	8	216.47	0.36	48.36	46.65	116.78	3.48
23	2.25	1600	100	8	222.04	0.37	65.42	54.81	160.85	3.71
24	2.25	1600	140	8	220.93	2.28	64.01	50.06	155.59	7.81
25	3	400	35.5	10	295.63	0.25	141.03	65.74	216.48	17.51
26	3	400	71	10	383.70	0.46	204.13	92.98	307.28	24.63
27	3	800	100	10	327.85	1.76	170.10	76.96	260.04	15.03
28	3	800	140	10	400.36	1.33	202.53	92.71	316.66	18.22
29	3	1120	35.5	12	325.09	0.32	85.52	76.17	217.82	5.00
30	3	1120	71	12	367.58	0.40	111.35	96.84	279.74	4.41
31	3	1600	100	12	371.64	1.07	112.14	101.65	293.94	4.20
32	3	1600	140	12	410.42	1.02	130.65	122.35	344.82	4.15

B 8 Experimental Model NL8-1.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	0.5	500	50	6	69.32	0.11	20.88	14.80	50.78	2.25
2	0.5	500	150	10	148.67	0.59	42.37	35.84	112.85	3.83
3	0.5	1000	50	6	62.57	0.23	19.82	12.50	47.26	2.02
4	0.5	1000	150	10	106.44	0.20	28.49	22.36	78.80	2.00
5	1.5	500	50	10	194.00	0.32	53.35	49.31	142.28	2.44
6	1.5	500	150	6	366.19	0.46	126.57	95.12	284.87	10.09
7	1.5	1000	50	10	126.11	0.24	36.21	31.53	95.53	1.65
8	1.5	1000	150	6	205.09	0.79	73.88	48.59	167.07	6.54

B 9 Experimental Model NL8-2.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	1	750	100	8	127.13	0.26	44.55	33.71	105.61	2.97
2	1	750	200	12	281.38	0.06	72.44	74.67	210.09	3.78
3	1	1500	100	8	173.66	0.19	39.65	29.99	124.90	2.23
4	1	1500	200	12	225.43	0.12	58.72	56.01	168.56	3.32
5	2.5	750	100	12	389.65	0.55	111.91	110.26	304.55	3.67
6	2.5	750	200	8	416.04	0.30	198.09	88.11	341.06	30.43
7	2.5	1500	100	12	312.18	0.53	88.25	83.04	248.56	4.02
8	2.5	1500	200	8	323.90	0.82	133.40	68.34	269.07	11.96

B 10 Experimental Model UL-add.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	0.5	560	100	10	135.41	0.70	38.48	34.30	107.52	2.49
2	1	1600	230	12	203.93	0.71	57.12	47.55	157.82	4.18
3	1.5	1120	35.5	8	131.85	0.20	36.82	28.22	90.84	3.75
4	1.5	1120	71	8	170.68	0.13	46.29	34.67	112.09	3.17
5	1.5	1600	140	8	158.02	0.91	53.28	35.62	113.54	4.61
6	1.5	1600	100	8	140.20	0.24	43.81	28.84	98.86	4.61
7	2	1120	50	7	168.23	0.04	61.91	39.07	134.29	3.29
8	2	800	100	10	243.53	0.38	95.53	61.98	203.69	3.56
9	2	1120	50	11	217.80	0.11	57.03	47.27	147.20	3.50
10	2	800	100	12	358.37	0.02	96.97	87.61	251.67	3.00
11	4	1120	50	10	369.64	3.84	145.10	69.35	256.29	17.46
12	4	800	100	10	371.18	2.08	204.83	77.37	307.88	23.83
13	4	1120	50	11	344.65	1.79	114.01	82.50	264.43	8.75
14	3.5	1600	140	6	278.08	0.10	103.47	71.39	247.42	1.91
15	3.5	800	280	11	805.34	3.02	406.14	183.75	665.62	47.58
16	3.5	1600	200	11	490.36	1.28	177.77	120.33	381.57	5.94

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
17	3	800	200	11	693.18	1.82	306.58	169.21	605.19	24.65
18	3	1120	280	11	700.59	1.88	292.99	151.50	566.45	22.43
19	1.5	1120	280	6	375.87	1.35	147.89	78.78	284.96	12.02
20	3	1120	140	6	333.87	0.56	150.92	74.73	273.25	10.22
21	1.5	1120	100	6	195.18	0.42	72.34	45.87	155.88	4.43

B 11 Experimental Model NL-add.

Exp #	Input Parameters				Output Parameters					
	a	n	f	d	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}
1	0.5	500	100	10	112.19	0.36	34.08	30.02	91.92	2.58
2	0.5	1500	100	6	68.36	0.21	19.78	15.70	53.49	1.08
3	0.5	500	200	12	172.20	0.42	53.90	39.08	124.47	5.12
4	1.5	1000	200	6	304.44	0.28	102.01	85.14	254.78	3.46
5	1	1000	100	8	126.35	0.18	37.81	31.23	97.05	1.77
6	1	750	100	12	169.59	0.32	47.42	44.33	131.34	2.95
7	0.5	750	100	12	99.04	0.50	26.05	20.97	70.06	2.63
8	1	1500	100	12	177.04	0.30	45.03	39.28	124.56	3.31
9	0.5	1000	50	10	60.90	0.31	17.78	12.05	45.63	1.84
10	0.5	1000	100	12	100.81	0.35	24.37	18.28	64.81	2.29
11	0.5	1000	200	6	120.61	0.20	34.04	31.39	95.70	1.51
12	1.5	500	50	6	224.35	0.12	60.86	56.21	157.02	1.36
13	1.5	500	100	10	331.43	0.21	87.96	91.23	242.43	2.06
14	1.5	1500	200	10	310.96	0.24	76.95	73.76	219.24	1.87
15	2.5	500	100	10	420.52	0.46	162.69	115.54	352.76	8.14
16	0.5	1500	200	10	132.49	0.15	28.67	25.85	86.23	1.19
17	0.5	500	150	10	163.72	0.37	49.26	43.75	131.03	2.54

Appendix C

ANN RESULTS

C 1 No. of Training patterns vs. Predictive Errors using RBN and UL59 model.

No. Of training Patterns	Predictive Force Errors						Excluding F_{Min}, FM-Min	
	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
2	1.438	9.967	1.536	2.614	1.871	0.643	1.865	0.575
3	0.764	18.512	0.686	0.994	0.827	1.518	0.817	0.145
4	0.373	9.233	0.407	0.507	0.447	0.834	0.433	0.105
5	0.368	9.184	0.347	0.523	0.379	0.597	0.404	0.175
6	0.391	7.823	0.346	0.486	0.400	0.574	0.406	0.148
7	0.493	9.760	0.515	0.621	0.523	0.639	0.538	0.121
8	0.448	9.132	0.460	0.510	0.462	0.617	0.470	0.101
9	0.585	10.180	0.630	0.654	0.577	0.766	0.612	0.100
10	0.513	9.995	0.563	0.565	0.510	0.752	0.538	0.094
11	0.523	10.368	0.593	0.567	0.524	0.774	0.552	0.095
12	0.443	9.679	0.490	0.483	0.458	0.719	0.468	0.079
13	0.436	8.517	0.474	0.471	0.448	0.664	0.457	0.078
14	0.450	9.294	0.520	0.476	0.463	0.723	0.477	0.077
15	0.415	9.061	0.490	0.439	0.435	0.724	0.444	0.075
16	0.401	8.362	0.480	0.413	0.422	0.735	0.429	0.071
17	0.425	7.947	0.444	0.475	0.456	0.870	0.450	0.087
18	0.424	7.911	0.444	0.474	0.455	0.868	0.449	0.090
19	0.424	9.213	0.444	0.474	0.455	0.927	0.449	0.087
20	0.430	8.985	0.464	0.479	0.460	1.076	0.458	0.096
21	0.370	8.981	0.379	0.416	0.397	0.935	0.391	0.090
22	0.372	8.760	0.380	0.417	0.398	0.873	0.392	0.087
23	0.305	8.725	0.333	0.325	0.301	0.787	0.316	0.058
24	0.304	8.859	0.331	0.325	0.300	0.790	0.315	0.057
25	0.380	8.368	0.406	0.424	0.401	0.761	0.403	0.077
26	0.380	8.650	0.406	0.424	0.401	0.801	0.403	0.078
27	0.284	8.715	0.278	0.339	0.307	0.740	0.302	0.080
28	0.281	8.625	0.278	0.338	0.304	0.740	0.300	0.079
29	0.282	8.487	0.284	0.345	0.305	0.729	0.304	0.079
30	0.278	8.566	0.282	0.340	0.301	0.728	0.300	0.079
31	0.276	8.460	0.284	0.341	0.300	0.715	0.300	0.079
32	0.275	8.361	0.280	0.337	0.297	0.702	0.297	0.079
33	0.272	8.254	0.276	0.334	0.292	0.686	0.294	0.078
34	0.271	8.134	0.274	0.333	0.290	0.669	0.292	0.078
35	0.270	8.058	0.270	0.332	0.288	0.649	0.290	0.078
36	0.272	8.001	0.274	0.335	0.290	0.664	0.293	0.078
37	0.279	7.935	0.285	0.341	0.295	0.681	0.300	0.079
38	0.279	7.820	0.281	0.337	0.292	0.669	0.297	0.078
39	0.278	7.751	0.276	0.330	0.287	0.655	0.293	0.077

No. Of training Patterns	Predictive Force Errors						Excluding F_{Min}, F_{M-Min}	
	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
40	0.278	7.711	0.276	0.331	0.288	0.648	0.293	0.077
41	0.278	7.692	0.276	0.332	0.290	0.638	0.294	0.077
42	0.278	7.660	0.277	0.336	0.292	0.630	0.296	0.077
43	0.268	7.541	0.279	0.304	0.285	0.642	0.284	0.078
44	0.275	7.571	0.294	0.310	0.292	0.698	0.293	0.078
45	0.270	7.538	0.276	0.301	0.286	0.650	0.283	0.077
46	0.269	7.467	0.278	0.297	0.286	0.661	0.283	0.076
47	0.268	7.936	0.286	0.293	0.283	0.701	0.282	0.076
48	0.236	4.629	0.200	0.227	0.213	0.534	0.219	0.072
49	0.220	4.490	0.190	0.208	0.190	0.538	0.202	0.068
50	0.171	3.748	0.178	0.179	0.163	0.669	0.173	0.074
51	0.165	3.758	0.177	0.174	0.166	0.661	0.171	0.079
52	0.165	3.724	0.183	0.174	0.166	0.685	0.172	0.078
53	0.168	3.576	0.193	0.177	0.170	0.718	0.177	0.081
54	0.169	3.704	0.191	0.178	0.173	0.714	0.178	0.080
55	0.226	3.902	0.200	0.200	0.189	0.677	0.204	0.080
56	0.220	3.803	0.200	0.195	0.186	0.676	0.200	0.080
57	0.216	3.711	0.199	0.187	0.183	0.676	0.196	0.079
58	0.181	3.747	0.151	0.179	0.166	0.649	0.169	0.069
59	0.179	3.643	0.135	0.171	0.160	0.640	0.161	0.070

C 2 No. of Training patterns [60, 90] vs. Predictive Errors using RBN and UL90 model.

No. Of training Patterns	Predictive Force Errors						Excluding F_{Min}, FM-Min	
	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
60	0.179	3.624	0.138	0.171	0.161	0.648	0.1622	0.0694
61	0.175	3.615	0.141	0.166	0.159	0.636	0.1602	0.0688
62	0.167	3.624	0.132	0.161	0.154	0.607	0.1534	0.0744
63	0.167	3.640	0.132	0.161	0.154	0.606	0.1536	0.0746
64	0.176	4.352	0.185	0.170	0.187	0.611	0.1796	0.0759
65	0.174	3.838	0.187	0.170	0.187	0.630	0.1795	0.0774
66	0.174	3.807	0.187	0.169	0.186	0.628	0.1790	0.0773
67	0.178	3.971	0.185	0.172	0.186	0.606	0.1803	0.0746
68	0.178	3.964	0.186	0.172	0.187	0.604	0.1808	0.0748
69	0.177	3.990	0.181	0.171	0.185	0.600	0.1787	0.0730
70	0.171	2.931	0.195	0.166	0.185	0.572	0.1793	0.0692
71	0.171	2.948	0.195	0.167	0.185	0.582	0.1796	0.0694
72	0.129	2.600	0.151	0.117	0.132	0.636	0.1321	0.0636
73	0.136	3.318	0.141	0.130	0.145	0.670	0.1379	0.0689
74	0.136	3.313	0.142	0.131	0.145	0.669	0.1386	0.0688
75	0.118	2.238	0.159	0.105	0.118	0.733	0.1248	0.0672
76	0.118	2.292	0.157	0.103	0.116	0.738	0.1236	0.0663
77	0.119	2.273	0.159	0.103	0.116	0.737	0.1241	0.0663
78	0.127	3.131	0.159	0.103	0.110	0.684	0.1248	0.0654
79	0.158	2.439	0.172	0.165	0.147	0.655	0.1604	0.0569
80	0.156	3.835	0.137	0.175	0.135	0.645	0.1508	0.0642
81	0.131	3.736	0.179	0.118	0.118	0.674	0.1366	0.0770
82	0.131	4.508	0.173	0.140	0.127	0.643	0.1425	0.0671
83	0.143	3.433	0.147	0.134	0.127	0.602	0.1377	0.0497
84	0.127	4.319	0.138	0.133	0.115	0.625	0.1282	0.0553
85	0.116	4.910	0.178	0.109	0.106	0.665	0.1272	0.0688
86	0.119	4.901	0.156	0.108	0.097	0.679	0.1199	0.0596
87	0.119	4.852	0.147	0.105	0.094	0.624	0.1162	0.0575
88	0.107	4.146	0.101	0.097	0.078	0.538	0.0956	0.0516
89	0.108	4.149	0.101	0.097	0.079	0.538	0.0962	0.0518
90	0.107	4.063	0.103	0.095	0.077	0.552	0.0954	0.0526

C 3 Spread value [0.02, 1] vs. Predictive Errors using RBN.

Spread Value	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
0.020	**0.364**	**4.490**	**0.485**	**0.397**	**0.396**	**0.580**	**0.410**	**0.093**
0.030	**0.364**	**4.490**	**0.485**	**0.397**	**0.396**	**0.580**	**0.410**	**0.093**
0.040	**0.364**	**4.490**	**0.485**	**0.397**	**0.396**	**0.580**	**0.410**	**0.093**
0.050	**0.364**	**4.489**	**0.485**	**0.397**	**0.396**	**0.580**	**0.410**	**0.093**
0.060	**0.364**	**4.484**	**0.484**	**0.397**	**0.396**	**0.579**	**0.410**	**0.093**
0.070	**0.364**	**4.475**	**0.484**	**0.397**	**0.396**	**0.577**	**0.410**	**0.093**
0.080	**0.364**	**4.462**	**0.483**	**0.397**	**0.396**	**0.575**	**0.410**	**0.093**
0.090	**0.363**	**4.448**	**0.483**	**0.397**	**0.396**	**0.572**	**0.410**	**0.093**
0.100	**0.363**	**4.435**	**0.482**	**0.397**	**0.396**	**0.568**	**0.409**	**0.093**
0.110	**0.363**	**4.424**	**0.481**	**0.397**	**0.395**	**0.565**	**0.409**	**0.093**
0.120	**0.363**	**4.415**	**0.480**	**0.397**	**0.395**	**0.561**	**0.409**	**0.092**

Spread Value	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	F_{M-Max}	F_{M-Min}	E_{Mean}	E_{Stdev}
0.130	0.362	4.406	0.479	0.396	0.395	0.556	0.408	0.092
0.140	0.362	4.396	0.477	0.396	0.394	0.550	0.407	0.092
0.150	0.361	4.385	0.475	0.395	0.393	0.543	0.406	0.091
0.160	0.360	4.415	0.475	0.392	0.391	0.551	0.405	0.091
0.170	0.359	4.423	0.475	0.389	0.388	0.546	0.403	0.092
0.180	0.355	4.302	0.469	0.389	0.387	0.538	0.400	0.091
0.190	0.351	4.274	0.462	0.385	0.383	0.527	0.395	0.090
0.200	0.348	4.244	0.456	0.380	0.379	0.516	0.391	0.089
0.210	0.344	4.214	0.448	0.375	0.374	0.505	0.385	0.088
0.220	0.340	4.183	0.440	0.370	0.368	0.495	0.380	0.086
0.230	0.336	4.007	0.433	0.364	0.362	0.511	0.374	0.085
0.240	0.331	3.967	0.423	0.357	0.356	0.500	0.367	0.084
0.250	0.326	3.928	0.412	0.350	0.350	0.489	0.360	0.082
0.260	0.317	3.983	0.394	0.328	0.332	0.507	0.343	0.078
0.270	0.313	3.939	0.380	0.319	0.324	0.496	0.334	0.077
0.280	0.308	3.865	0.363	0.308	0.313	0.494	0.323	0.078
0.290	0.303	4.006	0.354	0.301	0.306	0.500	0.316	0.076
0.300	0.304	3.740	0.347	0.305	0.305	0.489	0.315	0.072
0.310	0.280	3.740	0.349	0.291	0.294	0.502	0.303	0.069
0.320	0.271	3.654	0.332	0.282	0.284	0.489	0.292	0.067
0.330	0.269	3.573	0.323	0.276	0.275	0.493	0.286	0.067
0.340	0.263	3.491	0.308	0.267	0.266	0.482	0.276	0.065
0.350	0.249	3.428	0.311	0.251	0.260	0.441	0.268	0.071
0.360	0.238	3.366	0.303	0.237	0.246	0.459	0.256	0.071
0.370	0.226	3.272	0.284	0.223	0.231	0.442	0.241	0.067
0.380	0.213	3.181	0.271	0.204	0.212	0.454	0.225	0.062
0.390	0.203	3.094	0.259	0.191	0.200	0.438	0.213	0.060
0.400	0.194	3.008	0.247	0.181	0.190	0.428	0.203	0.059
0.410	0.202	2.851	0.236	0.188	0.196	0.421	0.206	0.053
0.420	0.195	2.772	0.222	0.181	0.185	0.420	0.196	0.052
0.430	0.183	2.764	0.219	0.167	0.166	0.399	0.184	0.052
0.440	0.185	2.652	0.207	0.170	0.175	0.397	0.184	0.048
0.450	0.186	2.325	0.199	0.168	0.172	0.370	0.181	0.049
0.460	0.173	2.408	0.214	0.163	0.163	0.375	0.178	0.061
0.470	0.163	2.195	0.210	0.155	0.154	0.368	0.171	0.057
0.480	0.176	2.089	0.199	0.179	0.164	0.377	0.180	0.052
0.490	0.180	1.998	0.209	0.179	0.169	0.392	0.184	0.056
0.500	0.164	2.274	0.191	0.171	0.158	0.403	0.171	0.050
0.510	0.160	2.242	0.186	0.169	0.156	0.400	0.168	0.049
0.520	0.157	2.212	0.182	0.168	0.154	0.398	0.165	0.050
0.530	0.152	1.955	0.180	0.155	0.141	0.444	0.157	0.055
0.540	0.147	1.848	0.157	0.133	0.129	0.398	0.141	0.045
0.550	0.164	1.804	0.161	0.174	0.154	0.452	0.163	0.046
0.560	0.152	1.381	0.152	0.146	0.128	0.494	0.145	0.054
0.570	0.149	1.855	0.154	0.120	0.128	0.354	0.138	0.052
0.580	0.144	1.611	0.149	0.152	0.141	0.389	0.146	0.052
0.590	0.141	1.593	0.145	0.103	0.114	0.461	0.126	0.048
0.600	0.141	1.284	0.142	0.128	0.117	0.334	0.132	0.054
0.610	0.132	1.355	0.151	0.109	0.115	0.378	0.127	0.050
0.620	0.135	1.419	0.146	0.123	0.118	0.386	0.131	0.062
0.630	0.111	1.680	0.159	0.098	0.100	0.450	0.117	0.066
0.640	0.124	1.672	0.156	0.091	0.107	0.423	0.119	0.066
0.650	0.145	1.118	0.124	0.098	0.102	0.357	0.117	0.060

Spread Value	F_{Max}	F_{Min}	F_{Mean}	F_{Stdev}	$F_{M\text{-}Max}$	$F_{M\text{-}Min}$	E_{Mean}	E_{Stdev}
0.660	0.132	1.766	0.180	0.069	0.086	0.440	0.117	0.063
0.670	0.131	1.718	0.179	0.070	0.086	0.432	0.116	0.063
0.680	0.117	1.473	0.148	0.100	0.103	0.313	0.117	0.070
0.690	0.138	2.125	0.174	0.130	0.128	0.305	0.142	0.069
0.700	0.136	2.397	0.166	0.162	0.130	0.433	0.149	0.072
0.710	0.113	1.487	0.155	0.093	0.107	0.348	0.117	0.066
0.720	0.140	1.782	0.172	0.105	0.114	0.314	0.133	0.069
0.730	0.134	2.122	0.141	0.137	0.109	0.338	0.130	0.060
0.740	0.159	1.562	0.178	0.114	0.115	0.402	0.141	0.081
0.750	0.171	1.416	0.180	0.148	0.153	0.279	0.163	0.064
0.760	0.154	1.815	0.204	0.100	0.125	0.461	0.146	0.090
0.770	0.146	1.400	0.157	0.120	0.125	0.303	0.137	0.064
0.780	0.149	1.335	0.182	0.112	0.131	0.329	0.143	0.068
0.790	0.124	0.547	0.135	0.090	0.103	0.444	0.113	0.072
0.800	0.100	1.472	0.119	0.109	0.093	0.225	0.106	0.060
0.810	0.104	1.679	0.125	0.087	0.081	0.287	0.099	0.057
0.820	0.130	2.188	0.136	0.116	0.106	0.280	0.122	0.055
0.830	0.096	1.766	0.079	0.093	0.073	0.326	0.085	0.053
0.840	0.110	2.095	0.127	0.108	0.098	0.377	0.111	0.066
0.850	0.113	1.140	0.131	0.122	0.103	0.305	0.117	0.057
0.860	0.086	1.964	0.117	0.091	0.083	0.346	0.094	0.061
0.870	0.109	1.462	0.119	0.118	0.101	0.267	0.112	0.077
0.880	0.111	1.613	0.130	0.092	0.084	0.355	0.104	0.066
0.890	0.133	0.347	0.100	0.114	0.091	0.301	0.110	0.054
0.900	0.133	0.347	0.096	0.113	0.089	0.299	0.108	0.054
0.910	0.164	1.275	0.149	0.157	0.151	0.399	0.155	0.065
0.920	0.164	1.283	0.148	0.157	0.151	0.404	0.155	0.065
0.930	0.155	0.835	0.152	0.170	0.156	0.391	0.158	0.066
0.940	0.082	1.034	0.108	0.102	0.071	0.228	0.091	0.061
0.950	0.116	0.874	0.176	0.135	0.110	0.369	0.134	0.074
0.960	0.144	1.936	0.112	0.140	0.112	0.400	0.127	0.061
0.970	0.117	1.599	0.082	0.099	0.072	0.286	0.092	0.057
0.980	0.146	1.553	0.154	0.108	0.108	0.294	0.129	0.065
0.990	0.101	1.132	0.117	0.105	0.079	0.397	0.101	0.056

Appendix D

SOURCE CODE IN MATLAB

D 1 Neural Modeling Program[16]

```
function [net,Y_Real_T,Y_Real_V,Y_Predicted_T,Y_Predicted_V,Err_T,Err_V]=getNN_model(NType,machine,t,fa)
noise=0.000001;
FileName='Fmodels';
ext='-mat';
[F,I,V]=getForceData(FileName,ext,machine,t);
I=getNoise(I,noise);
size(I,1)
switch lower(NType)

case 'fb'
   hi=fa(1,1);
   si=fa(1,2);
   peak=fa(1,3);
   tg=fa(1,4);
   tf=[4,2,4]
    btf=1;
   blf=1;
   PF='MSEREG';

[net,Yt,Yv]=getNfr(I,V,hi,si,peak,tg,tf,btf,blf,pf);
[Y_Real_T,Y_Real_V,Y_Predicted_T,Y_Predicted_V,Err_T,Err_V]=getAna(F,I,V,Yt,Yv);
plot(Y_Real_T(:,5));
hold on
plot(Y_Predicted_T(:,5),'red');


case 'rb'
   goal=fa(1,1);
   spread=fa(1,2);
    mn=20;
    df=1;
    PF='MSE' ;
    %MSEREG, MAE,mse,'sse';

[net,Yt,Yv]=getNrb(I,V,goal,spread,mn,df,PF);
[Y_Real_T,Y_Real_V,Y_Predicted_T,Y_Predicted_V,Err_T,Err_V]=getAna(F,I,V,Yt,Yv);

otherwise
     disp('No Such Neural Name(choose either FB or RB')
end

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [Fall,I,V]=getForceData(FileName,ext,op1,t)
load(FileName,ext);
switch lower(op1)
  case 'conv'
    disp('Looding Experiments conducted on Conventional Machine...')
   Fall=[FL8_1;FL9;FL27_1;FL27_2;FL32;FL25;FL8_2;Fv;Fx;addo];
   F=Fall;
          Fn=Scale(F,F);

          FL8_1=Fn(1:8,:);
          FL9=Fn(9:17,:);
          FL27_1=Fn(18:44,:);
          FL27_2=Fn(45:71,:);
```

```
        FL32=Fn(72:103,:);
        FL25=Fn(104:128,:);
        FL8_2=Fn(129:136,:);
          Fv=Fn(137:155,:);
    Fx=Fn(156:157,:);
    %tname=[{'FL8_1','FL9','FL27_1','FL27_2','FL32','FL25','FL8_2','Fx'}];
    %F=[FL8_1;FL9;FL27_1;FL27_2;FL32;FL25;FL8_2;Fx];
    F=[FL32;FL25;FL8_2;FL27_2;FL8_1;FL9;FL27_1;Fx];
    %F=[FL27_1;FL32;FL25;FL8_1(7,:);FL8_2(6,:);FL9(2,:);FL27_2(10:12,:)];
    %V=[Fv;FL27_2(1:9,:);FL27_2(13:27,:);FL9(3:9,:);FL9(1,:);FL8_2(1:5,:);FL8_2(7:8,:)];
    V=[Fv];
    I=[];


      for i=1:size(t,2)
            try
              I=[I;eval(char(t(1,i)))];
            catch
              disp('error in force model name')
        end

        else
          I=F(1:t,:);

%     if (size(t,2)<=size(tname,2)&size(t,2)>0)
%             for i=1:size(t,2)
%             I=[I;eval(char(tname(1,t(1,i))))];
%           end
%         else
%           disp('Only 8 models are available');
%         end
%
end

  case 'cnc'
    disp('Looding Experiments conducted on CNC Machine...')
    F=[CFL8_1;CFL8_2;CFv;CFx];
    Fall=F;
    Fn=Scale(F,F);

    CFL8_1=Fn(1:8,:);
    CFL8_2=Fn(9:16,:);
     CFv=Fn(17:29,:);
     CFx=Fn(30:33,:);
     V=CFv;
     I=[];
     tname=[{'CFL8_1','CFL8_2','CFx'}];
     if (size(t,2)<=size(tname,2)&size(t,2)>0)
            for i=1:size(t,2)
            I=[I;eval(char(tname(1,t(1,i))))];
          end
        else
          disp('Only 3 models are available');
        end



  otherwise
    disp('No Such Experiment Name')
end

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [dn]=Scale(F,d)
if (nargin ~= 2 | size(d,2) >10)
dn = [];
disp('number of matrix columns must be less than or equal 10');
```

```
return
end
%% Scale factors
[R_MIN,R_MAX,S_MIN,S_MAX ]=Scale_Factors(F);

for i=1:size(d,1)
for j=1:size(d,2)
  a=(S_MAX(1,j)-S_MIN(1,j))/(R_MAX(1,j)-R_MIN(1,j));
  dn(i,j)=((d(i,j)-R_MIN(1,j))*a)+S_MIN(1,j);
end
end

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [d]=unscale(F,dn)
[R_MIN,R_MAX,S_MIN,S_MAX ]=Scale_Factors(F);

for i=1:size(dn,1)

for j=1:size(dn,2)

  an=((R_MAX(1,j)-R_MIN(1,j))/(S_MAX(1,j)-S_MIN(1,j)));
  d(i,j)=((dn(i,j)-S_MIN(1,j))*an)+R_MIN(1,j);

end
end
function [dn]=unscale2(F,d)


for i=1:size(d,1)
for j=1:size(d,2)
  dn(i,j)=F(i,j)*(1+d(i,j));
end
end


% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [Y]=getNoise(F,noise)

noise=noise/3;

for i=1:size(F,1)
for j=1:size(F,2)
  Y(i,j)=F(i,j)*(1+(noise*getRand(3)));
end
end

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [Y]=randSort(F)
randomN=randperm(size(F,1));

for i=1:size(F,1)
  Y(i,:)=F(randomN(1,i),:);
end
% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [X]=getErr(Y,T);
if (nargin ~= 2 | size(Y,1) ~= size(T,1)|size(Y,2)~=size(T,2))
X = [];
disp('to compute the Errors the two matrix must have the same size');
return
end
%T=abs(T);
%Y=abs(Y);
for i=1:size(Y,1)
for j=1:size(Y,2)
```

```
  if Y(i,j)==0
    Y(i,j)=0.000001;
  end
  X(i,j)=abs((Y(i,j)-T(i,j))/Y(i,j));
end
end
Ymean=mean([X(:,1),X(:,3:5)],2);
Ystd=std([X(:,1),X(:,3:5)],0,2);
X=[X,Ymean,Ystd];
X=[X;mean(X,1)];

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [mx]=getMax(data);
for i=1:size(data,2);
mx(1,i)=max(data(:,i));
end

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [mn]=getMin(data);
for i=1:size(data,2);
mn(1,i)=min(data(:,i));
end

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [S] = val1(op,k)
for i=1:size(op,2)
 if (op(1,i)>k | op(1,i)<1)
   S=0;
   return
 else
   S=1;
 end
end

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
% This function stores the most known BLF functions in matlab
% BLF = Backprop weight/bias learning function, default = 'learngdm'.
function [S]=getBlf(op);
MyBlf=[{'learngdm','learngd'}];
if (nargin ~= 1 | val1(op,size(MyBlf,2))~=1)
S = char(MyBlf(1,1));
disp('Using the Default Backprop weight/bias learning function ');
return
end
S=char(MyBlf(1,op));
% This function stores the most known BTF functions in matlab
% BTF = Backprop network training function, default = 'trainlm'.
function [S]=getBtf(op);
MyBtf=[{'trainlm','trainbfg','trainrp','traingd','traingdm','traingdx','traincgf',' trainbr','trainscg'}];
if (nargin ~= 1 | val1(op,size(MyBtf,2))~=1)
S = char(MyBtf(1,1));
disp('Using the Default Backprop network training function ');
return
end
S=char(MyBtf(1,op));
% This function stores the most known TF functions in matlab
% TFi = Transfer function of ith layer, default = 'tansig' and i=1.

function [S]=getTf(hdNo,op);
MyTf=[{'tansig','logsig','radbas','purelin','tribas','poslin','satlin','satlins','hardlims'}];
if (nargin < 1)
  hdNo=1;
  op=ones(1,3);
end
if (nargin < 2)
  op=ones(1,hdNo+2);
```

```
end

if (size(op,2) ~=(hdNo+2) |val1(op,size(MyTf,2))~=1)
S = {'tansig'};
hdNo=1;
S = [S(ones(1,hdNo+2))];
disp('Using the Defult Transfer Functions');
return
end
for i=1:(hdNo+2)
S(1,i)=MyTf(1,op(1,i));
end
% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [net,St,Sv]=getNfr(I,V,hdNo,hsize,EpochsN,Goal,tf,btf,blf,PF);
input=4;
output=6;
P=I(:,1:input);
Pv=V(:,1:input);
T=I(:,input+1:input+output);
Tv=V(:,input+1:input+output);
if nargin < 3
  hdNo=1;
end
if nargin < 4
  hsize=round(size(I,1)*0.25);
end
if nargin < 5
  EpochsN=100;
end
if nargin < 6
  Goal=0.00001;
end
if nargin < 7
  TF=getTf(hdNo)
else
  TF=getTf(hdNo,tf);
end
if nargin < 8
  BTF =getBtf
else
  BTF=getBtf(btf);
end
if nargin < 9
  BLF =getBlf;
else
  BLF=getBlf(blf);
end
if nargin < 10 PF='mse' ; end

for i=1:hdNo
 hl(i)=hsize;
end
lsz=[size(P,2),hl,size(T,2)]
pr=zeros(size(P,2),2);
for i=1:size(P,2)
pr(i,2)=max(P(:,i));
pr(i,1)=min(P(:,i));
end

%net = newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)newfftd
net=newff(pr,lsz,TF,BTF,BLF,PF);
% net.trainParam.show = 25;
 net.trainparam.epochs=EpochsN;
  net.trainParam.mem_reduc =1;
%           %training rate
  net.trainParam.lr = 0.1;
```

```
%     % momentum factor
  net.trainParam.mc = 0.95;%
 net.trainparam.goal=Goal;

% net.adaptParam.passes = 10;
% [net,y,e]=adapt(net,P',T');
net.performFcn = PF;
if strcmp(PF,'MSEREG')==1;
   net.performParam.ratio=0.85;
end

[net lrncurv]=train(net,P',T');


St=[sim(net,P')]';
Sv=[sim(net,Pv')]';



% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [net,St,Sv]=getNrb(I,V,goal,spread,mn,df,PF);
input=4;
output=6;
P=I(:,1:input);
Pv=V(:,1:input);
T=I(:,input+1:input+output);
Tv=V(:,input+1:input+output);
if nargin < 3
  goal=1;
end
if nargin < 4
   spread=0.97;
end
if nargin < 5
   mn=round(size(I,1)*0.5);
end
if nargin < 6
   df=2;
end
if nargin < 7
   PF='mse' ;
end

%newrb(p,t,goal,spread,mn,df)
net=newrb(P',T',goal,spread,mn,df);
net.performFcn = PF;
St=[sim(net,P')]';
Sv=[sim(net,Pv')]';


% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [a]=getPlace(b);
x=min(b);
for i=1:size(b,1)
   for k=1:size(b,2)
      if x(1,k)==b(i,k)
         a(1,k)=i;
      end
   end
end

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [m]=bestE(data,n,pos);
if nargin < 3
   pos=size(data,2)-1;
end
```

```
for i=1:n
   [a]=getPlace(data(:,pos));
   m(i,:)=data(a,:);
   data(a,:)=[];
end

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
function [Y_Real_T,Y_Real_V,Y_Predicted_T,Y_Predicted_V,Err_T,Err_V]=getAna(F,I,V,St,Sv)
Yt=[I(:,1:4),St];
Yv=[V(:,1:4),Sv];
Y_Predicted_T=unscale(F,Yt);
Y_Predicted_V=unscale(F,Yv);
Y_Real_T=unscale(F,I);
Y_Real_V=unscale(F,V);
Err_T=getErr(Y_Predicted_T(:,5:10),Y_Real_T(:,5:10));
Err_V=getErr(Y_Predicted_V(:,5:10),Y_Real_V(:,5:10));

% <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
```

D 2 Force Treatment Program [16]

```
F=[];
load info.txt;
ExperimentN=input('Please Enter Number of Experiments of the model= ');
for i =1:ExperimentN
%dataName=input('Please Enter The experiment name ','s');
load(dataName,'-mat');
R(i,:)=getF(data,info(i,:));
end
R=[info(:,2:5),R];
F=[R];
save F;
%%%%%%%%%%%%%%%%%%%%%%%
function [R]=getF(data,parm);

   Sx =parm(1,6);
   Sy =parm(1,6);
   Sz =parm(1,7);
   RPM=parm(1,3);

Sx=Sx*5;
Sy=Sy*5;
Sz=Sz*5;
rev=round((60*RPM)/500);

X=data(2,:);
Y=data(3,:);
Z=data(4,:);
X=X.*Sx;
Y=Y.*Sy;
Z=Z.*Sz;
F=sqrt(X(1,:).^2+Y(1,:).^2+Z(1,:).^2);
plot(F);
xlabel('Time 1/500 sec');
  ylabel('Force N');
  title('Plot of Force vs time');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%option=input('Please Enter option #  ');
option=2;
if option ==2
   [X,Y,Z]=delUnwanted(X,Y,Z);
   [X,Y,Z]=delNoise(X,Y,Z);
end
if option ==1
  [X,Y,Z]=delNoise(X,Y,Z);
```

```
end
if option ==3
  [X,Y,Z]=delUnwanted(X,Y,Z);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


F=sqrt(X(1,:).^2+Y(1,:).^2+Z(1,:).^2);
plot(F);
  xlabel('Time 1/500 sec');
  ylabel('Force N');
  title('Plot of Force vs time');
Fmax=max(F);
Fmin=min(F);
Fmean=mean(F);
Fstd=std(F);
ll=1:rev;
b=(length(F)/rev);
j=0;
k=1000;
for i=1:b
   mx(i)=max(F((i-1)*rev+ll));
   if mx(i)>j
      j=mx;
      j1=i;
   end
   mn(i)=min(F((i-1)*rev+ll));
   if mn(i)<k
      k=mx;
      k1=i;
   end
end
Fmean_max=mean(mx);
Fmean_min=mean(mn);
R=[Fmax,Fmin,Fmean,Fstd,Fmean_max,Fmean_min,j1/b];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [X,Y,Z]=delNoise(X1,Y1,Z1);
F=sqrt(X1(1,:).^2+Y1(1,:).^2+Z1(1,:).^2);
plot(F)
[aaa,aaaa]=ginput(2);
a3=round((aaa(1,1)));
a4=round((aaa(2,1)));
Z=Z1(a3:a4)-(0.5*(mean(Z1(1:a3-1))+mean(Z1(a4+1:length(Z1)-1))));
X=X1(a3:a4)-(0.5*(mean(X1(1:a3-1))+mean(X1(a4+1:length(X1)-1))));
Y=Y1(a3:a4)-(0.5*(mean(Y1(1:a3-1))+mean(Y1(a4+1:length(Y1)-1))));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [X,Y,Z]=delUnwanted(X1,Y1,Z1);
F=sqrt(X1(1,:).^2+Y1(1,:).^2+Z1(1,:).^2);
plot(F)
[a,aa]=ginput(2);
a1=round((a(1,1)));
a2=round((a(2,1)));
Z=Z1(a1:a2);
X=X1(a1:a2);
Y=Y1(a1:a2);
```