The fuzzy-nets based approach in predicting the cutting power of end milling operations

by

Chuan-Teh Chang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Education and Technology

Major Professor: Joseph C. Chen

Iowa State University

Ames, Iowa

1997

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Chuan-Teh Chang

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Process planning is a major determinant of manufacturing cost. The selection of machining parameters is an important element of process planning. The development of a utility to show the cutting power on-line would be helpful to programmers and process planners in selecting machining parameters. The relationship between the cutting power and the machining parameters is nonlinear. Presently there is no accurate or simple algorithm to calculate the required cutting power for a selected set of parameters. Although machining data handbooks, machinability data systems, and machining databases have been developed to recommend machining parameters for efficient machining, they are basically for general reference and hard to use as well.

In this research, a self-organizing fuzzy-nets optimization system was developed to generate a knowledge bank that can show the required cutting power on-line for a short length of time in an NC verifier. The fuzzy-nets system (FNS) utilizes a five-step self-learning procedure. A generic FNS program consisting of fuzzification and defuzzification modules was implemented in the C++ programming language to perform the procedure. The FNS was assessed before an actual experiment was set up to collect data.

The performance of the FNS was then examined for end milling operations on a Fadal VMC40 vertical machining center. The cutting force signals were measured by a three-component dynamometer mounted on the table of the Fadal CNC machine with the workpiece mounted on it. Amplified signals were collected by a personal computer on which an Omega DAS-1401 analog-to-digital (A/D) converter was installed to sample the data on-line. Data sets were collected to train and test the system. The results showed that the FNS possessed a satisfactory range of accuracy with the intended applications of the model. The values of cutting power predicted by the FNS were more accurate than the formula values. Compared to the FNS system, dynamometers and amplifiers are very expensive. Thus, most of them could be replaced with the FNS.

CHAPTER 1. INTRODUCTION

The task of manufacturing is to produce product components that meet design specifications. On the other hand, how a component will be manufactured is determined by process planning which acts as a bridge between design and manufacturing. Thus, process planning refers to a set of instructions that are used to make a component so that the design specifications are met. Process planning is the major determinant of manufacturing cost. An automated or computer-aided process planning (CAPP) system can be used in process plans to reduce time and cost. The selection of machining parameters, e.g., cutting speed, feed, and depth of cut, is an important element of process planning.

The utilization of computers in manufacturing has been one of the most significant developments over the last couple of decades in improving the productivity and quality of manufacturing systems (Singh, 1996). One of the earliest applications of computers to control individual manufacturing functions at the shop floor level has been Numerical control (NC). Most NC machines in use today are metal-cutting machine tools (Singh, 1996).

An important factor in using NC machines effectively is the efficient collection and use of accurate, reliable machinability data (Parsons, 1971). Traditionally, a programmer communicates with a CAD/CAM interface to generate an NC program. However, it is uncertain whether the program can use the NC machine optimally. Overuse of the machine's cutting power will cause damage to the machine, tool, etc. On the other hand, underuse results in a lack of optimal productivity. Therefore, there is a need for the programmer to know the cutting power requirement of the programs.

A tool to show the cutting power on-line in the NC verifier would be helpful to programmers and process planners in selecting machining parameters. However, there is presently no accurate or simple algorithm to calculate the required cutting power when inputting machining parameters such as speed (Sc), feed (Fr), depth of cut (Dc), strength of workpiece material (Ws), strength of tool material (Ts), etc. Although machining data handbooks and machinability data systems have been developed to recommend machining parameters for efficient machining, they are basically for general reference and are hard to use as well. The machining database is not efficient in terms of time and space. In addition, the machining database assumes that all the machines are same. This assumption is just not true. Each machine has its own characteristics and capabilities. Therefore, intelligence is needed in the machine control system.

The drive for autonomy and intelligence in manufacturing and manufactured goods, coupled with increased complexity and high performance requirements, necessitates more sophisticated control systems such as intelligent controllers. There have been two main developments relevant to intelligent control: artificial neural networks (ANNs) and fuzzy logic. ANNs were developed to emulate the human brain's neuronal-synaptic mechanisms that store, learn, and retrieve information on a purely experiential basis, whereas fuzzy logic was developed to emulate human reasoning, using linguistic expressions (Brown & Harris, 1994; Pal & Srimani, 1996). An ANN has training capability but no reasoning capability, whereas fuzzy logic has reasoning capability but no training capability. To have both training and reasoning capabilities, the ANN and fuzzy logic are combined to form a fuzzy-nets system (Ralescu, 1994).

In this research, a self-organizing fuzzy-nets system (FNS) has been developed to generate a knowledge bank that can show the required cutting power to be on-line for a short length of time in an NC part program verifier.

Purpose of the Study

The purpose of this study was threefold: (1) to develop a fuzzy-nets system that can show the required cutting power of milling processes to be on-line for a short length of time in an NC verifier; (2) to evaluate whether the fuzzy rule bank was suitable to replace the machine database; and (3) to determine whether the fuzzy-nets system could become acceptable for industry through experimentation.

Hypotheses of the Study

Three hypotheses were generated to carry out the purpose of the study.

1. There is no significant difference in cutting power between the data calculated by the formula and the data calculated by the fuzzy-nets system.

   $$H_0\colon \mu_d = 0$$

2. There is no significant difference in cutting power between the data calculated by the fuzzy-nets system and the data collected from experimentation.

   $$H_0\colon \mu_d = 0$$

3. There is no significant difference in cutting power between the data calculated by the formula and the data collected from experimentation.

   $$H_0\colon \mu_d = 0$$

Assumptions of the Study

1. The geometry and condition of the tools used in the experiments will not change.
2. The speed (revolutions per minute) will not change once it is set.
3. The feed and depth of cut are as accurate as specified.

Limitations of the Study

1. The study was confined to end milling operations.
2. The workpiece materials were limited to aluminum.
3. The tools were high-speed steel end mills with 19.05 mm diameter and four flutes.

Procedure of the Study

The following procedure was followed in carrying out the study:

1. Identify the research problem.
2. Review the literature related to machining processes, neural networks, fuzzy logic, fuzzy-nets, and machining optimization..
3. Build fuzzy-nets training and testing system based on the theoretical data.
4. Test the fuzzy-nets system using simulation data.
5. Build fuzzy-nets training and testing system for experimental data.
6. Set up the experiment, including hardware and software.
7. Conduct the experiment and collect training and testing data for analysis.
8. Analyze the data.
9. Optimize or revise the software.

10. Repeat 7 - 9 until the results meet requirements.

11. Write a final report summary.

12. Take final oral examination.

Definitions of Terms

The following definitions were made to clarify the various terms used in this study.

Accelerometer: A device that measures the acceleration of a moving body and translates it into a corresponding electrical quantity.

Analog-To-Digital Converter (A/D, ADC): A hardware device that senses an analog signal and converts it to a representation in digital form.

Artificial Neural Networks (ANNs): Computational models that are composed of many nonlinear processing elements arranged in patterns similar to biological neuron networks (Tan, Quah, & Teh, 1996).

Assessment: The process of assessing the credibility of a simulation by performing different activities in each of the four assessment phases: preparation, planning, application, and evaluation (Knepell & Arangno, 1993).

BHN: The acronym of Brinell Hardness Number. Brinell hardness test is one of the earliest standardized methods of measuring hardness.

Computer Aided Design/ Computer Aided Manufacturing (CAD/CAM): A technology that uses computers to perform certain functions for design and production in which the database is shared by both functions.

Computer Numerical Control (CNC): An NC system that utilizes a dedicated, stored-program computer to perform some or all of the basic numerical control functions.

Conflict: Rules are conflicting if they have the same IF premise but a different THEN conclusion.

Credibility: The establishment of confidence in the validity of the model (Knepell & Arangno, 1993).

Database: Any collection of related data files.

Fuzzy Logic: Nonclassical logic which has more than two truth values.

Fuzzy Set: An extension of a classical (crisp) set that allows the elements to have partial membership.

Heuristic: Pertaining to exploratory methods of problem solving in which solutions are discovered by evaluation of the progress made toward the final result.

Knowledge base: An assembly of facts agreed upon by experts; the common knowledge they have acquired over years of work; and the rules of thumb (heuristics) that they apply to derive conclusions. A knowledge base is so organized and encoded that it can be interrogated via an expert system.

Linguistic Variable: A variable that takes on some linguistic values called terms. For example, the linguistic variable "speed" can take on the terms "slow," "medium," or "fast."

Machine Tool: A powered machine used to shape a part, typically by the action of a tool moving in relation to the workpiece.

Machining Parameter: A physical variable or condition in machining.

Machining Process: Any particular machining operation viewed as an indivisible activity for planning purposes.

Manufacturing: A series of interrelated activities and operations involving the design, materials selection, planning, production, quality assurance, management, and marketing of discrete consumer and durable goods.

Membership Function: A function which maps the elements of the universe onto numerical values in the interval [0, 1].

Metal-Remove Rate (MRR): A measurement of how fast material is removed from a workpiece.

Numerical Control (NC): A system in which actions are controlled by direct insertion of numerical data at some point. The system must automatically interpret at least some portion of this data.

NC (Part) Program: The numerical data required to produce a part.

On-Line: Operation where input data is fed directly from measuring devices into the CPU or MCU.

Orthogonal Cutting: The simplified cutting conditions used in the first stages of laboratory investigations. In orthogonal cutting, the tool edge is straight, normal to the cutting direction, and also normal to the feed direction.

Piezoelectric: The property of a material to generate a voltage when mechanical force is applied, or to produce a mechanical force when a voltage is applied, as in a piezoelectic crystal (Markus & Sclater, 1994).

Piezoelectric Transducer: A transducer whose output voltage is produced by deformation of a crystal or ceramic material that has piezoelectric properties.

Probe: A metal rod that projects into but is insulated from a waveguide or resonant cavity. It provides coupling to an external circuit for injection or extraction of energy.

Process: A systematic sequence of operations to produce a specific result.

Quartz: A natural or artificially grown piezoelectric crystal composed of silicon dioxide.

Real Time: Pertaining to computation performed while the related physical process is taking place so that results of the computation can be used in guiding the physical process.

Reality: An entity, situation, or system selected for analysis (Knepell & Arangno, 1993). (Also referred to as real-world system or entity.)

Resolution: The least interval that can be distinguished from one another.

Simulation: The representation of certain features of the behavior of a physical or abstract system by the behavior of another system, typically a physical or computer model.

Transducer: A device used to convert physical parameters such as temperature, pressure, weight, etc. into electrical signal.

Validation: Substantiation that a computer model, within its domain of applicability, possesses a satisfactory range of accuracy consistent with the intended application of the model (Knepell & Arangno, 1993).

Verification: Substantiation that the computer program implementation of a conceptual model is correct and performs as intended (Knepell & Arangno, 1993).

Workpiece: Any part in any stage of manufacture prior to its becoming a finished part.

Organization of the Dissertation

This dissertation consists of six chapters; each addresses a specific issue of the research. The following is an overview of each chapter of the remainder of the dissertation.

Chapter 2, Review of Literature, summarizes the literature review performed by the researcher. The subjects reviewed include machining processes, machining force and power calculations, neural networks, fuzzy set theory, fuzzy logic, sensors, assessment of the fuzzy-nets system, and optimization of machining processes. The main purpose of the review is to understand previous work and the trends that have emerged.

Chapter 3, Methodology, describes the learning procedure of the fuzzy-nets system. The procedure consists of five steps: (1) defining the fuzzy regions of the input and output spaces; (2) generating the fuzzy rules from given data pairs through experimentation; (3) resolving conflicting rules; (4) developing a combined fuzzy rule base; and (5) defuzzifying the output. The structure and implementation of fuzzy-nets system are also briefly addressed.

Chapter 4, Assessment of the Fuzzy-nets System, discusses the procedure to assess the fuzzy-nets system. Simulation and theoretical data are used in the evaluation process. The purpose of the assessment is to investigate and understand the end milling process, verify and validate the system, and reduce the experimental cost. The FNS is evaluated in terms of its range of accuracy with the intended applications of the model.

Chapter 5, Experimental Setup and Results, presents the procedure to conduct the actual experiment and data analysis. The results are presented and discussed.

Chapter 6, Conclusions and Recommendations, summarizes the first five chapters of the study, discusses the results of this research, and suggests questions for further research.

CHAPTER 2. REVIEW OF LITERATURE

The purpose of this study is to optimize CNC end milling operations performed on milling machines. The end milling operation uses a multi-toothed rotary tool to remove chips from a workpiece. The operator of a milling machine has to know how to use sensing devices and other utilities to monitor machining conditions, how to select the optimal machining parameters, and how to calculate the required cutting power.

Tools, machines, sensors, software, and other objects have been created to extend human abilities. Various areas of literature were reviewed to understand previous work on these objects and the trends that have emerged to further enhance human abilities.

Machining Processes

The manufacturing sectors of industries are the primary strength of an industrialized nation. Although increasingly larger segments of the population are employed in service industries, it is manufacturing that produces the wealth of the nation (Black, 1991). A manufacturing system is a collection or arrangement of operations and processes used to make desired products or components. Manufacturing processes can be classified as casting, forming, machining or material removal, joining, surface finishing, and heat treating.

Machining processes are the most important processes in a manufacturing system. In many cases, products from the primary forming processes must undergo further refinements in size and surface finish to meet their design specifications. To conform to the precise tolerances, the removal of small amounts of material is needed. Such secondary operations are called machining processes and they are usually performed on machine tools. Without machine tools, modern civilization could not exist. Today, every product known, from a paper clip to a space vehicle, is a product of machine tools. If not used directly in the manufacture of the product itself, machine tools are required to produce the machinery and equipment necessary for its processing. Machine tools determine how much a nation produces and how well its people live (Lascoe, Nelson, & Porter, 1973).

Machining processes can be classified under two main categories: chip producing and nonchip producing (nontraditional or chipless machining). There are two types of nontraditional processes. The first type is based on electrical phenomena, whereas the second type depends upon chemical dissolution (Niebel, Draper, & Wysk, 1989). The nontraditional processes have grown out of a need to machine ever more peculiar materials, often in a hardened condition and with very intricate designs (Lindberg, 1990).

There are four basic chip producing machining processes: turning, planing, drilling, and milling. The turning process produces surfaces of revolution by a combination of a single-point tool moving parallel to the axis of work rotation. The planing process generates a plane surface with a single-point tool by a combination of a reciprocating motion along one axis and a feed motion normal to that axis. The drilling process produces a hole in a workpiece by forcing a rotating drill against it. The milling process uses a multitoothed rotary tool to remove chips from a workpiece.

The milling process is performed on a milling machine which is the most versatile of all machine tools. Milling cutters have been developed to produce a multitude of contours in a finished part. As shown in Figure 2.1, there are two broad classifications of milling operations: peripheral (or plain) milling and end (or face) milling (Groover, 1996). The basic

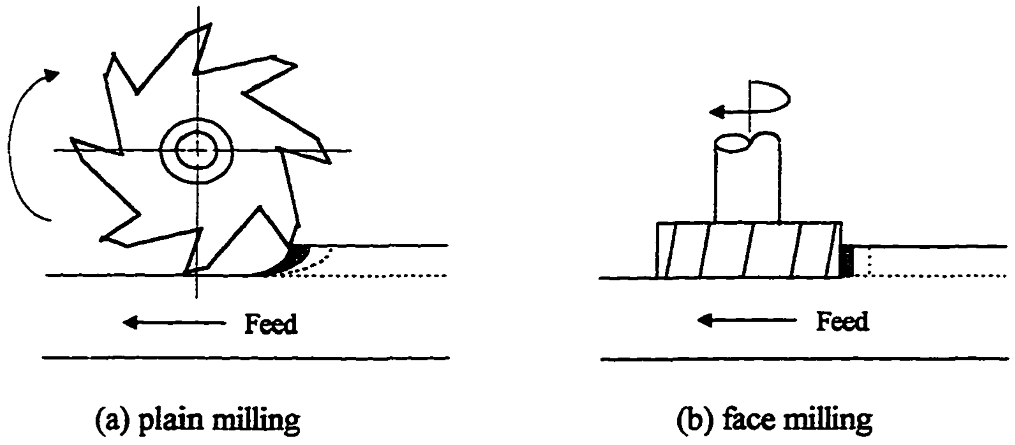
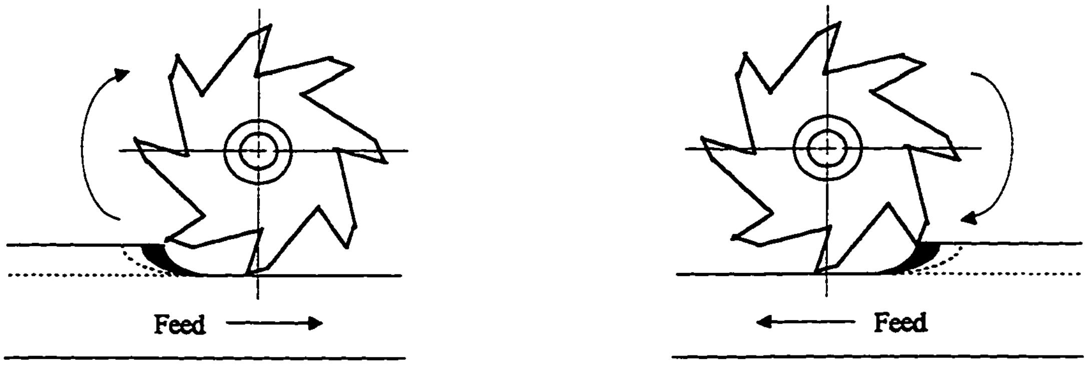


Figure 2.1. Two broad classifications of milling operations

plain-milling cutter produces a flat surface through the use of cutting teeth on its periphery that are parallel to the axis of rotation. An end mill has its cutting teeth located at the end as well as on its periphery and rotates around an axis that is normal to the surface being cut. It also produces a flat surface. A face-milling cutter is large in diameter and produces a flat surface.

Based on the direction of cutter rotation and workpiece feed, milling processes are classified into two basic categories: down milling and up milling. Figure 2.2 depicts the difference. When the cutter enters the material in the direction of feed, it is known as down milling; otherwise, it is known as up milling. Down milling and up milling produce opposite resultant forces. Down milling tends to push the workpiece against the table, whereas up milling tends to lift the workpiece off the table. In general, down milling produces the best results, but it may be disastrous to the cutting edge if used on metal that has a hard, scaly surface.



(a) up milling (b) down milling

Figure 2.2. Two basic categories of milling

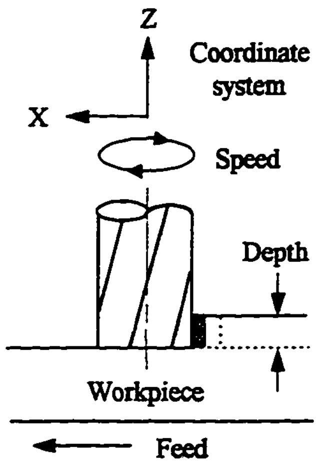
Force and Power Calculations

Machining force and power requirements are valuable in the selection of machining processes and machining parameters. Figure 2.3 shows the end milling operation and the machining parameters. Force and power are functions of machining parameters. When using the same machine, tool, and workpiece material, the greater the volume of material removed per unit time from a workpiece, the greater the power required.

Orthogonal cutting

The classical thin-zone mechanics model was first proposed by Merchant (1945). The mechanics were developed for orthogonal cutting. As shown in Figure 2.4, the forces applied against the chip are listed as follows (Amstead, Ostwald, & Begeman, 1987):

1. *Friction force F* - This force resists the flow of the chip along the rake face of the tool.
2. *Normal force to friction N* - This force is normal to the friction force.
3. *Shear force* F_s - This force causes shear deformation to occur in the shear plane.



Cutting Parameters:

1. Feed rate
2. Spindle speed
3. Depth of cut
4. Workpiece material
5. Tool material
6. Number of teeth of the tool

etc.

Figure 2.3. End milling operation and cutting parameters

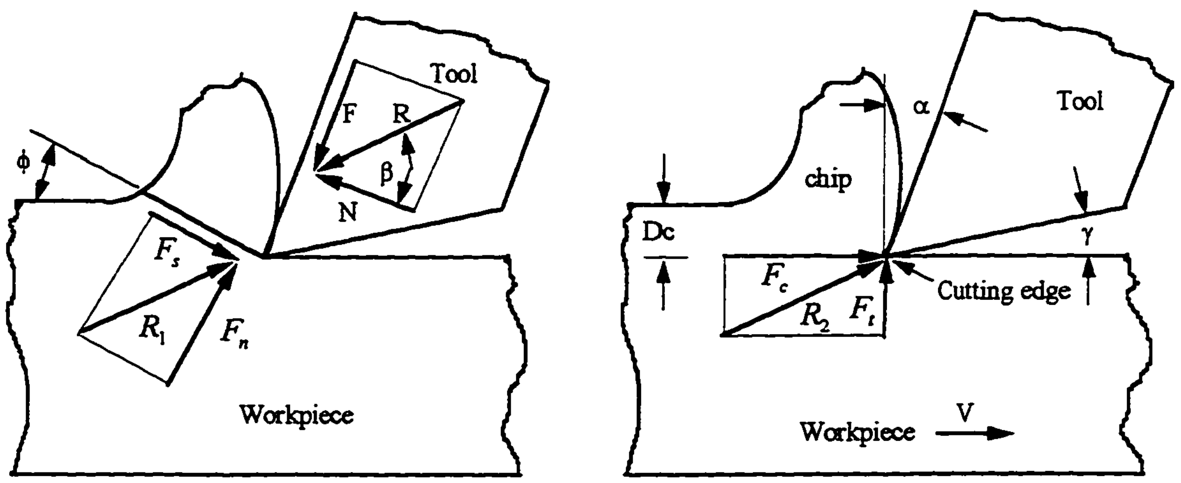


Figure 2.4. The geometry and forces in orthogonal cutting

4. *Normal force to shear* F_n - This force is normal to the shear force.

Where F and N are applied by the tool and F_s and F_n are acted by the workpiece, none of these force components can be directly measured in a machine operation. However, it is possible to measure two additional force components, F_c and F_t, with a dynamometer (Groover, 1996). These two components act on the tool:

1. *Cutting force* F_c - This force is in the direction of cutting, the same direction as the cutting speed V.
2. *Thrust force* F_t - This force is perpendicular to the cutting force.

Equations can be derived to relate the four forces to the two forces that can be measured. Using the force diagram in Figure 2.5, the following relationships can be defined:

$$F = F_c \bullet \sin\alpha + F_t \bullet \cos\alpha \tag{2.1}$$

$$N = F_c \bullet \cos\alpha - F_t \bullet \sin\alpha \tag{2.2}$$

$$F_s = F_c \bullet \cos\phi - F_t \bullet \sin\phi \tag{2.3}$$

$$F_n = F_c \bullet \sin\phi + F_t \bullet \cos\phi \tag{2.4}$$

where α is the rake angle, ϕ is the shear plane angle.

Various quantities can be determined in the force diagram. The forces F and N applied against the chip by the tool can be used to define the coefficient of friction μ between the tool and the chip:

$$\mu = F / N \tag{2.5}$$

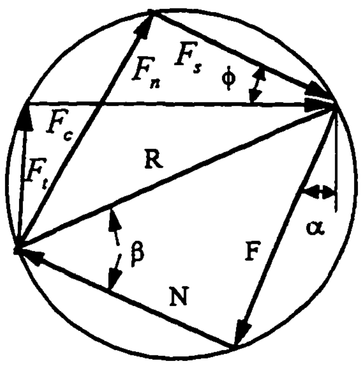


Figure 2.5. Relationship between forces in orthogonal cutting

$\mu = \tan\beta$ (2.6)

where β is the friction angle.

Based on the shear force, the shear stress τ, acting along the shear plane between the workpiece and the chip, can be defined as:

$\tau = F_s \,/\, A_s$ (2.7)

where A_s = area of the shear plane.

The shear plane area can be calculated by the equation:

$A_s = \mathrm{Dc} \bullet \mathrm{w} \,/\, \sin\phi$ (2.8)

where w is the width of the cutter.

One important relationship in metal cutting was derived by Merchant (1945) who expressed the shear stress in the following form by combining Equations 2.3, 2.7, and 2.8:

$\tau = (F_c \bullet \cos\phi - F_t \bullet \sin\phi) \,/\, (\mathrm{Dc} \bullet \mathrm{w} \,/\, \sin\phi)$ (2.9)

The shear plane angle ϕ can be determined by taking the derivative of the shear stress τ in Equation 2.9 with respect to ϕ and setting the derivative to zero. Solving for ϕ, we get the following relationship known as the Merchant equation:

$\phi = 45 + \alpha / 2 - \beta / 2$ (2.10)

The Merchant equation defines the relationship between rake angle, tool-chip friction, and shear plane angle. An increase in the rake angle and/or a decrease in the friction angle will cause the shear plane angle to increase. A higher shear plane angle results in lower cutting energy and cutting temperature.

The orthogonal cutting model can be used to approximate turning and certain other single-point machining operations as long as the feed is small relative to the depth of cut.

Thus, most of the cutting will take place in the direction of the feed, and cutting on the nose of the tool will be negligible (Groover, 1996).

Machining power requirements

As shown in Figure 2.6, the typical cutting force system in an oblique chip formation process has three force components (Amstead et al., 1987):

1. F_c = primary cutting force acting in the direction of the cutting speed. This force is generally the largest force and accounts for 99% of the power required by the process (DeGarmo, Black, & Kohser, 1988).
2. F_f = feed force acting the direction of the tool feed.
3. F_r = radial (or thrust) force acting perpendicular to the machined surface. The feed force and the radial force are negligible because the velocities of both components are usually small compared to cutting speeds.

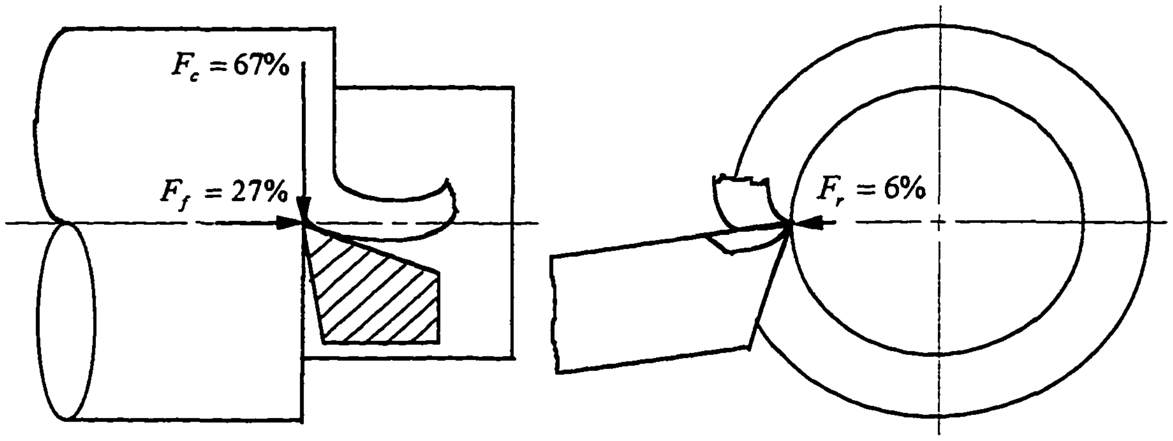


Figure 2.6. Distribution of forces acting on a single-point cutting tool

In orthogonal cutting, as was shown in Figure 2.4, the resultant force, R_2, applied to the chip by the tool lies in a plane normal to the tool cutting edge. F_c is the major cutting force, F_t is the thrust force, V is the cutting speed, and γ is the normal clearance angle.

The cutting energy per unit time, P_c or cutting power, can be calculated by the equation

$$P_c = F_c V \tag{2.11}$$

where P_c = cutting power, N-m/s or watts (or ft-lb/min); F_c = Newton (or lb); and V = m/s (or ft/m). The English units can be converted to horsepower using the constant 33,000 (ft-lb/min)/hp

$$\mathrm{HP} = F_c \mathrm{V} / 33{,}000 \tag{2.12}$$

where HP = cutting horsepower, hp.

The unit or specific horsepower concept (UHP) is often used to calculate the required cutting power. The unit horsepower is defined as

$$\mathrm{UHP} = \mathrm{HP} / \mathrm{MRR} \tag{2.13}$$

where MRR = material-removal rate, $in.^3$/min.

In the end milling operation, MRR is calculated by the equation

$$\mathrm{MRR} = \mathrm{W} \bullet \mathrm{H} \bullet \mathrm{Fr} \tag{2.14}$$

where W = width of cut, inches; H = depth of cut, inches; and Fr = feed rate, inches/min.

The unit horsepower can be expressed as the unit power U, also known as specific energy. The specific energy is determined by

$$\mathrm{U} = P_c / \mathrm{MRR} \tag{2.15}$$

where U = specific energy, N-m/mm^3 (in.-lb/$in.^3$); P_c = cutting power, N-m/s or watts (or ft-lb/min); and MRR = material-removal rate, mm^3/s (or $in.^3$/min).

Table 2.1 shows a listing of unit horsepower and specific energy values for selected workpiece materials (Groover, 1996), using sharp cutting tools and depth of cut Dc = 0.25 mm (0.010 in.).

Table 2.1. Unit horsepower and specific energy values

Material	Hardness Brinell (BHN)	Unit Horsepower (UHP) hp/($in.^3$/min)	Specific Energy (U) N-m/mm^3	in.-lb/$in.^3$
Carbon steel	150 - 200	0.6	1.655	240000
	201 - 250	0.8	2.206	320000
	251 - 300	1.0	2.758	400000
Aluminum alloys	100 - 150	0.3	0.827	120000

The specific horsepower with other factors can be used to estimate the motor horsepower required to perform a machining operation. The motor horsepower is determined by the equation

$$HP_m = \text{UHP} \bullet \text{MRR} \bullet \text{FCF} \bullet \text{WCF} / \text{E} \qquad (2.16)$$

where HP_m = motor horsepower, hp; UHP = unit horse power, hp/($in.^3$/min); MRR = material-removal rate, $in.^3$/min; FCF = feed correction factor; WCF = tool wear correction factor; and E = Efficiency of the machine. The efficiency factor accounts for the power required for friction and inertia in the machine and drive moving parts. Table 2.2 shows some feed correction factors. Correction factors may also be used to account for variations in cutting speed and rake angle.

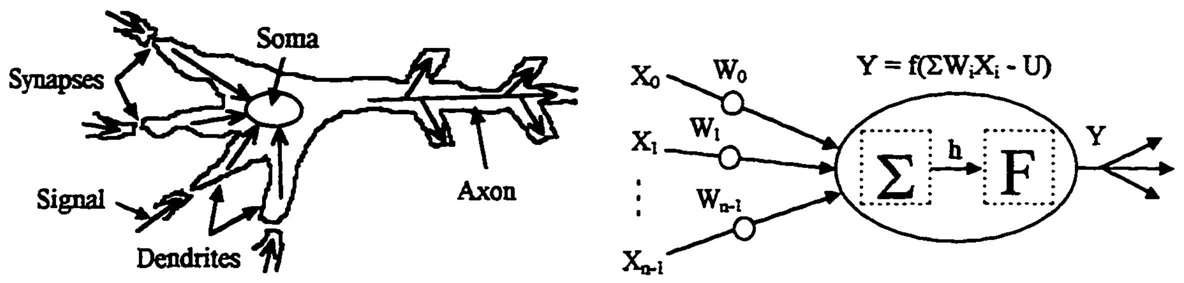
Table 2.2. Feed correction factors for unit horsepower and specific energy

Feed (mmpt)	0.025	0.075	0.125	0.175	0.225	0.275	0.325
FCF	1.6	1.4	1.25	1.18	1.06	0.95	0.92

Neural Networks

Artificial neural network models, or simply neural networks, are composed of many nonlinear computational elements (nodes) operating in parallel and arranged in patterns similar to biological neural networks (Lippmann, 1987). The brain is a major unit of the human nervous system. The human brain, containing billions of interconnected neurons, is a complex computing system capable of thinking, remembering, and solving problems.

A neuron is a special cell that processes information. It is composed of a cell body, or soma, and two types of branches: the axon and the dendrites (Jain, Mao, & Mohiuddin, 1996). A sketch of biological (real) and artificial neurons is shown in Figure 2.7. If the combined signal from all the dendrites is strong enough, the neuron fires, producing an output signal along the axon (Uhrig, 1995). The axon splits up and connects to thousands of dendrites of other neurons through synapses. The synapses are the basic memory units of the brain.



(a) biological neuron (b) artificial neuron

Figure 2.7. A sketch of neurons

Artificial neurons attempt to simulate the structure and function of real neurons. However, artificial neuron models are not exactly constrained by real neurons and are based only loosely on them. This stems from the following facts (Cichocki & Unbehauen, 1993):

1. We do not completely understand the behavior of complex biological nervous systems which are very complex.
2. Only part of the behavior of real neurons is essential to their information processing capability and part of the behavior builds up irrelevant side effects.
3. From a technical implementation point of view it will probably be impossible and also inefficient to simulate the full behavior of real neurons.
4. Artificial neural networks are designed in order to realize very specific computational problems and their architecture and features depend on the problems to be solved. (p. 41)

Network models

Neural network models are specified by the network topology, node characteristics, and learning rules (Lippmann, 1987). These rules specify an initial set of weights and indicate how the weights should be adapted during use to improve performance. The simplest node sums N weighted inputs and passes the result through a nonlinearity, activation function, as was shown in Figure 2.7 (b). Three representative activation functions are shown in Figure 2.8.

As shown in Figure 2.9, based on the topology, neural networks can be grouped into

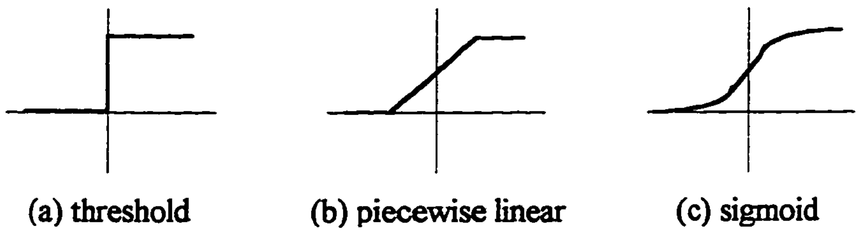


Figure 2.8. Three representative activation functions

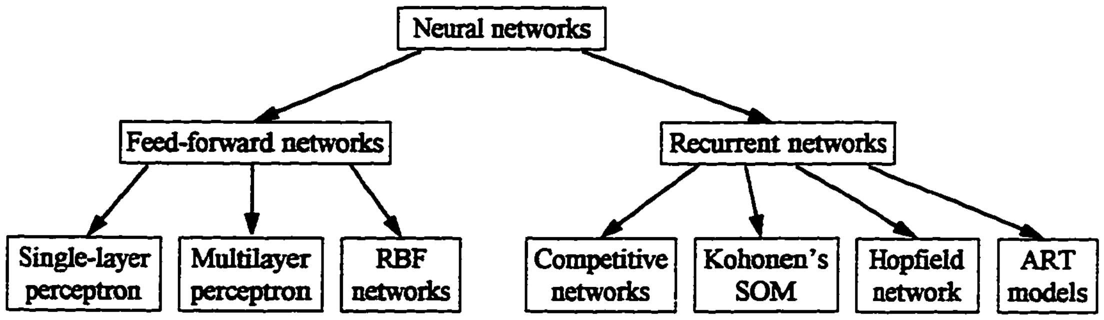


Figure 2.9. A taxonomy of network topologies

two categories (Jain et al., 1996): (1) feed-forward networks, in which graphs have no loops, and (2) recurrent (or feedback) networks, in which loops occur.

Different topologies yield different network behaviors. Generally speaking, feed-forward networks are static, that is, they produce only one set of output values from a given input. On the other hand, recurrent networks are dynamic. When a new input pattern is presented, the neuron outputs are computed. The inputs to each neuron are then modified by feedback signal and lead the network to enter a new state.

The multilayer perceptron is the most common family of feed-forward networks. Figure 2.10 shows a three-layer perceptron with two layers of hidden nodes that are not directly connected to the input and output nodes.

Learning

Learning ability is a fundamental trait of intelligence. Each network topology needs an appropriate learning process. A learning process can be viewed as the problem of updating the network topology and connection weights so that the network can efficiently perform a specific task (Jain et al., 1996).

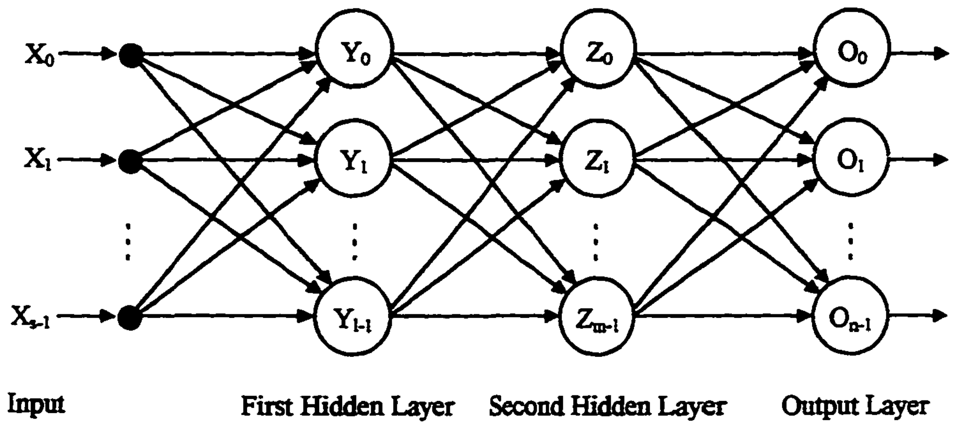


Figure 2.10. A three-layer perceptron network

To understand a learning process, one must know the learning rules and the learning paradigm. The learning rules specify how network weights are updated. A learning algorithm refers to a procedure in which rules are used for adjusting the weights. The learning paradigm is the model of the learning environment in which a neural network operates.

There are three main learning paradigms: supervised, unsupervised, and hybrid. Supervised learning requires a desired output for every input pattern. A system then compares the actual output with the desired output and converts the difference into an adjustment of the connection weights. In contrast, unsupervised learning classifies input patterns to derive the results by self-organization (Chu, 1993). Hybrid learning combines supervised and unsupervised learning.

Characteristics of neural networks

The characteristics that make neural network systems different from traditional computing and artificial intelligence are listed as follows (Uhrig, 1995):

1. *Learn by examples*. The ability of neural networks to learn from examples makes them attractive for applications in domains where explicit knowledge is not available. Instead of following a set of rules specified by human experts, neural networks learn underlying rules, like input-output relationships, from representative examples.
2. *Inherent parallelism*. Both the structural and processing sequences are parallel in neural networks. Computation is performed simultaneously over more than one node.
3. *Distributed associative memory*. Different from the traditional Von Neumann computing, the storage of an information unit is distributed across all memory units in the network. Associative memory, as the name implies, can be accessed by their content. When the trained network is presented with a partial input or distorted content, the network will choose the closest match to the input in the memory and generate a desired output.
4. *Fault tolerance*. The performance of the network only changes slightly if some processing elements are destroyed or disabled. This is because the information is distributed throughout the memory units of the network.
5. *Pattern recognition*. Neural networks have the ability to match a large amount of input information simultaneously and then generate a categorical output.

Applications of neural networks

Although neural computing is still at an early stage of development, the results have been impressive. Neural networks have been used for solving seven classes of problems: pattern classification, clustering (categorization), function approximation, prediction (forecasting), optimization, retrieval by content, and control (Jain et al., 1996). Examples of neural network applications currently in practice are handwriting and speech recognition, financial analysis, prediction of passenger demands and seat allocation, motor control, and voltage control (Widrow, Rumelhart, & Lehr, 1994). Future applications appear unlimited, but much development work remains to be done. The difficulty in achieving the potential of neural networks lies in the limited understanding of how the human brain functions (Badiru, 1992).

Fuzzy Set Theory

For a long time, philosophers have been conscious of the fact that any introduction of exactness is artificial and forced (Novak, 1989). The classes of objects encountered in the real world usually do not have precisely defined criteria of membership. To deal with these ill-defined classes, Zadeh (1965) introduced the fuzzy set theory. However, in the background one can see a hidden wish to improve the relationship between humanity and the computer (Terano, Asai, & Sugeno, 1992). Since its inception, fuzzy set theory has been applied to a wide variety of fields such as psychology, economics, engineering, law, medicine, decision-analysis, information retrieval, and artificial intelligence. A new trend is to combine fuzzy logic with neural networks whose learning capabilities allow one to tune membership functions more precisely and to eliminate useless rules (Dorf & Kusiak, 1994).

Fuzzy set

A fuzzy set is an extension of a classical (crisp) set. A classical set X is equated with its characteristic function

$$\mu_X : U \to \{0,1\} \tag{2.17}$$

which associates with each element of a universe of discourse U either 1 (full-membership) or 0 (nonmembership), i.e., $\mu(x) = 1$ if $x \in X$, $\mu(x) = 0$, otherwise. A fuzzy set allows its elements to have partial membership. The degree (grade) to which the generic element x belongs to F is characterized by a membership function

$$\mu_F : U \rightarrow [0,1] \tag{2.18}$$

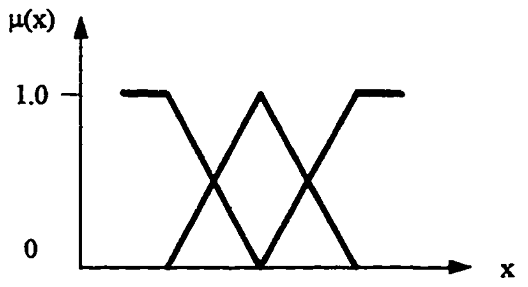
which takes on the values from the whole interval. Thus, a fuzzy set F in U may be represented by the set of ordered pairs:

$$F = \{(x, \mu_F(x)) \mid x \in U\} \tag{2.19}$$

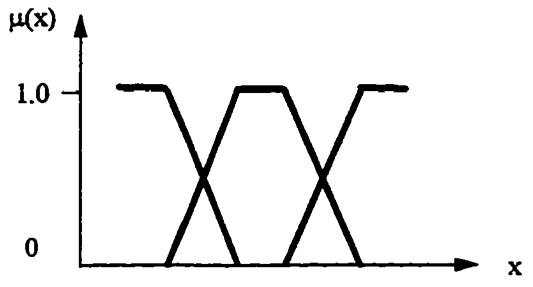
where x is a generic element, and $\mu_F(x)$ is the membership grade of x in the fuzzy set F. If U $= \{x_1, x_2, \ldots, x_n\}$, the pair $(x, \mu_F(x))$ is usually denoted by $\mu_F(x)/x$ and the fuzzy set is written as

$$F = \mu_F(x_1)/x_1 + \mu_F(x_2)/x_2 + \ldots + \mu_F(x_n)/x_n = \sum_{i=1}^{n} \mu_F(x_i)/x_i \tag{2.20}$$

where "+" and "Σ" are in the set-theoretic sense.



(a) triangular



(b) trapezoidal

Figure 2.11. Membership functions

Triangular and trapezoidal membership functions, as shown in Figure 2.11, are most frequently used by fuzzy engineers. Piecewise linear functions are easier to process with a computer. In principle, it is possible to use any type of membership functions (Kruse, Gebhardt, & Klawonn, 1994).

Basic operations of fuzzy sets

As in the classical (nonfuzzy) set theory, the basic operations in fuzzy set theory are complement, union, and intersection. Let μ_F and μ_G be the membership functions denoting the fuzzy set F and G, respectively. The following relations between two fuzzy sets and definitions of operations were originally proposed by Zadeh (1965):

- F is contained in G, $F \subseteq G$, iff $\mu_F \leq \mu_G$.
- F is equal to G, F = G, iff $\mu_F = \mu_G$.
- F is the complement of G, F = -G, iff $\mu_F = 1 - \mu_G$.
- The union of F and G, $F \cup G$, is such that $\mu_{F \cup G} = \max(\mu_F, \mu_G)$.
- The intersection of F and G, $F \cap G$, is such that $\mu_{F \cap G} = \min(\mu_F, \mu_G)$.

Linguistic approach

Zadeh (1973) formulated the principle of incompatibility which basically states that as the complexity of a problem increases, one's ability to analyze it in precise and yet relevant terms diminishes. Natural language is a powerful tool allowing human a comprehensive but imprecise description of reality. Therefore, Zadeh proposed a new approach, the linguistic approach, to the analysis of complex problems and systems.

As in conventional approaches in which the basic entity is a variable taking on some values such as 2 or 5.6, in the linguistic approach we have a linguistic (fuzzy) variable "speed," which takes on some linguistic values called terms such as "slow," "medium," or "fast," which are in turn defined as appropriate fuzzy sets (i.e., these terms are semantically equivalent to some fuzzy sets). Formally, a linguistic variable is characterized by the quintuple (x, T(x), U, G, M), where x is the name of the variable, T(x) is its term set, U is a universe of discourse, G is a syntactic rule for generating the values of x, and M is a semantic rule for associating with each value its meaning (Zimmermann, 1991). For example, if the cutting of a milling process is interpreted as a linguistic variable, x = speed, then its term set T(speed) can be expressed as T(speed) = {slow, medium, fast}, where each term in T(speed) is characterized by a fuzzy set in a universe of discourse U = [0, 60] mpm (surface speed, meter per minute). The terms "slow", "medium", and "fast" may be interpreted as "a speed below about 20 mpm," "a speed close to 30 mpm," and "a speed above about 40 mpm," respectively. These three terms can be illustrated as fuzzy sets whose membership functions are shown in Figure 2.12.

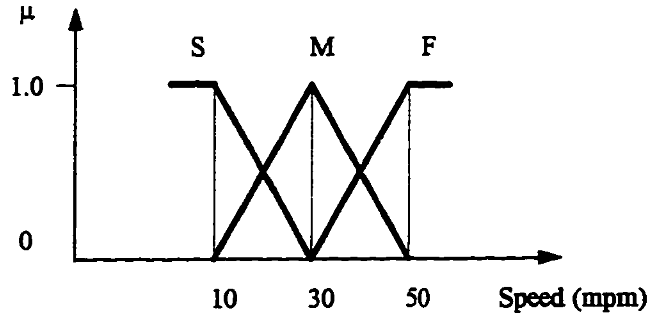


Figure 2.12. Membership functions representing the fuzzy sets small, medium, and fast

To design a fuzzy controller, the designer must identify the main control parameters and determine a term set T(x) which is at the right level of granularity for describing the values of each linguistic variable x (Berenji & Khedkar, 1992). In the milling process example, the term set T(speed) = {slow, medium, fast} may not be satisfactory in certain domains and require the use of a set of five terms {very slow, slow, medium, fast, very fast}.

Fuzzy Logic

A proposition is an assertion (statement). In classical logic, a proposition is either true or false. If a proposition is true, it has a truth value of true; otherwise, its truth value is false. A proposition variable denotes an arbitrary proposition with an unspecified true value. Propositions and proposition variables can be combined to form new assertions using logical connectives such as "NOT," "OR," "AND," "EQUALS," and "IMPLIES." Knowledge is represented by propositions and can be processed through reasoning by the application of various laws of logic including an appropriate rule of inference (Nguyen, Sugeno, Tong, & Yager, 1995). Knowledge processing may involve the following steps:

1. Simplify the knowledge base by applying various laws of logic.
2. Substitute into the knowledge base any new information including data and previous inferences.
3. Apply an appropriate rule of inference.

Fuzzy logic denotes the nonclassical logic which has more than two truth values. In classical logic, the "NOT," "OR," and "AND" operations (connectives) correspond to the classical set operations "complement," "union," and "intersection." Furthermore, the union of a set with the complement of a second set represents an "implication" of the first set by the second set. These logical operations have to be extended to fuzzy sets for use in fuzzy reasoning and fuzzy logic control.

In fuzzy logic control, the knowledge base typically consists of a set of "IF-THEN rules," where "implication" is the main connective used. The statement "P IMPLIES Q" ("P $\Rightarrow$ Q") is the same as "IF P THEN Q" and is false only when P is true and Q is false. The fuzzy conditional statements are used to characterize a relationship between linguistic variables. Using these statements, expert human operators can express the heuristic or the control knowledge that they use in controlling a process. For example,

1. IF (the speed is high) THEN (apply less force to the accelerator),
2. IF (higher speed is applied) THEN (the temperature of the tool tip will increase).

The first example shows how to use a fuzzy control rule to drive a car. The control rule sets forth the situations in which certain control actions should be taken. The second example shows how to use a IF-THEN rule to describe the behavior of a metal cutting process.

Fuzzy logic control system

In classical control, we have to specify a mathematical model of a system (process) that has to be controlled. It is often difficult or impossible to specify an accurate mathematical model of a process, especially a complex one. There should be an easier approach to control a process. An analogous example is that a person can ride a bicycle without knowledge of the existence of differential equations.

The biggest success using fuzzy systems in industrial and commercial applications has been achieved with fuzzy controllers. Used alternatively to classical control, fuzzy control is a method of defining non-linear table-based control systems, where the definition of the nonlinear transition function can be made without the need to specify each entry of the table individually (Kruse et al., 1994).

Figure 2.13 shows the architecture of a basic fuzzy (logic) control system (Kruse et al. 1994). The fuzzy control system consists of four major components: a fuzzification interface, a knowledge base, an inference mechanism, and a defuzzification interface. The fuzzy control procedure is executed in two phases (Nguyen et al., 1995). The first phase is to develop a fuzzy control algorithm according to the following four steps:

1. Develop a set of linguistic control rules.
2. Develop a set of discrete membership functions for process output variables and control input variables
3. Obtain the multidimensional array R_i of membership values for that rule by applying the fuzzy implication operation on each rule (i) in step 1 and using step 2.
4. Combine the relations R_i using fuzzy connectives to obtain the overall fuzzy rule base.

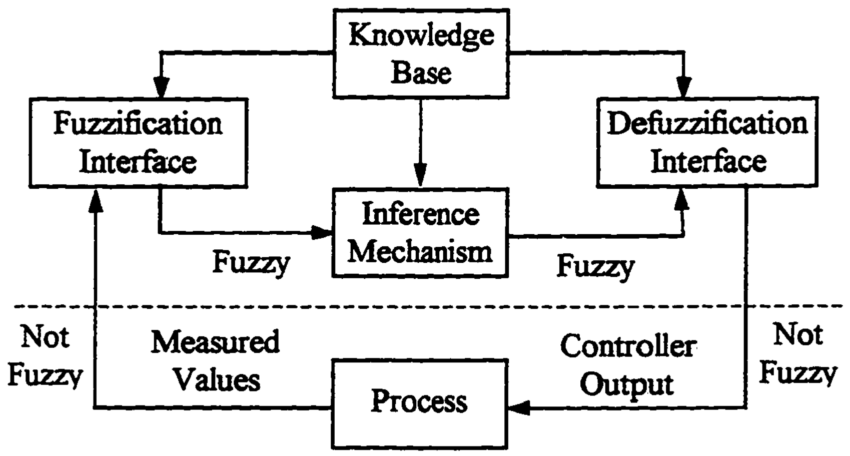


Figure 2.13. Architecture of a basic fuzzy logic control system

Then, control action may be determined in real time as follows:

1. Fuzzify the measured process variable values as fuzzy singletons.
2. Match the fuzzy measurements obtained in step 1 with the membership array of the fuzzy rule base using the compositional rule of inference.
3. Defuzzify the control inference obtained in step 2.

Development of fuzzy technology

As shown in Table 2.3, the general trend of fuzzy technology may be classified into four phases (Terano el al., 1992).

Table 2.3. Evolution of applications of fuzzy systems

	Phase 1	**phase 2 (present)**	**Phase 3**	**Phase 4**
Sub-stance	Industrial application of qualitative human knowledge	use of fuzzy logic to express macroscopic	Interpretation between man and computer through natural language	Intermediation among AI, neural networks, & human
Example	• Fuzzy control	• Fuzzy expert system • Non-engineering application (medicine, agriculture, management, society, ecology, etc.)	• Intelligent robot • Dialogue type decision support system	• Story Summarization • Human Interface • Translation • Support system for creative works

Sensors

A sensor or transducer is a device to detect, record, or measure a physical property. Sensory systems can be used to monitor a particular situation in the same way that a normal human being does in areas such as machining operations, tool conditions, machine conditions, and so on (Wild, 1994). To achieve greater quality and reliability in machining processes with minimal operator supervision depends to a large degree upon the development and implementation of automatic sensing techniques. These techniques are required to monitor the performance of machining processes and to compensate for uncertainties and irregularities of the work environment.

A dynamometer is a sensory system used for the measurement of forces acting on a tool. The dynamometer is a precise instrument for optimizing productivity. It is a piezoelectric force transducer. Piezoelectricity, discovered by Pierre and Jacques Curie in 1880, means pressure electricity (Allocca & Stuart, 1984). As shown in Figure 2.14, if a piezoelectric material is squeezed along a specified direction, an electric charge will be developed by the piezoelectric material. Of the numerous piezoelectric materials, quartz is by far the most suitable for measuring force because of its natural stability (Kistler, 1994). The piezoelectric properties of quartz are such that the crystals are sensitive to either pressure or shear forces. In this way, components of cutting force or torque are measured independently.

The piezoelectric force measuring system differs fundamentally from other methods. The forces acting on the quartz elements are directly converted into proportional electrical signals, and the resulting displacement amounts to only a few thousands of a millimeter. Consequently, quartz dynamometers are very rigid systems which offer high natural

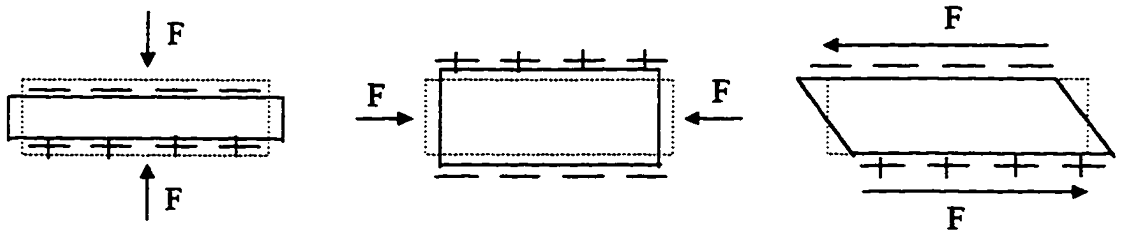


Figure 2.14. Different effects on a piezoelectric material

frequency, allowing precise measurements of very rapid events (Kistler, 1995). The development of three-component dynamometers for commercial use began in 1965. A three-component dynamometer consists of sensors with two shear quartz pairs (for F_x and F_y) and one pressure quartz pair (for F_z) assembled in a housing.

As it is impossible to measure cutting forces at the point of the tool and workpiece, the forces are measured away from the cutting point. Transducers and a platform are combined to measure one, two, or three forces and torques. A tool or workpiece is mounted on the platform.

Computer Simulation and Simulation Assessment

Simulation is the process by which understanding of the behavior of an existent (or to-be-constructed) physical (real) system is obtained by observing the behavior of a model representing the system (Kheir, 1988). The purposes of simulation include analysis, performance evaluation, tests of sensitivity, cost effectiveness, forecasting, safety, man-in-the-loop training, teaching, and decision making.

As shown in Figure 2.15, simulation is only one of several alternative ways to investigate the characteristics of a system (Law & Kelton, 1991). At one extreme, an analytic solution can be used for this purpose; however, in practice many real systems are too complicated to be modeled adequately by the analytic method. Sometimes, the system being modeled might be improperly distorted to fit a model amenable to the analytic solution, and one can wind up with the right solution for the wrong problem (Schriber, 1987). At the other

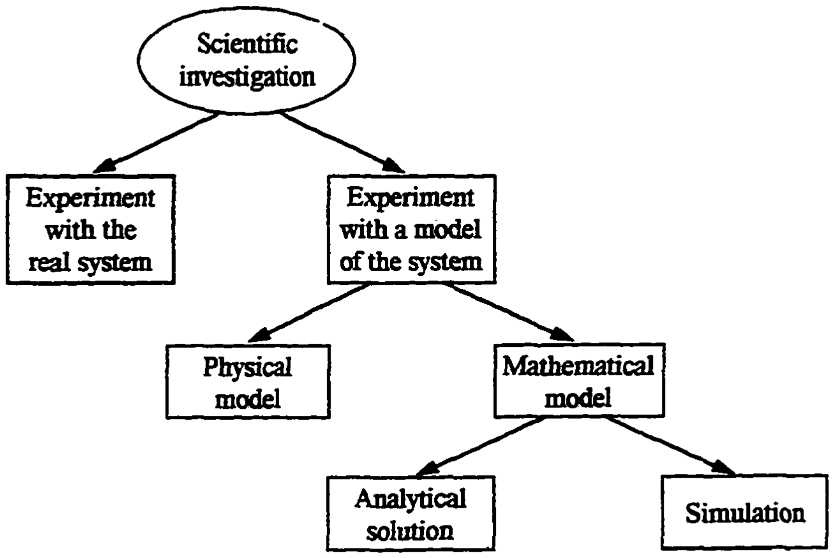


Figure 2.15. Ways to investigate the characteristics of a system

extreme, an experiment on the real system is possible in concept; however, the real system may not exist, so direct experiment is impossible. On the other hand, the real system may exist, but the experiment may be too expensive, time-consuming, and dangerous to perform.

Computer simulation combines the advantages and disadvantages of the analytic solution and real system experiment extremes. Law (1986) proposed a procedure to conduct a typical simulation study. The steps of the procedure are shown in Figure 2.16. The steps and their sequence may vary from study to study.

Application areas for simulation are numerous and diverse. Manufacturing processes and manufacturing systems represent an important application area for simulation. Our standard of living, in terms of material goods, depends primarily on our ability to manufacture products. Simulation provides an essential tool for improving productivity of these systems. It is used to address the following manufacturing issues (Schriber, 1987):

1. *The need for and the quantity of equipment and personnel.*

   Machines; carts, conveyors, pallets, fixtures, etc.; location and size of inventory buffers; evaluation of a change in product mix (impact of new product); evaluation of the effect of a new piece of equipment; and manpower requirements planning.

2. *Performance evaluation.*

   Throughput analysis, makespan (time in system for jobs) analysis, and bottleneck analysis.

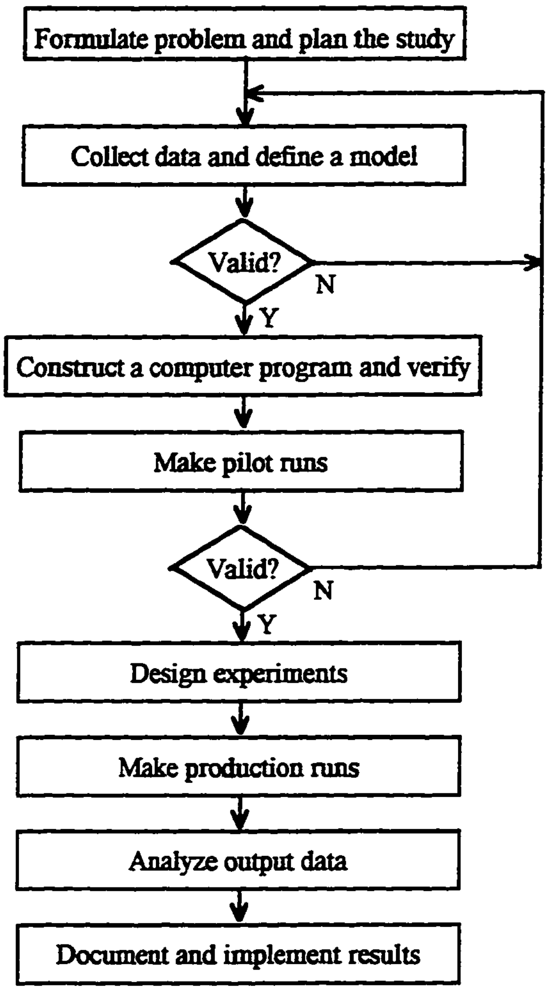

Formulate problem and plan the study
Collect data and define a model
Valid?
N
Y
Construct a computer program and verify
Make pilot runs
Valid?
N
Y
Design experiments
Make production runs
Analyze output data
Document and implement results

Figure 2.16. Steps in a simulation study

3. *Evaluation of operational procedures.*

Production scheduling; evaluation of policies for part or raw material inventory levels; evaluation of control strategies; reliability analysis (maintenance); and evaluation of quality control policies.

Simulation has established itself as a highly practical technique in problem solving. However, Knepell and Arangno (1993) stated that it is important to understand the following risks involved with simulation:

> A simulation may not adequately represent the real-world system. The data used to drive it may be inaccurate. It may be too difficult to model the operational environment or all the interactions that affect the real system. Output data may be flawed or subject to misinterpretation. Despite all their potential for saving money, simulations can be costly in terms of human effort and computer resource requirements. And of course, there are always questions about the credibility of the simulation tool and its output. (p. 1-1)

The credibility of a simulation model can be accomplished by systematically assessing the design, development, operation, and results of the model. Assessment is an integral part of modeling. Many assessment procedures can be incorporated into the development of a simulation to enhance the quality of the finished model.

In 1979, the Society for Computer Simulation Technical Committee on Model Credibility provided a framework for assessing simulations (Knepell & Arangno, 1993). The framework was expanded further by Robert Sargent (1991), as shown in Figure 2.17. Assessment activities can be derived from the framework. The fundamental building blocks of a simulation are the real-world problem entity being simulated, a conceptual model of that entity, and computer model of the conceptual model. The outer circle (conceptual model validity, software verification, and operational validity) along with the center (data validity)

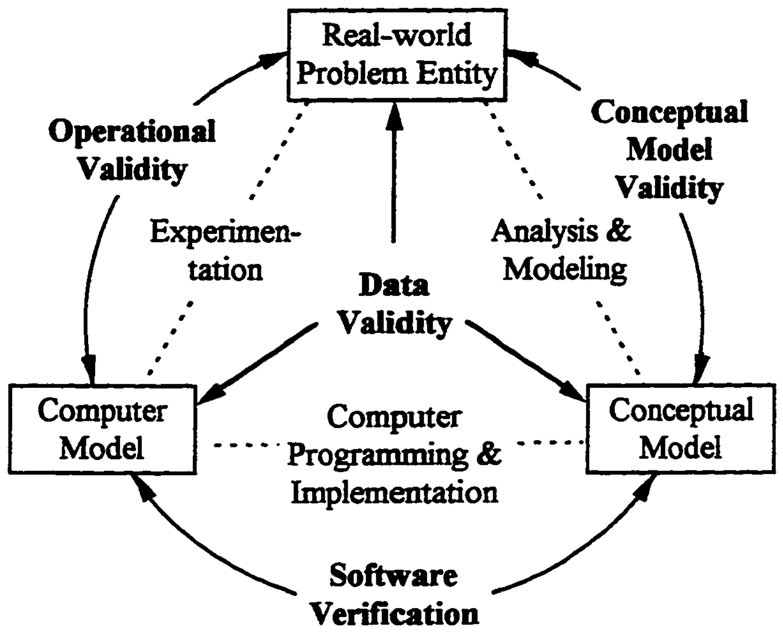


Figure 2.17. Sargent framework for model evaluation

validity, software verification, and operational validity) along with the center (data validity) are the technical processes that must be addressed to show that a model is credible. Assessment activities are spawned from each of the technical processes.

Optimization of Machining Processes

Manufacturing engineers have been attempting to develop a superior quality and highly productive operation in an unmanned manufacturing environment employing computer numerical control (CNC) machines. The existing computer-aided design/computer-aided manufacturing (CAD/CAM) systems for the programming of turning and milling processes are mainly geometry oriented and they do not offer optimization utilities. The optimization of these machining processes has been studied by many researchers, and some of the proposals are described below.

Chatter vibration is one of the most significant factors lowering the performance of a machine tool. Liao and Young (1996) proposed an on-line control method to suppress regenerative chatter during machining by regulating spindle speed. The dynamic cutting force signal collected from a dynamometer is passed through a low pass filter and then digitized. The digitized signal is converted to the corresponding power spectrum by the fast Fourier transform. A chatter frequency is identified when the intensity of a certain frequency, other than the spindle speed and tool rotating frequency, exceeds a critical value. Based on the chatter frequency, a new spindle speed is computed by keeping the phase between present and previous undulations to 90°. The new speed command is executed while the cutting proceeds. The results from simulation and experiments conducted in a CNC milling machine have shown that the chatter can be suppressed rapidly. The proposed strategy is very easy to implement and there is no need to alter any part of the machine tool.

The other important factor in metal machining is tool condition which exerts a strong influence on the surface finish and dimensional integrity of the workpiece and vibration levels of the machine tool. Rangwala and Dornfeld (1990) proposed a method to use neural networks to integrate information from multiple sensors to recognize the occurrence of tool wear in a turning operation. Leem, Dornfeld, and Dreyfus (1995) developed a customized neural network for sensor fusion of acoustic emission and force in on-line monitoring of tool wear. In a turning experiment, the customized neural network achieved high accuracy rates with robustness in the classifications of two and three levels of tool wear. Tool breakage is a severe tool failure which may result in workpiece and even machine damage. On-line methodologies for detecting tool breakage in a milling operation using neural networks were proposed by Ko, Cho, and Jung (1995); Tarng, Hseih, and Hwang (1994); and Tansel and McLaughlin (1993). In addition, Abdou and Yien (1995) conducted experiments to measure the cutting force and tool life in milling operations under dry conditions. A process optimization based on minimum production cost was applied to relate cutting force, tool life, and machinability criteria.

The unified mechanics of the cutting approach and modular software structure aimed at developing models for quantitative prediction of force components, torque, and power for practical machining operations were reviewed by Armarego and Deshpande (1993). This approach has been used to develop three predictive models and computer programs common to each of the following milling operations: peripheral milling, end milling, and slotting.

Bouzakis, Efstathiou, and Paraskevopoulou (1992) presented a computer supported procedure for the optimization of the cutting speed and feed rate in 3-axis milling. The input to that procedure is obtained from the NC code of a part. The tool motions, derived from the NC code, are grouped in subprocesses. The optimal feed rate and the cutting speed values along the tool path are then calculated using the developed models. These optimal cutting conditions are then automatically implemented into the NC code.

Mesquita, Krasteva, and Doytchinov (1995) also presented a model and an interactive program system (MECCANO2) for the multiple criteria selection of optimal machining conditions in multipass turning. Optimization is done for the most important machining conditions: cutting speed, feed, and depth of cut, with respect to various combinations of the criteria, minimum unit production cost, minimum unit production time, and minimum number of passes. The user can specify the values of the model parameters, criterion weights, and desired tool life. MECCANO2 provides graphical presentation of results which makes it very suitable for application in an educational environment.

Tarng, Cheng, and Kao (1995) developed a computer-aided cutting simulation system to model three-dimensional NC end milling operations. In this system, the varying axial and radial depths of cut in an NC tool path were identified by a modeling system using constructive solid geometry and boundary representation techniques. Once the axial and radial depths of cut are calculated, the dynamic cutting force is calculated from an end milling process model. As a result, the cutting performance in three-dimensional NC end milling operations can be verified and optimized.

Fang and Jawahir (1994) presented a new methodology for predicting total machining performance (TMP), i.e., surface finish, tool-wear rate, dimensional accuracy, cutting power, and chip breakability. In this new methodology, a series of fuzzy-set models were developed to give quantitative assessments of the TMP for any given set of input conditions including work material properties, tool geometries, chip breaker types, and cutting conditions.

Sakai and Ohkusa described some basic ideas that are necessary in dealing with automatic regulation of the cutting status in turning process automation (Sugeno, 1985). A human operator's manner and characteristics in manual handling are analyzed. They attempted to introduce a human operator's ability in establishing the function of automated supervision. They developed a fuzzy control system to perform the automated supervision function.

Summary of the Review of Literature

The most important processes in a manufacturing system are machining processes, either chip-producing or nonchip-producing. Milling is a chip-producing machining process performed on a milling machine, whereas end milling is the operation of machining flat surfaces using an end mill.

Machining force and power requirements are valuable in the selection of machining processes and machining parameters. When using the same machine, tool, and workpiece material, the greater the volume of material removed per unit time from a workpiece, the greater the power required.

The classical thin-zone mechanics model was developed for orthogonal cutting. The cutting force and the thrust force act on the tool and can be measured with a special sensing device called a dynamometer. The orthogonal cutting model can be applied to certain single-point machining operations as long as the feed is small relative to the depth of cut. The unit or specific horsepower concept is used often to calculate the required cutting power. The unit horsepower can be expressed as the unit power or specific energy.

The relationship between the cutting power and machining parameters is nonlinear. Sensors, simulation, artificial neural networks, and fuzzy logic systems are utilized to optimize machining processes.

Artificial neural networks (ANNs) are composed of many nonlinear computational elements operating in parallel and arranged in patterns similar to biological neural networks. The ANNs can be grouped broadly into feed-forward and recurrent (or feedback) networks.

In classical logic, a proposition is either true or false. Fuzzy logic, the extension of classical logic, has more than two truth values. A fuzzy logic control system consists of four major components: a fuzzification interface, a knowledge base, an inference mechanism, and a defuzzification interface. To have both the training and reasoning capabilities, the ANN and fuzzy logic are combined to form a fuzzy-nets system (FNS).

A sensor or transducer is a device to detect, record, or measure a physical property. To achieve greater quality and reliability in machining processes, automatic sensing techniques are required to monitor the performance of the processes and to compensate for uncertainties and irregularities of the work environment. A dynamometer is a piezoelectric force transducer. If a piezoelectric material is squeezed along a specified direction, an electric charge will be developed by the piezoelectric material. The forces acting on the quartz elements are directly converted into proportional electrical signals.

Simulation has established itself as a highly practical technique in problem solving. The purposes of simulation include analysis, performance evaluation, tests of sensitivity, cost effectiveness, forecasting, safety, man-in-the-loop training, teaching, and decision making. However, a simulation may not adequately represent the real-world system. The credibility of a simulation model can be accomplished by systematically assessing the design, development, operation, and results of the model.

The techniques described previously were applied by many researchers to monitor and optimize machining processes. The factors monitored and optimized were: chatter vibration, tool wear, tool breakage, chip breakability, cutting power, machining parameters, surface finish, dimension accuracy. Computer simulation also was used to investigate the characteristics of machining.

CHAPTER 3. METHODOLOGY

This chapter briefly discusses the architecture of the NC part program verifier developed in this research. The verifier comprises four major components: the character recognition system, the word recognition system, the fuzzy-nets system (FNS), and the tool path viewer. The focus of this study is on the FNS which is used to help programmers and process planners select optimal machining parameters. In this chapter, the basics of the FNS are described in detail, and the structure and implementation of the FNS model are then depicted.

The NC Part Program Verifier

The NC program has a particular structure that the controller can understand and it must follow a specific syntax. Writing NC programs can be an error-prone process. It is difficult to debug a program of any sizable length. A computer-assisted part programming language can be used to perform tedious and/or complex calculations necessary to prepare the program. However, even with computer-assisted part programming, some of the important commands of a program could be missing, incomplete, or incorrect. For example: if mistakes are made in spindle speed, tool size, fixture offset, depth of cut, feed rate, and/or tool path, they could cause damage to the tools and the machine and injuries to the operator and other people. The tool path should also be checked for errors before running the part program on the machine.

As shown in Figure 3.1, the NC part program verification system consists of four major components: (1) the character recognition system; (2) the word recognition system; (3) the fuzzy-nets system; and (4) the tool path viewer. The input to the verification system is a part program and the output is a correct part program.

Learning Procedure of the Fuzzy-Nets System

The focus of this study is on the fuzzy-nets system. The fuzzy-nets system is formed by combining artificial neural networks (ANN) and fuzzy logic (Ralescu, 1994). This new approach takes advantages of learning capability from the ANN and reasoning capability from fuzzy logic. A five-layer fuzzy-nets structure developed by Chen (1996) is shown in Figure 3.2. The fuzzy-nets system requires a self-learning procedure consisting of the following steps (Chen, 1996):

Step 1: Define the fuzzy regions of the input and output spaces.

The purpose of the fuzzy-nets model is to verify milling operations. The inputs to the model are speed, feed, and depth of cut. The input feature vector is defined as [Sc, Fr, Dc].

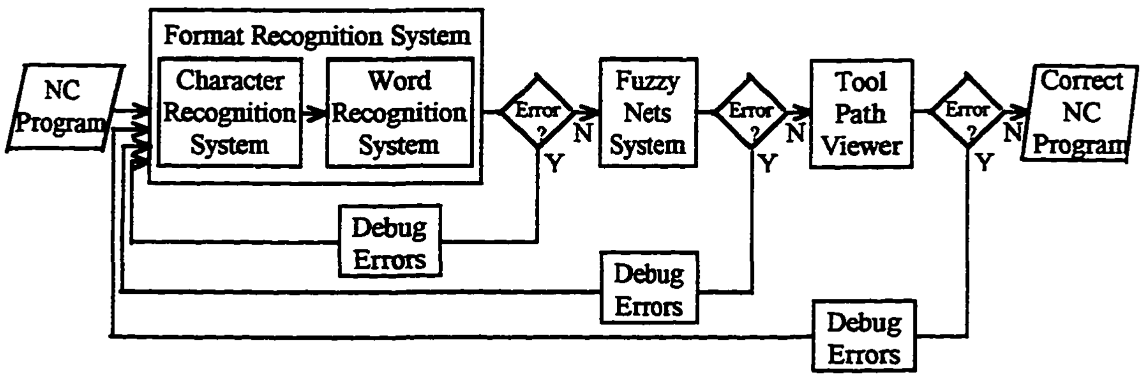
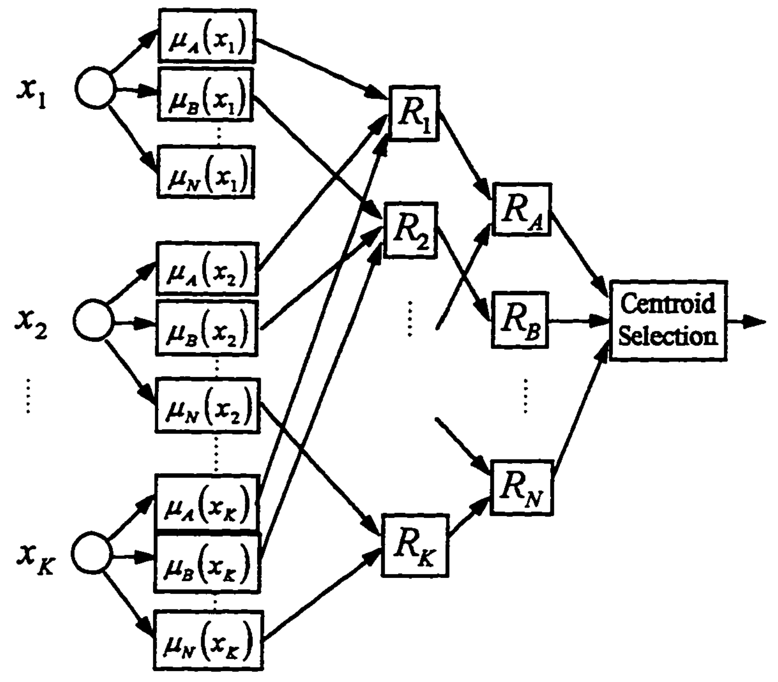


Figure 3.1. Architecture of the NC part program verifier



Layer 1 Layer 2 Layer 3 Layer 4 Layer 5

Figure 3.2. The structure of the five-layer fuzzy-nets classifier

The spread of an input feature I is calculated by the equation

$$s = \frac{X_{\max} - X_{\min}}{N-1} \tag{3.1}$$

where $X_{\max}$ = the maximum value of the input feature I; $X_{\min}$ = the minimum value of the input feature I; and N = the number of regions of the input feature I.

The center of each linguistic variable is determined by

$$(X_{\min}, X_{\min} + s, \ldots, X_{\min} + s * (N - 2), X_{\max}) \tag{3.2}$$

For example, the surface speed (Sc) is considered to be from 18 mpm to 58 mpm (surface speed, meter per minute). The shape of each membership function is triangular and the width of the spread of each triangular function is identical. Assume that the three regions of each variable are defined; then, the spread of the surface speed, denoted as s(Sc), is shown to be 20 mpm. Consequently, the center points of each linguistic variable (S, M, L) of Sc are (18, 38, 58), as shown in Figure 3.3.

Step 2: Generate the fuzzy rules from given data pairs through experimentation.

There are two ways to obtain the training data; one is from machining handbooks, while the other is from experiments. The data will consist of the following elements:

$$\left[Sc^{(i)}, Fr^{(i)}, Dc^{(i)}, Pc^{(i)}, \mu_{d^{(i)}}^{(i)}\right] \tag{3.3}$$

where i denotes the number of the training data set, and μ_d denotes a degree of this data set assigned by a human expert. The degree (μ_d) represents the usefulness of the data pair.

The degrees of each input and output feature are determined in different regions. The function used to calculate the degrees is given as:

$$\mu_{X_c X_s}(x_i) = \begin{cases} 1 - \dfrac{X_c - x_i}{X_s}, & x_i \in [X_c - X_s, X_c] \\ 1 - \dfrac{x_i - X_c}{X_s}, & x_i \in [X_c, X_c + X_s] \\ 0, & \textit{otherwise} \end{cases} \tag{3.4}$$

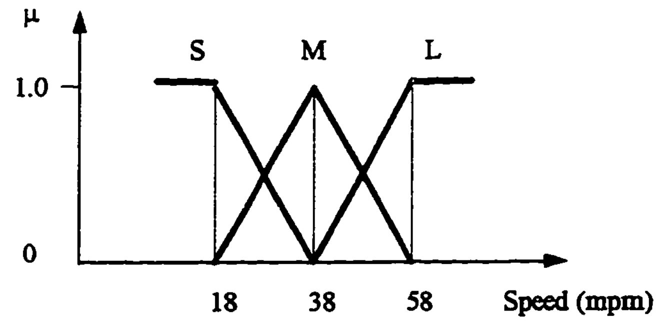


Figure 3.3. Center points of linguistic variables small, medium, and large

where X_c and X_s indicate the center point and the spread width of the linguistic level X, respectively.

For example, if the input vector [Sc, Fr, Dc] of an experiment cut has been set at [49 mpm, 0.14 mmpt, 1.7 mm] and the Pc value obtained from the A/D board was 240 W (Watts), as shown in Figure 3.4, the degrees of input and output values are as follows:

if x(Sc) = 49 mpm and X_S(Sc) = 10 mpm; then $\mu_{46,10}(49) = 0.7$ and $\mu_{56,10}(49) = 0.3$ (i.e., the Sc input value 49 mpm has degree 0.7 in M and degree 0.3 in L).

Similarly, the Fr input value 0.14 mmpt has degree 0.2 in S and degree 0.8 in M, the

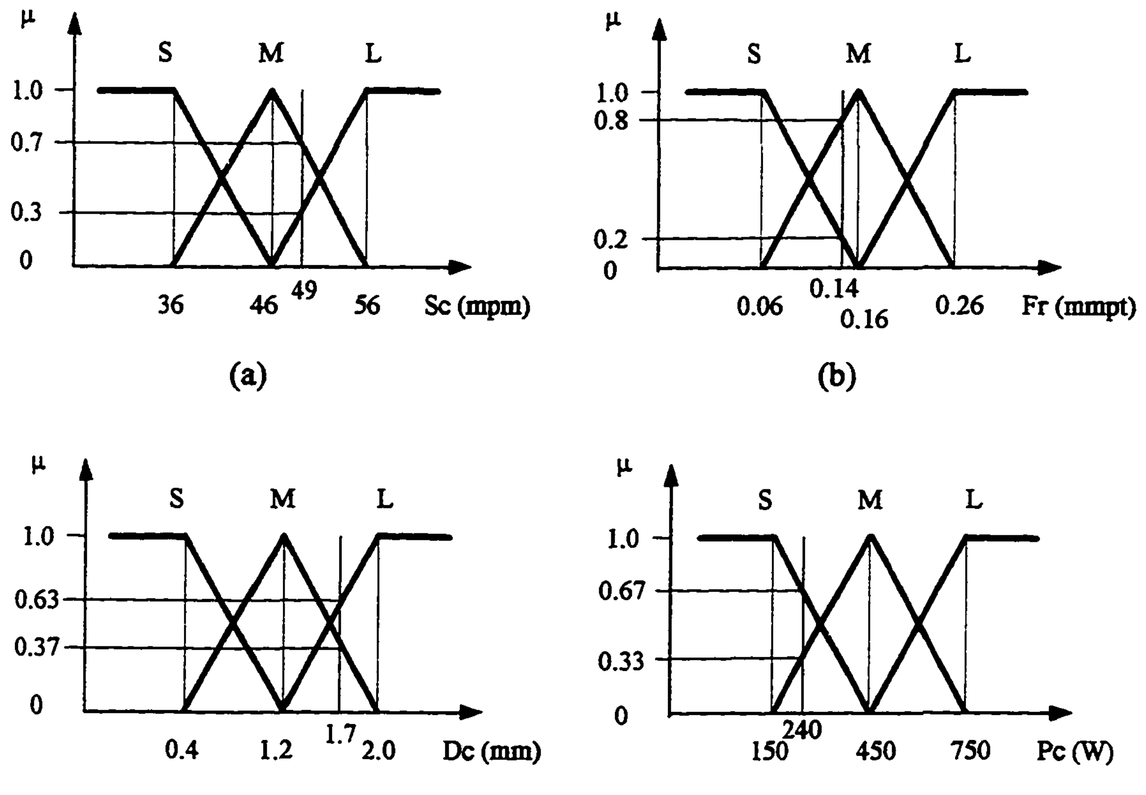


Figure 3.4. Degrees of input and output values

Dc input value 1.7 mm has degree 0.37 in M and degree 0.63 in L, and the Pc output value 240 W has degree 0.67 in S and degree 0.33 in M. After all the values have been assigned degrees in all regions, each value is assigned to the region with maximum degree. Then, Sc(49 mpm) is assigned to M (degree = 0.7), Fr(0.14 mmpt) is assigned to M (degree = 0.8), Dc(1.7 mm) is assigned to L (degree = 0.63), and Pc is assigned to S (degree = 0.67).

The input-output data pairs define the fuzzy classification rules for the knowledge base of the fuzzy logic system as:

$$\text{IF (Sc is } A_1 \text{ AND Fr is } B_1 \text{ AND Dc is } C_1\text{) THEN (the output is } Pc_1\text{).} \quad (3.5)$$

For example, if

$$[\text{Sc, Fr, Dc, Pc}] \Rightarrow [49 \text{ mpm}, 0.14 \text{ mmpt}, 1.7 \text{ mm}, 240 \text{ W}] \quad (3.6)$$

then

$$[\text{Sc}(0.7 \in \text{M}), \text{Fr}(0.8 \in \text{M}), \text{Dc}(0.63 \in \text{L}), \text{Pc}(0.67 \in \text{S})] \Rightarrow$$

$$\text{IF (Sc is M} \wedge \text{Fr is M} \wedge \text{Dc is L) THEN (Pc is S)} \quad (3.7)$$

where the symbol $\wedge$ represents the "AND" operation in the classical logic. A fuzzy rule will be generated for each input-output data pair.

Step 3: Resolve conflicting rules.

It is highly possible that there will be conflicting rules, i.e., rules that have the same IF premise but a different THEN conclusion. Top-down and bottom-up methodologies are proposed to resolve this conflict. Top-down methodology assigns a degree (d) to each rule. The degree of the following rule "IF Sc is M and Fr is M and Dc is L, THEN Pc is S," is defined as:

$$d(Rule) = \mu_M(Sc)\mu_M(Fr)\mu_L(Dc)\mu_S(Pc)\mu_D \quad (3.8)$$

where μ_D is the data pair degree assigned by the human expert. An example of two rules (j and k) is:

Rule j: "IF Sc is M and Fr is M and Dc is L, THEN Pc is S." (3.9)

Rule k: "IF Sc is M and Fr is M and Dc is L, THEN Pc is M." (3.10)

The following strategy is used to resolve the conflicting rules. If the magnitude of the deviation |d(rule k) - d(rule j)| > δ, where $0 < \delta < 0.1$ and δ is a user-defined parameter, then the rule with the maximum active value is the winner. Otherwise, a bottom-up procedure is required to resolve the problem. Using the bottom-up methodology will add two more regions to one feature of the input vector. For example, Sc initially is set up for five regions. If the differential degree of rule k and rule j is less than δ, then Sc is extended to seven regions. Thus, all of the previously trained input-output data pairs must be retrained. If any other rules conflict, two more regions are added to the output feature. If the conflicts are still not resolved, the number of regions of the next input feature and the output feature ((Fr, Pc), (Fr, Pc), (Dc, Pc)) is extended sequentially until all of the conflicting situations are resolved.

Step 4: Develop a combined fuzzy rule base.

The FNS is a 3-dimensional space classifier. A three-region fuzzy associative memory (FAM) bank is shown in Figure 3.5. The following strategy summarizes how the cells of the fuzzy rule base are filled. A combined fuzzy rule base assigns rules from the experimental data pairs. If more than one rule is in a cell indicating a conflict, top-down and bottom-up strategies are applied to resolve the problem. Since the linguistic rule is an "AND" rule in this case, only one rule will fill a cell. As described in Step 3, if rule j is the winner, then the region value S will fill the cell illustrated in Figure 3.5.

Step 5: Perform defuzzification.

The following defuzzification strategy is used to determine the output control Pc for the inputs (Sc, Fr, Dc). First, for given inputs (Sc, Fr, Dc), the antecedents of the fuzzy rule use the multiplication operation to determine the degree, μ_{output}, of the output control responding to the input, i.e.,

$$\mu^{i}_{Output^{i}} = \mu^{i}_{Input^{i}_{Sc}} \mu^{i}_{Input^{i}_{Fr}} \mu^{i}_{Input^{i}_{Dc}}, \tag{3.11}$$

where *Output*^{i} denotes the output regions of rule i, and *Input*^{i} denotes the input region of rule i of Sc, Fr, Dc. The centroid defuzzification is applied to determine the output, which is calculated based on

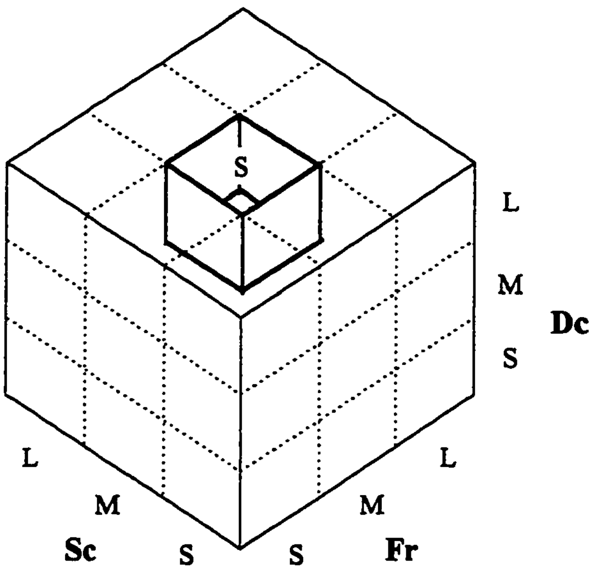


Figure 3.5. A three-region FAM bank

$$y = \left(\sum_{j}^{m} \mu_{Output}(Pc_j)c(Pc_j)\right) / \left(\sum_{j}^{m} \mu_{Output}(Pc_j)\right), \tag{3.12}$$

where $c(Pc_j)$ denotes the center value of the region, *Output*^{i}, and m is the number of fuzzy rules in the combined fuzzy rule base. Note that the center of a fuzzy region is defined as the point that has the smallest absolute value among all the points at which the membership function for this region has a membership value of one.

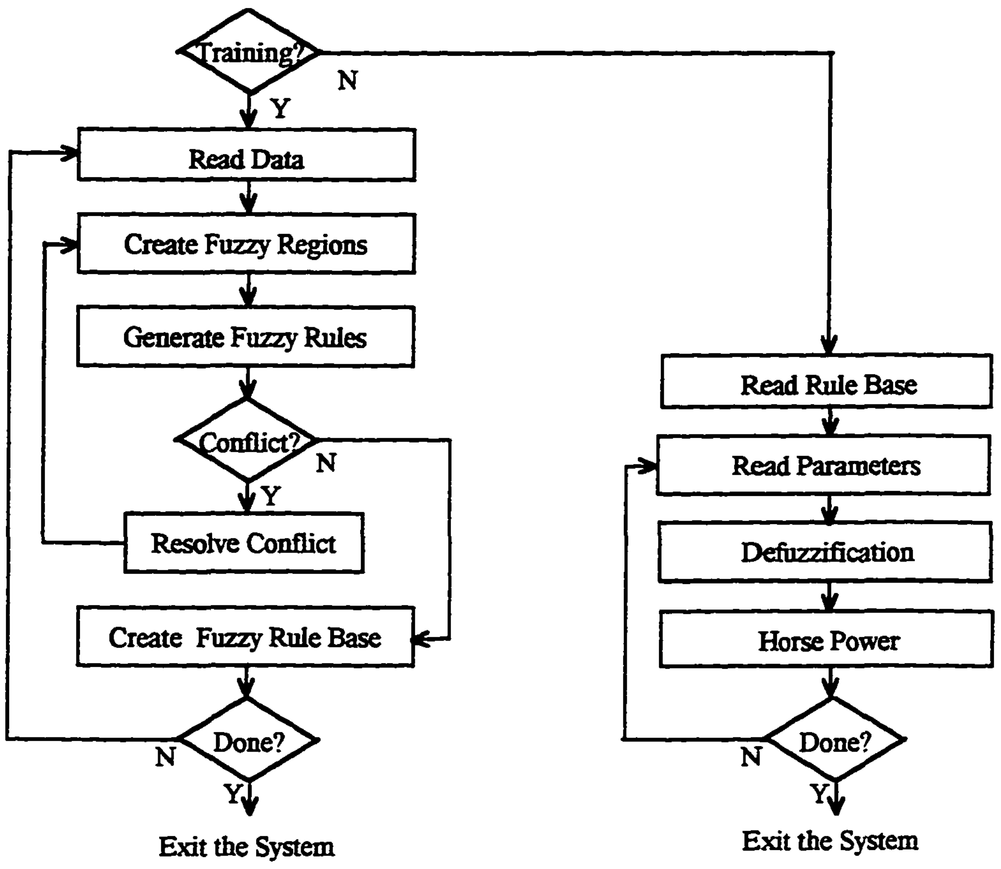
Structure of the Fuzzy-Nets System

The FNS fuzzifies inputs, constructs the fuzzy rule base, retrieves fuzzy rules from the rule (knowledge) base, defuzzifies the data, and reports errors. The overall structure of the fuzzy-nets system is shown in Figure 3.6. The system consists of two modules: fuzzification and defuzzification. The fuzzification module implements the first four steps of the self-learning procedure, and the defuzzification module implements the last step of the procedure. The system is generic and can be used for a majority of applications.

The inputs to the fuzzification module are the values of delta, the input variable count, initial region count, data set count, and the data sets. All the values are compiled into a file. The output is the required cutting power. The maximum number of input variables is five, and the maximum number of regions is eleven.

System implementation

The five-step self-learning procedure provides a basis for the implementation of the fuzzy-nets system. The task of developing the fuzzy-nets system is now much easier than



(a) fuzzification module (b) defuzzification module

Figure 3.6. Structure of the fuzzy-nets system

before due to advances in: (a) integrated development of environments for languages such as C++; (b) object-oriented programming (OOP); (c) powerful graphics software, techniques, and matching hardware, and (d) a variety of CASE (Computer-Aided Software Engineering) tools and powerful debuggers to further reduce cycle time (Prasad, 1992). The program is listed in Appendix A. Simulation and theoretical data and various input files are used to evaluate the system.

System Assessment and Experiment

For quality and reliability, the fuzzy-nets system needs to be assessed. The assessment of the system will be discussed in Chapter 4. After assessment, the system is used in an actual experiment which is depicted in chapter 5. Data collected from the experiment are analyzed and used to train and test the system. The testing results are also analyzed and interpreted.

Experimental design

The simulation and the experiment each contain three independent variables (machining parameters: depth of cut, cutting speed, and feed) and one dependent variable (cutting power). Based on the workpiece and tool materials, machine capabilities, and other factors, different combinations of the settings of the independent variables were selected for the experiment. The settings are shown in Table 3.1. The required cutting power for each combination of the machining settings was obtained from the theoretical model, the experiment, or the fuzzy-nets system.

Table 3.1. Settings for the independent variables

Machining parameters	**Settings**						
Depth of cut (mm)	0.360	1.800	3.240	4.680	6.120	7.560	9.000
Cutting speed (mpm)	39.000	43.400	47.800	52.200	56.600	61.000	65.400
Feed rate (mmpt)	0.025	0.075	0.125	0.175	0.225	0.275	0.325

CHAPTER 4. ASSESSMENT OF THE FUZZY-NETS SYSTEM

The fuzzy-nets system is a computer program developed to simulate and predict the cutting power of end milling operations. The required cutting power of a CNC machine is determined by variables such as speed, feed, depth of cut, strength of tool material, strength of workpiece material, etc. The relationship of the cutting power and these variables is nonlinear. The collection of real data from an experiment could be a relatively long process. The experiment consumes the operator's time, machine time, material, and so on. Thus, it is important to assess the fuzzy-nets system before conducting the experiment. The purpose of the system assessment is to investigate and understand the milling process, verify and validate the system, and reduce experimental costs.

The procedure to assess the fuzzy-nets system includes the following steps:

1. Study and determine the formula for calculating the cutting power.
2. Create training data sets, including input data sets, output data, and testing data.
3. Compile the training data sets and other data into an ASCII file.
4. Train the FNS using simulation and theoretical data.
5. Test the FNS using simulation data.
6. Analyze and evaluate the results.

Cutting Power Calculation

The cutting power P_c is calculated by the formula

$$P_c = \text{MRR} \bullet \text{U} \bullet \text{FCF} \bullet \text{WCF} \qquad (4.1)$$

where MRR = material removal rate, mm^3/s; U = unit power or specific energy, N-m/mm^3; FCF = feed correction factor; and WCF = tool wear correction factor. The value of WCF is 1.1 and the value of U is 0.8274 N-m/mm^3 (Groover, 1996). Table 4.1 shows some feeds and their corresponding correction factors, as was described in Chapter 2. The correction factors for other feeds can be determined by interpolation.

The material removal rate MRR is calculated by the formula

$$\mathrm{MRR} = \mathrm{W} \bullet \mathrm{H} \bullet \mathrm{Fr} \tag{4.2}$$

where W = width of cut, mm; H = depth of cut, mm; and Fr = feed rate, mm/s.

The feed rate (Fr) is calculated by the formula

$$\mathrm{Fr} = \mathrm{Ft} \bullet \mathrm{T} \bullet \mathrm{N} \tag{4.3}$$

where Ft = feed, mmpt; T = number of teeth, a constant; N = speed, rps.

The speed N is calculated by the formula

$$\mathrm{N} = \frac{S \bullet 1000}{\pi \bullet D \bullet 60} \tag{4.4}$$

where S = surface speed, mpm; π = the constant 3.1416; and D = tool diameter, mm.

By combining Equations 4.1, 4.2, 4.3, and 4.4, the cutting power P_c can be calculated by the following equation (assume T = 4)

$$P_c = \mathrm{H} \bullet \mathrm{Ft} \bullet \mathrm{S} \bullet 17.558 \bullet \mathrm{FCF} \bullet 1.1 \tag{4.5}$$

Table 4.1. Feed correction factors for unit horsepower and specific energy

Feed (mmpt)	0.025	0.075	0.125	0.175	0.225	0.275	0.325
FCF	1.6	1.4	1.25	1.18	1.06	0.95	0.92

Preliminary Training and Testing of the Fuzzy-Nets System

Three hundred and forty-three input-output data sets are created to train the fuzzy-nets system, as shown in Appendix B. The input variables are depth of cut (millimeters), speed (meters per minute), and feed (millimeters per tooth), and the output variable is cutting power (watts). The cutting tool is a high speed steel end mill with 19.05 mm (0.75 in.) diameter and four flutes, and the workpiece material is 6061 aluminum. The data set degree of one is assigned to train the system. The initial value of δ, the user-defined parameter, is 0.09. The initial numbers of regions for the input variables and the output variable are three and twenty-five, respectively. At the end of training, the number of regions for the input variables are increased to seven. Figure 4.1 shows the central points of the linguistic variables. Three hundred and forty-three rules are generated through the training procedure for the three hundred and forty-three cells in the rule base.

Table 4.2 shows the rule base for the preliminary testing; rows 1 to 7 contain rules for "Dc is S3;" rows 8 to 14 contain rules for "Dc is S2;" ...; and rows 43 to 49 contain rules for

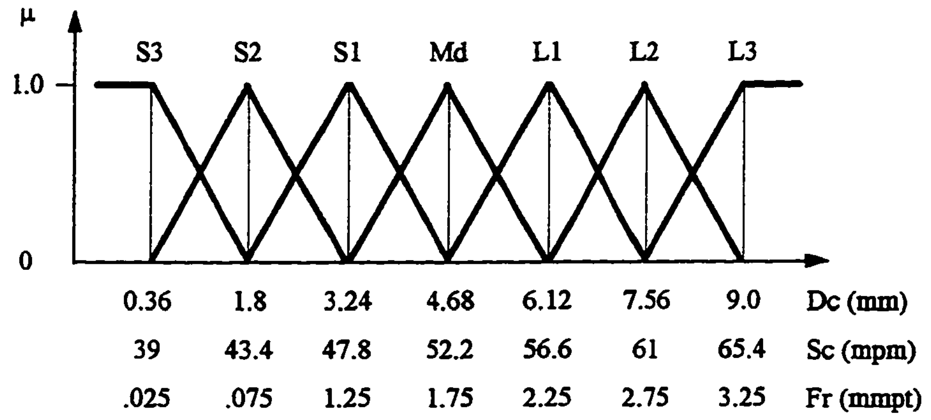

μ
1.0
0
S3
S2
S1
Md
L1
L2
L3
0.36 1.8 3.24 4.68 6.12 7.56 9.0 Dc (mm)
39 43.4 47.8 52.2 56.6 61 65.4 Sc (mpm)
.025 .075 1.25 1.75 2.25 2.75 3.25 Fr (mmpt)

Figure 4.1. Central points of linguistic variables extremely_small (S3), very_small (S2), small (S1), medium (Md), large (L1), very_large (L2), and extremely_large (L3)

Table 4.2. Preliminary FAM banks of the fuzzy-nets system

Row#	Dc	Fr							Sc
		S3	S2	S1	Md	L1	L2	L3	
1		S12	S12	S12	S12	S12	S12	S12	S3
2		S12	S12	S12	S12	S12	S12	S11	S2
3		S12	S12	S12	S12	S12	S11	S11	S1
4	S3	S12	S12	S12	S12	S11	S11	S11	Md
5		S12	S12	S12	S12	S11	S11	S11	L1
6		S12	S12	S12	S11	S11	S11	S11	L2
7		S12	S12	S12	S11	S11	S11	S11	L3
8		S12	S11	S11	S10	S10	S10	S9	S3
9		S12	S11	S10	S10	S10	S9	S9	S2
10		S12	S11	S10	S10	S9	S9	S9	S1
11	S2	S12	S11	S10	S9	S9	S9	S8	Md
12		S12	S11	S10	S9	S9	S8	S8	L1
13		S11	S10	S10	S9	S8	S8	S8	L2
14		S11	S10	S10	S9	S8	S8	S7	L3
15		S11	S10	S9	S9	S8	S8	S7	S3
16		S11	S10	S9	S8	S7	S7	S6	S2
17		S11	S10	S9	S8	S7	S7	S6	S1
18	S1	S11	S10	S8	S7	S7	S6	S5	Md
19		S11	S9	S8	S7	S6	S6	S5	L1
20		S11	S9	S8	S6	S6	S5	S4	L2
21		S11	S9	S8	S6	S5	S5	S3	L3
22		S11	S9	S8	S7	S6	S6	S5	S3
23		S11	S9	S8	S6	S5	S5	S4	S2
24		S11	S9	S7	S6	S5	S4	S3	S1
25	Md	S11	S9	S7	S5	S4	S3	S2	Md
26		S11	S8	S6	S5	S3	S3	S1	L1
27		S11	S8	S6	S4	S3	S2	M	L2
28		S10	S8	S6	S3	S2	S1	M	L3
29		S11	S9	S7	S5	S4	S4	S2	S3
30		S11	S8	S6	S5	S3	S3	S1	S2
31		S10	S8	S6	S4	S3	S2	M	S1
32	L1	S10	S7	S5	S3	S2	S1	L1	Md
33		S10	S7	S5	S2	S1	M	L2	L1
34		S10	S7	S4	S2	M	L1	L3	L2
35		S10	S6	S4	S1	L1	L2	L4	L3

Table 4.2. (continued)

Row#	**Dc**	**Fr**							**Sc**
		S3	**S2**	**S1**	**Md**	**L1**	**L2**	**L3**	
36		S10	S8	S6	S4	S2	S2	M	**S3**
37		S10	S7	S5	S3	S1	M	L1	**S2**
38		S10	S7	S4	S2	M	L1	L3	**S1**
39	**L2**	S10	S6	S4	S1	L1	L2	L4	**Md**
40		S10	S6	S3	M	L2	L3	L5	**L1**
41		S10	S5	S2	L1	L3	L4	L7	**L2**
42		S9	S5	S2	L2	L4	L6	L8	**L3**
43		S10	S7	S5	S2	S1	M	L2	**S3**
44		S10	S6	S4	S1	L1	L2	L4	**S2**
45		S10	S6	S3	M	L2	L3	L6	**S1**
46	**L3**	S10	S5	S2	L1	L3	L5	L7	**Md**
47		S9	S5	S1	L2	L5	L6	L9	**L1**
48		S9	S4	M	L3	L6	L8	L10	**L2**
49		S9	S4	L1	L5	L7	L9	L12	**L3**

"Dc is L3." For example, if "Dc is S3" and "Sc is L2" and "Fr is Md," then "P_c is S11."

After training, 20 input testing data sets are used to evaluate the performance of the system. Table 4.3 shows the preliminary testing results. The predicted cutting power (PP) is calculated by the fuzzy-nets system and the expected cutting power (EP) is calculated by the formula described above. The comparisons between the expected cutting power and the predicted cutting power are shown in Figure 4.2.

Analysis and Evaluation of the Preliminary Testing Results

It was hypothesized that there is no significant difference in cutting power between the data calculated by the formula and the data calculated by the fuzzy-nets system, i.e., H_0: $\mu_d = 0$. The mean ($\bar{d}$) and standard deviation (s.d.) of the 20 difference measurements

Table 4.3. Preliminary testing results of the fuzzy-nets system

Input Sets				Outputs (W)		
#	Dc (mm)	Sc (mpm)	Fr (mmpt)	EP	PP	EP - PP
1	0.4	63	0.24	120	164	-44
2	0.68	51	0.3	188	226	-38
3	1.08	58	0.2	271	280	-9
4	1.48	65	0.1	246	230	16
5	1.7	49	0.14	277	284	-7
6	2.12	62	0.22	599	634	-35
7	2.52	55	0.18	563	580	-17
8	2.92	48	0.08	300	276	24
9	3.56	57	0.12	595	621	-26
10	3.96	64	0.09	597	568	29
11	4.36	50	0.06	369	328	41
12	5	63	0.09	742	711	31
13	5.4	51	0.05	399	389	10
14	5.8	47	0.06	461	453	8
15	6.44	48	0.07	593	581	12
16	6.84	55	0.06	636	639	-3
17	7.24	62	0.04	534	506	28
18	7.88	53	0.06	707	712	-5
19	8.28	60	0.04	591	566	25
20	8.68	49	0.05	616	579	37

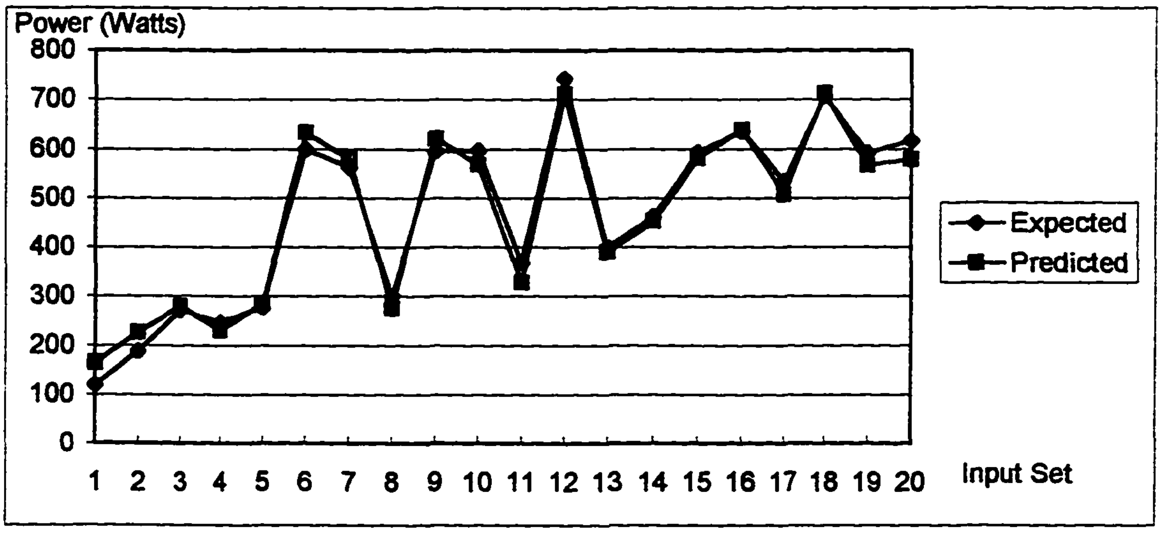


Figure 4.2. Comparisons between expected and predicted cutting powers

are $\bar{d}$ = 3.85 and s.d. = 25.994, respectively. Then

$$t = \frac{\bar{d} - 0}{s.d./\sqrt{n}} = \frac{3.85}{25.994 / \sqrt{20}} = 0.662$$

The critical value of t for a two-tailed statistical test is 2.539, (α = .02; and degrees of freedom = 19). Since the observed value of t does not exceed 2.539, it can be concluded that there is insufficient evidence to indicate that the cutting power calculated by the formula is different from the cutting power calculated by the fuzzy-nets system, at the .02 level of significance. In other words, the FNS possesses a satisfactory range of accuracy with the intended applications of the model.

CHAPTER 5. EXPERIMENTAL SETUP AND RESULTS

The assessment of the fuzzy-nets system (FNS) was described in Chapter 4. Simulation and theoretical data and various input files were used to evaluate the system. The results showed that the FNS possesses a satisfactory range of accuracy with the intended applications of the model. To achieve the objectives of this study, an actual experiment was designed to collect cutting force data during end milling operations for training and testing the FNS.

The experiment was conducted in the CNC (computer numerical control) Laboratory in the Department of Industrial Education and Technology at Iowa State University. The primary purpose of this experiment was to evaluate whether the fuzzy rule bank is suitable to replace the machine database and to determine whether the fuzzy-nets system could be acceptable for industry through experimentation.

The procedure to conduct the end milling experiment includes the following steps:

1. Study and determine the formula for calculating the cutting power.
2. Create training data sets, including input data sets, output data, and testing data.
3. Set up the experiment, including hardware and software.
4. Compile the training data sets and other data into an ASCII file.
5. Conduct the experiment and collect training and testing data for analysis.
6. Train and test the FNS using experimental data.
7. Analyze the data.
8. Evaluate the results.

Experimental Setup

The performance of the fuzzy-nets system was examined for end milling operations. As shown in Figure 5.1 and Appendix C, the experimental setup comprises the following hardware and software components:

Hardware components

1. *Fadal Vertical Machining Center*

The Vertical Machining Center used for this experiment is the Fadal Model 904-1 (VMC40) machine. This CNC machine consists of three major components: a pendant (CNC 32 MP), a tool changer, and a machine tool. The computer of the machine runs the MS DOS 5.0 operating system. A CNC part program can be input to the machine control unit (MCU) using the floppy drive or the keyboard on the pendant. The 15 horsepower (HP) machine has three linear axes (X, Y, and Z) and two rotary axes (A and B) (Fadal, 1992). It can operate in

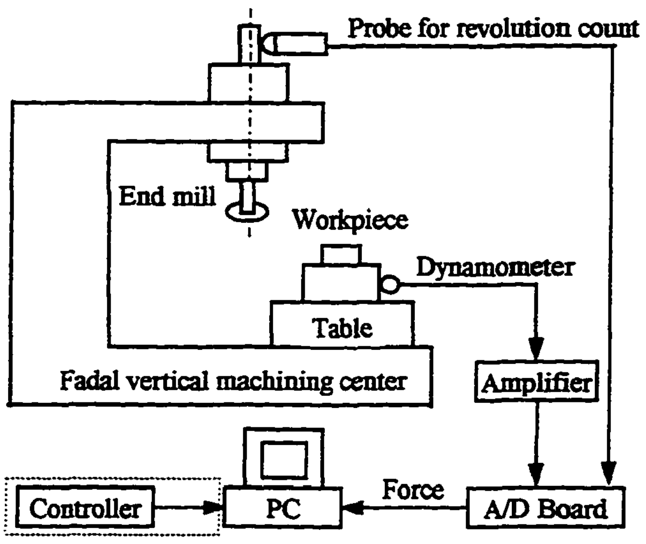


Figure 5.1. Experimental setup

either metric or inch mode. The linear resolution is 0.00254 mm (0.0001 in). The maximum spindle speed is 10,000 rpm (Fadal, 1994). The tool changer can hold up to 21 tools.

2. *Probe*

A probe was used to count the revolutions of the spindle. Since there were two protuberances on the spindle, the probe detected the protuberances and sent out two sets of signals for every revolution (Figure 5.2). The sampling rate and the spindle speed are 1000 samples per second and 10 revolutions per second, respectively.

3. *Morse High-Speed Steel (HSS) End Mill*

The cutting tools used for this experiment are Morse double-ended HSS end mills with 19.05 mm (0.75 in) diameter, four flutes, and a 30° right-hand cut and right-hand helix angle. These cutters have teeth on the end face as well as on the periphery.

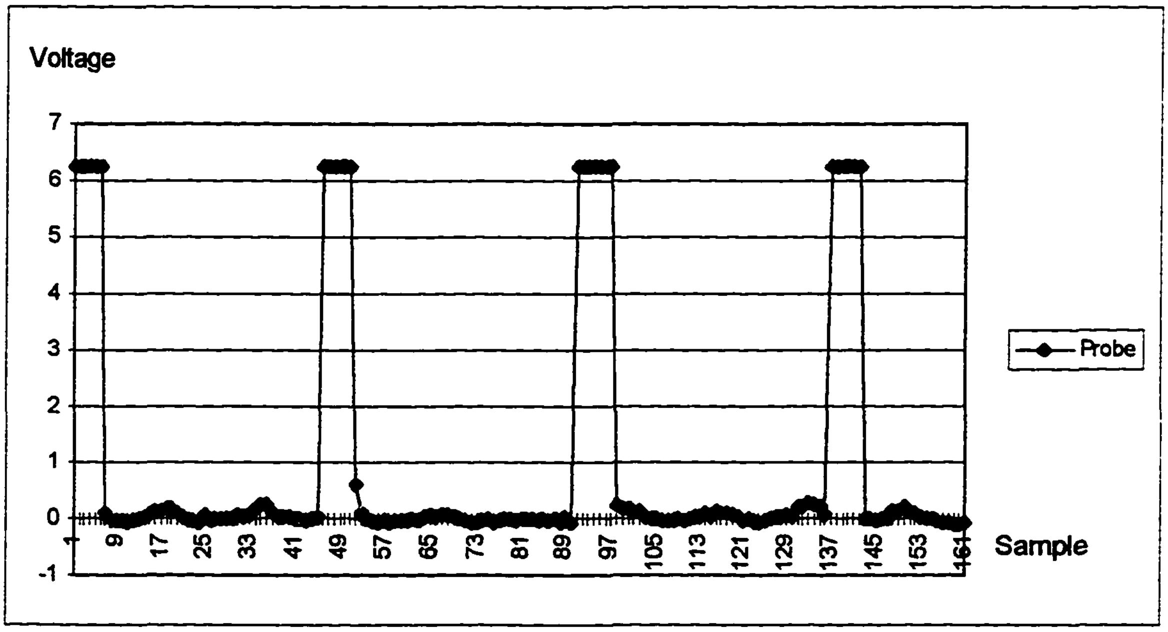


Figure 5.2. Revolution counts of the spindle

4. *Workpiece*

The workpiece material used for the experiment is 6061-T6 aluminum, which is the most versatile of the heat treatable aluminum alloys. The 6061 aluminum is used for a variety of products and applications from truck bodies and frames to screw machine parts and structural components. The dimensions of each workpiece before machining are shown in Figure 5.3.

5. *Workpiece Holder*

The material of the workpiece holder is 1018 steel. Figure 5.4 shows the orthographic drawing (three views) of the workpiece holder which was designed by the researcher. The holder was designed for efficiency in terms of workpiece setup time, stability, and durability.

6. *Kistler Quartz 3-Component Dynamometer Type 9257B*

As shown in Appendix D, the multicomponent dynamometer provides dynamic and quasi-static measurement of the three orthogonal components of a force (Fx, Fy, and Fz) acting from any direction onto the top plate. The dynamometer has a high rigidity and high natural frequency. The high resolution enables very small dynamic changes to be measured in

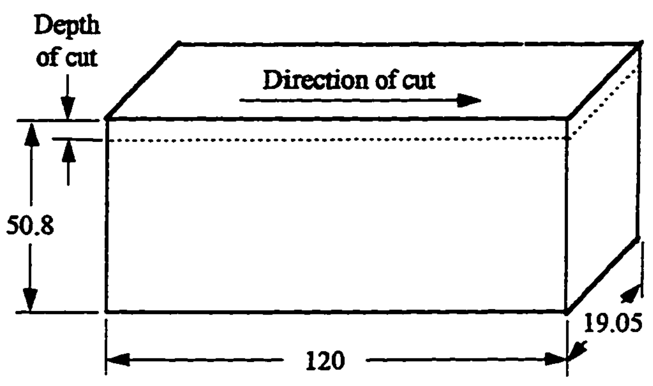


Figure 5.3. Initial dimensions of a workpiece

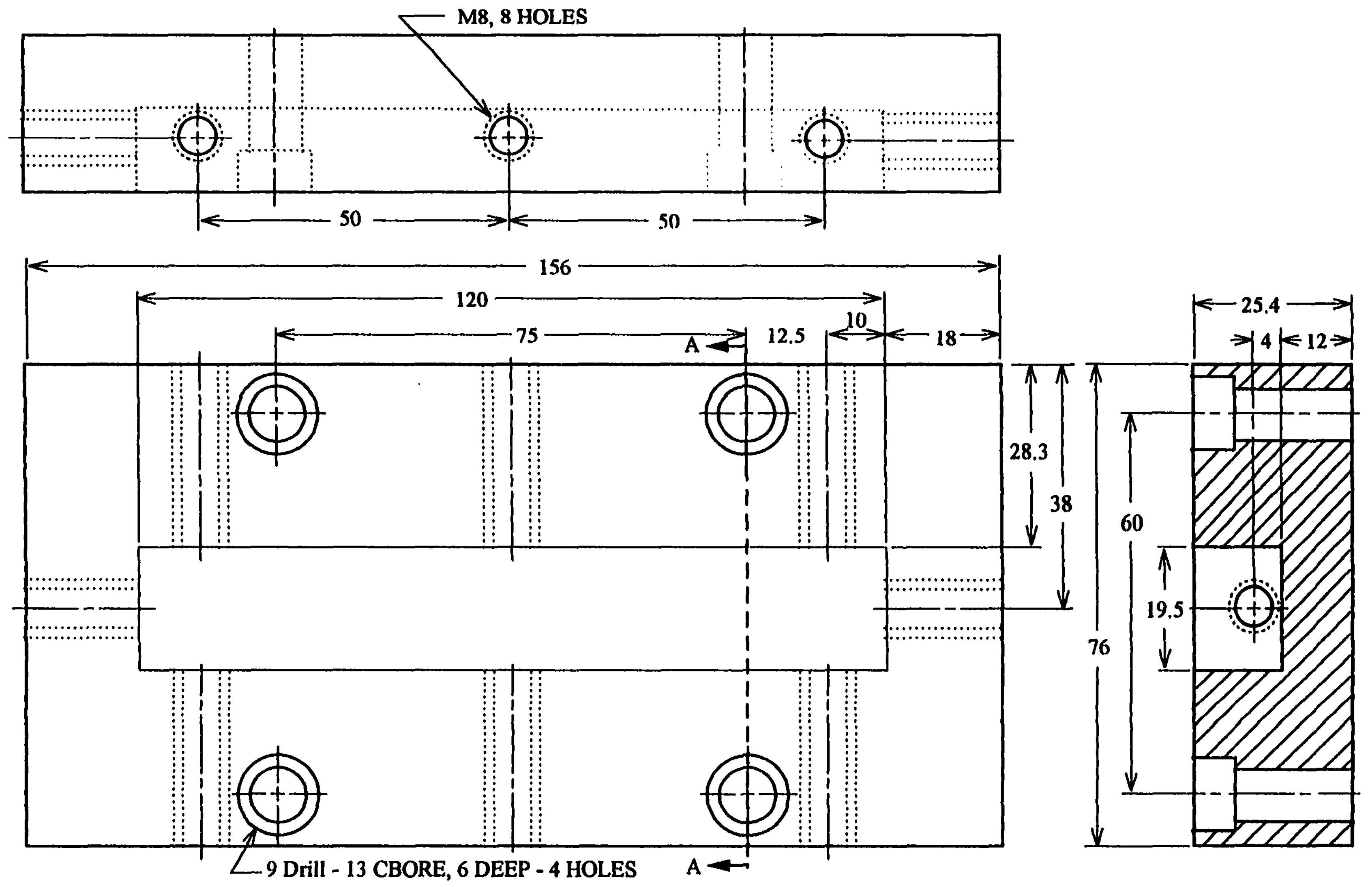


Figure 5.4. Orthographic drawing of the workpiece holder

large forces. The specifications of the dynamometer are shown in Appendix E.

The force to be measured is introduced via a top plate and distributed between four three-component force sensors arranged between the base and top plates. Each of the sensors has three pairs of quartz plates, one sensitive to pressure in the z direction and the other two to shear in the x and y directions, respectively. In these four force sensors, the force introduced is broken down into three components. For the force measurement in three components the individual signals are led together in the connecting cable. Depending on the direction of the force, positive or negative charges occur at the connectives. Negative charges give positive voltages at the output of the charge amplifier, and vice versa.

The dynamometer is rustproof and protected against an ingress of splashwater and coolant. The top plate has a special thermal insulation coating, which renders the dynamometer largely insensitive to temperature influences (Kistler, 1994).

7. *Kistler dual mode amplifier with three channels Type 5814B1*

The dual mode amplifier is a three channel amplifier and constant current supply as shown in Appendix F. The unit converts a piezoelectric transducer signal into a proportional output voltage. The dual mode allows the amplifier to be used with either a charge (high impedance) or a voltage (low impedance) transducer (e.g., Piezotron, ICP compatible, etc.) (Kistler, 1995).

Push buttons are provided on the front panel for user control of the amplifier. The select button selects either transducer sensitivity or scale adjustments. Three other push buttons select: (1) operate/reset, (2) time constant (short, medium, or long), and (3) mode (charge or voltage).

8. *Analog-to-digital (A/D) converter*

The A/D converter was an Omega DAS-1401 high performance analog-to-digital (A/D) board installed in a computer to sense an analog signal and convert it to a representation in digital form.

9. *Apex 486 personal computer*

This computer was used to sample, collect, analyze, and store the data from the experiment.

Software components

1. *CNC part program*

A CNC part program was created to operate the Fadal vertical machining center (Appendix G). It performed the following functions:

a. Start/stop the machine.

b. Use the metric mode.

c. Set the depth of cut, feed rate, spindle speed, and direction of rotation.

d. Turn on/off manual feed/spindle-speed control.

e. Specify fixture and tool numbers.

f. Synchronize workpiece cutting and data collection.

2. *LabTech data collection software*

The LabTech data collection software is window-based (with graphic user interface, or GUI). The user can set sampling rate and duration, channels, filenames, and so on to desired values. The data collected from different channels are stored in different files. The dynamic data exchange (DDE) utility of this software allows data to be transferred between applications. A drawback of the software is that it takes several seconds to open it. Hence, an idle mechanism was added to the CNC part program to synchronize workpiece cutting and data collection.

Experimental Process

A series of end milling operations were performed on a Fadal VMC40 vertical machining center using a CNC part program written by this researcher to collect data for this study. The cutting force signal was measured by a three-component dynamometer mounted on the table of the CNC machine with the workpiece mounted on it. The output voltage signal of the charge amplifier was collected by a personal computer on which an Omega DAS-1401 high performance analog to digital (A/D) board was installed to sample the data on-line. Data sets were collected to train and test the fuzzy-nets system.

Instrument calibration and setup evaluation

To ensure reliable operation of an instrument, it must be well maintained and recalibrated after a specified period of time or an uncontrolled overloading. The three-component dynamometer and amplifier were brand-new so they did not need any factory recalibration. However, the amplifier and the base and top surfaces of the dynamometer were inspected for visible damage before they were used. A weight scale, a probe, and test cuts were used to validate the experimental setup.

Production cuts and data collection

The workpieces were deburred and their surfaces were smoothened before they were secured in the holder. This process was done to ensure accurate cutting results. The cutters used for this experiment were high-speed steel (HSS) end mills with 19.05 mm (0.75 in) diameter, four flutes, and a 30° helix angle. Vibration during cutting could cause the experimental components to move or become loose, especially the workpiece and the end mill. The other source of the problem was the z-axis force created by the helix shape of the end mill. The vibration could become severe when the feed rate is high and the cut is deepening. Therefore, they were inspected periodically for movement and damage.

The three orthogonal components of the force (Fx, Fy, and Fz) acting on the workpiece were measured by the Kistler three-component dynamometer which was connected to a Kistler amplifier. The amplifier converted the piezoelectric transducer signals from the dynamometer into proportional output voltages. The voltage signals were sent to the Omega DAS-1401 A/D board which was installed in a personal computer.

The digital data from the A/D board were then acquired and stored by the LabTech software into three files for the three force components. One more file could be created for the revolution count data from the probe. The data are also displayed on the computer monitor for inspection. The execution of the LabTech program would overwrite the files created by the previous execution. Hence, the files were copied to different files before the next execution of the LabTech program. The sampling rate and duration of data acquisition were 1,000 samples per second and 1.6 seconds respectively. These settings were limited by the RAM of the computer.

A timing problem could occur during the production cuts. A cut took greater or less than 1.6 seconds depending on the feed rate, and it took several seconds to open the LabTech program. These two events should be synchronized so that the data are not lost. The problem was solved by delaying the start of either events or suspending the feed for a specified length of time by the CNC part program.

The material-removal rate (MRR) is a measurement of how fast material is removed from a workpiece. A higher MRR consumes less processing time, but requires greater power. Different formulas are used for different processes. For end milling, the MRR is calculated by the formula

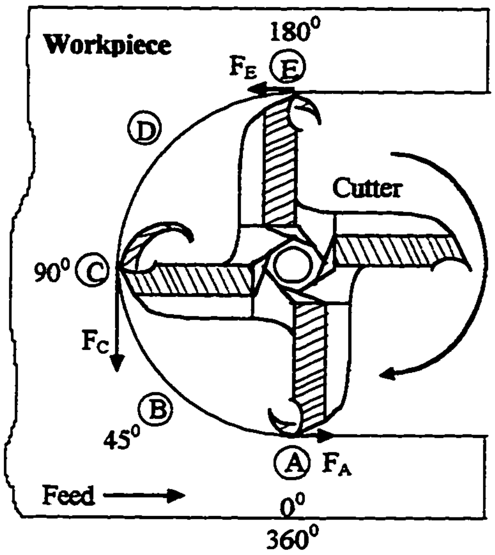
$$MRR = W \bullet Dc \bullet Ft \bullet T \bullet N$$

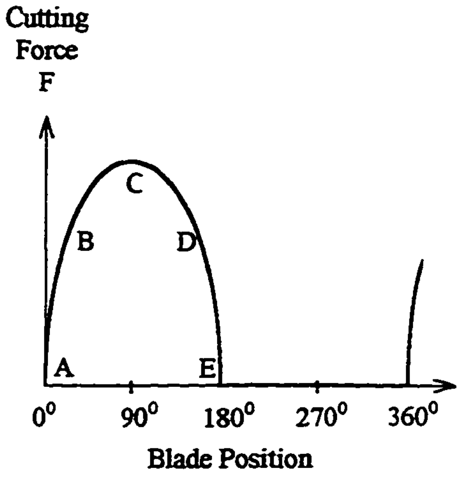
where W = width of cut, mm; Dc = depth of cut, mm; Ft = feed, mm/t; T = number of teeth; and N = spindle speed, rpm. Assume that W, T, and N are constants, then the MRR is determined by the product of Dc and Ft. For this experiment, the product of Dc and Ft is limited to $0.765\ mm^2$.

Two hundred and twenty-three combinations of the machining parameters (depth of cut, feed rate, and spindle speed) were selected to make production cuts. The machining settings were manually changed in the CNC part program for each run. Sometimes each set of the machining settings needed more than one run. Cutting fluids were not used in this experiment.

Force analysis of end milling operations

Milling is an interrupted cutting process wherein entering and leaving the workpiece subjects the cutter to impact loading, cyclic heating, and cyclic forces. As shown in Figure 5.5a, the cutting force, F, builds rapidly as the cutter enters the workpiece at A and progresses to B, peaks as the blade crosses the direction of feed at C, decreases to D, and then drops to zero upon exit (DeGarmo, Black, & Kohser, 1988). Figure 5.5b displays the cutting force





(a) conventional milling (b) corresponding cutting force diagram

Figure 5.5. The interrupted nature of the milling process

diagram for a single blade of the cutter during operation, and Figure 5.6 shows the X and Y force components measured by the dynamometer during the cut as well as their resultant R.

Training and Testing of the Fuzzy-Nets System

Two hundred and three input-output data sets are created to train the fuzzy-nets system, as shown in Appendix H. The input variables are depth of cut (millimeters), speed (meters per minute), and feed (millimeters per tooth), and the output variable is cutting power (watts). The cutting tool is a high-speed steel end mill with 19.05 mm (0.75 in.) diameter and four flutes, and the workpiece material is 6061 aluminum. The data set degree of one is assigned to train the system. The initial value of delta, the user defined parameter, is 0.09. The initial numbers of regions for the input variables and the output variable are three and twenty-five, respectively. At the end of training, the number of regions for the input variables

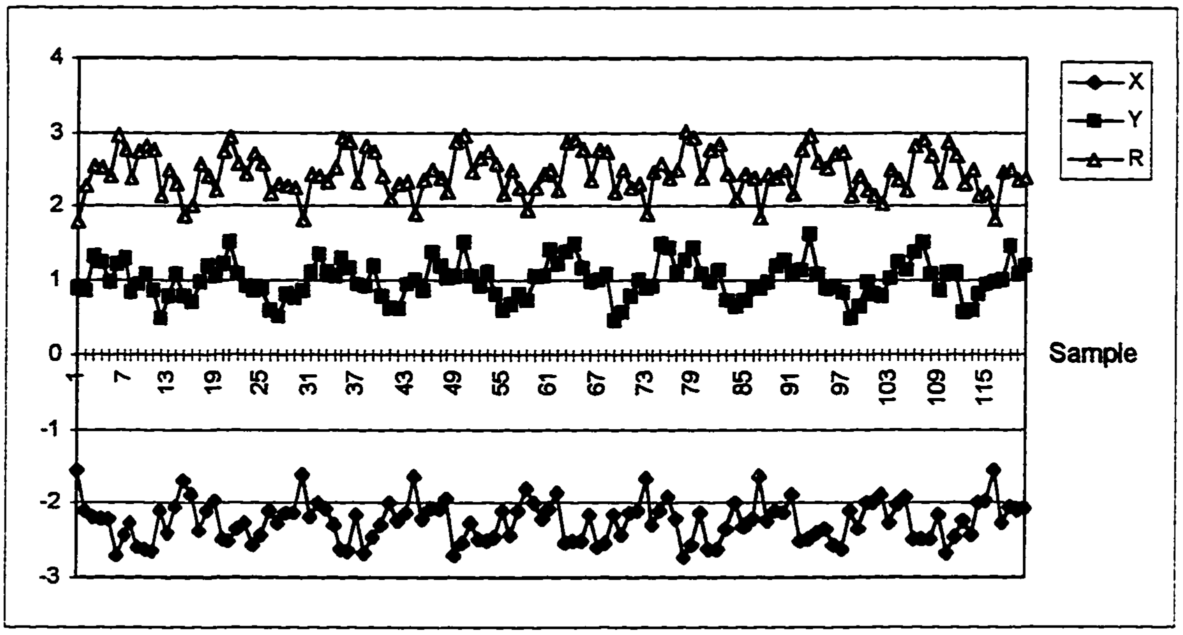


Figure 5.6. Force diagram of a production cut

are increased to seven. Because the product of Dc and Ft is limited to 0.765 mm^2, only two hundred and three rules are generated through the training procedure for the three hundred and forty-three cells in the rule base. Thus, one hundred and forty cells are not filled.

Table 5.1 shows the rule base for the testing; rows 1 to 7 contain rules for "Dc is S3;" rows 8 to 14 contain rules for "Dc is S2;" ...; and rows 43 to 49 contain rules for "Dc is L3." For example, if "Dc is S3" and "Sc is L2" and "Fr is Md," then "P_c is S11."

After training, 23 input testing data sets are used to evaluate the performance of the system. Table 5.2 shows the testing results. The measured cutting power (MP) is converted from the forces measured by the dynamometer; the predicted cutting power (PP) is calculated by the fuzzy-nets system; and the theoretical cutting power (FP) is calculated by the formula described above. The comparisons between the measured cutting power, the theoretical cutting powers, and the predicted cutting power are shown in Figure 5.7.

Table 5.1. FAM banks of the fuzzy-nets system

Row#	Dc	Fr							Sc
		S3	S2	S1	Md	L1	L2	L3	
1		S12	S12	S12	S11	S11	S11	S11	S3
2		S12	S12	S11	S11	S11	S11	S11	S2
3		S12	S12	S11	S11	S11	S11	S11	S1
4	S3	S12	S12	S11	S11	S11	S11	S11	Md
5		S12	S11	S11	S11	S11	S11	S10	L1
6		S12	S11	S11	S11	S11	S10	S10	L2
7		S12	S11	S11	S11	S10	S10	S10	L3
8		S11	S10	S9	S8	S6	S5	S5	S3
9		S11	S9	S8	S7	S6	S5	S4	S2
10		S11	S9	S8	S6	S5	S4	S3	S1
11	S2	S10	S9	S7	S6	S4	S3	S2	Md
12		S10	S8	S7	S5	S4	S2	S1	L1
13		S10	S8	S6	S5	S3	S1	M	L2
14		S10	S8	S6	S4	S2	S1	L1	L3
15		S10	S8	S6	S4	S2			S3
16		S10	S7	S5	S3	S1			S2
17		S9	S6	S4	S2	L1			S1
18	S1	S9	S6	S3	S1	L2			Md
19		S9	S6	S3	M	L3			L1
20		S9	S5	S2	L1	L4			L2
21		S8	S4	S1	L2	L5			L3
22		S9	S6	S3					S3
23		S9	S5	S2					S2
24		S8	S4	S1					S1
25	Md	S8	S3	L1					Md
26		S7	S3	L1					L1
27		S7	S2	L2					L2
28		S7	S1	L4					L3
29		S8	S4	S1					S3
30		S8	S3	L1					S2
31		S7	S2	L2					S1
32	L1	S7	S1	L4					Md
33		S6	M	L5					L1
34		S6	L1	L6					L2
35		S5	L2	L8					L3

Table 5.1. (continued)

Row#	Dc	Fr S3	Fr S2	Fr S1	Fr Md	Fr L1	Fr L2	Fr L3	Sc
36		S8	S1						S3
37		S7	M						S2
38		S6	L2						S1
39	L2	S6	L3						Md
40		S5	L5						L1
41		S4	L6						L2
42		S4	L7						L3
43		S7	L1						S3
44		S6	L3						S2
45		S5	L4						S1
46	L3	S4	L6						Md
47		S4	L8						L1
48		S2	L10						L2
49		S1	L12						L3

Table 5.2. Testing results of the fuzzy-nets system

Input Sets				Outputs (W)					
#	Dc (mm)	Sc (mpm)	Fr (mmpt)	Measured	Formula	Predicted	MP - PP	MP - FP	PP - FP
1	0.4	63	0.24	118.712	119.963	122	-3.288	-1.251	2.037
2	0.68	51	0.3	202.056	187.879	177	25.056	14.177	-10.879
3	1.08	58	0.2	286.455	270.998	263	23.455	15.457	-7.998
4	1.48	65	0.1	258.786	246.182	251	7.786	12.604	4.818
5	1.7	49	0.14	285.228	276.816	274	11.228	8.412	-2.816
6	2.12	62	0.22	650.163	598.702	618	32.163	51.461	19.298
7	2.52	55	0.18	580.405	562.787	550	30.405	17.618	-12.787
8	2.92	48	0.08	315.746	299.937	331	-15.254	15.809	31.063
9	3.56	57	0.12	610.778	594.925	567	43.778	15.853	-27.925
10	3.96	64	0.09	605.254	596.929	606	-0.746	8.325	9.071
11	4.36	50	0.06	400.478	368.83	398	2.478	31.648	29.17
12	5	63	0.09	772.97	741.922	730	42.97	31.048	-11.922
13	5.4	51	0.05	440.16	398.925	422	18.16	41.235	23.075
14	5.8	47	0.06	475.988	461.207	469	6.988	14.781	7.793
15	6.44	48	0.07	614.127	593.443	607	7.127	20.684	13.557
16	6.84	55	0.06	692.283	636.486	678	14.283	55.797	41.514
17	7.24	62	0.04	582.899	534.044	613	-30.101	48.855	78.956
18	7.88	53	0.06	730.929	706.597	769	-38.071	24.332	62.403
19	8.28	60	0.04	676.503	591.055	698	-21.497	85.448	106.945
20	8.68	49	0.05	649.758	616.088	673	-23.242	33.67	56.912
21	1.8	39	0.245	352.939	337.491	381	-28.061	15.448	43.509
22	1.8	46.5	0.245	427.803	402.394	421	6.803	25.409	18.606
23	1.8	54	0.245	507.948	467.296	505	2.948	40.652	37.704

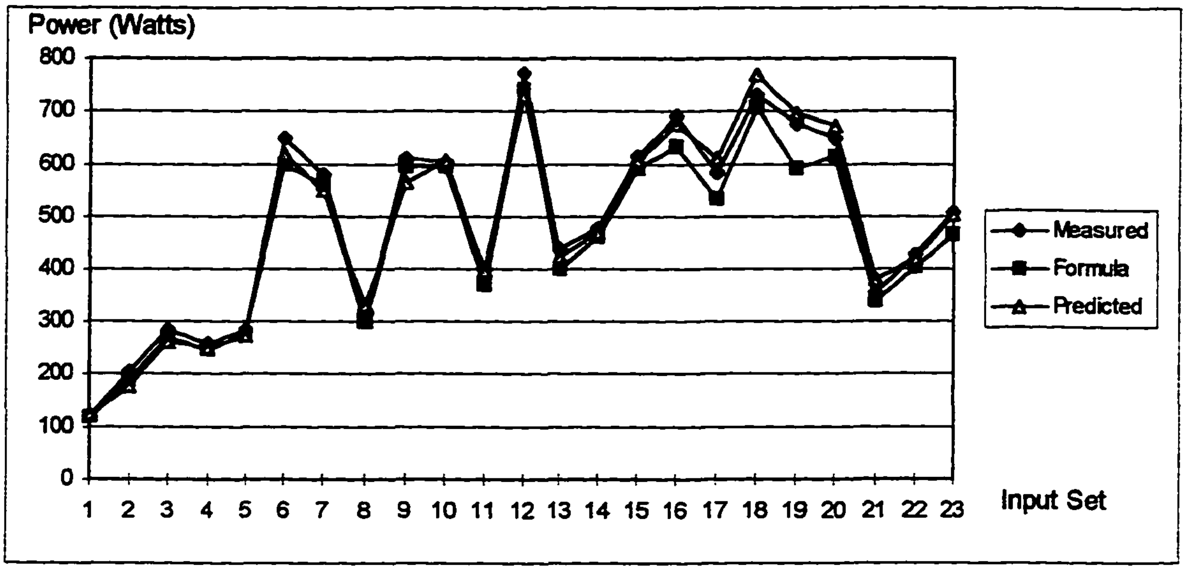


Figure 5.7. Comparisons between measured, formula, and predicted cutting powers

Analysis and Evaluation of the Testing Results

Three hypotheses were formulated to carry out the purpose of the experiment.

Null Hypothesis 1. There is no significant difference in cutting power between the data calculated by the fuzzy-nets system and the data collected from experimentation.

$$H_0 \colon \mu_d = 0$$

The mean ($\bar{d}$) and standard deviation (s.d.) of the 23 difference measurements are $\bar{d} = 5.016$ and s.d. = 22.954, respectively. Thus

$$t = \frac{\bar{d} - 0}{s.d./\sqrt{n}} = \frac{5.016}{22.954 / \sqrt{23}} = 1.048$$

The critical value of t for two-tailed statistical test is 2.508 (α = .02; degrees of freedom (df) = 22). Since the observed value of t does not exceed 2.508, we conclude that there is insufficient evidence to indicate that the cutting power collected from experimentation is different from the cutting power calculated by the fuzzy-nets system, at the .02 level of significance. In other words, the FNS possesses a satisfactory range of accuracy with the intended applications of the model.

Null Hypothesis 2. There is no significant difference in cutting power between the data collected form the experimentation and the data calculated by the formula.

$$H_0:\ \mu_d = 0$$

The mean ($\bar{d}$) and standard deviation (s.d.) of the 23 difference measurements are $\bar{d}$ = 27.281 and s.d. = 19.606, respectively. Thus

$$t = \frac{\bar{d} - 0}{s.d./\sqrt{n}} = \frac{27.281}{19.606/\sqrt{23}} = 6.673$$

The critical value of t for two-tailed statistical test is 6.673, (α = .02; df = 22). Since the observed value of t exceeds 2.508, we conclude that there is sufficient evidence to indicate that the cutting power collected from experimentation is different from the cutting power calculated by the formula.

Null Hypothesis 3. There is no significant difference in cutting power between the data calculated by the formula and the data calculated by the fuzzy-nets system.

$$H_0:\ \mu_d = 0$$

The mean ($\bar{d}$) and standard deviation (s.d.) of the 23 difference measurements are $\bar{d}$ = 22.265 and s.d. = 32.43, respectively. Thus

$$t = \frac{\bar{d} - 0}{s.d./\sqrt{n}} = \frac{22.265}{32.43/\sqrt{23}} = 3.293$$

The critical value of t for two-tailed statistical test is 3.293, (α = .02; df = 22).

Since the observed value of t exceeds 2.508, we conclude that there is sufficient evidence to indicate that the cutting power calculated by the fuzzy-nets system is different from the cutting power calculated by the formula.

Accuracy comparison

The accuracy of data is expressed in terms of error, i.e.,

$$Error\ (\%) = \frac{|Predicted - Measured|}{Measured} \bullet 100$$

where predicted: formula or FNS. The means of errors in power calculation are 5.43 % for the formula and 4.1 % for the FNS. From the statistical test and the accuracy comparison, we conclude that the power calculated by the FNS is more accurate than the power calculated by the formula.

CHAPTER 6. CONCLUSIONS AND RECOMMENDATIONS

The selection of machining parameters is an important element of process planning. The development of a utility to show the cutting power on-line would be helpful to programmers and process planners in selecting machining parameters. The relationship of the cutting power and the machining parameters is nonlinear. Presently there is no accurate or simple algorithm to calculate the required cutting power for a selected set of parameters. In this research, a self-organizing fuzzy-nets system was developed to generate a knowledge bank that can show the required cutting power on-line for a short length of time in an NC verifier. The other objectives of this study were to evaluate whether the fuzzy rule bank could replace the machine database and to determine whether the fuzzy-nets system could become acceptable for industry through experimentation.

Three hypotheses were formulated to carry out the purpose of the study:

1. There is no significant difference in cutting power between the data calculated by the formula and the data calculated by the fuzzy-nets system.
2. There is no significant difference in cutting power between the data calculated by the fuzzy-nets system and the data collected from experimentation.
3. There is no significant difference in cutting power between the data calculated by the formula and the data collected from experimentation.

The fuzzy-nets system (FNS) utilizes a five-step self-learning procedure. A generic FNS program consisting of fuzzification and defuzzification modules was implemented in the C++ programming language to perform the procedure. The FNS was assessed before an actual experiment was set up to collect data. The FNS possessed a satisfactory range of accuracy with the intended applications of the model.

The performance of the FNS was then examined for end milling operations on a Fadal VMC40 vertical machining center. The cutting force signals were measured by a three-component dynamometer mounted on the table of the Fadal CNC machine with the workpiece mounted on it. Amplified signals were collected by a personal computer on which an Omega DAS-1401 analog-to-digital converter was installed to sample the data on-line. Data sets were collected to train and test the system. The results were analyzed and evaluated.

Comparisons to Fang and Jawahir's Study

As described in the review of literature, various researchers have attempted to find a method for predicting cutting power and/or optimizing machining performance and they have demonstrated great success. In a recent research, Fang and Jawahir (1994) developed a new methodology for predicting the total machining performance (TMP) in finish turning using integrated fuzzy-set models of machinability parameters. The total machining performance is evaluated in terms of surface finish, tool-wear rate, dimensional accuracy, cutting power, and chip breakability. In the development of the new methodology, a machining reference database is first established from experiments. Secondly, a knowledge pool is developed based on a series of machining experiments and the existing knowledge of the major influencing factors on the TMP. Then, a fuzzy-set method is introduced to quantify the effects of these factors. Finally, several fuzzy-set models are developed to quantitatively assess the TMP for any given set of input conditions. The comparisons between their study (TMP) and this study (FNS) are as follows:

1. The operation ranges of the FNS is larger. In the TMP study, the range of feed for finish turning operations is from 0.04 mmpr (millimeter per revolution) to 0.2 mmpr and the range of depth is from 0.25 mm to 0.8 mm. In the FNS study, the range of feed for end milling operations is from 0.1 mmpr to 1.3 mmpr and the range of depth is from 0.36 mm to 9 mm.
2. The space to store the data and knowledge base is less for the FNS. A machining reference database and a knowledge pool are required in the TMP study. However, what the FNS needs is the fuzzy rule base.
3. The performance of the FNS was evaluated using simulation and experimental data in various tests with combinations of input conditions including depth of cut, feed rate, and cutting speed. The accuracy of prediction is expressed in terms of error. For example, if the expected cutting power (EP) and the predicted cutting power (PP) are 520 watts and 550 watts, respectively, then the error is calculated by the equation

   $$\text{Error (\%)} = \frac{|PP - EP|}{EP} \bullet 100 = \frac{|550 - 520|}{520} \bullet 100 = 5.8\ \%.$$

   In this research, the means of errors in power prediction are 6.7 % for the simulation tests and 4.1 % for the experimental tests, whereas in the TMP model the mean of errors is 13.5 %. The FNS model is thus significantly more accurate than the TMP model.

The above comparisons show that the FNS model is a better alternative to the TMP approach in the prediction of cutting power requirements. Based on the comparisons, the conclusions of this study and recommendations for further research are summarized below.

Conclusions

This research studied the relationship between the cutting power and the machining parameters. Although machining data handbooks, machinability data systems, and machining databases have been developed to recommend machining parameters for efficient machining, they are basically for general reference and are hard to use as well. In addition, the machining database is not efficient in terms of time and space. Each machine has its own characteristics and capabilities. Thus, the fuzzy-nets system was developed to predict the cutting power of end milling operations. The major conclusions drawn from this study are as follows:

1. The fuzzy-nets system was assessed for its performance. Simulation and theoretical data were used in the evaluation process. Twenty difference measurements were used to test the system. The results show that the FNS possesses a satisfactory range of accuracy with the intended applications of the model.
2. Data handbooks and formula are only for general references. The power requirements obtained from the experiment are more accurate than the formula.
3. The data predicted by the FNS are as accurate as the data collected from the experiment.
4. The data predicted by the FNS are more accurate than the data calculated by formulas.
5. The new fuzzy-nets based methodology can show the process planners and CNC part programmers the required cutting power in a short length of time on line, so the new approach is a better alternative to the traditionally known methods for selecting machining parameters.
6. The FNS can be incorporated into a CAD/CAM package to recommend optimal machining parameters.

7. The FNS predicts the required cutting power within a satisfactory range of accuracy. Compared to the FNS, dynamometers and amplifiers are very expensive. Most of these instruments in a plant could be replaced with the FNS.

Recommendations for Further Research

Due to the assumptions and limitations described in Chapter 1, applications of the FNS to the real world could have some constraints. To reduce the constraints, the following areas are recommended for further research:

1. The cutting tools utilized in the experiment were high-speed steel (HSS) end mills with such attributes as 19.05 mm (0.75 in) diameter, four flutes, and a 30° right-hand cut and right-hand helix angle. Tools with different material (e.g., carbide, ceramic, etc.) and/or attributes could be studied to understand their impact on the power requirements.
2. Different materials have different physical and chemical properties. The workpiece materials used for this research were aluminum. Other commonly used materials such as carbon steel and cast iron could be investigated.
3. Other manufacturing processes such as turning, grinding, and so on could be considered to examine the performance of the FNS.

APPENDIX A.   FNS PROGRAM

```
/*********************************\
 *
 *     FNS.H -- user interface program (borlandc c++)
 *            By Ted C. Chang
 *
\*********************************/

//tc0319
//#define INCL_TRAIN

#include <process.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <math.h>
#include <conio.h>
#include <float.h>
#include <ctype.h>

//#define FSC_C_COL       4
//#define FSC_C_DATASET   270

#define FSC_C_FILE       2
#define FSC_C_FV         2
#define FSC_C_INDATA     3  // name + nick_name + fDelta
//#define FSC_C_REGION    9

//#define FSC_C_VAR      2 //tc 10

#define FSC_CCH_LINE   256
#define FSC_CCH_NAME    18
#define FSC_CCH_FILENAME 128
#define FSC_CCH_FLOAT   12
#define FSC_CCH_NICK     3
#define FSC_CCH_NUMBER  12
#define FSC_CCH_SLINE   128
//tc #define FSC_CCH_TIME    32

#define FSC_I_INFILE     0
#define FSC_I_INITREGION 999
#define FSC_I_OUTFILE    1

#define FSC_I_DEPTH      0
#define FSC_I_SPEED      1
#define FSC_I_FEED       2
#define FSC_I_OUTDATA    3
#define FSC_I_OUTDEGREE  4
```

```
#define FSC_I_LEFT      0
#define FSC_I_MIDDLE     1
#define FSC_I_RIGHT      2

#define FSC_MAX_REGIONS 11
#define FSC_MAX_OUT_REGIONS 25

#define FSC_C_DS_DEBUG 10 // DS: Data Sets


#define FSC_SZ_INFILE   "FNOIN.DAT"
#define FSC_SZ_OUTFILE  "FNOOUT.DAT"

#define FSC_C_FCF 7   /* feed correction factors */
#define FSC_F_WEAR 1.1  /* tool wear factor */

typedef FILE *PFILE;
//typedef struct _iobuf *PFILE;
typedef int *PINT;
typedef float *PFLOAT;
typedef void *PVOID;

typedef struct _FV /* fuzzy value */
{
 int  iRegion;
 float fDegree;
} FV, *PFV, _far *PRULE;

typedef struct _VAR *PVAR;
typedef struct _VAR
{
 char  szName[FSC_CCH_NAME + 1];
 char  szNick[FSC_CCH_NICK + 1];
 float fMax, fMin;
 int  cRegion; /* cInterval + 1 */
 float fData;
 FV  afv[FSC_C_FV]; /* small and large */
 float fInterval; /* (fMax - fMin) / cInterval */
 int  cEmt;  /* descentant count */

// PFLOAT pfCenter; /* central point of a region */
// int  iRegion; /* current region */
// float fDegree; /* current degree */
// PVAR left, right; /* left/right child */
} VAR;

typedef struct _VI /* var info */
{
 PVAR pvar;
 int  cvarMax; /* max # input vars */
 int  cvarAct; /* actual */
 int  cvarIn;  /* input vars */
} VI, *PVI;

typedef struct _DATASET *PDATASET;
typedef struct _DATASET
{

 PFLOAT pfIn;  /* input data set, output data, degree */

//  float afIn[FSC_C_VAR + 2];
//  float fOut;   /* output data */
//  float fDegree; /* representing the data set.  Assigned by an expert. */
//  int iExpert;  /* The input classifier */
//  int iExpOut;  /* The input classifier */
//  PDATASET next;
} DATASET;

typedef struct _DSI /* data set info */
{
 PDATASET pds;
 PFLOAT  pfIn;
 int      cdsMax;  /* max # data sets */
 int      cdsAct;  /* actual */
 int      cData;   /* # data in a data set */
} DSI, *PDSI;

#ifdef INCTMP

typedef struct _RULE /* rule struct */
{
 int  iRegion;
 float fDegree;
} RULE, _far *PRULE;

typedef struct _RI /* rule info struct */
{
} RI, _far *PRI;

typedef struct _TREE *PTREE;
typedef struct _TREE /* defuzzification */
{
 int  iRegion;
 float fDegree;
 PTREE left, right; /* left/right child */
} TREE, *PTREE;

typedef struct _DFI /* defuzzification info */
{
 DF dfSmall;
 DF dfLarge;
} DFI, *PDFI;
```

```
#endif /* INCTMP */

typedef struct _FSC
{
 PFILE apfile[FSC_C_FILE];
 float  fDelta;
 int   fRetrain;
 int   cRegion; /* inial region count */
 int   iConflict; /* current conflict input var */
 VI    vi;
 DSI   dsi;
 PRULE  aprule[FSC_MAX_REGIONS];
 float  afOutCenter[FSC_MAX_OUT_REGIONS];
 PFLOAT pfIn;  /* input test data set */
// PFV   **pppfv; /* (pow(2,InVars))(InVars) */
 PFV   **pppfv; /* (pow(2,InVars)) */
 PFV   *ppfv;  /* (pow(2,InVars))(InVars) */
} FSC, *PFSC;

typedef struct _FCF /* feed correction factor */
{
 float fFeed;
 float fFactor;
} FCF, *PFCF;

int FscExit(PVOID pv, int iStatus);
char *FscGetLine(char *pszText, int cchMax, int *pcchText, PFILE pfile);
int FscDisplayHelp(int cArg, char *apszArg[]);
int FscGetFilename(char *pszInOut, char *pszFilename, int cchFilename);
int FscOpenFiles(int cArg, char *apszArg[], PFILE *ppfile, int cfile);
int FscGetInteger(PFILE pfile, int *piNum);
int FscGetFloat(FILE *pfileData, float *pfDelta);
int FscGetWord(char **ppszLine, int *pichWord, char *pszWord, int cchWord);
int FscGetVarStruct(char *pszLine, PVAR pvar, int cData);
int FscGetVarStructList(PFSC pfsc);
int FscGetDataSet(char *pszLine, PDATASET pds, int cData);
int FscGetOutRegion(PFSC pfsc, int ifv);
int FscCalcCenters(PFSC pfsc, PFLOAT pfCenter, int ivar);
int FscGetEmtCount(PFSC pfsc);
int FscGetDataSetList(PFSC pfsc);
int FscGetMinMax(PFSC pfsc);
int FscBuildFvArray(PFSC pfsc);
int FscGetFvArray(PFSC pfsc, int ids, float fIn, float fMin, float fMax, float fInterval, PFV pfv);
int FscGetFvArray2(PFSC pfsc, int ids, float fIn, float fMin, float fMax, float fInterval, PFV pfv);
int FscDefuzzify(PFSC pfsc);
float FscGetFcf(float fFeed);


/*** fns.h ***/
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\
 *
 *     FNS.CPP -- user interface program (borlandc c++)
 *            By Ted C. Chang
 *
\* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <math.h>
#include <conio.h>
#include <float.h>
#include <malloc.h> // _fcalloc
#include "fno.h"

#define  FSC_C_DATASET  8 /* 270 training input-output data sets */
#define  FSC_C_VAR      4  /* variables to be measured  */
#define  INPUT_PTS     20  /* the forces input for getting training peaks */
#define  MAXREGION      10  /* training points */

#define  max(a,b) (a>b?a:b)

/*** function prototype ***/

int FscRetrain(PFSC pfsc, PFLOAT pfDelta);
int FscGetOutputData(PFSC pfsc);
int FscInitGrades(PFSC pfsc);
int FscGetGrades(PFLOAT, PFLOAT, PFSC, int);

/*** global vars ***/

PFILE pfileLog;
char  szTmp[FSC_CCH_SLINE + 1];
char  *pszTmp;
int   cchTmp;

/*** main ***********************************************************************/
int main(int cArg, char *apszArg[])
{
 int   i, j, k, m;
 FSC  fsc;
 PFSC  pfsc = &fsc;
 int   cvarMax;  /* number of input vars */
 int   cvarIn;   /* number of actual input vars */
 int   cvarAct;  /* number of actual input vars */
 int   ivar;     /* index */
 int   cData;  /* cvarMax + 2, i.e., input vars, output var, & degree */
 int   cdsTrain;
 int   idsTrain;
 float  fDelta;
 int     cRegion;  /* initial number of regions for each var */

 /* open the debug file */
 pfileLog = fopen("fno.log", "w");

fprintf(pfileLog, "*** FUZZY SYSTEM TRAINING ***\n\n");

 FscDisplayHelp(cArg, apszArg);

 FscOpenFiles(cArg, apszArg, pfsc->apfile, FSC_C_FILE);

 /* get data */
 FscGetFloat(pfsc->apfile[FSC_I_INFILE], &fDelta);
 FscGetInteger(pfsc->apfile[FSC_I_INFILE], &cvarMax);
 FscGetInteger(pfsc->apfile[FSC_I_INFILE], &cRegion);

 pfsc->fDelta = fDelta;
 pfsc->vi.cvarMax = cvarMax;
 pfsc->cRegion = cRegion;

 pfsc->vi.pvar = (PVAR) calloc(cvarMax, sizeof(VAR));
 if (!pfsc->vi.pvar)
   FscExit(pfsc, 0);
 FscGetVarStructList(pfsc);
 cvarAct = pfsc->vi.cvarAct;
 cvarIn = pfsc->vi.cvarIn;

 cData = cvarMax;
 pfsc->dsi.cData = cData;

 FscGetInteger(pfsc->apfile[FSC_I_INFILE], &cdsTrain);
 pfsc->dsi.cdsMax = cdsTrain;

 pfsc->dsi.pds = (PDATASET) calloc(cdsTrain, sizeof(DATASET));
 if (!pfsc->dsi.pds)
   FscExit(pfsc, 0);

 pfsc->dsi.pflIn = (PFLOAT) calloc(cdsTrain * cData, sizeof(float));
 if (!pfsc->dsi.pflIn)
   FscExit(pfsc, 0);

 pfsc->pflIn = (PFLOAT) calloc(cvarIn, sizeof(float));
 if (!pfsc->pflIn)
   FscExit(pfsc, 0);

 int cfvTmp = pow(2, cvarIn);

 pfsc->pppfv = (PFV **) calloc(cfvTmp, sizeof(PFV));
 if (!pfsc->pppfv)
   FscExit(pfsc, 0);

 pfsc->ppfv = (PFV *) calloc(cvarIn * cfvTmp, sizeof(PFV));
```

```
if (!pfsc->ppfv)
  FscExit(pfsc, 0);

for (i = 0; i < cfvTmp; i++)
{
  pfsc->pppfv[i] = pfsc->ppfv + cvarIn * i;
}/* FOR */

PVAR      pvar;
PDATASET pdsTrain;
PFLOAT   pflnTrain;
PRULE    *pprule;

pvar = pfsc->vi.pvar;
pdsTrain = pfsc->dsi.pds;
pflnTrain = pfsc->dsi.pfln;
pprule = pfsc->aprule;

for (idsTrain = 0; idsTrain < cdsTrain; idsTrain++)
{
  pdsTrain[idsTrain].pfln = pflnTrain + cData * idsTrain;
}/* FOR */

for (i = 0; i < FSC_MAX_REGIONS; i++)
{
  pfsc->aprule[i] = (PRULE) _fcalloc(9 * 9 * 9 * 9, sizeof(FV));
  if (!pfsc->aprule[i])
  {
    fprintf(pfileLog, "main:fcalloc failed: sizeof(FV)=%d", sizeof(FV));
    FscExit(pfsc, 0);
  }/* IF */
}/* FOR */

FscGetDataSetList(pfsc);

if (cArg >= 2 && 0 == strcmp(strupr(apszArg[1] + 1), "FNO"))
{
  FscGetOutputData(pfsc);
}/* IF */


PFILE  pfileOut;
int    cEmt0;

pfileOut = pfsc->apfile[FSC_I_OUTFILE];

pfsc->fRetrain = 0;
pfsc->iConflict = 0;

for (i = 0; i < cvarMax; i++)
{
  pvar[i].cRegion = (i != cvarMax - 2) ? cRegion : 25;
```

```
 }/* FOR */

 FscGetMinMax(pfsc);


 pfsc->fRetrain = 1;
 while (pfsc->fRetrain)
 {
  if (pfsc->iConflict == 1 && pvar[0].cRegion == FSC_MAX_REGIONS + 2)
  {
   pvar[0].cRegion = FSC_MAX_REGIONS;
   if (!FscRetrain(pfsc, &fDelta))
     break;
  }/* IF */

  pfsc->fRetrain = 0;
  FscGetEmtCount(pfsc);
  FscInitGrades(pfsc);


  /* begin the sequence of training */
  for (m = 0; m < cdsTrain; m++)
  {
   pfsc->fRetrain = FscGetGrades(pdsTrain[m].pfIn, &fDelta, pfsc, m);
   if (pfsc->fRetrain == 1)
     break;
  }/* FOR */
 }/* WHILE */

 /*** after the training, keep all the rules  ***/

 fprintf(pfileOut,"The updated fuzzy rule bases:\n\n");

 for (i = 0; i < cvarIn; i++)
 {
  fprintf(pfileOut, "main: i=%d, acR[i]=%d\n", i, pvar[i].cRegion);
 }/* FOR */

 cEmt0 = pfsc->vi.pvar[0].cEmt;


#if 1
 for (i = 0; i < pvar[0].cRegion; i++)
 {
  for (j = 0; j < cEmt0; j++)
  {
//DO NOT delete the following line
    if (pprule[i][j].fDegree > 0.0)
      fprintf(pfileOut,"(%d %d) (%d %5f)\n", i, j, pprule[i][j].iRegion,pprule[i][j].fDegree);
  }/* FOR */
 }/* FOR */
```

```
#endif

  fclose(pfileOut);

  /* test the fuzzy system */
  FscDefuzzify(pfsc);

  fcloseall();

  return (1);
}/* main */


/*** display max regions msg ***/
int FscRetrain(PFSC pfsc, PFLOAT pfDelta)
{
  int   ivar;
  float fDeltaTmp;

  printf("\nThe program reached the max number of regions.\n");
  printf("Please enter the following two items:\n");
  printf("  1) a smaller \"DELTA\" to Continue  OR  'X' to Exit.\n");
  printf("  2) <ENTER>\n\n");

  pszTmp = FscGetLine(szTmp, sizeof(szTmp), &cchTmp, stdin);

  if (toupper(szTmp[0]) != 'X')
  {
    fDeltaTmp = atof(szTmp);
    if (fDeltaTmp > 0.0 && fDeltaTmp < pfsc->fDelta)
    {
      pfsc->iConflict = 0;
      pfsc->fDelta = fDeltaTmp;
      *pfDelta = fDeltaTmp;
      for (ivar = 0; ivar < pfsc->vi.cvarIn; ivar++)
      {
        pfsc->vi.pvar[ivar].cRegion = pfsc->cRegion;
      }/* FOR */

      return(1);
    }/* IF */
    else
    {
      printf("Invalid delta.\n");
    }/* ELSE */
  }/* ELSE */

  return(0);
}/* FscRetrain */


/*** get output data ***/
```

```
int FscGetOutputData(PFSC pfsc)
{
  PDATASET pds = pfsc->dsi.pds;
  int cdsAct = pfsc->dsi.cdsAct;
  int ids;
  int iRet = 0;

  if (cdsAct == 0)
    return (0);

 fprintf(pfileLog, "B GetOutputD\n");

  for (ids = 0; ids < cdsAct; ids++)
  {
    float ffcf;

    ffcf = FscGetFcf(pds[ids].pfIn[FSC_I_FEED]);

#ifdef INCL_ENGLICH

    pds[ids].pfIn[FSC_I_OUTDATA] = 140000.0 * pds[ids].pfIn[FSC_I_SPEED] *
      pds[ids].pfIn[FSC_I_FEED] * 0.2 / 33000.0;
//     pow(pds[ids].pfIn[FSC_I_FEED], 0.8) * pow(5.0, 0.9) / 1000.0;

#endif /* INCL_ENGLICH */

/*** metric system (kw) ***/
//   pds[ids].pfIn[FSC_I_OUTDATA] = 16.088 * pds[ids].pfIn[FSC_I_SPEED] *
//   pds[ids].pfIn[FSC_I_FEED] * 2.0 * 5.0 * 1.14 / 1000.0;

/*** Formula *** 1996.10.2 ************************************/
// (s * 1000) / (wid * 3.1416) * (mmpt * 4 teeth) * wid * depth / (1000 * 60) * ukw
// = 1 / (3.1416 * 60) * s * mmpr * depth * ukw
// (full slot: wid = dia), no feed correction factor, no wear factor

/*** Formula *** 1997.2.18 *** Alum alloy ********************************/
// (s * 1000) / (wid * 3.1416) * (mmpt * 4 teeth) * wid * depth / (1000 * 60) * ukw
// = 1 / (3.1416 * 15) * s * mmpt * depth * ukw * ffcf * FSC_F_WEAR

pds[ids].pfIn[FSC_I_OUTDATA] = pds[ids].pfIn[FSC_I_DEPTH] *
    pds[ids].pfIn[FSC_I_SPEED] * pds[ids].pfIn[FSC_I_FEED] *
    0.0212206 * 0.8274 * ffcf * FSC_F_WEAR;

  }/* FOR */

 fprintf(pfileLog, "E GetOutputD\n");

  return(iRet);
}/* FscGetOutputData */


/******************************************/
```

```
int FscInitGrades(PFSC pfsc)
{
 int i, j; //tc , iEmt;
 PRULE *pprule;

 pprule = pfsc->aprule;

 for (i = 0; i < pfsc->vi.pvar[0].cRegion; i++)
 {
  for (j = 0; j < pfsc->vi.pvar[0].cEmt; j++)
  {
   pprule[i][j].iRegion = FSC_I_INITREGION;
   pprule[i][j].fDegree = 0.0;
  }/* FOR */
 }/* FOR */

 return(1);
}/* FscInitGrades */


/*****************************************/
int FscGetGrades(PFLOAT pfIn, PFLOAT pfDelta, PFSC pfsc, int iDs)
{
 int i, iEmt;
 float fProdGrade:
 PVAR pvar = pfsc->vi.pvar;
 int cvarIn = pfsc->vi.cvarIn;
 int iRgnTmp;
 PRULE *pprule:
 int iOutRgn;
 int iConflict;


 pprule = pfsc->aprule;
 fProdGrade = 1.0;

 for (i = 0; i < (cvarIn + 1); i++)
 {
  FscGetFvArray(pfsc, iDs, pfIn[i], pvar[i].fMin, pvar[i].fMax, pvar[i].fInterval, pvar[i].afv);


  /* grade degree for i element of the input vector */
  fProdGrade = fProdGrade * pvar[i].afv[0].fDegree;
 }/* FOR */

 iOutRgn = pvar[cvarIn].afv[0].iRegion;

 /* define the rule for this data pair */

 iRgnTmp = pvar[0].afv[0].iRegion;

 for (iEmt = 0, i = 1; i < cvarIn; i++)
```

```
  {
   iEmt += pvar[i].afv[0].iRegion * pvar[i].cEmt;

  }/* FOR */


  if ((fabs(pprule[iRgnTmp][iEmt].fDegree - fProdGrade) < *pfDelta) &&
    (pprule[iRgnTmp][iEmt].iRegion != FSC_I_INITREGION) &&
    (pprule[iRgnTmp][iEmt].iRegion != iOutRgn))
  {
   pfsc->fRetrain = 1;
   iConflict = pfsc->iConflict;
   pvar[iConflict].cRegion += 2;
   pvar[iConflict].fInterval = (pvar[iConflict].fMax - pvar[iConflict].fMin) /
     (pvar[iConflict].cRegion - 1);

   if (pvar[cvarIn].cRegion < pvar[iConflict].cRegion &&
     pvar[cvarIn].cRegion < FSC_MAX_REGIONS)
   {
    pvar[cvarIn].cRegion += 2;
    pvar[cvarIn].fInterval = (pvar[cvarIn].fMax - pvar[cvarIn].fMin) /
      (pvar[cvarIn].cRegion - 1);
   }/* IF */

   pfsc->iConflict += 1;

   if (pfsc->iConflict == cvarIn)
   {
    *pfDelta -= (pfsc->fDelta / 3.0);
    pfsc->iConflict = 0;
   }/* IF */


  fclose(pfileLog);
  pfileLog = fopen("fno.log", "a");

  }/* IF */

  else if ((pprule[iRgnTmp][iEmt].iRegion == FSC_I_INITREGION) ||
  fabs(pprule[iRgnTmp][iEmt].fDegree - fProdGrade) >= *pfDelta)
  {

   pprule[iRgnTmp][iEmt].iRegion = iOutRgn;
   pprule[iRgnTmp][iEmt].fDegree = fProdGrade;
  }/* ELSE */

  return (pfsc->fRetrain);
}/* FscGetGrades */


/*** fns.cpp ***/
```

```
/**********************************\
 *
 * FNSIN.CPP -- user interface program (borlandc c++)
 *
 *           By Ted C. Chang              *
 *
\**********************************/

#include <process.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <math.h>
#include <conio.h>
#include <float.h>
#include <malloc.h>  // ffree
#include <ctype.h>
#include "fno.h"


extern PFILE pfileLog;

FCF afcf[FSC_C_FCF] = {{0.025,1.6}, {0.075,1.4}, {0.125,1.25}, {0.175,1.18}, {0.225,1.06},
  {0.275,0.95}, {0.325,0.92}}; /* array of feed correction factors */

int FscExit(PVOID pv, int iStatus)
{
 PFSC pfsc = (PFSC) pv;
 int   irule;

 if (pfsc)
 {

  if (pfsc->vi.pvar)
   free(pfsc->vi.pvar);

  if (pfsc->dsi.pds)
   free(pfsc->dsi.pds);

  if (pfsc->dsi.pfIn)
   free(pfsc->dsi.pfIn);

  for (irule = 0; irule < FSC_MAX_REGIONS; irule++)
  {
   if (pfsc->aprule[irule])
        _ffree(pfsc->aprule[irule]);
  }/* FOR */

  if (pfsc->pfIn)
   free(pfsc->pfIn);
   if (pfsc->pppfv)
     free(pfsc->pppfv);

   if (pfsc->ppfv)
     free(pfsc->ppfv);

  }/* IF */

  fcloseall();
  exit(iStatus);

  return (iStatus);
}/* FscExit */


char *FscGetLine(char *pszText, int cchMax, int *pcchText, PFILE pfile)
{
 int fFull;  /* full line ? */
 char *pszLine;
 int fDone = 0;
 int chRead;
 int cchText;


 while (!fDone)
 {
  *pszText = 0;
   pszLine = fgets(pszText, cchMax, pfile);
   if (!pszLine)
     return (NULL);

   cchText = strlen(pszText);
   fDone = (!(cchText == 1 && (pszText[0] == '\n' || pszText[0] == '\r')) &&
          !(cchText >= 2 && pszText[0] == '/' && pszText[1] == '/'));

  fFull = pszText[cchText - 1] == '\n' || pszText[cchText - 1] == '\r';

  if (fFull)
  {
   pszText[cchText - 1] = 0;
   cchText--;
  }/* IF */
  else
  {
   /* read extra chars */
   while ((chRead = getc(pfile)) != '\n' && chRead != '\r' && chRead != EOF)
         ;
  }/* ELSE */
 }/* WHILE */

 *pcchText = cchText;
 return (pszText);
}/* FscGetLine */


int FscDisplayHelp(int cArg, char *apszArg[])
{
 int fHelp = 0;
 int fDefault = 0;
 int fInOut = 0;
 int iArg;
 char chOpt;

 for (iArg = 1; iArg < cArg; iArg++)
 {
  if (strlen(apszArg[iArg]) >= 2 && (apszArg[iArg][0] == '/' || apszArg[iArg][0] == '-'))
  {
   chOpt = toupper(apszArg[iArg][1]);
   if (chOpt == 'H')
   {
        fHelp = 1;
   }/* IF */
   else if (chOpt == 'D')
   {
        fDefault = 1;
   }/* ELSEIF */
   else if (strlen(apszArg[iArg]) >= 3 && (chOpt == 'I' || chOpt == 'O')
              && apszArg[iArg][2] == ':')
   {
        fInOut = 1;
   }/* ELSEIF */
  }/* IF */
 }/* FOR */

 if (!fDefault && !fInOut)
  printf("\nFNO [(/|-)(h|H)[elp]] [(/|-)(fno|FNO)] [(/|-)(d|D)] [(/|-)I:filename] [(/|-)O:filename]\n");

 if (fHelp)
 {
  printf("\nSWITCHES:\n");
  printf("  1) / or - :: leading character of a switch parameter.\n");
  printf("  2) h or H :: display help information.\n");
  printf("  3) fno or FNO :: machining optimization.\n");
  printf("  4) d or D :: open default filenames, i.e., fscin.dat & fscout.dat.\n");
  printf("  5) I: or O:filename :: input/output filename.\n");
  printf("\nEXAMPLES:\n");
  printf("  1) FSC /h\n");
  printf("  2) FSC /D\n");
  printf("  3) FSC -I:fscin.dat -O:fscout.dat\n");
  printf("  4) FSC /I:fscin.dat\n");
  printf("  5) FSC /h -O:fscout.dat\n\n");
 }/* IF */
```

```
  return (1);
 }/* FscDisplayHelp */


int FscGetFilename(char *pszInOut, char *pszFilename, int cchFilename)
{
  char *pszLine;
  int cchFn;

  printf("\nPlease enter the following two items:\n");
  printf("  1) %s filename to Continue OR 'X' to Exit.  (length <= 128 chars).\n", pszInOut);
  printf("  2) <ENTER>\n\n");

  pszLine = FscGetLine(pszFilename, cchFilename, &cchFn, stdin);

  if (!pszLine)
    return (0);
  else if (toupper(pszFilename[0]) == 'X')
  {
    FscExit(NULL, 0);
  }/* ELSE */

  return (1);
}/* FscGetFilename */


int FscOpenFiles(int cArg, char *apszArg[], PFILE *ppfile, int cfile)
{
  char szInFile[FSC_CCH_FILENAME + 1] = "";
  char szOutFile[FSC_CCH_FILENAME + 1] = "";
  int  ifile = 0;
  int cpfile;
  int iArg;

  for (ifile = 0; ifile < cfile; ifile++)
  {
    ppfile[ifile] = NULL;
  }/* FOR */

  for (iArg = 1; iArg < cArg; iArg++)
  {
    if (strlen(apszArg[iArg]) == 2 && (apszArg[iArg][0] == '/' || apszArg[iArg][0] == '-') &&
        toupper(apszArg[iArg][1]) == 'D')
    {
      strcpy(szInFile, FSC_SZ_INFILE);
      strcpy(szOutFile, FSC_SZ_OUTFILE);
    }/* IF */
    else if (strlen(apszArg[iArg]) >= 3 && (apszArg[iArg][0] == '/' || apszArg[iArg][0] == '-') &&
        apszArg[iArg][2] == ':')
    {
      if (toupper(apszArg[iArg][1]) == 'I')
```

```
        strcpy(szInFile, apszArg[iArg] + 3);
    else if (toupper(apszArg[iArg][1]) == 'O')
        strcpy(szOutFile, apszArg[iArg] + 3);
  }/* IF */
 }/* FOR */

 if (szInFile[0])
  ppfile[FSC_I_INFILE] = fopen(szInFile, "r");

 if (szOutFile[0])
 {
  if (strcmp(szInFile, szOutFile))
   ppfile[FSC_I_OUTFILE] = fopen(szOutFile, "w");
  else
  {
   printf("\nInput and output filenames should be different !!!\n");
   printf("Press a key to continue . . .\n");
  }/* ELSE */
 }/* IF */

cpfile = 0;
while (cpfile != cfile)
{
 for (ifile = 0, cpfile = 0; ifile < cfile; ifile++)
 {
  if (ppfile[ifile] != NULL)
  {
   cpfile++;
  }
  else
  {
   if (ifile == FSC_I_INFILE)
   {
        FscGetFilename("input", szInFile, FSC_CCH_FILENAME);
        ppfile[ifile] = fopen(szInFile, "r");
   }
   else if (ifile == FSC_I_OUTFILE)
   {
        FscGetFilename("output", szOutFile, FSC_CCH_FILENAME);
        if (strcmp(szInFile, szOutFile))
        {
         ppfile[ifile] = fopen(szOutFile, "w");
        }
        else
        {
         printf("\nInput and output filenames should be different !!!\n");
        }/* ELSE */
     }/* ELSE IF */
  }/* IF */
 }/* FOR */
}/* WHILE */
```

```
  return (1);
 }/* FscOpenFiles */


 int FscGetInteger(PFILE pfile, int *piNum)
 {
  char szNum[FSC_CCH_NUMBER + 1] = "";
  int cchNum;

  FscGetLine(szNum, FSC_CCH_NUMBER, &cchNum, pfile);

  *piNum = atoi(szNum);

  return (1);
 }/* FscGetInteger */


 int FscGetFloat(FILE *pfileData, float *pfData)
 {
  char szLine[FSC_CCH_LINE + 1] = "";
  int iRet = 0;
  int cchLn;

  if (FscGetLine(szLine, FSC_CCH_LINE + 1, &cchLn, pfileData))
    iRet = sscanf(szLine, "%f", pfData);

  return(iRet);
 }/* FscGetFloat */


 /* get a word from a buffer of a line */
 int FscGetWord(char **ppszLine, int *pichWord, char *pszWord, int cchWord)
 {
  char *pszLine = *ppszLine;
  int ichWord;

  if (!ppszLine || !pszWord || cchWord <= 0)
    return(0);

  /* <SPACE>, <TAB>, and <COMMA> are delimiters */
  while (*pszLine == ' ' || *pszLine == '\t' &&  *pszLine == ',')
  {
   pszLine++;
  }/* while */

  *pszWord = 0;
  ichWord = 0;
  /* <SPACE>, <TAB>, and <COMMA> are delimiters */
  while (*pszLine != 0 &&
   *pszLine != ' ' && *pszLine != '\t' &&  *pszLine != ',' &&  ichWord < cchWord)
  {
   *pszWord = *pszLine;
   pszLine++;
   pszWord++;
   ichWord++;
  }/* while */

  *pszWord = 0;
  *pichWord = ichWord;
  *ppszLine = pszLine;

  return(*pszLine);
}/* FscGetWord */


/* get input struct */
int FscGetVarStruct(char *pszLine, PVAR pvar, int cData)
{
  char szWord[FSC_CCH_NAME + 1];
  int ichWord;
  int iRet = 0;
  int iData;

  if (!pszLine || !pvar || cData <= 0)
   return(0);

  ichWord = 0;
  iData = 0;
  while (iData < cData)
  {
   iRet = FscGetWord(&pszLine, &ichWord, szWord, FSC_CCH_NAME);

   if (ichWord == 0)
   {
    break;
   }/* IF */

   if (iData == 0)
   {
    strcpy(pvar->szName, szWord);
   }
   else if (iData == 1)
   {
    strcpy(pvar->szNick, szWord);
   }/* ELSE IF */
   else
   {
    pvar->fInterval = atof(szWord);
   }/* ELSE */

   if (iRet == 0 || iRet == '\n' || iRet == '\r')
    break;
   iData++;
  }/* while */

  return(iRet);
}/* FscGetVarStruct */


/* get fv out region */
int FscGetOutRegion(PFSC pfsc, int ifv)
{
  PVAR pvar = pfsc->vi.pvar;
  PFV *ppfv = pfsc->pppfv[ifv];
  PRULE *pprule;
  int cvarIn = pfsc->vi.cvarIn;
  int ivar;
  int iRegion;
  int iEmt;

  pprule = pfsc->aprule;
  for (iEmt = 0, ivar = 1; ivar < cvarIn; ivar++)
  {
   iEmt += ppfv[ivar]->iRegion * pvar[ivar].cEmt;
  }/* FOR */

  iRegion = pprule[ppfv[0]->iRegion][iEmt].iRegion;

  return(iRegion);
}/* FscGetOutRegion */


/* calculate centers */
int FscCalcCenters(PFSC pfsc, PFLOAT pfCenter, int ivar)
{
  PVAR pvar = pfsc->vi.pvar;
  float fInterval;
  int cRegion;
  int iRegion;
  int iEmt;

  cRegion = pvar[ivar].cRegion;
  fInterval = (pvar[ivar].fMax - pvar[ivar].fMin) / (cRegion - 1);

  for (iRegion = 0; iRegion < cRegion; iRegion++)
  {
   pfCenter[iRegion] = pvar[ivar].fMin + fInterval * iRegion;
  }/* FOR */

  return(1);
}/* FscCalcCenters */


/* get element count */
```

```
int FscGetEmtCount(PFSC pfsc)
{
 PVAR pvar = pfsc->vi.pvar;
 int cvarIn = pfsc->vi.cvarIn;
 int ivar;
 int iRet = 0;

 for (ivar = cvarIn - 1; ivar > -1; ivar--)
 {
  if (ivar == cvarIn - 1)
  {
   pvar[ivar].cEmt = 1;
  }/* IF */
  else
  {
   pvar[ivar].cEmt = pvar[ivar + 1].cEmt * (pvar[ivar + 1].cRegion);
  }/* ELSE */
 }/* FOR */

 return(iRet);
}/* FscGetEmtCount */


/* get var struct list */
int FscGetVarStructList(PFSC pfsc)
{
 PFILE pfileData = pfsc->apfile[FSC_I_INFILE];
 PVAR pvar = pfsc->vi.pvar;
 int cvarMax = pfsc->vi.cvarMax;
 int ivar;
 int cData = FSC_C_INDATA;
 char szLine[FSC_CCH_LINE + 1];
 char *pszLine;
 int cchLn;
 int iRet = 0;


 for (ivar = 0; ivar < cvarMax; ivar++)
 {
  /* init var struct */
  pvar[ivar].szName[0] = 0;
  pvar[ivar].szNick[0] = 0;
  pvar[ivar].fMax = 0.0;
  pvar[ivar].fMin = 0.0;
  pvar[ivar].fInterval = 0.0;
  pvar[ivar].cRegion = (ivar != cvarMax - 2) ? pfsc->cRegion : (pfsc->cRegion + 4);
  pvar[ivar].cEmt = 0;
  pvar[ivar].afv[0].iRegion = 999;
  pvar[ivar].afv[0].fDegree = 0.0;
  pvar[ivar].afv[1].iRegion = 999;
  pvar[ivar].afv[1].fDegree = 0.0;
  pszLine = FscGetLine(szLine, FSC_CCH_LINE + 1, &cchLn, pfileData);
  if (!pszLine)
  {
   fprintf(pfileLog, "# Input Vars Read Incomplete\n");
   break;
  }/* IF */
  else
   fprintf(pfileLog, "ivar=%d, pszL = %s\n", ivar, pszLine);

  FscGetVarStruct(pszLine, pvar + ivar, cData);
 }/* FOR */

 pfsc->vi.cvarAct = ivar;
 pfsc->vi.cvarIn = ivar - 2;

 FscGetEmtCount(pfsc);

 return(iRet);
}/* FscGetVarStructList */


/* get a data set */
int FscGetDataSet(char *pszLine, PFLOAT pflIn, int cData)
{
 char szWord[FSC_CCH_FLOAT + 1];
 int ichWord;
 int iRet = 0;
 int iData;

 if (!pszLine || !pflIn || cData <= 0)
  return(0);

 ichWord = 0;
 iData = 0;
 while (iData < cData)
 {
  iRet = FscGetWord(&pszLine, &ichWord, szWord, FSC_CCH_FLOAT);

  if (ichWord == 0)
  {
   break;
  }/* IF */

  pflIn[iData] = atof(szWord);

  iData++;

  if (iRet == 0 || iRet == '\n' || iRet == '\r')
   break;
 }/* while */
 return(iData);
}/* FscGetDataSet */
```

```
/* get data sets */
int FscGetDataSetList(PFSC pfsc)
{
 PFILE pfileData = pfsc->apfile[FSC_I_INFILE];
 PDATASET pds = pfsc->dsi.pds;
 int cdsMax = pfsc->dsi.cdsMax;
 int cData = pfsc->dsi.cData;
 int iData;
 int ids;
 char szLine[FSC_CCH_LINE + 1];
 char *pszLine;
 int cchLn;
 int iRet = 0;


 fprintf(pfileLog, "Begin GetDataSetList:\n");

 for (ids = 0; ids < cdsMax; ids++)
 {
  /* init float list */
  for (iData = 0; iData < cData; iData++)
   pds[ids].pflIn[iData] = 0.0;

  pszLine = FscGetLine(szLine, FSC_CCH_LINE + 1, &cchLn, pfileData);
  if (!pszLine)
  {
   fprintf(pfileLog, "# Data Sets Read Incomplete\n");
   break;
  }/* IF */
//tc DO NOT DELETE
//  else
//    fprintf(pfileLog, "[%d] %s\n", ids, pszLine);

  FscGetDataSet(pszLine, (pds + ids)->pflIn, cData);
 }/* FOR */

 pfsc->dsi.cdsAct = ids;

 fprintf(pfileLog, "End GetDataSetList:\n");

 return(iRet);
}/* FscGetDataSetList */


/* get min & max */
int FscGetMinMax(PFSC pfsc)
{
 PVAR pvar = pfsc->vi.pvar;
 int  ivar;
 PDATASET pds = pfsc->dsi.pds;
 int cdsAct = pfsc->dsi.cdsAct;
 int cData = pfsc->vi.cvarAct;  /* input + output vars */
 int ids;
 int iRet = 0;

 if (cdsAct == 0)
  return (0);

 /* get max/min number */
 for (ivar = 0; ivar < cData; ivar++)
 {
  pvar[ivar].fMax = pds[0].pfIn[ivar];
  pvar[ivar].fMin = pds[0].pfIn[ivar];
 }/* FOR */

 for (ids = 1; ids < cdsAct; ids++)
 {
  /* get max/min number */
  for (ivar = 0; ivar < cData; ivar++)
  {
   if (pvar[ivar].fMax < pds[ids].pfIn[ivar])
        pvar[ivar].fMax = pds[ids].pfIn[ivar];
   if (pvar[ivar].fMin > pds[ids].pfIn[ivar])
        pvar[ivar].fMin = pds[ids].pfIn[ivar];
  }/* FOR */
 }/* FOR */

 /* get max/min number */
 for (ivar = 0; ivar < cData; ivar++)
 {
  pvar[ivar].fInterval = (pvar[ivar].fMax - pvar[ivar].fMin) / (pvar[ivar].cRegion - 1);

 }/* FOR */

 return(iRet);
}/* FscGetMinMax */


/* build fv array for defuzzfication */
int FscBuildFvArray(PFSC pfsc)
{
 PVAR pvar = pfsc->vi.pvar;
 int cvarIn = pfsc->vi.cvarIn;
 int ivarIn;
 PFV **pppfv;
 int ifvCycle, ifv, cfvCycle, cfv;
 int iRet = 0;

 pppfv = pfsc->pppfv;
 cfv = pow(2, cvarIn);
 for (ivarIn = 0; ivarIn < cvarIn; ivarIn++)
 {
    cfvCycle = pow(2, cvarIn - ivarIn - 1);
    ifv = 0;
    while (ifv < cfv)
    {
     for (ifvCycle = 0; ifvCycle < cfvCycle; ifvCycle++)
     {
          pppfv[ifv][ivarIn] = pvar[ivarIn].afv;

          ifv++;
     }/* FOR */
     for (ifvCycle = 0; ifvCycle < cfvCycle; ifvCycle++)
     {
          pppfv[ifv][ivarIn] = pvar[ivarIn].afv + 1;
          ifv++;
     }/* FOR */
    }/* WHILE */
  }/* FOR */

  return(iRet);
}/* FscBuildFvArray */


/*** get fv array. store the bigger in the first ***/
int FscGetFvArray(PFSC pfsc, int ids, float fIn, float fMin, float fMax, float fInterval, PFV pfv)
{
  int  iRegion = 0;
  int  ifv = 0;
  float fDegree = 0.0;
  float fCenter = 0.0;
  float fDiff = 0.0;

if (ids < FSC_C_DS_DEBUG)
  fprintf(pfileLog, "B GetFvA\n");

//tc0728
 if (fMin == fMax)
 {
  if (ids < FSC_C_DS_DEBUG)
  {
   fprintf(pfileLog, "\n!!! Unreasonable Values: fMin == fMax\n");
   fprintf(pfileLog, "You need the \"fno\" switch if machining.\n");
  }

   FscExit(pfsc, 0);
 }else if (fIn == fMax)
 {
   fDiff = fMax - fMin;
   iRegion = fDiff / fInterval;

//tc
if (ids < FSC_C_DS_DEBUG)
 fprintf(pfileLog, "iR=%d,fV=%6.3f,fM=%6.3f,fM=%6.3f,fI=%6.3f\n",
  iRegion, fIn, fMax, fMin, fInterval);

  pfv[0].iRegion = iRegion;
  pfv[0].fDegree = 1.0;
  pfv[1].iRegion = iRegion - 1;
  pfv[1].fDegree = 0.0;

 }
 else if (fIn == fMin)
 {
pfv[0].iRegion = 0;
  pfv[0].fDegree = 1.0;
  pfv[1].iRegion = 1;
  pfv[1].fDegree = 0.0;

 }
 else
 {

 fDiff = fIn - fMin;
/*** 1997.2.6 *** - 0.001 ***/
 iRegion = fDiff / (fInterval - 0.001);

 if (fIn == fCenter)
 {

//tc 1997.2.5
  pfv[0].iRegion = iRegion;
  pfv[0].fDegree = 1.0;
  pfv[1].iRegion = iRegion - 1;
  pfv[1].fDegree = 0.0;
 }else

 for (ifv = 0; ifv < FSC_C_FV; ifv++)
 {
  iRegion += ifv;
  pfv[ifv].iRegion = iRegion;

  fCenter = fMin + fInterval * iRegion;

 if (fIn > fCenter)
   fDegree = 1 - (fIn - fCenter) / fInterval;
  else
   fDegree = 1 - (fCenter - fIn) / fInterval;

  if (fDegree < 0.0)
   fDegree = 0.0;

fprintf(pfileLog, "GetFv: fD=%6.3f\n",fDegree);

  pfv[ifv].fDegree = fDegree;
 } /* FOR */
```

```
  if (pfv[0].fDegree < pfv[1].fDegree)
  {
   iRegion = pfv[0].iRegion;
   fDegree = pfv[0].fDegree;
   pfv[0].iRegion = pfv[1].iRegion;
   pfv[0].fDegree = pfv[1].fDegree;
   pfv[1].iRegion = iRegion;
   pfv[1].fDegree = fDegree;
  }/* IF */
 }/* ELSE */

 if (ids < FSC_C_DS_DEBUG)
  fprintf(pfileLog, "E GetFvA\n");

  return(1);
}/* FscGetFvArray */


/* data set is valid ? */
int FscIsDataSetValid(PFSC pfsc)
{
 PVAR pvar = pfsc->vi.pvar;
 int cvarIn = pfsc->vi.cvarIn;
 int ivarIn;

 for (ivarIn = 0; ivarIn < cvarIn; ivarIn++)
 {
  if (pfsc->pfIn[ivarIn] < pvar[ivarIn].fMin || pfsc->pfIn[ivarIn] > pvar[ivarIn].fMax)
   return(0);
 }/* FOR */

 return(1);
}/* FscIsDataSetValid */


/* get feed correct factor for a given: feed */
float FscGetFcf(float fFeed)
{
 int ifcf;
 float ffcf;

 /* given: feed; calc: factor */
 for (ifcf = 0; ifcf < FSC_C_FCF; ifcf++)
 {
  if (ifcf < (FSC_C_FCF - 1) && fFeed >= afcf[ifcf].fFeed &&
    fFeed < afcf[ifcf + 1].fFeed)
  {
   ffcf = afcf[ifcf].fFactor - ((afcf[ifcf].fFactor - afcf[ifcf + 1].fFactor) /
    (afcf[ifcf + 1].fFeed - afcf[ifcf].fFeed) * (fFeed - afcf[ifcf].fFeed));
   break; /* done */
  }
```

```
    else
    {
     ffcf = afcf[FSC_C_FCF - 1].fFactor;
    }
   }/* FOR */

   return(ffcf);
  }/* FscGetFcf */


/* Defuzzify */
int FscDefuzzify(PFSC pfsc)
{
 PVAR pvar = pfsc->vi.pvar;
 int cvarIn = pfsc->vi.cvarIn;
 int ivarIn;
 int ifv, cfv;
 int iRgn;
 float fDegree = 0;
 float fSum1 = 0;
 float fSum2 = 0;
 char szLine[FSC_CCH_LINE + 1] = "";
 char *pszLine;
 int cchLn;
 int iRet = 0;
 PFV **pppfv;

//tcdebug
 float fOutData;

 cfv = pow(2, cvarIn);
 pppfv = pfsc->pppfv;
 FscCalcCenters(pfsc, pfsc->afOutCenter, pfsc->vi.cvarIn);

 printf("\n*** FUZZY SYSTEM TESTING ***\n");

//tc
 fprintf(pfileLog, "\n\n*** FUZZY SYSTEM TESTING ***\n\n");

 while (1)
 {
 printf("\nPlease enter the following two items:\n");
 printf("  1) input data set to Continue OR 'X' to Exit.  (length <= 128 chars).\n");
 printf("  2) <ENTER>\n\n");

 pszLine = FscGetLine(szLine, FSC_CCH_LINE + 1, &cchLn, stdin);

 if (!pszLine)
 {
  continue;
 }
 else if (toupper(pszLine[0]) == 'X')
 {
  FscExit(pfsc, 0);
 }/* ELSE */

 iRet = FscGetDataSet(pszLine, pfsc->pfIn, cvarIn);
 if (iRet < cvarIn)
 {
  printf("Incomplete data set !!!\n");
  continue;
 }/* IF */

 if (!FscIsDataSetValid(pfsc))
 {
  printf("Invalid data set !!!\n");
  continue;
 }/* IF */

 for (ivarIn = 0; ivarIn < cvarIn; ivarIn++)
 {
  FscGetFvArray(pfsc, FSC_C_DS_DEBUG, pfsc->pfIn[ivarIn], pvar[ivarIn].fMin,
     pvar[ivarIn].fMax, pvar[ivarIn].fInterval, pvar[ivarIn].afv);

  pvar[ivarIn].afv[0].iRegion,pvar[ivarIn].afv[0].fDegree,pvar[ivarIn].afv[1].iRegion,
  pvar[ivarIn].afv[1].fDegree);

 }/* FOR */

 FscBuildFvArray(pfsc);

 fSum1 = fSum2 = 0;
 for (ifv = 0; ifv < cfv; ifv++)
 {
  iRgn = FscGetOutRegion(pfsc, ifv);

//tc to be checked
  if (iRgn == FSC_I_INITREGION)
  {
   continue;
  }/* IF */

  fDegree = 1.0;
  for (ivarIn = 0; ivarIn < cvarIn; ivarIn++)
  {
   fDegree *= pppfv[ifv][ivarIn]->fDegree;

  }/* FOR */

  fSum1 += (fDegree * pfsc->afOutCenter[iRgn]);

  fSum2 += fDegree;
 }/* FOR */
```

```
  if (fSum2 <= 0.0)
  {
   printf("\nNom = %f, Denominator (%f) <= 0.0\n", fSum1, fSum2);
  }
  else
  {
   float ffcf;

   printf("\ny = %f\n", fSum1 / fSum2);

   ffcf = FscGetFcf(pfsc->pfIn[FSC_I_FEED]);


  /* metric system (kw) */
   fOutData = pfsc->pfIn[FSC_I_DEPTH] * pfsc->pfIn[FSC_I_SPEED] *
     pfsc->pfIn[FSC_I_FEED] * 0.0212206 * 0.8274 * ffcf * FSC_F_WEAR;

//F=0.8274*A5*C5*1000

//tc
 fprintf(pfileLog, "Defuzzify: %6.3f,%6.3f,%6.3f,%6.3f,F=%6.3f,P=%6.3f,\n",
  pfsc->pfIn[0],pfsc->pfIn[1],pfsc->pfIn[2],ffcf,fOutData,fSum1/fSum2);
  }/* ELSE */

 }/* WHILE */

  return(iRet);
}/* FscDefuzzify */


/*** fnsin.cpp ***/
```

APPENDIX B. TRAINING DATA FOR ASSESSMENT

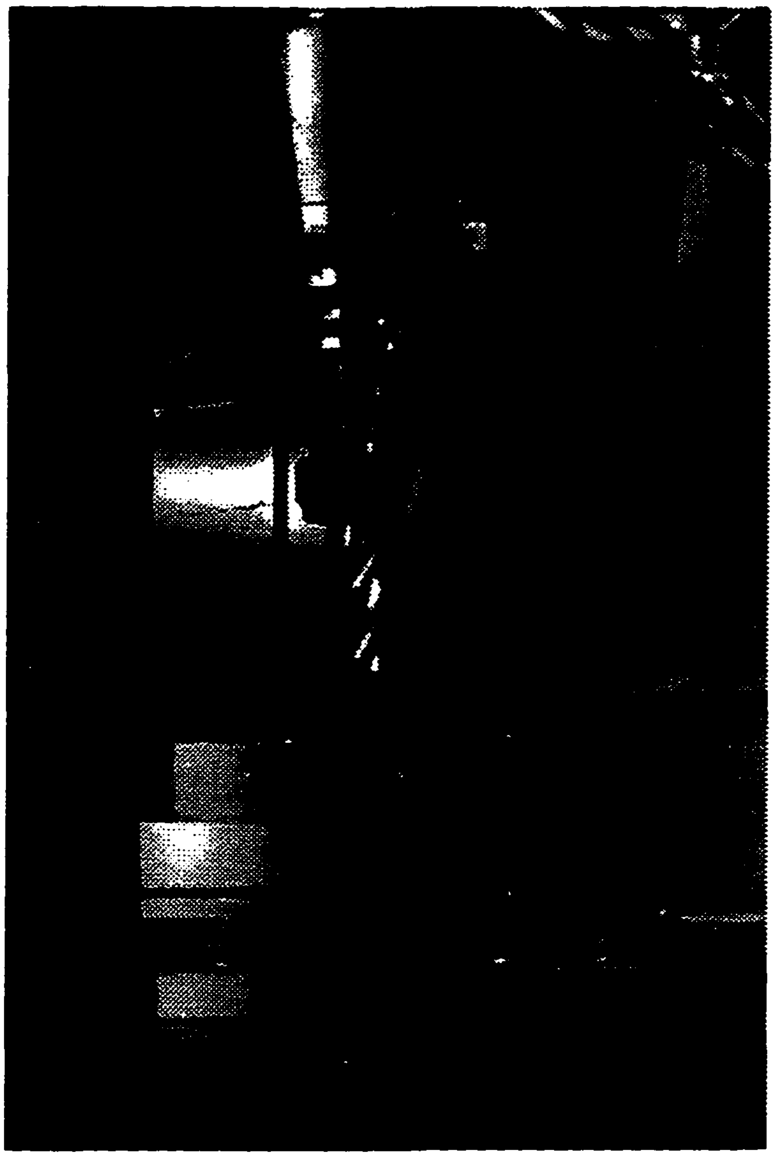
```
//*** input.dat -- for the fns ***

//delta
0.09

//number of (input variables + y + mu)
5

//initial number of regions for each variable
3

//input variables + y + mu
Depth Dc
Speed Ns
Feed Fd
//Cut_Depth Dc
//Workpiece_strength Ws
Out_Value Ov
Ds_Degree Dd

//number of data sets
343

//data sets

//Depth Speed Feed Output DataSetDegree

0.36 39 0.025 0.999 1
0.36 39 0.075 0.999 1
0.36 39 0.125 0.999 1
0.36 39 0.175 0.999 1
0.36 39 0.225 0.999 1
0.36 39 0.275 0.999 1
0.36 39 0.325 0.999 1

0.36 43.4 0.025 0.999 1
0.36 43.4 0.075 0.999 1
0.36 43.4 0.125 0.999 1
0.36 43.4 0.175 0.999 1
0.36 43.4 0.225 0.999 1
0.36 43.4 0.275 0.999 1
0.36 43.4 0.325 0.999 1

0.36 47.8 0.025 0.999 1
0.36 47.8 0.075 0.999 1
0.36 47.8 0.125 0.999 1
0.36 47.8 0.175 0.999 1
0.36 47.8 0.225 0.999 1
0.36 47.8 0.275 0.999 1
0.36 47.8 0.325 0.999 1

0.36 52.2 0.025 0.999 1
0.36 52.2 0.075 0.999 1
0.36 52.2 0.125 0.999 1
0.36 52.2 0.175 0.999 1
0.36 52.2 0.225 0.999 1
0.36 52.2 0.275 0.999 1
0.36 52.2 0.325 0.999 1

0.36 56.6 0.025 0.999 1
0.36 56.6 0.075 0.999 1
0.36 56.6 0.125 0.999 1
0.36 56.6 0.175 0.999 1
0.36 56.6 0.225 0.999 1
0.36 56.6 0.275 0.999 1
0.36 56.6 0.325 0.999 1

0.36 61 0.025 0.999 1
0.36 61 0.075 0.999 1
0.36 61 0.125 0.999 1
0.36 61 0.175 0.999 1
0.36 61 0.225 0.999 1
0.36 61 0.275 0.999 1
0.36 61 0.325 0.999 1

0.36 65.4 0.025 0.999 1
0.36 65.4 0.075 0.999 1
0.36 65.4 0.125 0.999 1
0.36 65.4 0.175 0.999 1
0.36 65.4 0.225 0.999 1
0.36 65.4 0.275 0.999 1
0.36 65.4 0.325 0.999 1

1.8 39 0.025 0.999 1
1.8 39 0.075 0.999 1
1.8 39 0.125 0.999 1
1.8 39 0.175 0.999 1
1.8 39 0.225 0.999 1
1.8 39 0.275 0.999 1
1.8 39 0.325 0.999 1
```

```
1.8 43.4 0.025 0.999 1
1.8 43.4 0.075 0.999 1
1.8 43.4 0.125 0.999 1
1.8 43.4 0.175 0.999 1
1.8 43.4 0.225 0.999 1
1.8 43.4 0.275 0.999 1
1.8 43.4 0.325 0.999 1

1.8 47.8 0.025 0.999 1
1.8 47.8 0.075 0.999 1
1.8 47.8 0.125 0.999 1
1.8 47.8 0.175 0.999 1
1.8 47.8 0.225 0.999 1
1.8 47.8 0.275 0.999 1
1.8 47.8 0.325 0.999 1

1.8 52.2 0.025 0.999 1
1.8 52.2 0.075 0.999 1
1.8 52.2 0.125 0.999 1
1.8 52.2 0.175 0.999 1
1.8 52.2 0.225 0.999 1
1.8 52.2 0.275 0.999 1
1.8 52.2 0.325 0.999 1

1.8 56.6 0.025 0.999 1
1.8 56.6 0.075 0.999 1
1.8 56.6 0.125 0.999 1
1.8 56.6 0.175 0.999 1
1.8 56.6 0.225 0.999 1
1.8 56.6 0.275 0.999 1
1.8 56.6 0.325 0.999 1

1.8 61 0.025 0.999 1
1.8 61 0.075 0.999 1
1.8 61 0.125 0.999 1
1.8 61 0.175 0.999 1
1.8 61 0.225 0.999 1
1.8 61 0.275 0.999 1
1.8 61 0.325 0.999 1

1.8 65.4 0.025 0.999 1
1.8 65.4 0.075 0.999 1
1.8 65.4 0.125 0.999 1
1.8 65.4 0.175 0.999 1
1.8 65.4 0.225 0.999 1
1.8 65.4 0.275 0.999 1
1.8 65.4 0.325 0.999 1

3.24 39 0.025 0.999 1
3.24 39 0.075 0.999 1
3.24 39 0.125 0.999 1
3.24 39 0.175 0.999 1
3.24 39 0.225 0.999 1
3.24 39 0.275 0.999 1
3.24 39 0.325 0.999 1

3.24 43.4 0.025 0.999 1
3.24 43.4 0.075 0.999 1
3.24 43.4 0.125 0.999 1
3.24 43.4 0.175 0.999 1
3.24 43.4 0.225 0.999 1
3.24 43.4 0.275 0.999 1
3.24 43.4 0.325 0.999 1

3.24 47.8 0.025 0.999 1
3.24 47.8 0.075 0.999 1
3.24 47.8 0.125 0.999 1
3.24 47.8 0.175 0.999 1
3.24 47.8 0.225 0.999 1
3.24 47.8 0.275 0.999 1
3.24 47.8 0.325 0.999 1

3.24 52.2 0.025 0.999 1
3.24 52.2 0.075 0.999 1
3.24 52.2 0.125 0.999 1
3.24 52.2 0.175 0.999 1
3.24 52.2 0.225 0.999 1
3.24 52.2 0.275 0.999 1
3.24 52.2 0.325 0.999 1

3.24 56.6 0.025 0.999 1
3.24 56.6 0.075 0.999 1
3.24 56.6 0.125 0.999 1
3.24 56.6 0.175 0.999 1
3.24 56.6 0.225 0.999 1
3.24 56.6 0.275 0.999 1
3.24 56.6 0.325 0.999 1

3.24 61 0.025 0.999 1
3.24 61 0.075 0.999 1
3.24 61 0.125 0.999 1
3.24 61 0.175 0.999 1
3.24 61 0.225 0.999 1
3.24 61 0.275 0.999 1
3.24 61 0.325 0.999 1

3.24 65.4 0.025 0.999 1
3.24 65.4 0.075 0.999 1
3.24 65.4 0.125 0.999 1
3.24 65.4 0.175 0.999 1
3.24 65.4 0.225 0.999 1
3.24 65.4 0.275 0.999 1
3.24 65.4 0.325 0.999 1

4.68 39 0.025 0.999 1
4.68 39 0.075 0.999 1
4.68 39 0.125 0.999 1
4.68 39 0.175 0.999 1
4.68 39 0.225 0.999 1
4.68 39 0.275 0.999 1
4.68 39 0.325 0.999 1

4.68 43.4 0.025 0.999 1
4.68 43.4 0.075 0.999 1
4.68 43.4 0.125 0.999 1
4.68 43.4 0.175 0.999 1
4.68 43.4 0.225 0.999 1
4.68 43.4 0.275 0.999 1
4.68 43.4 0.325 0.999 1

4.68 47.8 0.025 0.999 1
4.68 47.8 0.075 0.999 1
4.68 47.8 0.125 0.999 1
4.68 47.8 0.175 0.999 1
4.68 47.8 0.225 0.999 1
4.68 47.8 0.275 0.999 1
4.68 47.8 0.325 0.999 1

4.68 52.2 0.025 0.999 1
4.68 52.2 0.075 0.999 1
4.68 52.2 0.125 0.999 1
4.68 52.2 0.175 0.999 1
4.68 52.2 0.225 0.999 1
4.68 52.2 0.275 0.999 1
4.68 52.2 0.325 0.999 1

4.68 56.6 0.025 0.999 1
4.68 56.6 0.075 0.999 1
4.68 56.6 0.125 0.999 1
4.68 56.6 0.175 0.999 1
4.68 56.6 0.225 0.999 1
4.68 56.6 0.275 0.999 1
4.68 56.6 0.325 0.999 1

4.68 61 0.025 0.999 1
4.68 61 0.075 0.999 1
4.68 61 0.125 0.999 1
4.68 61 0.175 0.999 1
4.68 61 0.225 0.999 1
4.68 61 0.275 0.999 1
4.68 61 0.325 0.999 1

4.68 65.4 0.025 0.999 1
4.68 65.4 0.075 0.999 1
4.68 65.4 0.125 0.999 1
```

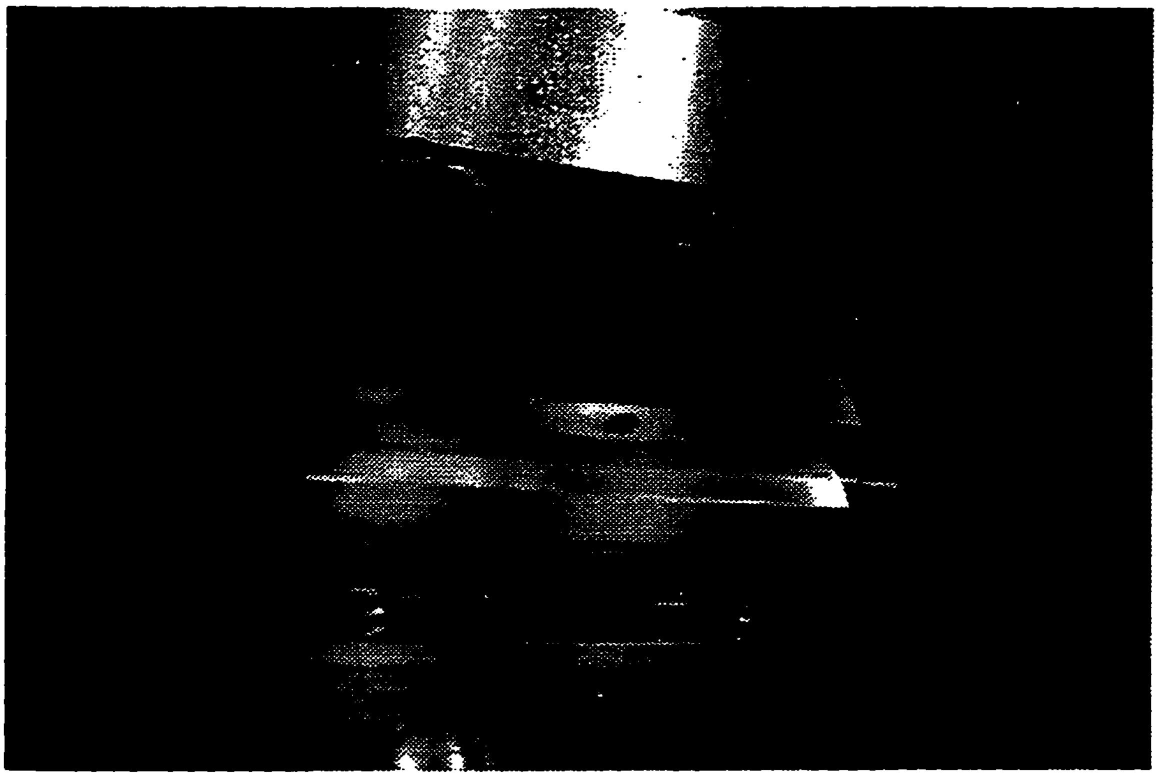
```
4.68 65.4 0.175 0.999 1
4.68 65.4 0.225 0.999 1
4.68 65.4 0.275 0.999 1
4.68 65.4 0.325 0.999 1

6.12 39 0.025 0.999 1
6.12 39 0.075 0.999 1
6.12 39 0.125 0.999 1
6.12 39 0.175 0.999 1
6.12 39 0.225 0.999 1
6.12 39 0.275 0.999 1
6.12 39 0.325 0.999 1

6.12 43.4 0.025 0.999 1
6.12 43.4 0.075 0.999 1
6.12 43.4 0.125 0.999 1
6.12 43.4 0.175 0.999 1
6.12 43.4 0.225 0.999 1
6.12 43.4 0.275 0.999 1
6.12 43.4 0.325 0.999 1

6.12 47.8 0.025 0.999 1
6.12 47.8 0.075 0.999 1
6.12 47.8 0.125 0.999 1
6.12 47.8 0.175 0.999 1
6.12 47.8 0.225 0.999 1
6.12 47.8 0.275 0.999 1
6.12 47.8 0.325 0.999 1

6.12 52.2 0.025 0.999 1
6.12 52.2 0.075 0.999 1
6.12 52.2 0.125 0.999 1
6.12 52.2 0.175 0.999 1
6.12 52.2 0.225 0.999 1
6.12 52.2 0.275 0.999 1
6.12 52.2 0.325 0.999 1

6.12 56.6 0.025 0.999 1
6.12 56.6 0.075 0.999 1
6.12 56.6 0.125 0.999 1
6.12 56.6 0.175 0.999 1
6.12 56.6 0.225 0.999 1
6.12 56.6 0.275 0.999 1
6.12 56.6 0.325 0.999 1

6.12 61 0.025 0.999 1
6.12 61 0.075 0.999 1
6.12 61 0.125 0.999 1
6.12 61 0.175 0.999 1
6.12 61 0.225 0.999 1
6.12 61 0.275 0.999 1
6.12 61 0.325 0.999 1

6.12 65.4 0.025 0.999 1
6.12 65.4 0.075 0.999 1
6.12 65.4 0.125 0.999 1
6.12 65.4 0.175 0.999 1
6.12 65.4 0.225 0.999 1
6.12 65.4 0.275 0.999 1
6.12 65.4 0.325 0.999 1

7.56 39 0.025 0.999 1
7.56 39 0.075 0.999 1
7.56 39 0.125 0.999 1
7.56 39 0.175 0.999 1
7.56 39 0.225 0.999 1
7.56 39 0.275 0.999 1
7.56 39 0.325 0.999 1

7.56 43.4 0.025 0.999 1
7.56 43.4 0.075 0.999 1
7.56 43.4 0.125 0.999 1
7.56 43.4 0.175 0.999 1
7.56 43.4 0.225 0.999 1
7.56 43.4 0.275 0.999 1
7.56 43.4 0.325 0.999 1

7.56 47.8 0.025 0.999 1
7.56 47.8 0.075 0.999 1
7.56 47.8 0.125 0.999 1
7.56 47.8 0.175 0.999 1
7.56 47.8 0.225 0.999 1
7.56 47.8 0.275 0.999 1
7.56 47.8 0.325 0.999 1

7.56 52.2 0.025 0.999 1
7.56 52.2 0.075 0.999 1
7.56 52.2 0.125 0.999 1
7.56 52.2 0.175 0.999 1
7.56 52.2 0.225 0.999 1
7.56 52.2 0.275 0.999 1
7.56 52.2 0.325 0.999 1

7.56 56.6 0.025 0.999 1
7.56 56.6 0.075 0.999 1
7.56 56.6 0.125 0.999 1
7.56 56.6 0.175 0.999 1
7.56 56.6 0.225 0.999 1
7.56 56.6 0.275 0.999 1
7.56 56.6 0.325 0.999 1

7.56 61 0.025 0.999 1
7.56 61 0.075 0.999 1
7.56 61 0.125 0.999 1
7.56 61 0.175 0.999 1
7.56 61 0.225 0.999 1
7.56 61 0.275 0.999 1
7.56 61 0.325 0.999 1

7.56 65.4 0.025 0.999 1
7.56 65.4 0.075 0.999 1
7.56 65.4 0.125 0.999 1
7.56 65.4 0.175 0.999 1
7.56 65.4 0.225 0.999 1
7.56 65.4 0.275 0.999 1
7.56 65.4 0.325 0.999 1

9 39 0.025 0.999 1
9 39 0.075 0.999 1
9 39 0.125 0.999 1
9 39 0.175 0.999 1
9 39 0.225 0.999 1
9 39 0.275 0.999 1
9 39 0.325 0.999 1

9 43.4 0.025 0.999 1
9 43.4 0.075 0.999 1
9 43.4 0.125 0.999 1
9 43.4 0.175 0.999 1
9 43.4 0.225 0.999 1
9 43.4 0.275 0.999 1
9 43.4 0.325 0.999 1

9 47.8 0.025 0.999 1
9 47.8 0.075 0.999 1
9 47.8 0.125 0.999 1
9 47.8 0.175 0.999 1
9 47.8 0.225 0.999 1
9 47.8 0.275 0.999 1
9 47.8 0.325 0.999 1

9 52.2 0.025 0.999 1
9 52.2 0.075 0.999 1
9 52.2 0.125 0.999 1
9 52.2 0.175 0.999 1
9 52.2 0.225 0.999 1
9 52.2 0.275 0.999 1
9 52.2 0.325 0.999 1

9 56.6 0.025 0.999 1
9 56.6 0.075 0.999 1
9 56.6 0.125 0.999 1
9 56.6 0.175 0.999 1
9 56.6 0.225 0.999 1
9 56.6 0.275 0.999 1
9 56.6 0.325 0.999 1
```

```
9 61 0.025 0.999 1
9 61 0.075 0.999 1
9 61 0.125 0.999 1
9 61 0.175 0.999 1
9 61 0.225 0.999 1
9 61 0.275 0.999 1
9 61 0.325 0.999 1

9 65.4 0.025 0.999 1
9 65.4 0.075 0.999 1
9 65.4 0.125 0.999 1
9 65.4 0.175 0.999 1
9 65.4 0.225 0.999 1
9 65.4 0.275 0.999 1
9 65.4 0.325 0.999 1
```

APPENDIX C. DIGITAL IMAGE OF THE EXPERIMENTAL SETUP



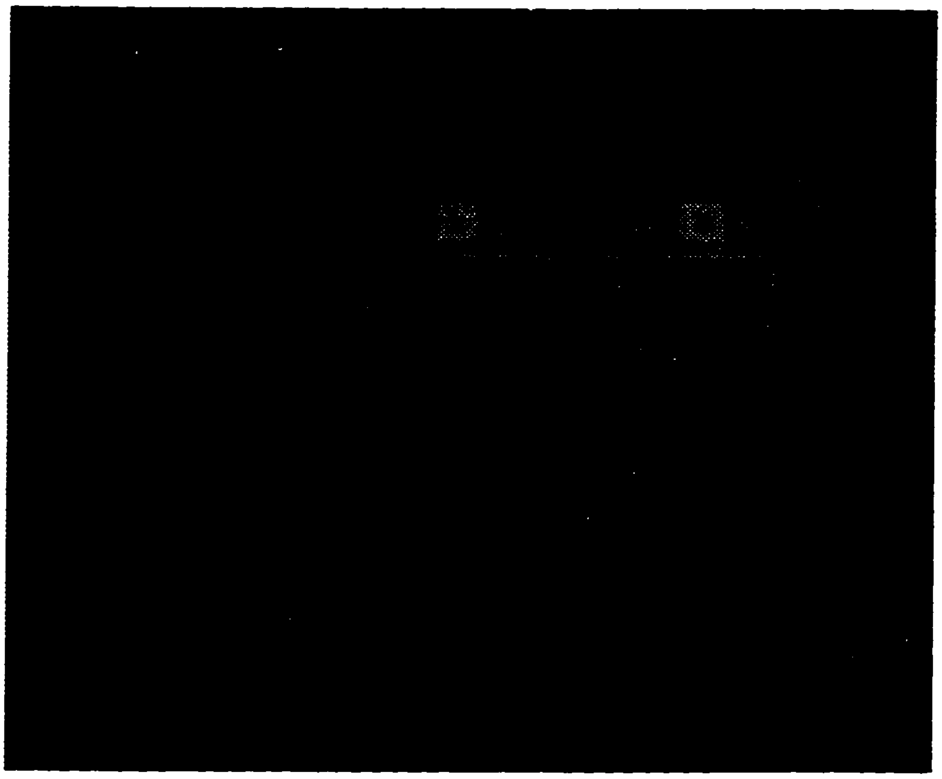
APPENDIX D. DIGITAL IMAGE OF THE WORKPIECE HOLDER AND THE DYNAMOMETER



APPENDIX E. SPECIFICATIONS OF THE DYNAMOMETER

Description	Items	Unit	Range
Force application with and max. 25mm above top surface	Fx, Fy, Fz	kN	-5 ... 5
Range when turning, force application at point A. Fx and Fy $\leq$ 0.5 Fz	Fz	kN	-5 ... 5
Calibrated partial range 1	Fx, Fy	N	0 ... 500
	Fz	N	0 ... 1000
Calibrated partial range 1	Fx, Fy	N	0 ... 50
	Fz	N	0 ... 100
Overload	Fx, Fy, Fz	kN	-7.5 / 7.5
with Fx and Fy $\leq$ 0.5 Fz	Fz	kN	-7.5 / 15
Response threshold		N	<0.01
Sensitivity	Fx, Fy	pC/N	≈-7.5
	Fz	pC/N	≈-3.5
Linearity (all ranges)		%FSO	≤±1
Hysteresis (all ranges)		%FSO	≤0.5
Crosstalk		%FSO	≤±2
Rigidity	cx, cy	kN/µm	>1
	cz	kN/µm	>2
Natural frequency	f_0(x,y,z)	kHz	≈3.5
Natural frequency (mounted on flanges)	f_0(x,y)	kHz	≈2.3
	f_0(z)	kHz	≈3.5
Operating temperature range		°C	0 ... 70
Temperature coefficient of sensitivity		%/°C	-0.02
Capacitance (of channel)		pF	≈220
Insulation resistance at 20 °C		Ω	$>10^{13}$
Ground insulation		Ω	$>10^{8}$
Protection class with cable Type 1687B5, 1689B5, 1677A5, 1679A5		-	IP 67
Weight		kg	7.3
Dimensions	170 x 140 x 60 (mm)		

APPENDIX F. DIGITAL IMAGE OF THE AMPLIFIER



APPENDIX G. CNC PART PROGRAM

```
N10 O1997 (* pow97.nc - metric
N20 G90 G80 G40 G17 M49 (* turn off manual feed/rpm control
N30 (* T5 M6 (* Dia 0.75 in = 19.05 mm
N40 G01 E28 X40.0 Y72.0 Z36.0 F1600.0
N50 Z13.08 F1200.0 (*** Z APPR = Z Cur + 20
N60 X57.0 (*** start pos 52 - 12 = 40
N70 M0 (* suspend to reset amplifier
N80 S600 M3
N90 (* G42 D10
N100 Z-6.72 F400.0 (*** Z APPR = Z Cur + 0.2; 20 / 400 * 60 = 3s
N110 Y30.0 F1600.0
N120 Y9.53 F800.0
N130 G4 P500
N140 Z-6.92 F30
N150 S652 F65.0
N160 X81.0 (* X63.0
N170 S600 F60.0
N180 (* G4 P500
N190 (* X81.0 (*** X135.0 = 120 + 9.53 + 5 safety
N200 Z20.0 F1600.0
N210 G40 G00 (* X-36.0 Z20.0
N220 M5
N230 G91 G28 X0.0 Y0.0 Z0.0
N240 M30
```

APPENDIX H. TRAINING DATA FROM THE EXPERIMENT

```
//*** input.dat -- for the fsc system ***

//delta
0.09

//number of (input variables + y + mu)
5

//initial number of regions for each variable
3

//input variables + y + mu
Depth Dc
Speed Ns
Feed Fd
//Cut_Depth Dc
//Workpiece_strength Ws
Out_Value Ov
Ds_Degree Dd

//number of data sets
343

//data sets

//Depth Speed Feed Output DataSetDegree

0.36 39 0.025 0.016194 1
0.36 39 0.075 0.032568 1
0.36 39 0.125 0.044359 1
0.36 39 0.175 0.055766 1
0.36 39 0.225 0.064243 1
0.36 39 0.275 0.069261 1
0.36 39 0.325 0.076256 1

0.36 43.4 0.025 0.017874 1
0.36 43.4 0.075 0.036861 1
0.36 43.4 0.125 0.048635 1
0.36 43.4 0.175 0.061836 1
0.36 43.4 0.225 0.070883 1
0.36 43.4 0.275 0.075986 1
0.36 43.4 0.325 0.083616 1

0.36 47.8 0.025 0.019524 1
0.36 47.8 0.075 0.041279 1
0.36 47.8 0.125 0.052763 1
0.36 47.8 0.175 0.06786 1
0.36 47.8 0.225 0.077401 1
0.36 47.8 0.275 0.082491 1
0.36 47.8 0.325 0.090725 1

0.36 52.2 0.025 0.022641 1
0.36 52.2 0.075 0.043783 1
0.36 52.2 0.125 0.060556 1
0.36 52.2 0.175 0.069709 1
0.36 52.2 0.225 0.084281 1
0.36 52.2 0.275 0.089176 1
0.36 52.2 0.325 0.098812 1

0.36 56.6 0.025 0.023703 1
0.36 56.6 0.075 0.045554 1
0.36 56.6 0.125 0.063596 1
0.36 56.6 0.175 0.079442 1
0.36 56.6 0.225 0.090319 1
0.36 56.6 0.275 0.097303 1
0.36 56.6 0.325 0.108508 1

0.36 61 0.025 0.024902 1
0.36 61 0.075 0.049827 1
0.36 61 0.125 0.068335 1
0.36 61 0.175 0.085234 1
0.36 61 0.225 0.096741 1
0.36 61 0.275 0.104462 1
0.36 61 0.325 0.115986 1

0.36 65.4 0.025 0.026008 1
0.36 65.4 0.075 0.054204 1
0.36 65.4 0.125 0.073046 1
0.36 65.4 0.175 0.09097 1
0.36 65.4 0.225 0.103076 1
0.36 65.4 0.275 0.111562 1
0.36 65.4 0.325 0.123325 1

1.8 39 0.025 0.065426 1
1.8 39 0.075 0.14278 1
1.8 39 0.125 0.212323 1
1.8 39 0.175 0.269653 1
1.8 39 0.225 0.335132 1
1.8 39 0.275 0.389474 1
1.8 39 0.325 0.434224 1

1.8 43.4 0.025 0.074409 1
1.8 43.4 0.075 0.159276 1
```

```
1.8 43.4 0.125 0.237231 1
1.8 43.4 0.175 0.301308 1
1.8 43.4 0.225 0.374856 1
1.8 43.4 0.275 0.435328 1
1.8 43.4 0.325 0.485128 1

1.8 47.8 0.025 0.083717 1
1.8 47.8 0.075 0.175851 1
1.8 47.8 0.125 0.262332 1
1.8 47.8 0.175 0.333213 1
1.8 47.8 0.225 0.414968 1
1.8 47.8 0.275 0.481571 1
1.8 47.8 0.325 0.536419 1

1.8 52.2 0.025 0.107422 1
1.8 52.2 0.075 0.208126 1
1.8 52.2 0.125 0.293818 1
1.8 52.2 0.175 0.381529 1
1.8 52.2 0.225 0.453323 1
1.8 52.2 0.275 0.526056 1
1.8 52.2 0.325 0.585953 1

1.8 56.6 0.025 0.105144 1
1.8 56.6 0.075 0.216814 1
1.8 56.6 0.125 0.310722 1
1.8 56.6 0.175 0.392983 1
1.8 56.6 0.225 0.49518 1
1.8 56.6 0.275 0.574044 1
1.8 56.6 0.325 0.63899 1

1.8 61 0.025 0.113787 1
1.8 61 0.075 0.235187 1
1.8 61 0.125 0.333994 1
1.8 61 0.175 0.42528 1
1.8 61 0.225 0.532008 1
1.8 61 0.275 0.617004 1
1.8 61 0.325 0.686998 1

1.8 65.4 0.025 0.122498 1
1.8 65.4 0.075 0.253778 1
1.8 65.4 0.125 0.357138 1
1.8 65.4 0.175 0.457829 1
1.8 65.4 0.225 0.568597 1
1.8 65.4 0.275 0.659723 1
1.8 65.4 0.325 0.734766 1

3.24 39 0.025 0.13382 1
3.24 39 0.075 0.257115 1
3.24 39 0.125 0.38281 1
3.24 39 0.175 0.482187 1
3.24 39 0.225 0.597481 1

3.24 43.4 0.025 0.147122 1
3.24 43.4 0.075 0.29907 1
3.24 43.4 0.125 0.427696 1
3.24 43.4 0.175 0.538285 1
3.24 43.4 0.225 0.666586 1

3.24 47.8 0.025 0.160058 1
3.24 47.8 0.075 0.34365 1
3.24 47.8 0.125 0.472927 1
3.24 47.8 0.175 0.594728 1
3.24 47.8 0.225 0.736037 1

3.24 52.2 0.025 0.173187 1
3.24 52.2 0.075 0.376512 1
3.24 52.2 0.125 0.509441 1
3.24 52.2 0.175 0.642453 1
3.24 52.2 0.225 0.796769 1

3.24 56.6 0.025 0.18082 1
3.24 56.6 0.075 0.380837 1
3.24 56.6 0.125 0.554319 1
3.24 56.6 0.175 0.698543 1
3.24 56.6 0.225 0.865867 1

3.24 61 0.025 0.204893 1
3.24 61 0.075 0.421347 1
3.24 61 0.125 0.594699 1
3.24 61 0.175 0.750134 1
3.24 61 0.225 0.930466 1

3.24 65.4 0.025 0.230412 1
3.24 65.4 0.075 0.46343 1
3.24 65.4 0.125 0.634686 1
3.24 65.4 0.175 0.801334 1
3.24 65.4 0.225 0.994673 1

4.68 39 0.025 0.185449 1
4.68 39 0.075 0.386639 1
4.68 39 0.125 0.544515 1

4.68 43.4 0.025 0.209731 1
4.68 43.4 0.075 0.42892 1
4.68 43.4 0.125 0.604607 1

4.68 47.8 0.025 0.234694 1
4.68 47.8 0.075 0.470929 1
4.68 47.8 0.125 0.664428 1

4.68 52.2 0.025 0.253856 1
4.68 52.2 0.075 0.531299 1
4.68 52.2 0.125 0.74261 1

4.68 56.6 0.025 0.281987 1
4.68 56.6 0.075 0.548978 1
4.68 56.6 0.125 0.778101 1

4.68 61 0.025 0.302761 1
4.68 61 0.075 0.593715 1
4.68 61 0.125 0.840649 1

4.68 65.4 0.025 0.32337 1
4.68 65.4 0.075 0.638749 1
4.68 65.4 0.125 0.903494 1

6.12 39 0.025 0.230924 1
6.12 39 0.075 0.467137 1
6.12 39 0.125 0.663248 1

6.12 43.4 0.025 0.260384 1
6.12 43.4 0.075 0.528364 1
6.12 43.4 0.125 0.746601 1

6.12 47.8 0.025 0.290536 1
6.12 47.8 0.075 0.59132 1
6.12 47.8 0.125 0.831682 1

6.12 52.2 0.025 0.310386 1
6.12 52.2 0.075 0.64443 1
6.12 52.2 0.125 0.906917 1

6.12 56.6 0.025 0.338463 1
6.12 56.6 0.075 0.691261 1
6.12 56.6 0.125 0.975873 1

6.12 61 0.025 0.375192 1
6.12 61 0.075 0.748428 1
6.12 61 0.125 1.055166 1

6.12 65.4 0.025 0.413425 1
6.12 65.4 0.075 0.80609 1
6.12 65.4 0.125 1.134953 1

7.56 39 0.025 0.271215 1
7.56 39 0.075 0.641747 1

7.56 43.4 0.025 0.312919 1
7.56 43.4 0.075 0.725255 1

7.56 47.8 0.025 0.356875 1
7.56 47.8 0.075 0.811014 1

7.56 52.2 0.025 0.375421 1
7.56 52.2 0.075 0.871364 1
```

```
7.56 56.6 0.025 0.431257 1
7.56 56.6 0.075 0.969003 1

7.56 61 0.025 0.448645 1
7.56 61 0.075 1.028195 1

7.56 65.4 0.025 0.463705 1
7.56 65.4 0.075 1.085059 1

9 39 0.025 0.31236 1
9 39 0.075 0.758317 1

9 43.4 0.025 0.354596 1
9 43.4 0.075 0.850867 1

9 47.8 0.025 0.398251 1
9 47.8 0.075 0.944834 1

9 52.2 0.025 0.455717 1
9 52.2 0.075 1.052613 1

9 56.6 0.025 0.496138 1
9 56.6 0.075 1.143347 1

9 61 0.025 0.56443 1
9 61 0.075 1.261953 1

9 65.4 0.025 0.637011 1
9 65.4 0.075 1.384847 1
```

REFERENCES

Abdou, G. and Yien, J. (1995). Analysis of Force Patterns and Tool Life in Milling Operations. *International Journal of Advanced Manufacturing Technology, 10*(1), 11-18.

Allocca, J. A. & Stuart, A. (1984). *Transducers: theory and applications.* Reston, VA: Reston Publishing Company.

Amstead, B. H., Ostwald, P. F., & Begeman, M. L. (1987). Manufacturing Processes. New York: John Wiley & Sons.

Armarego, E. J. A. and Deshpande, N. P. (1993). Force prediction models and CAD/CAM software for helical tooth milling processes. I. Basic approach and cutting analyses. *International Journal of Production Research, 31*(8), 1991-2009.

Badiru, A. B. (1992). Expert systems applications in engineering and manufacturing. Englewood Cliffs, NJ: Prentice-Hall.

Berenji & Khedkar, (1992). Learning and tuning fuzzy logic controllers through reinforcement. *IEEE Transactions on Neural Networks, 3*(5), 724-740.

Black, J T. (1991). *The design of the factory with a future*. New York: McGraw-Hill, Inc.

Bouzakis, K. -D., Efstathiou, K., & Paraskevopoulou, R. (1992). NC-Code Preparation with Optimum Cutting Conditions in 3-Axis Milling. *Annals of the CIRP, 41*(1), 513-516.

Brown, M. and Harris, C. (1994). *Neurofuzzy adaptive modelling and control*. New York: Prentice Hall.

Chen, J. C. (1996). A fuzzy-nets tool-breakage detection for end-milling operations. *International Journal of Advanced Manufacturing Technology, 12*(1), 153-164.

Chu, C. H. (1993). Manufacturing cell formation be competitive learning. *International Journal of Production Research, 31*(4), 829-843.

Cichocki, A. & Unbehauen, R. (1993). *Neural networks for optimization and signal processing*. New York: John Wiley & Sons.

DeGarmo, E. P., Black, J T. & Kohser, R. A. (1988). Materials and processes in manufacturing. New York: Macmillian Publishing Company.

Dorf, R. C. and Kusiak, A. (1994). *Handbook of design, manufacturing and automation.* New York: John Wiley & Sons, Inc.

Fadal Engineering Co., Inc. (1992). *VMC maintenance manual.* Chatsworth, CA: Author.

Fadal Engineering Co., Inc. (1994). *VMC users manual.* Chatsworth, CA: Author.

Fang, X. D. and Jawahir, I. S. (1994). Predicting total machining performance in finish turning using integrated fuzzy-set models of the machinability parameters. *International Journal of Production Research, Vol. 32*(4), 833-849.

Groover, M. P. (1996). Fundamentals of modern manufacturing: materials, processes, and systems. Upper Saddle River, NJ: Prentice Hall.

Jain, A. K., Mao, J. & Mohiuddin, K. M. (1996). Artificial neural networks: a tutorial. *Computer, 29*(3), 31-44.

Kheir, N. A. (1988). System modeling and computer simulation. New York: Marcel Dekker, Inc.

Kistler Instrument Corp. (1994). *Operating instructions: quartz 3-component dynamometer type 9257B.* Amherst, NY: Author.

Kistler Instrument Corp. (1995). *Operating instructions: dual mode amplifier type 5010B.* Amherst, NY: Author.

Knepell, P. L. & Arangno, D. C. (1993). Simulation validation: a confidence assessment methodology. Los Alamitos, CA: IEEE Computer Society Press.

Ko, T. J., Cho, D. W., & Jung, M. Y. (1995). On-line monitoring of tool breakage in face milling using a self-organized neural network. *Journal of Manufacturing System, 14*(2), 80-90.

Kruse, R., Gebhardt, J., & Klawonn, F. (1994). *Foundations of fuzzy systems.* New York: John Wiley & Sons.

Lascoe, O. D., Nelson, C. A., & Porter, H. W. (1973). *Machine shop operations and setups.* Chicago: American Technical Society.

Law, A. M. (1986). Introduction to simulation: a powerful tool for analyzing complex manufacturing systems. *Industrial Engineering, 18*(5), 46-63.

Law, A. M. & Kelton, W. D. (1991). *Simulation modeling and analysis.* New York: McGraw-Hill, Inc.

Leem, C. S., Dornfeld, D. A. & Dreyfus, S. E. (1995). A customized neural network for sensor fusion in on-line monitoring of cutting tool wear. *Journal of Engineering for Industry, 117*(2), 152-159.

Liao, Y. S. & Young, Y. C. (1996). A new on-line spindle speed regulation strategy for chatter control. *International Journal of Machine Tools and Manufacture, 36*(5), 651-660.

Lindberg, Roy A. (1990). *Processes and materials of manufacture*. Boston: Allyn and Bacon.

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine, 4*(2), 4-22.

Markus, J. & Sclater, N. (1994). *McGraw-Hill electronics dictionary*. New York: McGraw-Hill, Inc.

Martini, K. H. (1983). Multicomponent dynamometers using quartz crystals as sensing elements. *ISA Transactions, 22*(1), 35-46.

Merchant, M. E. (1945). Mechanics of metal cutting process—II. plasticity conditions in orthogonal cutting. *Journal of Applied Physics, 16*(6), 318-324.

Mesquita, R., Krasteva, E., & Doytchinov, S. (1995). Computer-Aided Selection of Optimum Machining Parameters in Multipass turning. *International Journal of Advanced Manufacturing Technology, 10*(1), 19-26.

Niebel, Benjamin W., Draper, Alan B., & Wysk, Richard A. (1989). *Modern manufacturing process engineering*. New York: McGraw-Hill Publishing Company.

Novak, Vilem (1989). *Fuzzy sets and their applications*. Bristol, England: Adam Hilger.

Nguyen, Hung T., Sugeno, Michio, Tong, Richard, & Yager, Ronald R. (1995). *Theoretical aspects of fuzzy control*. New York: John Wiley & Sons, Inc.

Pal, S. K. & Srimani, P. K. (1996). Neurocomputing: motivation, models, and hybridization. *Computer, 29*(3), 24-30.

Parsons, N. R. (1971). N/C machinability data systems. Dearborn, MI: the Society of Manufacturing Engineers.

Prasad, B. S. V. (1992). Designing programming station software for CNC profile cutting. *Computers in Industry, 18(1)*, 67-76.

Ralescu, A. L. (1994). *Applied Research in fuzzy technology*. Boston: Kluwer Academic Publishers.

Rangwala, S. & Dornfeld, D. A. (1990). Sensor integration using neural networks for intelligent tool condition monitoring. *Journal of Engineering for Industry, 112*(3), 219-228.

Sargent, R. G. (1991). Simulation model verification and validation. *Winter Simulation Conference Proceedings*, 37-47.

Schriber, T. J. (1987). The nature and role of simulation in the design of manufacturing systems. *Proceedings of the European Simulation Multiconference, Vienna, Austria*, 5-18.

Singh, N. (1996). *Systems approach to computer-integrated design and manufacturing*. New York: John Wiley & Sons, Inc.

Sugeno, M. (1985). *Industrial applications of fuzzy control*. New York: Elsevier Science Publishers B.V.

Tan, C. L., Quah, T. S., & Teh, H. H. (1996). An artificial neural network that models human decision making. *Computer, 29(3), 64-70.*

Tansel, I. N. & McLaughlin, C. (1993). Detection of tool breakage in milling operation - part II: the neural network approach. *International Journal of Machine Tools and Manufacture, 33*(4), 545-558.

Tarng, Y. S., Cheng, C. I., & Kao, Y. K. (1995). Modeling of three-dimensional numerically controlled end milling operations. *International Journal of Machine Tools and Manufacture, 35*(7), 939-950.

Tarng, Y. S., Hseih, Y. W., & Hwang, S. T. (1994). Sensing tool breakage in face milling with a neural network. *International Journal of Machine Tools and Manufacture, 34*(3), 341-350.

Terano, T., Asai, K., & Sugeno, M. (1992). *Fuzzy systems theory and its applications*. Boston: Academic Press, Inc.

Uhrig, R. E. (1995). Introduction to artificial neural networks. Proceedings of International Conference on Industrial Electronics, Control and Instrumentation, 33-37.

Widrow, B., Rumelhart, D. E., & Lehr, M. A. (1994). Neural networks: applications in industry, business and science. *Communications of the ACM, 37*(3), 93-105.

Wild, P. (1994). *Industrial sensors and applications for condition monitoring*. London: Mechanical Engineering Publications Limited.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control, 8*(3), 338-353.

Zadeh, L. A. (1973). Outline of a new approach to the analysis complex systems and decision processes. *IEEE Transactions on Systems, Man, Cybernetics, 3*(1), 28-34.

Zimmermann, H. J. (1991). *Fuzzy set theory and its applications*. Boston: Kluwer Academic publishers.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Joseph C. Chen, for his time, encouragement, and support. He is one of the most energetic and knowledgeable professors I know.

I would also like to thank my committee members, Dr. Larry L. Bradshaw, Dr. Tao Chang, Dr. Dale Grosvenor, and Dr. Mack C. Shelley for their time and support.

Thanks are extended to Dr. John N. Riley and Dr. William G. Miller for their time and continuous support. Dr. Riley had been a committee member until his full retirement. Dr. Miller was my M.S. major professor.

I am very grateful to my parents, Mao-Kun and Chin, for instilling me the value of education. I would like to thank my wife, Miao-Ching, and children, Jane and Justin, for their patience, understanding, and independence. I would also like to thank my siblings and in-laws for their support and encouragement.

I would also like to acknowledge my friends, other members of the IED&T faulty and staff, and the many other people who have contributed to my study and dissertation.