MONITORING AND DIAGNOSIS FOR CONTROL

OF AN

INTELLIGENT MACHINING PROCESS

by

Theo Ian van Niekerk

A thesis submitted in compliance with the full requirements for the

Doctor Technologiae: Engineering

in the

Faculty of Electrical and Mechanical Engineering

Port Elizabeth Technikon

July 2001

Promoter

Prof Zvi Katz

I wish to acknowledge the contributions by the following:

My promoter, Prof Zvi Katz, for his guidance and encouragement.

All the team members in the Manufacturing Technology Research Centre; Prof Eugene du Preez for his encouragement and support; Dr Pat McGrath and William Rall for the application of strain gauges; Prof Hugh Jeffery for the use of the lathe and for his encouragement and support; Dr Hattingh for the use of the strain gauge amplifier and guidance; Frank Adlam and Hannalie van Niekerk for assistance with regard to lecturing duties. Research assistants: Grant Kruger for his assistance with the graphical interface of the Java Applet; Howard Loftus for assembly and pcb manufacture of microphone amplifier; Mr Brahm de Villiers for assisting in selecting and setting up the DSP equipment.

The National Research Foundation and research committee of the Port Elizabeth Technikon for financial assistance. This enabled the purchasing of hardware of software required for the experimental set-up and to present various aspects of the research at two international conferences.

The library staff of the Port Elizabeth Technikon for their friendly assistance. Mr Coos Bosma for providing assistance with statistical analysis (Statistica V6.0). Mr Piet Boonzaier for language editing.

All members of rectorate of the Port Elizabeth Technikon for providing climate and means for performing my research.

My family Eleanor, Ian and Emile for their love and support throughout the duration of my research.

The Lord Jesus Christ for the persons and systems he timeously put in place, and for giving me the strength and dedication to complete the project to the best of my ability.

I Theo van Niekerk hereby declare that:

- the work done in this thesis is my own;
- all sources used or referred to have been documented and recognized; and
- this thesis has not been previously submitted in full or partial fulfillment of the requirements for an equivalent or higher qualification at any other educational institution.

Signature **03/07/2001**

1 **Date**

Abstract

A multi-level modular control scheme to realize integrated process monitoring, diagnosis and control for intelligent machining is proposed and implemented. PC-based hardware architecture to manipulate machining process cutting parameters, using a PMAC interface card as well as sensing processes performance parameters through sampling, and processing by means of DSP interface cards is presented. Controller hardware, to interface the PC-based PMAC interface card to a machining process for the direct control of speed, feed and depth of cut, is described. Sensors to directly measure on-line process performance parameters, including cutting forces, cutting sound, tool-workpiece vibration, cutting temperature and spindle current are described. The indirect measurement of performance parameter surface roughness and tool wear monitoring, through the use of NF sensor fusion modeling, is described and verified. An object based software architecture,

with corresponding user interfaces (using Microsoft Visual C++ Foundation Classes and implemented C++ classes for sending motion control commands to the PMAC and receiving processed on-line sensor data from the DSP) is explained. The software structure indicates all the components necessary for integrating the monitoring, diagnosis and control scheme. C-based software code executed on the DSP for real-time sampling, filtering and FFT processing of sensor signals, is explained.

Making use of experimental data and regression analysis, analytical relationships between cutting parameters (independent) and each of the performance parameters (dependent) are obtained and used to simulate the machining process. A fuzzy relation that contains values determined from statistical data (indicating the strength of connection between the independent and dependent variables) is proposed. The fuzzy relation forms the basis of a diagnostic scheme that is able to intelligently determine which independent variable to change when a machining performance parameter exceeds control limits. The intelligent diagnosis scheme is extensively tested using the machining process simulation.

Table of Contents	**Page**

3

4 Chapter 3 - Experimental setup: Machine Controls, Sensors and Software Components

5 Chapter 4 – Multi-Sensor Fusion Models for Tool Wear Classification and Surface Roughness Measurement

List of Symbols

9

Machining Process

d_y	Depth of cut, [mm]
D_{AVE}	Average diameter of the workpiece, (DO-DI)/2, [mm]
D_O	Outer diameter of workpiece, before machined, [mm]
D_I	Inner diameter of workpiece, after machined, [mm]
$f1_X$	Feed/revolution of the tool, how far the tool travels per revolution of the workpiece, [mm/rev]
$f2_X$	Feedrate of the tool, linear speed of the tool along the workpiece length, [mm/min]
F_C, F_Z	Cutting force in the z-direction, [Newton]
F_F, F_X	Feed force in the x-direction, [Newton]
I_S	Spindle current, [mA]
L	Length of workpiece, [mm]
N	Rotational speed of the workpiece, $V_C/(\pi\ D_{AVE})$, [RPM]
P_C	Power in cut, [Watts]
R_a	Surface roughness, [μm]
r_e	Cutting tool nose radius

T_C Cutting torque, [Nm]

T^n Tool life, [minutes]

T_t Mean temperature in cutting tool tip, [^{0}C]

S_C Cutting sound, [mV]

V_C Cutting speed, surface speed of the workpiece, [m/min]

V_y Cutting tool-workpiece vibration, [mV]

VB,V_b Tool wear land, [mm]

Statistical Analysis

r^2 Coefficient of determination, the degree of relationship that exist between variables

ß$_i$ Transformed regression coefficient, in order to compare contribution of regression coefficients on a standardized base

Digital Signal Processing

b_k FIR filter coefficients

δ_p Peak passband deviation

δ_s Stopband deviation

δ_f Transition width, stopband edge frequency – passband edge frequency, [Hz]

f_c Cut-off frequency of an ideal low pass filter, [Hz]

F_s Sampling frequency, [Hz]

L Range of discrete buffer, buffer size

N Filter length, number of filter coefficients, N = M+ 1

M Order of FIR filter, = N-1

μ Mean of a discrete set of value

T_s Sampling period, [seconds]

ω_p Passband edge angular frequency, [radians/sec]

ω_s Stopband edge angular frequency, [radians/sec]

σ Standard deviation, square root of average squared deviation from the mean

Abbreviations

$\mu_A(x)$ Degree of membership of variable (x) in fuzzy set (A)

$\overline{R}_{\bar{A}X\bar{B}}$ Fuzzy relation used to map fuzzy sets between different universes of discourse

$\mu_R(x,y)$Elements of fuzzy relation, where each element correspond to the strength of connection

AC	Adaptive Control
ACC	Adaptive Control Constraints
ACO	Adaptive Control Optimization
ADC	Analog-to-Digital Converter

AI	Artificial Intelligence
ANOVA	Analysis of Variance
API	Application Program Interface
CNC	Computer Numerical Control
CNF	Certainty factors associated with facts and with rules
CoM	Centre-of-Maximum
DFT	Discrete Fourier Transform
DLL	Dynamic Link Library
DoS	Degree of Support
DSP	Digital Signal Processing
FAM	Fuzzy Associative Map
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FL	Fuzzy Logic
FR	Fuzzy Relation
GUI	Graphical User Interface
LPF	Low-Pass Filter
MBF	Membership Function
MFC	Microsoft Foundation Classes
MoM	Mean-of-Maximum
NC	Numerical Control
NF	Neuro-Fuzzy
NN	Neural Network
OOP	Object Orientated Programming
PC	Personal Computer
PMAC	Programmable Multi-Axis Controller
PSW	Personal Web Server

RMS	Root Mean Square
TCP	Transport Control Protocol
WWW	World Wide Web

List of Figures

List of Tables

Chapter 1

Introduction

There is a myriad of machines which cut, sand, drill, face, turn, bend, grind, and much more. The underlying concept is the same: The criteria for optimal performance is the rate of metal removal of material from the workpiece. In addition, international competitiveness requires high product quality in combination with reduced throughput time at minimal cost. By making use of basic monitoring and diagnostic systems, effective machining time has been increased from 10% to 65% [1]. A major obstacle hampering the progress towards the development of unmanned machining centers is the limited use of on-line monitoring and diagnostic systems in practice. Monitoring and diagnostic systems that rely on the on-line acquisition of machining process sensor data will enhance the implementation of intelligent machining [2]. Harber et al [3] define intelligent machining as:

“A computationally efficient procedure developed combining one or more intelligent techniques (fuzzy logic, neural networks e.g.) and expert criteria (operator knowledge), with one or more higher resolution levels (hierarchical levels), which basically manipulate cutting conditions (spindle speed, feed) and should be monitoring tool status and finished surface quality, as well as increasing productivity through higher metal removal rate“

Signals from the machining process may be used in multi-sensor monitoring systems to measure surface roughness and tool wear indirectly, using intelligent systems. Intelligent systems consist of algorithms developed to emulate certain characteristics of the human being’s intelligent biological systems [4]. It is considered to be a powerful way to achieve superior performance by putting engineering expertise into products with the added advantage of making the design process faster, easier and more transparent [5].

The successful and reliable monitoring of surface roughness and tool wear will not only play a crucial role in achieving advanced automation, but its values may be used in controlling the quality of the manufactured part. Machining is a complex process and cutting parameters feed, speed and depth of cut modulate several process parameters, that in turn influence the quality of the manufactured product. For example, excessive cutting tool-workpiece vibration levels may negatively influence part tolerances. Similarly, excessive cutting temperature may cause damage to a machined surface. Currently post processing quality control procedures identify product deficiencies and after evaluating all the process parameters the process engineer may then decide what cutting parameter to change. A high productivity at the demanded process quality requires a process integrated quality assurance [1]. Allowable process parameter limits for tool temperature, cutting forces, spindle current, cutting sound, tool-workpiece vibration, surface roughness and cutting power may be set. When exceeded, an intelligent diagnostic system using on-line sensor data may reason and decide which cutting parameter, feed, speed, depth of cut to change in order to ensure that product quality is maintained.

To react and implement the control action, the machine control system needs to respond within a relatively short period of time. Therefore, advanced monitoring, intelligent diagnosis and parameter (machine) control systems need to be integrated. Due to its traditional closed architecture, conventional computer numerical control (CNC) machines cannot efficiently respond to sensor data provided by sensor based monitoring and diagnosis systems. CNC systems are machine control oriented whilst the success of intelligent machining greatly depends on how effectively performance parameters may be changed to ultimately produce a quality product more efficiently.

1.1 Aim

To implement intelligent machining by integrating sensor-based monitoring, intelligent diagnostic and machine control systems that are able to flexibly maintain machining process performance parameters within acceptable limits.

1.2 Objectives

The following objectives were accordingly specified for this project:

- To perform literature research into theoretical concepts and physical components for the implementation of intelligent machining.
- To identify signals that characterize the machining process performance and hence develop appropriate sensory systems to interface to computer based analog-to-digital converter (ADC) system.
- To identify motor and control equipment to drive a spindle and x-y coordinate system from a personal computer (PC)-based multi-axis control interface card.

- To select PC-based signal processing hardware and development software, and hence develop code that is able to sample, filter and process the identified analog signals.
- To select PC-based multi-axis machine control hardware and development software, and hence develop code that is able to execute motion control operations.
- To propose a logical framework that shows and connects all system components for intelligent machining.
- To propose and implement hardware architecture and software components to perform multi-axis control and signal processing.
- To develop a windows based object oriented software application framework with appropriate user and communication interfaces that integrate signal processing, monitoring, intelligent diagnosis and machine control components.
- To obtain experimental data by varying cutting parameters (input) and measure process performance parameters (output) for different tool wear.
- Make use of Statistica to analyze the experimental data to determine which sensor data is sensitive to tool wear and surface roughness. Create multi-sensor fusion models for tool wear and surface finish measurement, using neuro-fuzzy (NF) technology.
- Make use of the experimental data and Statistica's regression analysis module to determine empirical relationships between the dependent and independent variables. Use these relationships to model the machining process for simulation purposes.
- To develop an intelligent diagnostic system that is able to maintain machining process parameters within acceptable limits. To test the performance of the intelligent diagnostic system using the machining process simulation.

1.3 Hypothesis

PC-based digital signal processing, multi-axis machine control and Internet system hardware may be integrated with available motion control and sensor technology, to realize machine level open architecture with enhanced flexibility and modularity, which will realize an intelligent machine controller.

Visual C++ with its rich set of Microsoft foundation classes (MFC) and object oriented language features may be utilized to develop an object-oriented software framework to integrate and coordinate sensor data sampling and processing, monitoring and intelligent diagnosis and machine control functions. The application framework will further enhance flexibility, modularity, promote user-process interaction and software re-configurability.

Multiple sensors make it possible to reflect the complex machining process. Multi-sensor fusion systems by means of NF (intelligent systems) technology may be used in monitoring systems to measure tool wear and surface roughness indirectly which will enhance in-process quality control of the machined product.

Machining process knowledge may be represented using fuzzy relation, and fuzzy inferencing may be used to decide which cutting parameter to change in order to maintain machining performance parameters within acceptable limits. This will realize intelligent machining, enable quality control based on product properties, which will ultimately lead to higher throughput and less wastage of raw materials.

1.4 Methodological Justification

In order to accomplish the objectives, the fundamental research issues covered in this project include:

- **Statistical Analysis**

Data analysis requires the use of statistical methods, which includes curve fitting, and uses the method of least squares for producing multiple linear/non-linear regression equations to simulate machining process and multiple correlation coefficient, which is used to determine the degree of relationship for constructing knowledge based systems.

- **Digital Signal Processing**

Signals carry information and need to be processes to extract (completely or partially) the information contained in them, depending on the application of interest. Signal processing is concerned with the mathematical representation of the signal in the domain of the original dependent variable i.e. time domain, or in a transformed domain i.e. frequency domain, and with the algorithmic manipulation of the signal to extract the information being carried. To implement signal-processing techniques fundamental mathematical research into discrete domain systems includes Finite Impulse Response Filtering (FIR) and Discrete Fourier Transform (DFT).

- **Neural Network and Fuzzy Logic**

Machining processes are highly complex, and precise mathematical models may not always be the most effective method used in monitoring systems. Neural networks (NN) provide a strong tool for learning and, combined with multiple sensors data, result in advanced monitoring systems. Fuzzy logic (FL) allows the representation of decision and evaluation processes in an algorithmic (rule-based) form. NF technology has the learning capability of NN and a FL based rule structure. This increases the understanding into the working of knowledge-based systems as well as making the modification for enhancement possible. FL is considered to be a powerful way to achieve superior performance by putting engineering expertise into products, which may include many control parameters, with the added advantage of making the design process faster, easier, and more transparent.

- **Open Architecture**

Open architecture is a philosophy in the design and implementation of machine tools, production processes and control. Open architecture is a competitive area of manufacturing and it will meet manufacturing requirements in supplying more competitive products for the global market. Open architecture controllers must use standard computing architectures, standard operating systems, must be programmable in standard languages, and its application software must be open and extendable to allow users to integrate custom control algorithms.

1.5 Delimitations

The research will establish a sound experimental basis to serve future sensor based research projects for industrial machining centers. The machining process to be used in this research project, however, will be limited to an EMCO Compact 5 CNC training lathe. The open system architecture controller will focus on implementation aspects of intelligent machining and will not include a completed and operational system.

1.6 Significance of Research

Production quality and performance are concerns in machining processes. Poor production performances in machining are often caused by product wastage incurred by the application of excessive cutting power, torque, cutting forces, tool-workpiece vibration and high temperature. These process parameters may also contribute to excessive tool wear and breakage. Intelligent machining will improve production quality and performance of machining as it is able to detect and react to process parameters that exceed defined limits, thereby ensuring that product quality is not compromised which will ultimately lead to less wastage.

Intelligent machining is an advanced approach in manufacturing, strongly related to the efforts in developing re-configurable manufacturing equipment. Advances in PC-based hardware which include digital signal processing and programmable multi axis machine-control interface cards, and development software such as Visual C++ with MFC may be utilized to realize re-configurable manufacturing systems. Windows based graphical software interfaces will enable advanced machining process-human interaction.

1.7 Organization of Thesis

Chapter 2 describes the relevant theoretical concepts, corresponding components and technology relating to intelligent machining. It includes a logical framework that shows and connects all system components for intelligent machining. Chapter 3 provides a detailed description of the experimental setup, including implementation aspects of the sensor and motor control equipment, hardware architecture and software components to perform multi-axis control and signal processing, and an object oriented software framework to integrate all the system components. Chapter 4 presents and analyses NF-based multi-sensor fusion models for on-line tool wear and surface roughness monitoring. Chapter 5 describes an intelligent diagnostic scheme to realize intelligent machining and includes simulation and testing. Chapter 6 is the conclusion, which includes a discussion on future development.

Appendix A contains an example indicating the method for finding filter coefficients of a low-pass filter (LPF) to meet the specifications as used in this project. Appendix B contains experimental of machining process data for tool wear 0 mm and 0.2 mm. Appendix C contains source code for the

DSP target. Appendix D contains source code for the class library created to enable motion control commands from within a Windows application.

Chapter 2

Intelligent Machining:

Relevant Concepts, Components, Framework and Technology

Figure 2.1 shows that the machining process is automatically controlled via three independent machine control variables, namely cutting speed, feed and depth of cut. These variables modulate the dependent variables of the process (performance measures), such as, workpiece surface roughness, tool-workpiece vibration, cutting power, tool temperature, cutting forces, spindle current and cutting sound and contribute to tool wear. It shows the components and concepts researched in implementing intelligent machining, namely: digital signal processing (DSP) for sensor measurement, intelligent systems for monitoring and intelligent diagnostic, multi-axis control for machine control capability and regression and correlation analysis for process modeling.

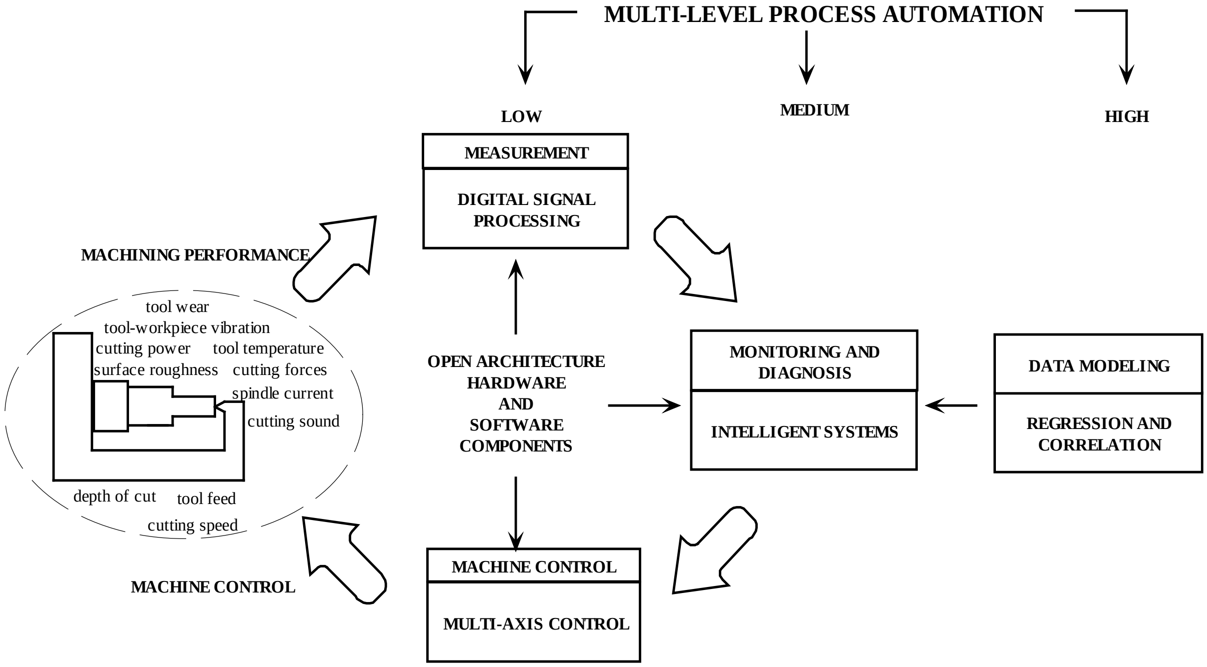


Figure 2.1: Relevant components and concepts in the monitoring, diagnosis and control for intelligent machining.

This chapter commences with a definition of intelligence and hence describes the intelligent machining. It introduces the fundamental mathematical relationship that exists between the independent (input) and dependent variables (output). These relationships will be used as a foundation for developing a steady state machining process model for single point turning, using experimental data (Chapter 5). The methods used to obtain the process relationships from experimental data are explained. The fundamentals of DSP which are used to measure and process sensor signals are introduced. The fundamental concepts to realize intelligent systems, in particular uncertainty, neural networks, fuzzy logic and NF are explained. These artificial intelligence concepts which are explained will be used in the monitoring of tool wear and surface roughness (Chapter 4), and in the intelligent diagnosis for machining parameter control (Chapter 5). PC-based hardware and software technology used in this project to implement an open architecture-based intelligent machine controller is introduced. Finally, a framework for intelligent machining is explained.

2.1 Intelligent Machining

Intelligence is the ability of a human being to acquire knowledge and apply it by means of thinking and reasoning [6]. Artificial intelligence is a discipline which studies how humans solve problems intelligently, and how machines can emulate this human problem-solving ability [7]. Alternatively stated: how to make machines smarter by investing them with human intelligence. Expert systems, fuzzy logic and neural networks systems belong to a paradigm of so called intelligent systems. Harber et al [3] define intelligent machining as:

“A computationally efficient procedure developed combining one or more intelligent techniques (fuzzy logic, neural networks e.g.) and expert criteria (operator knowledge), with one or more higher resolution levels (hierarchical levels), which basically manipulate cutting conditions (spindle speed, feed) and should be monitoring tool status and finished surface quality, as well as increasing productivity through higher metal removal rate“

The monitoring of tool status and surface roughness by means of intelligent systems will enhance automated machining (Chapter 4). However, the primary difference between automated machining and intelligent machining is that an intelligent system (applied in the latter) is capable of making decisions based on significant information from the machining process. Intelligent control of machining process parameters can be treated as a decision-making problem [1]. The diagnostic process can be formulated in a manner similar to the one in which a human being would proceed, for example:

(i) Select the alternatives at a given decision point.

(ii) Select the applicable criteria to evaluate the different alternatives.

(iii) Calculate or estimate the selection parameters for each of the proposed alternatives.

(iv) Through decision rules select the best alternative.

2.2Machining Process

Figure 2.2 shows the machining process parameters, including cutting forces, cutting power, surface roughness, tool-workpiece temperature, tool-workpiece vibration, cutting sound and cutting torque / spindle current that characterize the systems performance. Key factors that affect the machining performance parameters include tool wear and machine control parameters.

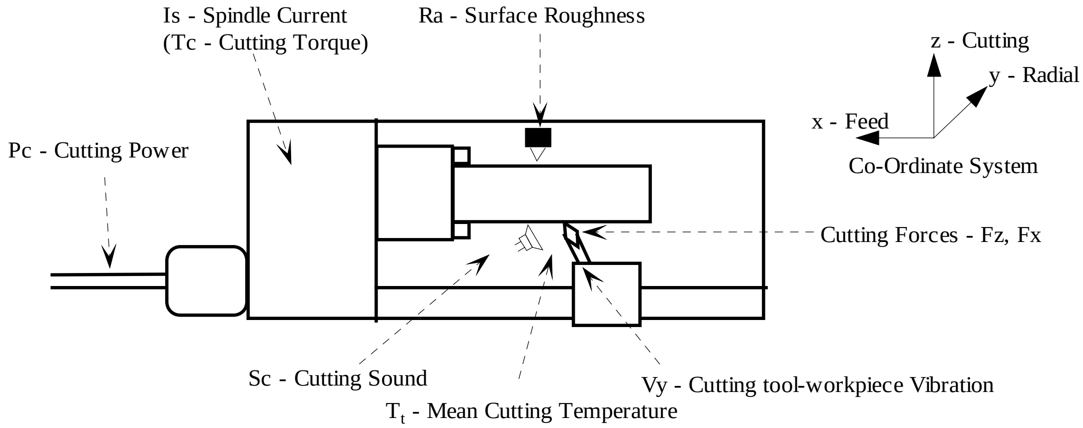


Figure 2.2: Parameters that characterize machining process performance.

The following subsections show and describe the basic mathematical relationships between the dependent performance variables and cutting parameters (independent variables) as well as describe how the particular dependent parameter/s influence the cutting tool / product quality / machine tool.

2.2.1 Cutting Forces, Torque and Power

The cutting force, acting in the direction of the cutting speed, supplies the energy for cutting and depends mainly on the work material, feed and depth of cut [8]:

$$F_Z = C_f \quad f1_X^a d_Y^b \quad [N] \qquad 2.1$$

Constants a and b depend on the cutting tool-workpiece combination. If F_Z and diameter, $D_{AVE,}$ is known the cutting torque, T_C, is given as:

$$T_C = F_Z \frac{D_{AVE}}{2} \quad [Nm] \qquad 2.2$$

The basic equation for cutting power, P_C is given as:

$$P_C = C_P F_Z V_C \quad [Watts] \qquad 2.3$$

P_C is also calculated as:

$$P_C = T_C \frac{2\Pi N}{60} \quad [Watts] \qquad 2.4$$

The on-line measurement of cutting forces, torque / spindle current and power will enable intelligent machining to [8, 9, 10]:

- Manage the supply of torque and power available from machine tool in order to meet on-line load requirements.
- Avoid excessive damage to machine elements and maintain desired tolerances for machined part.
- React to excessive increase of forces, torque and/or power resulting from tool wear.
- Protect the workpiece from the application of high cutting force / torque which may cause excessive distortion.
- Reduce excessive feed force that may cause the tool to deflect and result in surface waviness error [11].
- Reduce axial force so that it does not exceed the work holding pre-load, otherwise the workpiece will loose its rigidity [12].

2.2.2 Cutting Temperature

The energy dissipated in cutting operations is converted into heat, which, in turn raises the temperature in the cutting zone. The mean temperature in turning on a lathe is found to be proportional to the cutting speed and feed as follows [9]:

$$T_t = C_t V_C^a f 1_X^b \tag{2.5}$$

Constants a and b depend on the tool-workpiece combination.

The on-line measurement of cutting tool temperature will enable intelligent machining to [8, 9, 13, 14]:

- Reduce the rate of wear, as tool wear has been shown to be strongly temperature dependent. It adversely affects the strength, hardness, and wear resistance of the cutting tool.
- Increase tool life as temperature is inversely related to tool life.
- Improve accuracy as increased heat causes dimensional changes in the part being machined, making control of dimensional accuracy difficult.
- Reduce thermal damage to the machined surface as it adversely affects properties like fatigue life and corrosion resistance.
- Avoid the critical temperature of the tool-workpiece combination, as it will cause the two materials to interfuse. Chip particles welded to the surface of the tool are swept away and tear out minute chunks of tool material.
- Avoid the increase in machine tool temperature as it may cause distortion of the machine and result in poor dimensional control of the workpiece.

2.2.3 Tool Wear

RF Taylor recognized that tool wear is dependent on the cutting velocity and developed the following equation using data from tool life test [8]:

$$V_C T^n = C_T \tag{2.6}$$

V_C is the cutting speed, T^n the tool lifetime in minutes (the time recorded to develop a certain wear land, VB), n is an exponent that depends on cutting tool and workpiece materials as well as cutting conditions. C_T is a constant and represents a cutting speed for a tool life of one minute.

Although cutting speed is the most significant process variable in tool life the depth of cut and feed are also significant, hence from Equation 2.6 Taylor's expression is expanded as:

$$V_C T^n d_Y^a f 1_X^b = C_T \qquad 2.7$$

Flank wear land, as shown in Figure 2.3, has been commonly used in the measure of tool wear.

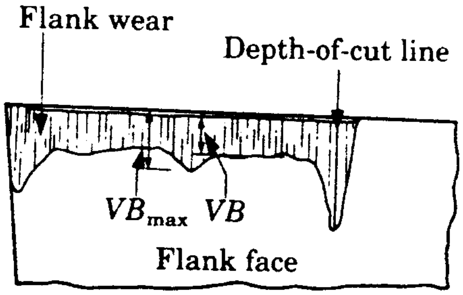


Figure 2.3: Flank wear land of a cutting tool.

To determine V_B the tool life test must be stopped and a measurement made, using optical instruments, like a scanning electron microscope, at suitable magnification levels. Signals from sensed dependent variables, influenced by tool wear, may be processed into a frequency spectrum by means of FFT. The power spectral densities that are most sensitive to tool wear are selected and fed into a previously trained artificial neural network to determine the state of the cutting tool [15]. On-line monitoring of tool wear is important as:

- Tool wear land will reach a limit before tool breakage / chipping occur, which in turn may cause severele damage to the machine tool and surface roughness of the workpiece.
- It increases the cutting forces, which in turn may cause plastic deformation of the workpiece.

- It negatively influences the dimensional accuracy of the workpiece.
- It may influence the tool-workpiece interaction which in turn may contribute to increased vibration

2.2.4 Surface Roughness

Roughness refers to relatively finely spaced surface irregularities as produced by the action of a cutting tool during a machining operation. The tool leaves a spiral profile - feed marks - on the machined surface as it moves across the workpiece, and this is given by [16]:

$$R_a = \frac{1}{8r_e} f 1_X^2 \quad [\mu m] \tag{2.8}$$

The higher the feed and the smaller the tool-nose radius (r_e), the more prominent the feed marks. Feed seems to affect surface roughness much more than depth of cut. The on-line monitoring of surface roughness will reduce part-manufacturing cost. The measurement of surface roughness is done by manual inspections of the work surfaces using profilometers. Manual inspection is time consuming and very costly. Furthermore the on-line measurement will in turn enable intelligent machining to:

- Maintain the quality of the machined product [17].
- Ensure that surface residual stresses that contribute to part failure, may be kept at a minimum. Residual stresses on the surface of a component are mainly, like surface roughness, influenced by feed [18]. By ensuring that the surface roughness is maintained below a threshold the residual stresses may be kept at a minimum.

2.2.5 Material Removal Rate

The material removal rate (MRR) is the volume of material removed per unit time, and given as:

$$MRR = V_C f1_X d_Y \, [mm^3/\sec] \qquad 2.9$$

Knowledge of MRR is important as the main criteria for optimal performance is the rate of metal removal of material from the work piece. The cutting time for a work piece of length L can be calculated as:

$$t = \frac{L}{f1_X N} \, [\min] \qquad 2.10$$

2.2.6 Tool-Workpiece Vibration

Metal cutting operations is inherently cyclic and excessive vibration may be caused by a periodic applied force, present in the machine tool (forced vibration), or by a disturbance in the cutting zone (self-excited vibration) [8,9,19]. The basic solution in reducing forced vibrations is to isolate or remove the forcing element.

A relationship exists between the fundamental frequency of a workpiece and the spindle speed [20]. The situation often occurs that the machining process is stable in the cutting zone, but once it reaches the middle position of the workpiece, excessive vibration (chatter) begins to develop. Cutting forces build up as the tool penetrates the material and deflect the tool. When shearing occurs to form the chip, the forces momentarily drop and the tool springs back. Vibration increases when the cutting forces get out of phase with the tool forces. The relationship between the cutting force and the amplitude of the tool vibration is given as [21]:

$$A_z = C_Z F_Z^a N^b \, [mm] \qquad 2.11$$

Where A_z is the amplitude of the tool, C_Z is a constant, F_Z is the cutting force, N is the spindle speed and a and b are exponential constants.

The on-line measurement of tool-workpiece vibration will enable intelligent machining to:

- Improve the surface quality, dimensional accuracy, productivity and even safety [22, 23].
- Reduce damage to machine tool components that may result from excessive vibrations.
- Reduce premature tool wear and chipping.

2.2.7 Cutting Sound

Sound and vibration occur as a result of the machining process. In general, the range of frequencies that are important in acoustics and associated vibrations, lie in the audible range of 20 to 20 000Hz. However, disturbances above 1000 Hz are generally reduced using passive techniques, for example, machine tool design [24]. Whereas active sound and vibration control has found its use in the 50 to 1000 Hz range [25]. The on-line measurement of sound may be used in monitoring of on-line tool conditions: Tool wear, tool chipping and tool breakage [26] and enable intelligent machining to:

- Reduce excessive vibration in the cutting process [27].
- Reduce objectionable noise generated.

2.3Data Modeling by means of Multiple Regression and Correlation

Given experimental data regression analysis provides the basis for predicting the values of a dependent variable (Y) from values of one or more other independent (X_1, X_2 ..) variables. These relationships are used for modeling and simulation of machining process to test intelligent decision making (Chapter 5). Correlation analysis enables us to assess the strength or degree of the relationships amongst the variables. It is used to find the membership of signal feature to a specific machine control action. The subsequent signal feature-control action relation is used for intelligent diagnosis (Chapter 5). Correlation analysis is also used in the monitoring of tool wear and surface roughness to determine which signal features influence theses parameters (Chapter 4).

A sample regression line describes the average relationship between X_i and Y variables in the sample data. The equation of this line, known as the sample regression equation, provides estimates of the mean value of Y for each value of X_i. Of all the curves approximating a given set of data points, the curve having the property that: $D_1^2 + D_2^2 + ... + D_N^2$ is a minimum, is called a best fitting curve. Where D_i is the deviation from the best-fit curve to a data point. A curve having this property is said to fit data in the least square and is called a least square curve. A multiple regression equation is an equation for estimating a dependent variable, say Y_1, from the independent variables X_2, X_3 and is called a regression equation of Y_1 on X_2, X_3. A multiple linear regression equation would be in the form:

$$Y_1 = f(X_2,X_3) = b_{1.23} + b_{12.3}X_2 + b_{13.2} X_3 \qquad 2.12$$

If X_3 is kept constant the graph of Y_1 versus X_2 is a straight line with slope $b_{12.3}$. If we keep X_2 constant the graph of Y_1 versus X_3 is a straight line with slope $b_{13.2}$. The subscripts after the dot indicate the variables held constant in each case. Due to the fact that Y_1 varies partially because of variation in X_2 and partially because of variation in X_3, we call $b_{12.3}$ and $b_{13.2}$ partial regression coefficients. From Equation 2.12, $b_{1.23}$, $b_{12.3}$ and $b_{13.2}$ are determined by solving the following normal equations [28, 29]:

$$\sum Y_1 = b_{1.23} N + b_{12.3} \sum X_2 + b_{13.2} \sum X_3$$

$$\sum Y_1 X_2 = b_{1.23} \sum X_2 + b_{12.3} \sum X_2^2 + b_{13.2} \sum X_2 X_3$$

$$\sum Y_1 X_3 = b_{1.23} \sum X_3 + b_{12.3} \sum X_2 X_3 + b_{13.2} \sum X_3^2 \qquad 2.13$$

Machining relations, as seen from Equations 2.1, 2.5, 2.8 and 2.11, are non-linear and generally expressed as:

$$Y = C\ X_1^a X_2^b \qquad 2.14$$

Equation 2.14 may be linearized, using a functional transformation, by taking log on both sides as follows [30]:

$$Log_{10}(Y) = Log_{10}(C) + a\ Log_{10}(X_1) + b\ Log_{10}(X_2) \qquad 2.15$$

The coefficients for Equation 2.14 may then be calculated using the method described in Equation 2.13. Correlation is the degree of relationship between variables, which seeks to determine how well a linear or other equation describes or explains the relationship between variables. The degree of relationship that existing between three or more variables is called multiple correlations. The ratio of the explained variation to the total variation is called the coefficient of determination, given as:

$$r^2 = \frac{\exp lained \operatorname{var} iation}{total \operatorname{var} iation} = \frac{\sum\left(Y_{est} - \bar{Y}\right)^2}{\sum\left(Y - \bar{Y}\right)^2} \qquad 2.16$$

The coefficient of determination may be interpreted as the proportion of variation in the dependent variable Y that has been accounted for, or “explained,” by the relationship Y and X expressed in the regression line. To determine the linear partial correlation coefficient between variables Y_1 and X_2, ignoring X_3 [31]:

$$r_{12} = \frac{N\ \sum Y_1 X_2 - \left(\sum Y_1\right)\left(\sum X_2\right)}{\sqrt{\left[N\ \sum Y_1^2 - \left(\Sigma Y_1\right)^2\right]\left[\left[N\ \sum X_2^2 - \left(\Sigma X_2\right)^2\right]\right]}} \qquad 2.17$$

The coefficient of linear multiple correlation of Y_1 on X_2 and X_3 may be calculated from the partial coefficients:

$$R_{1.23}=\sqrt{\frac{r_{12}^{2}+r_{13}^{2}-2r_{12}r_{13}r_{23}}{1-r_{23}^{2}}} \qquad 2.18$$

Note that the coefficients of multiple correlations $R_{1.23}$ are larger than either of the coefficients r_{12} or r_{13}. This is always true since, by taking into account additional relevant independent variables, we should arrive at a better relationship between variables.

The coefficients of partial correlation, designated $r_{y1.2}$ would indicate the partial correlation between Y and X_1 after the effect of X_2 on Y had been removed. The square of this coefficient measures the reduction in variance brought about by introducing X_1 after X_2 has already been accounted for. Sometimes it is difficult to compare the differences in net regression because the independent variables are stated in different units.

To improve comparability, we can state the regression equation in a different form, giving each of the variables in units of its own standard deviation. The transformed regression coefficients are called BETA coefficients. In term of BETA coefficients, the linear regression equation for three variables would be:

$$\frac{Y}{s_Y}=\alpha+\beta_1\frac{X_1}{s_{X_1}}+\beta_2\frac{X_2}{s_{X_2}} \qquad 2.19$$

Thus, the ß$_i$ coefficients are equal to [32]:

$$\beta_i=b_i\frac{s_{X_i}}{s_Y} \qquad 2.20$$

ß$_i$ measures the number of standard deviations that that Y changes with each change of one standard deviation in X_i.

2.4Digital Signal Processing

Advanced monitoring and diagnostic systems, to enhance intelligent machining, employ multiple sensors, and the signals from these sensors are sampled and signal processed. Signal processing is used to determine signal features, and may include: A fast fourier transform (FFT) algorithm to produce a frequency spectrum from where power spectral densities may be analyzed by way of calculating its mean, root mean square etc. Signal features that are sensitive to tool condition, tool wear, machine state classification etc, may be extracted as part of the particular monitoring system [15, 33]. In this project the primary function of DSP is to sample and determine the rms value of the sensor signals for on-line monitoring (Chapter 4) and diagnosis (Chapter 5) purposes. However, in addition the cutting forces, cutting sound, tool-workpiece vibration, tool temperature, spindle motor current sensor signals sampled and signal processed were further analyzed to determine which additional signal features are sensitive to tool wear (Chapter 4).

Signal processing is concerned with the mathematical representation of a signal in the domain of the original dependent variable i.e. time domain, or in a transformed domain i.e. frequency domain, and with the algorithmic manipulation of the signal to extract the information being carried. Figure 2.4 show a block diagram of the signal processing functions implemented including continuous to discrete domain conversion, Finite Impulse Response Filtering (FIR), discrete Fourier Transform (DFT), and signal data extraction.

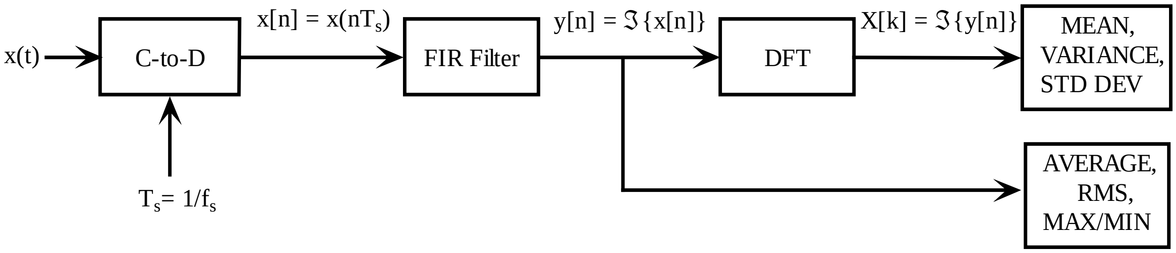


Figure 2.4: Signal processing functions.

The continuous domain signal x(t) is sampled, using an analog-to-digital conversion, at regular intervals of T_s to obtain discrete signal x[n]. To eliminate unwanted signal components the discrete domain signal x[n] is passed through a low-pass FIR filter of bandwidth 0 to $1/(2F_s)$ to obtain y[n] from where the frequency spectrum components X[k] are obtained using a DFT. Finally signal features is extracted from y[n] and X[k]. The following subsections describe the mathematical concepts of the functional blocks, whilst Chapter 3 describes the software implementation there off.

2.4.1 Sampling Process

11 If signal x(t) contains no frequencies higher than $F_S/2$ hertz, where F_s is the sampling rate, it is completely determined by the set of its values at regularly spaced intervals of period $T_s = 1/F_S$[34].

In the sampled series x[n]: x[0] corresponds to the input value at t=0, x[1] is the value at t = T_s, x[2] is the value at t = $2T_s$, and so on. The process of uniformly sampling a signal in the time domain results in a periodic spectrum in the frequency domain with a period equal to the sampling rate [35].

2.4.2 FIR Filter

The general definition of a FIR filter is [36]:

$$y[n] = \sum_{k=0}^{M} b_k \, x[n-k] \tag{2.21}$$

The filter coefficients b_k in Equation 2.21 are identical to the impulse response values h[k] of the filter, and may be written as:

$$y[n] = \sum_{k=0}^{M} h[k] \; x[n-k] \tag{2.22}$$

The operation performed in Equation 2.22 is known as a finite convolution sum and expressed as:

$$y[n] = x[n] \otimes h[n] \tag{2.23}$$

The design of a digital filter includes: Filter specification, coefficient calculation, realization and implementation.

2.4.2.1 Filter Specification

Filtering of the sensors in this project was limited to that of low pass filter whose tolerance specification scheme, specified in the frequency domain, is shown in Figure 2.5. Because of the symmetry and periodicity of the magnitude response $\left|H(e^{j\omega})\right|$, it is sufficient to give the specifications only for 0□ ω □□.

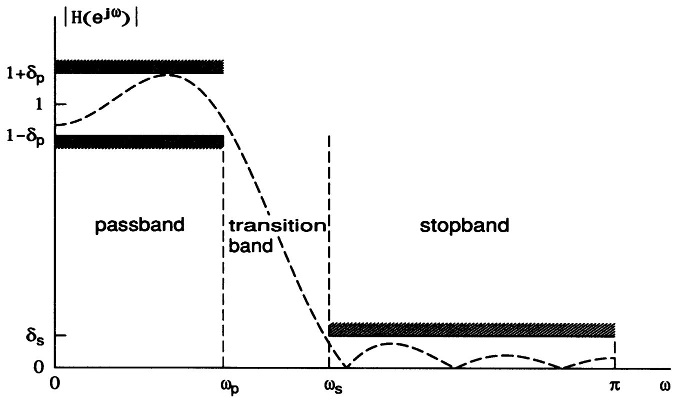


Figure 2.5: Magnitude-frequency response specification for a low pass filter [37].

In the low pass case, the desired magnitude response is usually given by:

$$D(\omega) = 1 \quad for \quad \omega \in [0, \omega_p], passband\ of\ filter$$
$$0 \quad for \quad \omega \in [\omega_p, \pi], stopband\ of\ filter$$

The specification includes a transition band (ω_s - ω_p) of nonzero width in which the filter response changes from unity in the pass band to zero in the stop band. The amplitudes of the allowable ripples expressed in decibel as, an [37]:

$$A_p = 20\log_{10}\left(\frac{1+\delta_p}{1-\delta_p}\right) dB \qquad and$$
$$A_s = -20\log_{10}(\delta_s) dB \qquad 2.24$$

2.4.2.2 Coefficient Calculation

The objective of FIR coefficient calculation methods is to obtain values of h[n] such that the resulting filter meets the design specifications expressed in Section 2.4.2.1. A popular approach is to use the infinite-duration response coefficients of an ideal filter, $h_D[n]$, and then to truncate and smooth the response by using a window function (w[n]), hence $h[n] = h_D[n]w[n]$. The impulse response coefficients for $h_D[n]$ filter is given as [38, 39]:

$$h_D[n] = 2 f_c \frac{\sin(n\,\omega_c)}{n\,\omega_c} \qquad n = 0$$
$$h_D[n] = 2 f_c \qquad n \neq 0 \qquad 2.25$$

And Hamming window coefficients as:

$$w_H[n] = 0.54 + 0.46\cos(\frac{2\pi n}{N}) \quad for \quad -(N-1)/2 < n < (N-1)/2 \quad (N\,odd)$$

$$-N/2 < n < N/2 \qquad (N even) \qquad 2.26$$

$$w_H[n] = 0 \qquad elsewhere$$

The transition width for a filter designed with the Hamming window and filter length N is determined from,

$$\Delta f = 3.3/N \qquad 2.27$$

The maximum stop band attenuation possible with the Hamming window is given as about 53dB, and the minimum peak pass band ripple is about 0.194dB, which is sufficient for this project. APPENDIX A SHOWS AN EXAMPLE IN OBTAINING THE FILTER COEFFICIENTS OF AN FIR LOWPASS FILTER TO MEET THE TYPICAL SPECIFICATIONS AS USED IN THIS PROJECT:

Pass band edge frequency: 10kHz   Transition width: 420 Hz

Sampling frequency: 5.0kHz

2.4.2.3 Realization structure

From Equation 2.22 the FIR filter may also be characterized by the transfer function (H(z)), the z— transform of the impulse response h[k], given by [36]:

$$H(z) = \sum_{n=0}^{M} h[k]\,z^{-k} \qquad 2.28$$

The realization structure for the FIR filter is essentially a block diagram representation of the transfer function. Although the implementation of Equation 2.28 may lead to several variations, the transversal structure, shown in Figure 2.6, is most often selected as it leads to the most efficient implementation [40].

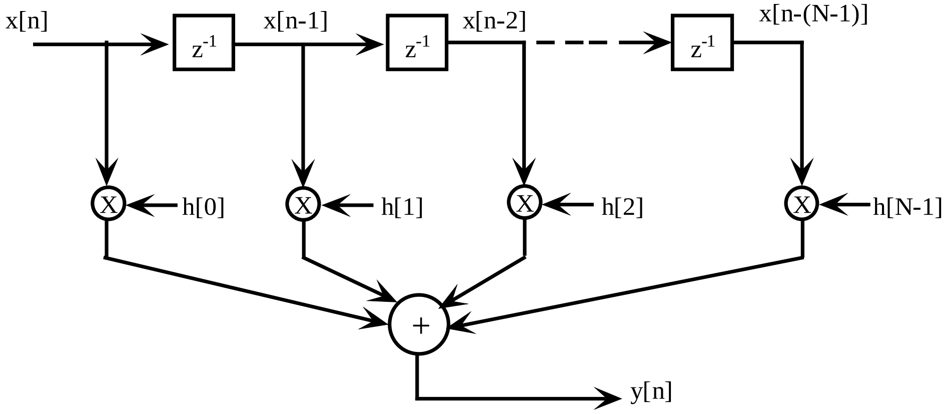


Figure 2.6: Transversal structure for the implementation of a FIR filter.

The symbol Z^{-1} represents a delay of one sample of time (T_s), also known as the unit delay. For the transversal structure, the computation of each output sample, y[n], requires: N-1 memory locations to store N-1 input samples, N memory locations to store the N co-efficient, N multiplications, and N-1 additions. The FIR filter can be adapted to construct a linear phase response by mirroring the values of the coefficients around the center tap, so that: h[0] = h[N], h[1] = h[N-1] etc.

2.4.2.4 Implementation

The final stage is to implement the filter for real-time operation, and the key issue is to produce software code of the chosen filter structure. The Texas Instruments TMS320C30 DSP processor, used in this project, has an architecture and instruction sets optimized for FIR filtering operations [41]. The DSP technology used in this project is introduced in Section 2.6 and the software implementation of the FIR operations for this project is covered in Chapter 3.

2.4.3 Frequency Spectrum

When a signal is non-repetitive (aperiodic), it can be expressed as the infinite sum (integral) of sinusoids, which are not harmonically related. The corresponding spectrum is continuous and is described mathematically by the FT [42]. The DFT is useful for the analysis of discrete-frequency representation of discrete-time sequence. The FFT is an efficient algorithm that can be used to obtain the discrete-frequency representation with fewer computations than the DFT. The DFT of data with a finite number of nonzero sample values, x[n] defined over the range 0<n<(L-1), is given by [41]:

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{L-1} x[n]\ e^{-j(2\pi kn/L)} \qquad\qquad 0 \le k \le (L-1) \\
&= \sum_{n=0}^{L-1} x[n] W_L^{kn}
\end{aligned}
$$

where

$$W_L^{kn} = e^{-j(2\pi/L)} \qquad 2.29$$

The number of multiplications required calculating X[k] is proportional to L^2. W_L^{kn} , also known as twiddle factors, is a periodic function with a limited number of distinct values. A highly efficient algorithm for computing the DFT, known as a FFT, makes use of this feature to reduce the number of multiplications in determining X[k].

Equation 2.28, X[k], may further be decomposed as [41, 43]:

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{L/2-1} x[2n]\ W_{L/2}^{nk} + W_L^k \sum_{n=0}^{L/2-1} x[2n+1]\ W_{L/2}^{nk} \\
&= G[k] + W_L^k H[k]
\end{aligned}
\qquad 2.30
$$

which expresses the original L-point DFT in terms of two L/2-pointDFT, G[k] (transform of even-numbered points in x[n]) and H[k] (transform of odd-numbered points in x[n]). The block diagram shown in Figure 2.7 shows how each L/2-point subsequence may further be decomposed into two shorter L/4-point subsequences.

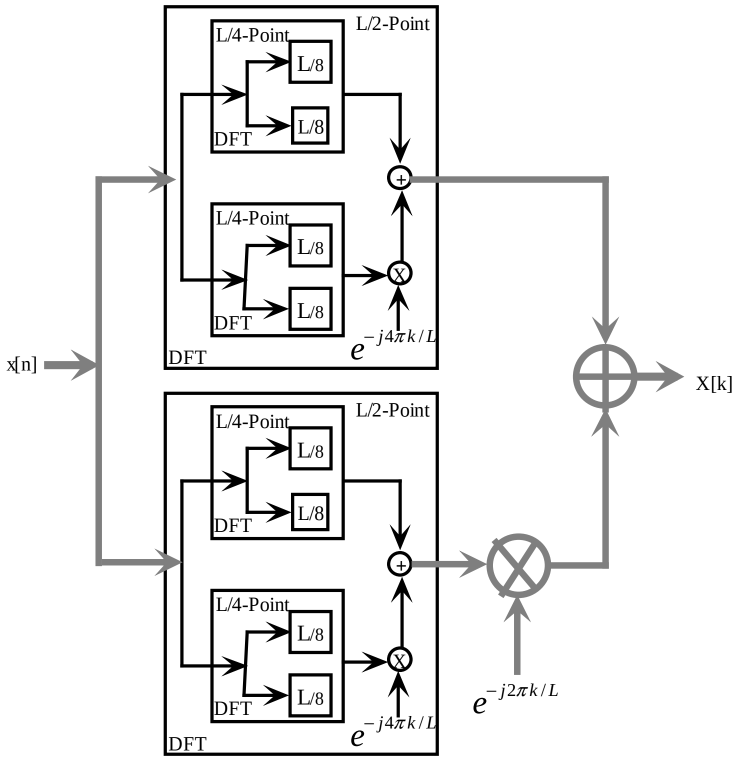


Figure 2.7: Block diagram of a radix-2 FFT algorithm [36].

The process can continue until, in the limit, we are left with a series of 2-point subsequences, each requiring a very simple 2-point DFT, leading to the most commonly used radix-2, decimation-in-time. Using a direct DFT the amount of complex multiplications is in the order L^2, however, if L is an inter power of 2, and the FFT decomposition proceeds right down to 2-popint transforms, there are $\log_2 L$ stages of FFT computations giving a total of:

No of Complex multiplications to perform FFT = $L\log_2 L$ 2.31

For $L=512=2^9$ the speed advantage is nearly 57.

2.4.4 Statistical Processing of Signal Spectrum

Statistical properties of the signal spectrum are used in system identification, properties may include [43, 44]:

- Mean value

$$\mu = \frac{\sum_{k=0}^{n-1} x^k}{n} \qquad 2.32$$

- Variance and standard deviation

$$\sigma^2 = \frac{\sum_{k=0}^{n-1} (x_k - \mu)^2}{n-1} \qquad 2.33$$

The square root of the variance is called the standard deviation (σ).

2.5 Intelligent System Components

Artificial intelligence is a discipline which studies the way in which humans solve problems intelligently, and how machines can emulate this human problem solving ability. Expert systems, FL and NN systems belong to a new paradigm of so-called intelligent systems. A so-called intelligent system gives appropriate problem-solving responses to problem inputs, even if such inputs are new and unexpected. Humans are such intelligent systems. At this moment, there is a considerable mismatch between humans and machines, in as much as humans reason in inaccurate, multi-valued, fuzzy ways while machines are based on bi-valent, binary reasoning. Eliminating this mismatch would make machines more intelligent, that is, they would be enabled to reason in a fuzzy manner, like humans. The following subsections describe the fundamental concepts of NN, FL and NF to realize models for the indirect measurement of tool wear and surface roughness respectively (Chapter 4). It introduces the concept of a fuzzy relation, which is used by the intelligent diagnosis (Chapter 5).

2.5.1 Uncertainty

Formulae describe deterministic processes - one where there is no uncertainty in the physics of the process (i.e. the right formula) and there is no uncertainty in the parameters of the process (i.e. the coefficients are known with precision. Information from physical processes virtually always contains uncertainty. There is uncertainty that arises from imprecision, from the inability to perform adequate measurements, from lack of knowledge or from vagueness. For example, uncertainty may be defined as the lack of adequate information to make a decision. Uncertainty is therefore a problem, as it may prevent us from making the best decision and may even cause a biased decision. A human operator might not have a deep understanding of the plant dynamics that he is controlling, but he knows what action to take whenever he observes certain conditions, such as combinations from instrument readings. Therefore we say that the human operator has the ability to overcome the uncertainties of the controlled system dynamics. Certainty factors (CNF) (or confidence factors) are one of the most common methods of dealing with uncertainty in rule-based systems [45].

Certainty factors may be associated with facts and with rules, for example:

Fact: <condition> = TRUE CNF <value>

Rule: If <condition> then <action> CNF <value> 2.34

The condition and action is known with a degree of certainty.

2.5.2 Fuzzy System

It is particularly noticed that expert system-based approaches for on-line machining condition monitoring, although effective due to the ability of dealing with the uncertainties, are often inadequate for the fast reaction requirements in a low-level machine control, especially when involving a complex knowledge base [33]. Fuzzy logic seems to be a unique method of dealing with uncertainties, especially in the control of physical systems. A fuzzy system consists of four principal components as shown in Figure 2.8: A fuzzifier, rule base, inference engine and defuzzifier [46].

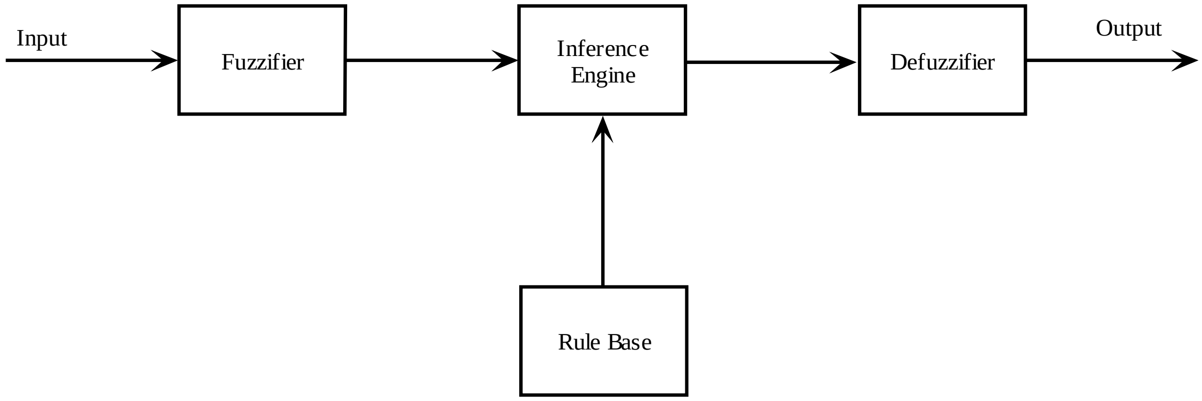


Figure 2.8: General components of a fuzzy logic system.

2.5.2.1 Fuzzifier

The assignment of linguistic values, defined by membership functions, to a sensor input, which yields "fuzzified" values of the original signal input. Using the triangular membership function a linguistic variable "small", "medium" and "large" may be illustrated as:

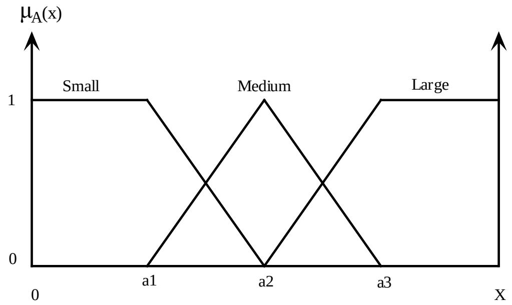


Figure 2.9: Triangular membership function for sensor data.

For fuzzy "small", its fuzzy function may be expressed as:

$$\mu_{A\,small}(x) = \begin{cases} 1 & x \le a_1 \\ \dfrac{x - a_2}{a_1 - a_2} & a_1 \le x \le a_2 \\ 0 & otherwise \end{cases} \qquad \textbf{2.35}$$

For fuzzy "medium", the function is given as:

$$\mu_{A\,medium}(x) = \begin{cases} \dfrac{x - a_1}{a_2 - a_1} & a_1 \le x \le a_2 \\ \dfrac{x - a_3}{a_2 - a_3} & a_2 \le x \le a_3 \\ 0 & otherwise \end{cases} \qquad \textbf{2.36}$$

And for fuzzy "large", the function is given as:

$$\mu_{A\,l\arg e}(x) = \begin{cases} \dfrac{x - a_2}{a_3 - a_2} & a_2 \le x \le a_3 \\ 1 & a_3 \le x \\ 0 & otherwise \end{cases} \qquad \textbf{2.37}$$

Where a_1, a_2, a_3 are parameters to determine the positions of the membership functions, as well as affect the shape of the membership functions. The fuzzifier computes for each sensed input, through the above fuzzy membership functions, values indicating as to what degree the input belongs to the "small", "medium" and "large" linguistic terms. A fuzzy membership function expressed generally as a fuzzy set with finite input values:

$$\overline{A}_i = \left\{ \mu_{Ai1}(x_1), \mu_{Ai2}(x_2), \mu_{Ai3}(x_3), \ldots, \mu_n(x_n) \right\} \qquad \textbf{2.38}$$

Where $\mu_{Aij}(x_j)$ is the membership values for all the possible sensed input values.

2.5.2.2 Knowledge Base and Inference Engine

The knowledge base may be represented as a fuzzy relation or as a linguistic fuzzy rule base with membership functions as a database.

- Fuzzy Relation (FR)

In control systems relationships are defined between system inputs and outputs. These mappings are between variables defined on different universes of discourse through the statement:

$$\bar{A} \xrightarrow{R} \bar{B}$$

If A(x) THEN B(y) through relation R

Where the condition set A is linked to the result set B through relation R. A relation R of universe of discourse, is defined as a subset of the Cartesian product:

$$\bar{R} \subset \bar{A} \mathrm{X}\ \bar{B}$$

and illustrated in matrix form as follows:

$$\bar{R}_{\bar{A}X\bar{B}} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccc} y_1 & y_2 & y_3 \end{array} \\ \left\{ \begin{array}{ccc} \mu_{11} & \mu_{12} & \mu_{13} \\ \mu_{21} & \mu_{22} & \mu_{23} \\ \mu_{31} & \mu_{32} & \mu_{33} \end{array} \right\} \end{array} \qquad 2.39$$

Where x$_i$ is the elements of A and y$_i$ is elements of B. The $\mu_R(x,y)$ values are the membership values for each element in the relation R and corresponds to the strength of connection or correlation between A and B in the mapping [47]. It is also called mapping intensity function. The determination of the relation R is called system identification.

The decision-making logic is embodied by an inference structure that has the capability of simulating human decision-making. The membership vector of B can be calculated from the given relation matrix R and input membership vector A, as follows:

$$\mu_B(y) = \sum \mu_A(x) \cdot \mu_R(x,y) \qquad 2.40$$

Where □ represents the Boolean sum, where the membership vector completely defines a set, therefore we can re-write Equation 2.39 as:

$$\overline{B}(y) = \overline{A}(x) \, o \, \overline{R}(x,y) \qquad 2.41$$

Where "°" is the compositional operator which is usually *max-min* or *max-product*. If the compositional operator is max-min the membership of vector B(y) is calculated as:

$$\mu_{\bar{B}}(y) = \max_{x \subset E}\left[\min\left[\mu_{\bar{A}}(x) \; ; \mu_{\bar{R}}(x,y)\right]\right] \qquad 2.42$$

If the compositional operator is max-product the membership of vector B(y) is calculated as:

$$\mu_{\bar{B}}(y) = \max_{x \subset E}\left[\mu_{\bar{A}}(x) \, * \, \mu_{\bar{R}}(x,y)\right] \qquad 2.43$$

Instead of using the max function, the output fuzzy sets are weighted and logically summed.

- Fuzzy Rule Base

A classical fuzzy logical inference may be expressed as using the following max-min rule structure:

$$\begin{array}{l}
If \;\; x_1 \;\; is \;\; A_{i1} \;\; and \;\; x_2 \;\; is \;\; A_{i2} \;\; \ldots \;\; and \;\; x_m \;\; is \;\; A_{im}\,, \\
Then \;\; y \;\; is \;\; B_i \\
or \\
If \;\; x_1 \;\; is \;\; A_{i1} \;\; and \;\; x_2 \;\; is \;\; A_{i2} \;\; \ldots \;\; and \;\; x_m \;\; is \;\; A_{im}\,, \\
Then \;\; y \;\; is \;\; B_i
\end{array} \qquad \mathbf{2.44}$$

The computation of fuzzy rules is called fuzzy rule inference, and consists of the aggregation and composition of a rule's membership factors. Aggregation determines the degree to which the If-part of the rule is fulfilled. The sensed input values (x_1, x_2, x_3..) have been fuzzified by having assigned to them values from their associated fuzzy membership functions (A_{i1}, A_{i2}, A_{i3}…). The AND connective between the If-part implies either an intersection (min function) or an algebraic multiplication (product) between the fuzzy sets assigned to the input variables, whilst the OR connective between the If-part implies a union to connect the individual rules, as follows:

$$AND: \mu_{IF} = \min_i(\mu_i)$$

$$OR: \mu_{IF} = \max_i(\mu_i) \quad \textbf{2.45}$$

With Fuzzy Associative Map (FAM) inference, each rule is assigned a degree of support (DoS) representing the individual importance of the rule. The then part of a fuzzy rule is modified to:

$$\mu_{THEN} = \mu_{IF} * DoS \quad \textbf{2.46}$$

If more than one rule produces the same conclusion, an operator must aggregate the results of these rules, hence rule composition as follows:

$$MAX - results\ aggregation: \mu_{RESULT} = \max_i(\mu_{THEN,RULEi}) \quad \textbf{2.47}$$

2.5.2.3 Defuzzifier

Defuzzifier consists of deriving a single control action from an inferred fuzzy control action. Each control membership function, as shown in Figure 2.9, may include more than one valid evaluated output term. The defuzzification method is used to determine a compromise between all the different output terms.

The Center-of-Maximum (CoM) and Mean-of-Maximum (MOM) methods are selected for different types of applications and are used to calculate a single control output as follows:

$$CoM-Defuzzification\!: Y=\frac{\sum_j \left(\mu_{RESULT\_TERMj} {*}Y_{\,j}\right)}{\sum_j \mu_{RESULT\_TERMj}}$$

2.48

$$MoM-Defuzzification\!: Y=Y_{\,j}\left(\mu_{RESULTTERM\_MAX}\right)$$ 2.49

2.5.3 Neural Networks

Figure 2.10 shows the architecture of a feed-forward artificial ANN, in which neurons (simple asynchronous processing elements) are configured in layers, with each neuron able to send a signal, along weighted connections, to other neurons [48].

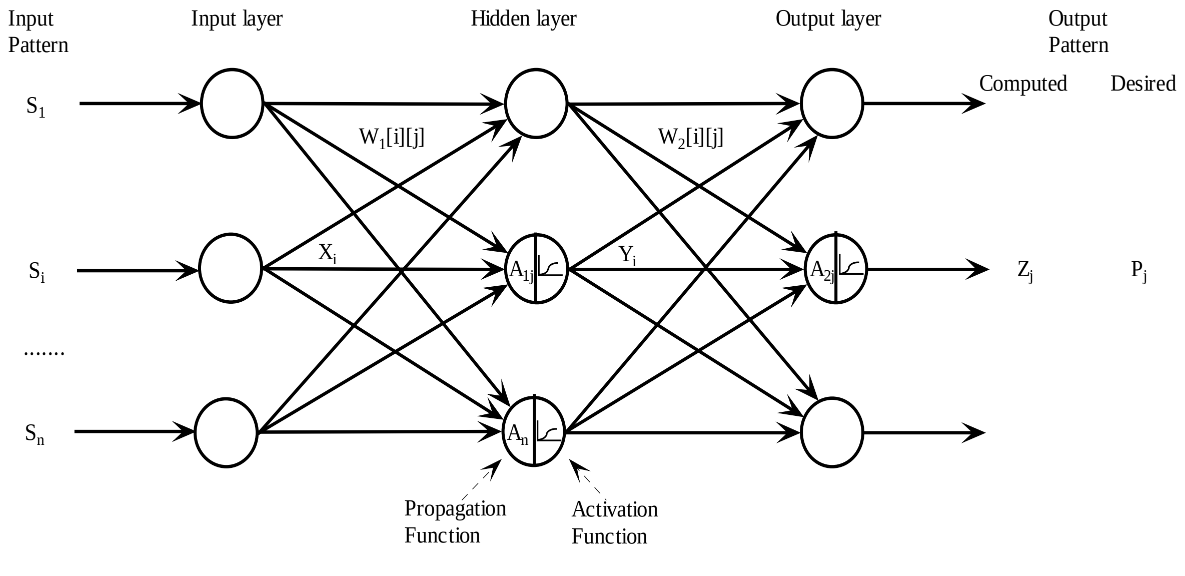


Figure 2.10: Basic feed-forward neural network processing elements.

The propagation functions A_{1j}, A_{2j} combine input signals X_i, Y_i from sending neurons, respectively. The means of combination is a weighted sum, with the weights of nodes given by matrixes W_1[i][j]

and W_2[i][j]. The total activity received by the neuron A_i and A_j is expressed by the propagation function:

$$A_{1j} = \sum_{i=1} X_i W_1[i][j] + \theta_{1j} \qquad \text{and} \qquad A_{2j} = \sum_{i=1} Y_i W_2[i][j] + \theta_{2j} \qquad 2.50$$

Where θ , is an offset added to the weighted sum. The so-called activation function computes the output signal for probabilistic type neurons using:

$$Z_j = f_j(A_j) = \left(1 + e^{Aj}\right)^{-1} \qquad 2.51$$

The feed-forward ANN does not have feedback connections, but errors are back-propagated during training, an iterative process, to adjust connection weights and threshold values until the desired – calculated output value is less than a selected threshold for a specific training data set. The ith component of the error at the output layer and hidden layer is:

$$e = P_i - Z_i \quad \textbf{and} \quad t_i = Y_i(1 - Y_i)\left(\sum_j W_2[i][j]e_j\right)$$

2.52

Adjustment for weight between ith neuron in hidden layer and jth output neuron:

$$\Delta W_2[i][j] = \beta_0 Y_i e_j \quad \textbf{and} \quad \Delta W_1[i][j] = \beta_h X_i t_j$$

2.53

where β_o and β_h is the learning rate parameters.

2.5.4 Neuro-Fuzzy

In most sensor applications classification criteria are often expressed by sample data. This is typical for decision support problems, diagnosis or pattern recognition examples, and data analysis. Traditional artificial intelligence (AI) has transparent mechanisms, often expressed in terms of logical operations and rule-based representations, that are meaningful in modeling real systems. Although NN has exciting possibilities, it does not use structured knowledge with symbols as used by humans to express reasoning processes [49]. NF technology allows for the automated generation of fuzzy logic systems based on neural network trained data. NF combines the advantages of fuzzy systems—the transparent representation of knowledge and the ability to cope with uncertainties—

with the advantages of neural nets, the ability to learn. Figure 2.11 shows a general 5 – layer structure of a NF model, as well as indication of how to map an NN to a fuzzy logic system [50].

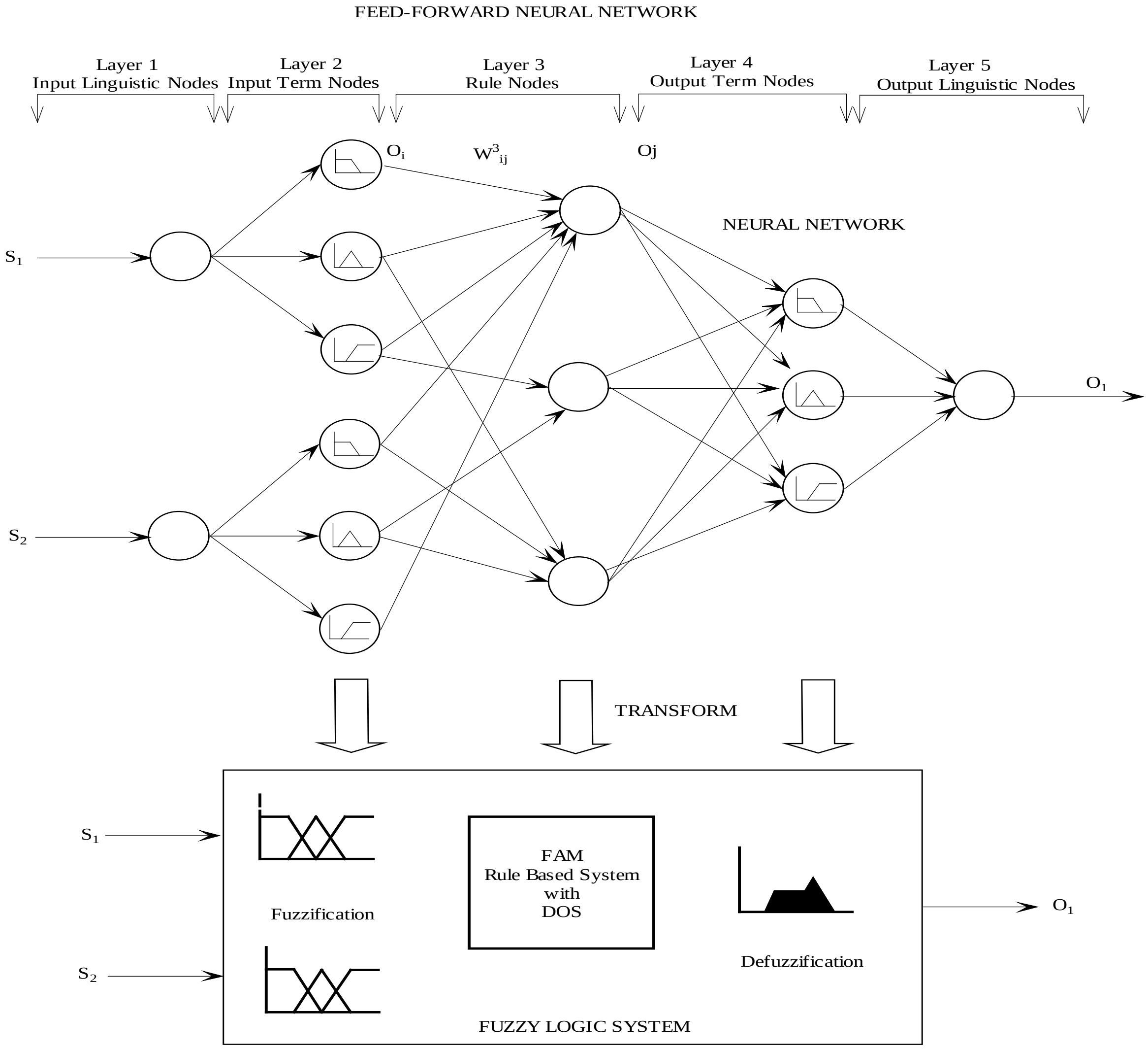


Figure 2.11: Neuro-fuzzy structure.

The linguistic nodes in layers one and five represent the input (S_1-S_n) and output (O_1-O_m) linguistic variables respectively. Nodes in layers two and four are term nodes acting as membership functions (MBFs) to represent the terms of the given linguistic variable. Each neuron of the third layer represents one fuzzy rule (rule nodes). Layer three links define the precondition of the rule and layer-four links incorporate the rule consequences. Initially these layers are fully connected representing all possible rules. A fuzzy system uses different units of computation for the input, hidden and output layers. A standard error back propagation algorithm cannot be used to calculate suitable weights for the neural network. The nodes in layer 1 simply transmit the input values to the next layer, O_i^2 = S$_i$ with unity weights W$_1$ = 1. Layer 2 nodes perform membership functions, e.g. triangular shaped functions, as shown in Figure 2.9 (Z-Lambda..Lambda-S) and Equations 2.35-2.37 to determine link weights.

Fuzzy rules, from Equation 2.44 are implemented in Layer 3 and Layer 4. Layer 3 performs fuzzy AND (min) operation with initial weights W_{ij}^3 =1, whilst nodes in layer 4 integrate the fired rules, having the same consequence by using the fuzzy OR (sum) operator also with W_{ij}^4 = 1. Rules may be represented by Fuzzy-Associative Maps (FAM), which is a fuzzy logic rule with an associated weight, known as rule firing strength (or Degree of Support DoS). Based on the rule firing strength, output $O_i^3(t)$ and the output from the nodes $O_i^4(t)$, the task is to decide the correct consequence link of each rule by competitive learning. The following learning law is used to update weights, where the basic is, learn if win:

$$W_{ij}^3\,(t) = O_j^4\left(-W_{ij}^3 + O_i^3\right) \qquad 2.54$$

Layer 5 performs de-fuzzification using one of Equations 2.48 or 2.49. NF training modules provide methods, based on the above description, for supervised learning. The method employed combines error back propagation with the idea of competitive learning. After a system output is computed by forward propagation, an error is identified by comparing the system output with the sample data [51]. This error is then used to determine the fuzzy rules most suited for influencing system behavior. Using the selected rule, the plausibility of the fuzzy rule is modified before subsequent data sets are processed.

2.5.5 Multi-Sensor Fusion

Figure 2.12 shows the interaction between sensors and the direct (solid-line) and indirect (dash-line) measurement of machining variables.

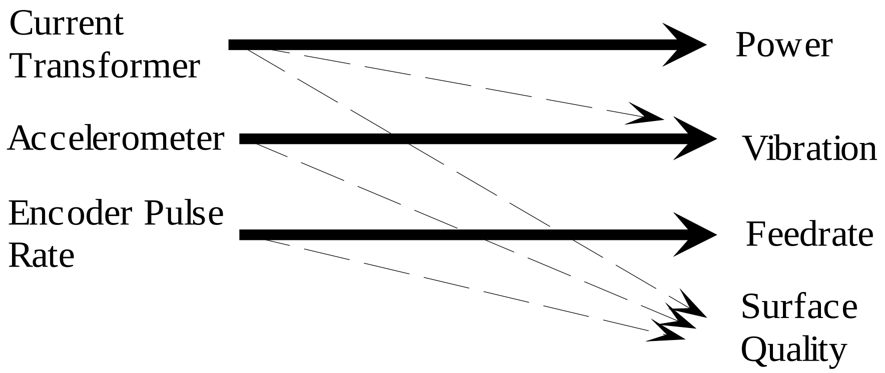


Figure 2.12: Parallel sensor - measured variable interaction.

The current transformer connected in-line with one of the phases of the spindle motor can indicate directly the spindle current and could be used to determine power consumption as well as partially reflect the system's vibration and surface quality. A combination of the current transformer, accelerometer and encoder pulse rate values, each contributing partially to the classification of surface roughness, may be used to measure surface roughness indirectly with greater accuracy by means of a sensor fusion model. A multi-sensor fusion model is basically a mathematical function developed to extract corroborative and relevant information on a particular manufacturing operation. In this project NF-based multi-sensor fusion models for the indirect measurement of surface roughness and tool wear is generated from experimental data (Chapter 4).

2.6 Characteristics for Intelligent Machining Controller

Over the last 60 years the use of automatic control theory and technology has allowed many industrial processes to operate automatically under certain operating conditions. Most machining processes are stochastic, nonlinear, complex and ill-defined and are open to control by means of intelligent systems. Process automation tasks are performed in hierarchical levels as shown in Figure 2.13 [52]:

- **Process level**: Measurement of the input variables and manipulation of the output variables and require a fast reaction time.
- **Control level**: Feedback and feed forward control where various variables are adjusted according to conditions or reference variables.
- **Supervision**: Indicates undesired or unpredicted process states and to take appropriate actions such as fail-safe, shutdown, or re-triggering of redundancy or reconfiguration schemes.
- **Management**: Performance optimization, coordination of general management in order to meet economic demands or scheduling and dedicated to tasks that do not require fast responses and act

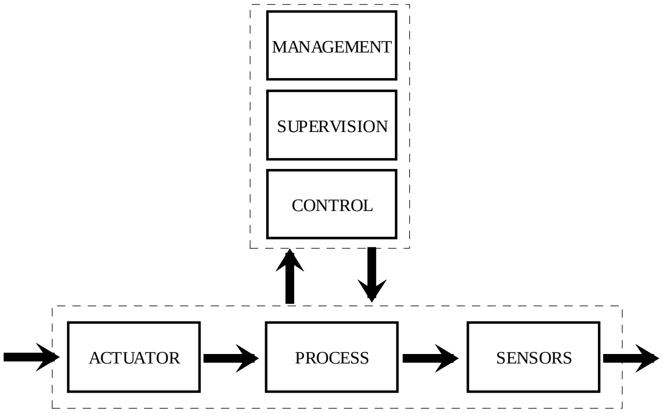


Figure 2.13: Multilevel process automation [1].

Numerical Control (NC) and CNC machine tools have been widely applied in industry. Productivity and production quality has been increased accordingly by means of these facilities [1].

Figure 2.14 shows the general architecture for machine controllers, which perform process and control level functions.

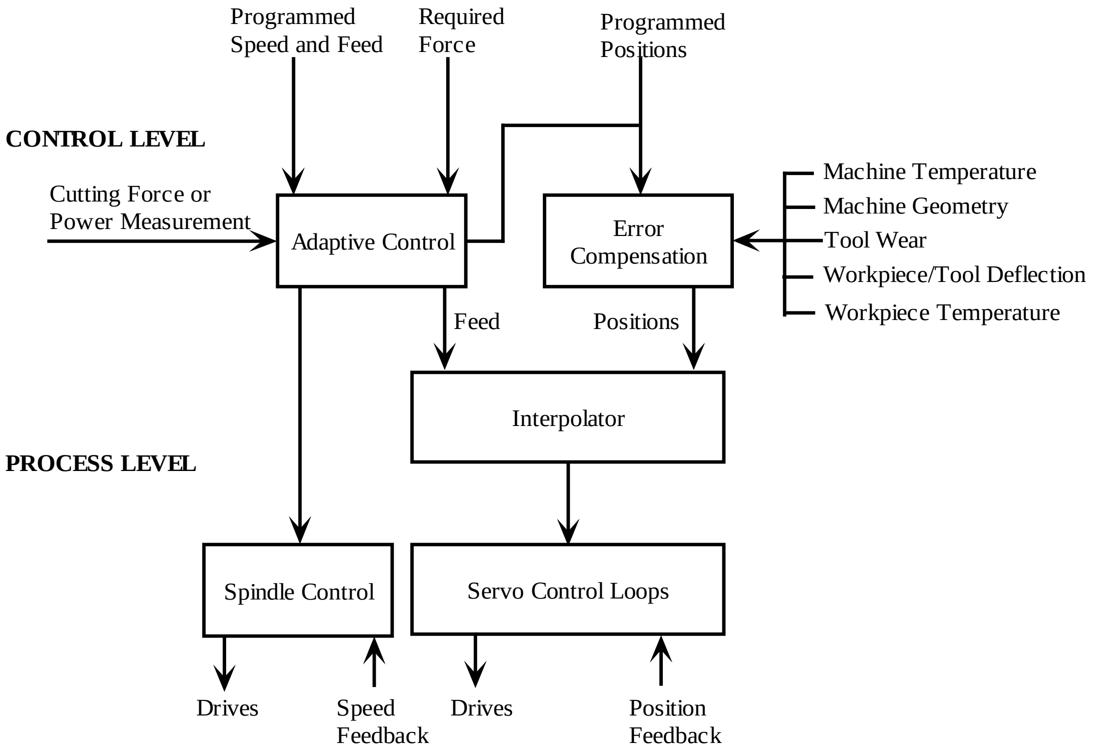


Figure 2.14: Hierarchical levels in CNC controllers [53].

At the process level NC and more recently CNC machine tools have been widely applied in the machining industry to manipulate the processes machining variables, depth of cut, feed and spindle speed. As a logical extension to CNC systems, the control level involves Adaptive Control (AC) of the machining process, which includes the following two major functions:

- Enhanced productivity by applying adaptive control techniques such as Adaptive Control Optimization (ACO) and Adaptive Control Constraints (ACC). The adaptation strategy is used to vary the machining variables in real time as cutting progresses. ACO performs optimization with respect to maximum production rate and/or minimum cost and ACC controls with respect to forces or with respect to vibrations [54, 55, 56, 57].
- Enhanced part precision by applying real-time geometric error compensation techniques such as Geometric Adaptive Compensation (GAC) for imprecise machine geometry, tool

wear etc [58]. The compensation strategy modifies the geometric data supplied by the part program, the depth of cut.

Conventional CNC machines have the following limitations because of their closed architecture [59, 60];

- They cannot efficiently provide real-time monitoring of a machining process by means of sensor feedback.
- The control of the machining process is not achieved adaptively in terms of on-line sensory data.
- The integration of task planning with control activities, and optimization of system performance, are not realized efficiently.

Furthermore, in order to deal with machining complexity an "intelligent machining controller" should have a suitable architecture. Open architecture is a philosophy in design and implementation of machine tool, production processes and control. It creates an open environment for manufacturing and enables manufacturing systems to changes and reconfiguration system hardware and software. An open architecture in the design and implementation of intelligent machine tools needs to embrace the following characteristics [61, 62, 63, 64]:

- Sensor based. The combination of multiple sensors makes it possible to reflect the complexity of the manufacturing process. Sensory data are not only for control, but also for process modeling, real-time simulation and performance monitoring and evaluation.

- Knowledge based. Human expertise, work experience, and testing experiment. Fuzzy logic is powerful in modeling human expertise and experience knowledge, as well as the highly non-linear manufacturing process. Since fuzzy knowledge inference is embedded within the modeling, monitoring and control, system flexibility and intelligence would be much enhanced.

- Integration. System integration is realized from different points of view. The processes of modeling, monitoring and control are integrated. On the other hand, sensory data and knowledge inference are integrated for on-line monitoring and remote decision-making via the Internet.
- Modular. A modular design is achieved in the interface and control of the system. It may be extended to other parts of the system, such as the inference algorithm. The interface access to the Internet is also designed as a module.
- Openness. Systems developed incorporating those features mentioned above would be open to changes in respect of machine setup, machining process, and control algorithm and operation.

2.7 PC-Based Technology for Open System Architecture

The hardware and software selected to develop an open architecture based machine tool controller should [65, 66]:

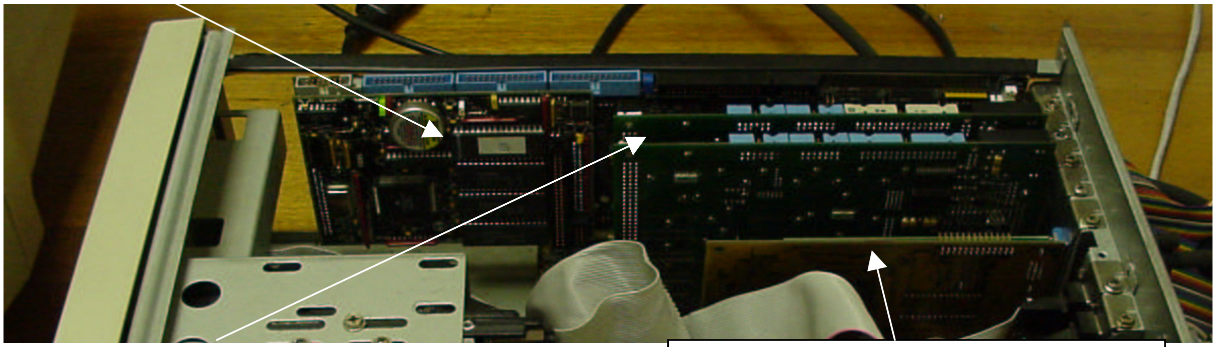
- Make use of standard computing architectures like VME or ISA/PCI bus standard processors like Motorola 68x0, PowerPC, or Intel 1x86/Pentuim-based systems.
- Be based on standard operating systems like Unix or Windows NT. Common operating systems for each level of the factory facilitate communications, programming efforts, and the protection of standardized data structures.
- Be programmable in standard languages like Microsoft Visual Basic and Visual C++ or C/C++ and X-Windows. Object-oriented, high-level languages that are comparable to plans and subsequent to machinery instructions are necessary requirements for the transfer of knowledge from one level of the factory to the next. This principle includes

standardized data structures that must pass unambiguously down through the factory hierarchy; and

- Be open and extendable so as to let the user integrate custom control algorithms. Open-architecture computer platforms are needed at all levels of the factory, with the key emphasis today being on improvement of factory floor machinery such as machine tools, robots, and common manufacturing devices.

Recent technological advances in PC-based DSP and Programmable Multi-Axis Machine Control (PMAC) products, as well as software interfacing Active-X controls and dynamic link libraries (DLL) to facilitate communication between these hardware components, and an object oriented windows based software application, enable the realization of PC-based open system architecture to implement the open architecture machine control as shown in Figure 2.15.

1.1 Multi-Axis Control Interface Card



Two DSP Interface Cards to Sample and Process Sensor Signals.

Ethernet Interface Card
To Open Machining Process for Internet Monitoring.

Figure 2.15: PC-based PMAC, DSP and Ethernet interface cards.

2.7.1 PCI32 a 32-bit Floating Point DSP with PCI bus Interface

Two PCI32 interfaces, shown in Figure 2.16, featuring the high performance Texas Instruments TMS320C32 32-bit floating point DSP capable of up to 60 MIPS were selected to be used in this project to sample analog sensor based systems.

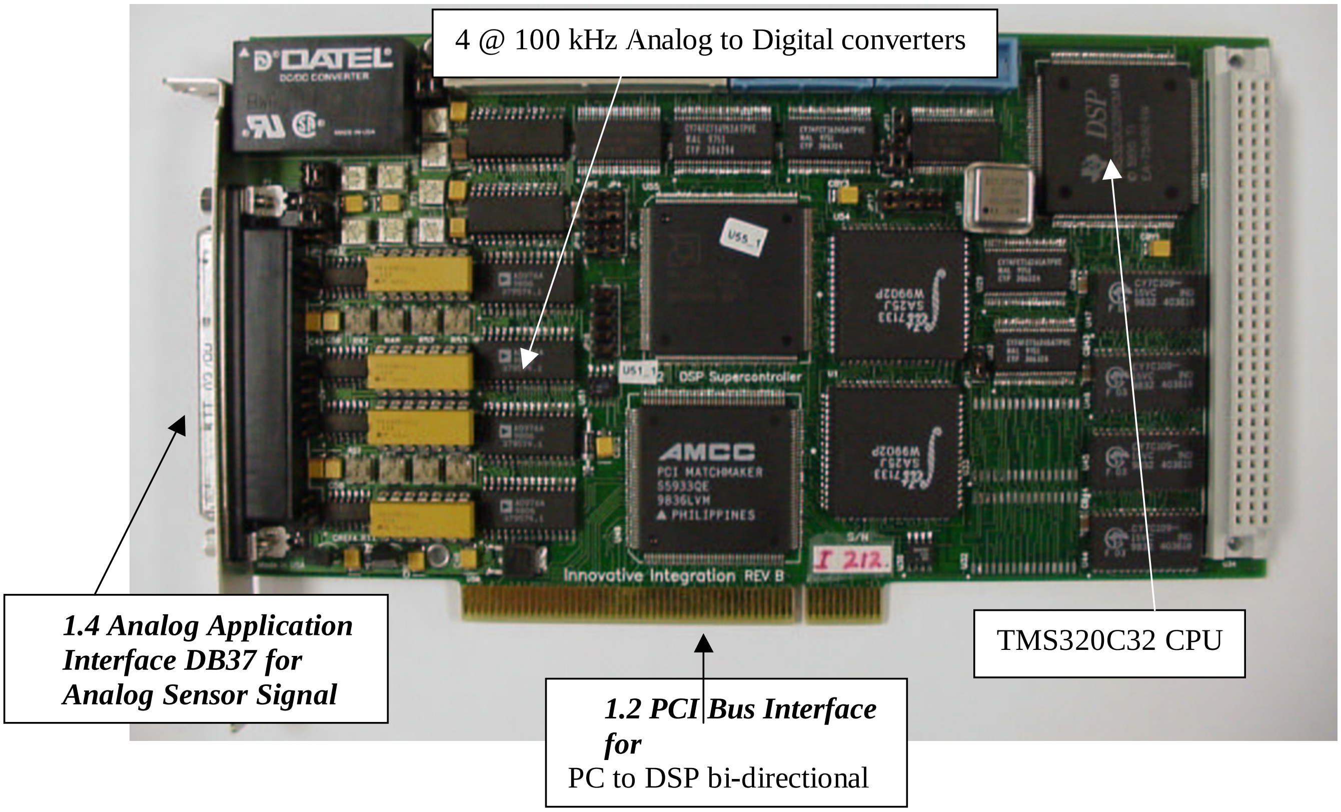

4 @ 100 kHz Analog to Digital converters
TMS320C32 CPU
1.4 Analog Application Interface DB37 for Analog Sensor Signal
1.2 PCI Bus Interface for
PC to DSP bi-directional
DATEL
DSP Supercontroller
AMCC
PCI MATCHMAKER
Innovative Integration REV B

Figure 2.16: PCI32 DSP interface.

The PCI32 plugs into a standard 32-bit PCI bus slot. The PCI bus interface includes dual-ported memory capable of burst transfers at rates to 40 Mbytes/sec on most platforms. This 8 Kbytes dual port RAM provides a superior interface and allows multiple cards to be installed in systems with full driver support under Windows 95 and NT. The PCI32 may be programmed in C or Assembler using tools available in a Software Development Package. Components within the package, installed and used in this project, which fully support development of custom DSP applications include [67]:

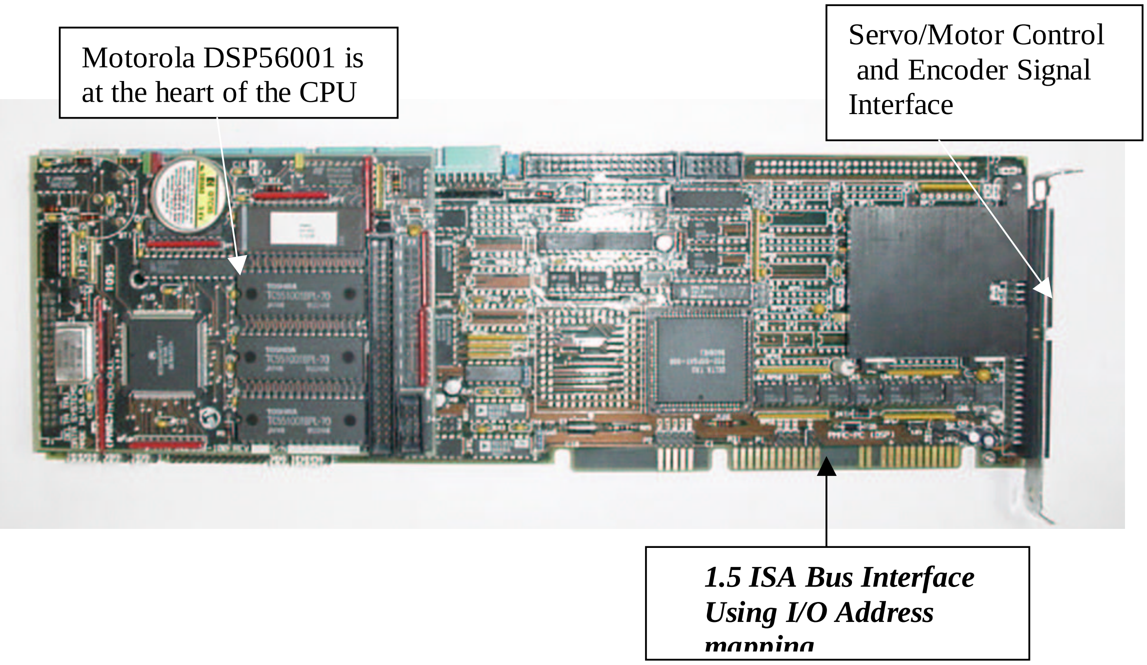
- Texas Instruments Floating Point C Compiler/Assembler toolset

- Codewright an integrated code generation environment.
- DSP Peripheral library that support all on-board peripherals and DSP functions.
- Custom 32-bit Windows 95/NT compatible dynamic link library, which utilizes a custom 32-bit Ring 0/Kernal-mode device driver for host PC software application development.
- Host support applets for automation program download.

The abovementioned software was utilized to develop a standalone PCI32 DSP application that is able to sample and signal; process the analog sensor signals which in turn may be accessed by the PC host application for advanced monitoring and decision making. The implementation aspects, including software code, are discussed in Chapter 3.

2.7.2 PMAC-PC Programmable Multi-Axis Controller with ISA interface

The Delta Tau Data Systems PMAC-PC, shown in Figure 2.17, is a high-performance servo motion controller capable of commanding up to eight axes of motion simultaneously with a high level of sophistication [68].



Motorola DSP56001 is at the heart of the CPU

Servo/Motor Control and Encoder Signal Interface

1.5 ISA Bus Interface Using I/O Address mapping

Figure 2.17: PMAC-PC multi-axis control card.

PMAC is a very flexible controller, suitable for many different types of applications, with different types of amplifiers, motors, encoders and sensors. The card may be configured for a specific application, using both hardware and software features and is therefore ideally suited for performing the multi-axis control function in open system based architecture.

Delta Tau developed PTalkDT [69], a software interface to its 32-bit software driver PComm32. PTalkDT is in the form of an ActiveX Control, a new and upcoming form of library that is very popular with Windows programming. It is designed to provide robust and efficient communication to PMAC from Windows based applications.

Many of the commands given to the PMAC, using the PTTalk ActiveX control, are on-line commands, which are executed immediately by the PMAC. There are three basic classes of on-line commands including: Motor-Specific Commands Coordinate system-specific commands and global commands.

PTTalk Active X control software was installed and extensively used to develop a Windows based software interface that is able to configure the PMAC, send motion control commands and receive coordinate positions from the PMAC. The implementation aspects, including software code, are discussed in Chapter 3.

2.7.3 Object Orientated Programming (OOP), Visual C++ and Visual J++

A programming language must support abstraction, encapsulation, inheritance, polymorphism and modularity before it deserves to be called object oriented [70]. The C++ language is based in, and extends the C programming language, by supporting OOP features [71]. By using these advanced capabilities one may achieve self-configurable software systems [72]. Using object-oriented techniques to develop software, helps to construct systems that closely model reality such as components and functions within an open architecture based machining controller. Each object knows how to handle its job well, and it collaborates with other objects to accomplish a common goal.

One particularly useful use of OOP is to create reusable application frameworks. An application framework is an integrated collection of object-oriented software components that offer all that is needed for a generic application [73]. Microsoft Foundation Classes is an application framework specifically tailored for creating applications for Microsoft Windows operating system. The Microsoft Foundation Class Library is built on top of the Win32 application-programming interface (API). This API is a set of functions exposed by the operating system for use by applications. Through MFC, base classes are exposed that represent common objects in the Windows operating system, such as windows and menus.

MFC does not encapsulate the entire API, just the main structural components and components that are commonly used. Because MFC is written in C++, MFC programmers can easily use the Win32 API to make native calls to the operating system. Figure 2.18 show the relationship between MFC, the Windows Base Operating Services, and the Windows Operating System Extensions.

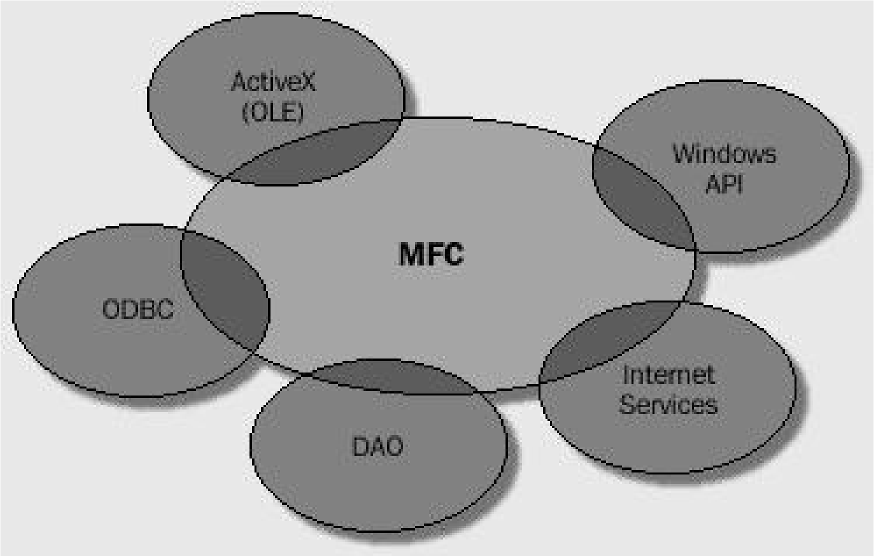


Figure 2.18: Relationship between MFC and Windows API [74].

By providing this simplified interface to the Windows API, MFC offers a number of advantages over using the Win32 API:

- MFC provides a higher-level abstraction of Windows, thus reducing complexity.
- One can learn Windows-based programming much faster than you would by working directly with the Windows API.
- One can quickly develop an application framework from which more complex applications are created.
- One has access to object-oriented techniques that are supported by the C++ language.
- MFC uses the more robust language features of C++, such as stronger type checking, exception handling, and intelligent object construction and destruction.
- MFC supplies additional library support for safe dynamic memory use, type validation, and debugging.

Microsoft Developer Studio, used in this project, is the development environment for Microsoft Visual C++ and MFC. In order to provide flexibility to meet various programming needs, Developer Studio integrates several other development tools such as Microsoft Visual J++. Besides providing an integrated, flexible environment, Developer Studio offers class navigation tools that are designed to simplify object-oriented programming for Windows.

Java applets are designed to work easily within the World Wide Web (WWW) of computers through commonly available, user-friendly software called browsers which are Java-enabled. Although it is a rather recent addition to the host of high-level computer languages available to the programmer it incorporates the latest in OOP features and capabilities [75]. The designers focused on security, network-awareness, multitasking and hardware abstraction, which have brought Java near-instant acclaim in the programming field. In this project a Java Applet will be developed and embedded within a web page. The applet will be enabled remotely through a web browser from whereby a connection-oriented service will be performed with the Visual C++ application. Implementation aspect of the remote monitoring for machining is be explained in Chapter 3.

2.8 System Framework for Intelligent Machining

Figure 2.19 shows a proposed system framework for the implementation of a sensor-integrated monitoring, intelligent diagnosis and control of the machining process. In adaptive control system cutting force, torque and/or power sensors are used to provide the feedback information, with the control of the cutting force being the most popular. It seems clear from the research conducted that, at the lower level, adaptive control with self-regulating ability applied to one dependent variable Fz is achievable by changing f1x and Vc. With a multi-input / multi-output control strategy one may want to control more than one independent variable, say Pc and Ra. The independent variables, Vc, f1x and dy are set in an optimal manner in order to maintain Pc and Ra set points [76]. A drawback of these control strategies is that it is based on the modeling of the original process and may only be applied within a limited range from the set point. The ideal is, of course, to be able to maintain all the dependent variables at variable set points. However, this has proven to be unattainable since it implies a tremendously complicated multi-input and multi-output control system. A simplified arrangement has been agreed upon [64]:

- Adaptive control of feed with respect to cutting force;
- Adaptive control of feed with respect to maximum productivity; and
- Advanced process monitoring and diagnosis

This is indeed what the framework proposes. Signals are sampled and processed using the DSP. Machining process parameters Ra and Vb that the sensor system is not able to measure directly, are determined indirectly using multi-sensor fusion modeling.

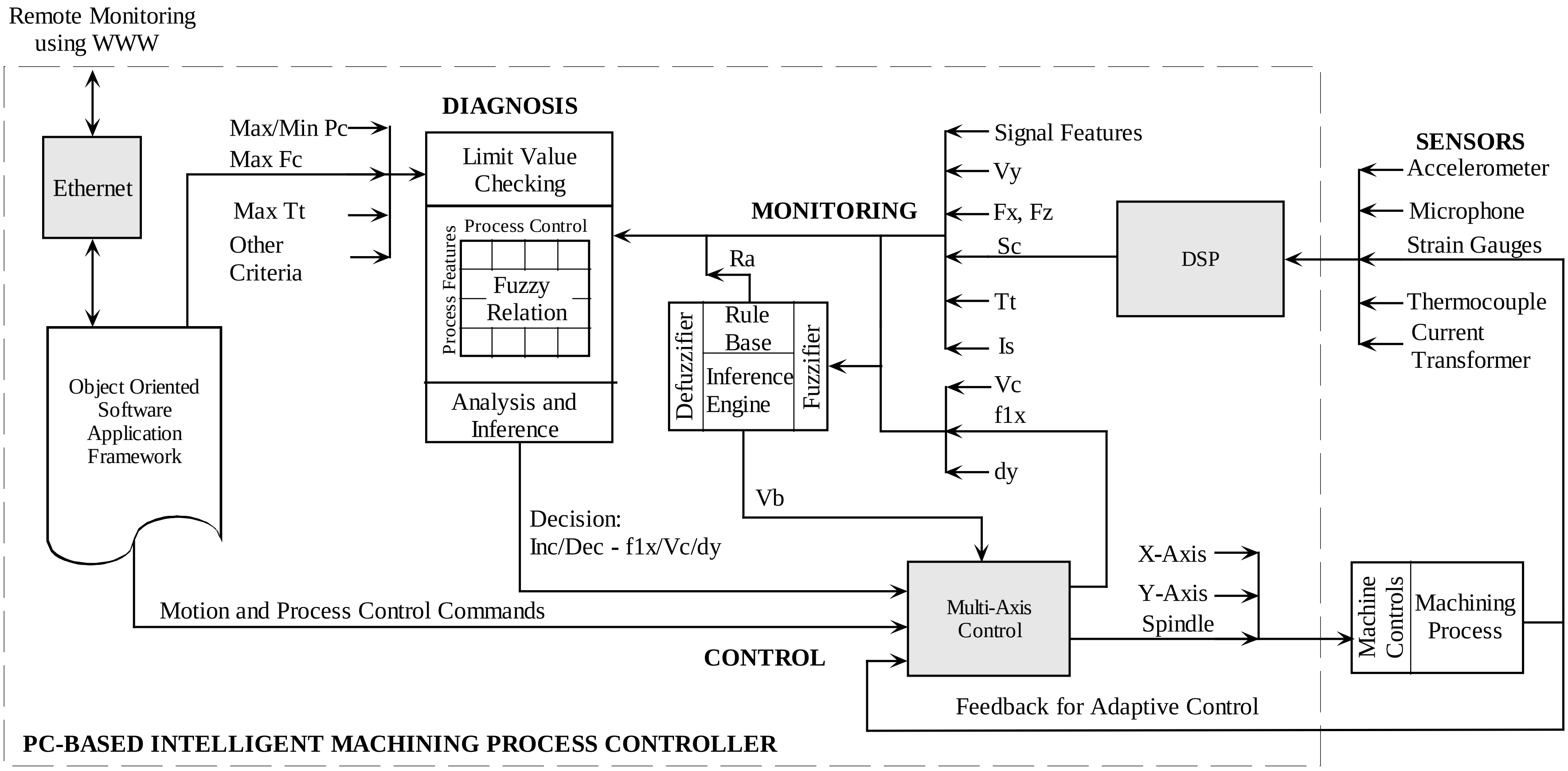


Figure 2.19: Framework for intelligent machining.

The object oriented software application interface serves as a graphical user interface and performs overall module integration and coordination. It sends motion and process control commands to the multi-axis controller and provides process parameter constraints to the diagnostic system. Once a process parameter exceeds its limit, it is up to the diagnostic systems to ensure that the specific parameter returns to its stable state. Using a fuzzy relation the diagnostic system will decide intelligently which one of the three process control parameters to change in order to achieve an overall stable machining process.

2.9 Conclusion

The monitoring of tool wear and surface roughness by means of intelligent systems will enhance automated machining. Neuro-Fuzzy modeling may be used as a basis for developing fuzzy logic models for the indirect measurement of tool wear and surface roughness. Fuzzy logic models, based on experimental data, for this purpose is analyzed and explained in Chapter 4.

The primary difference between automated machining and intelligent machining is that an intelligent system applied in the latter is capable of making decisions based on significant information from the machining process. A fuzzy relation that indicates the strength of connection between process features and process control action is used as part of a diagnostic system to decide intelligently which decision to make when a machining process parameter is exceeded. Chapter 5 includes empirical machining process input/output relationships, obtained from regression analysis of experimental data, for modeling and simulation in order to test the intelligent diagnostic system.

A framework for sensor-integrated monitoring, diagnosis and control for intelligent machining process control is proposed. Chapter 3 describes the experimental set-up including all hardware and software components to implement the proposed system framework on a PC-based system.

Chapter 3

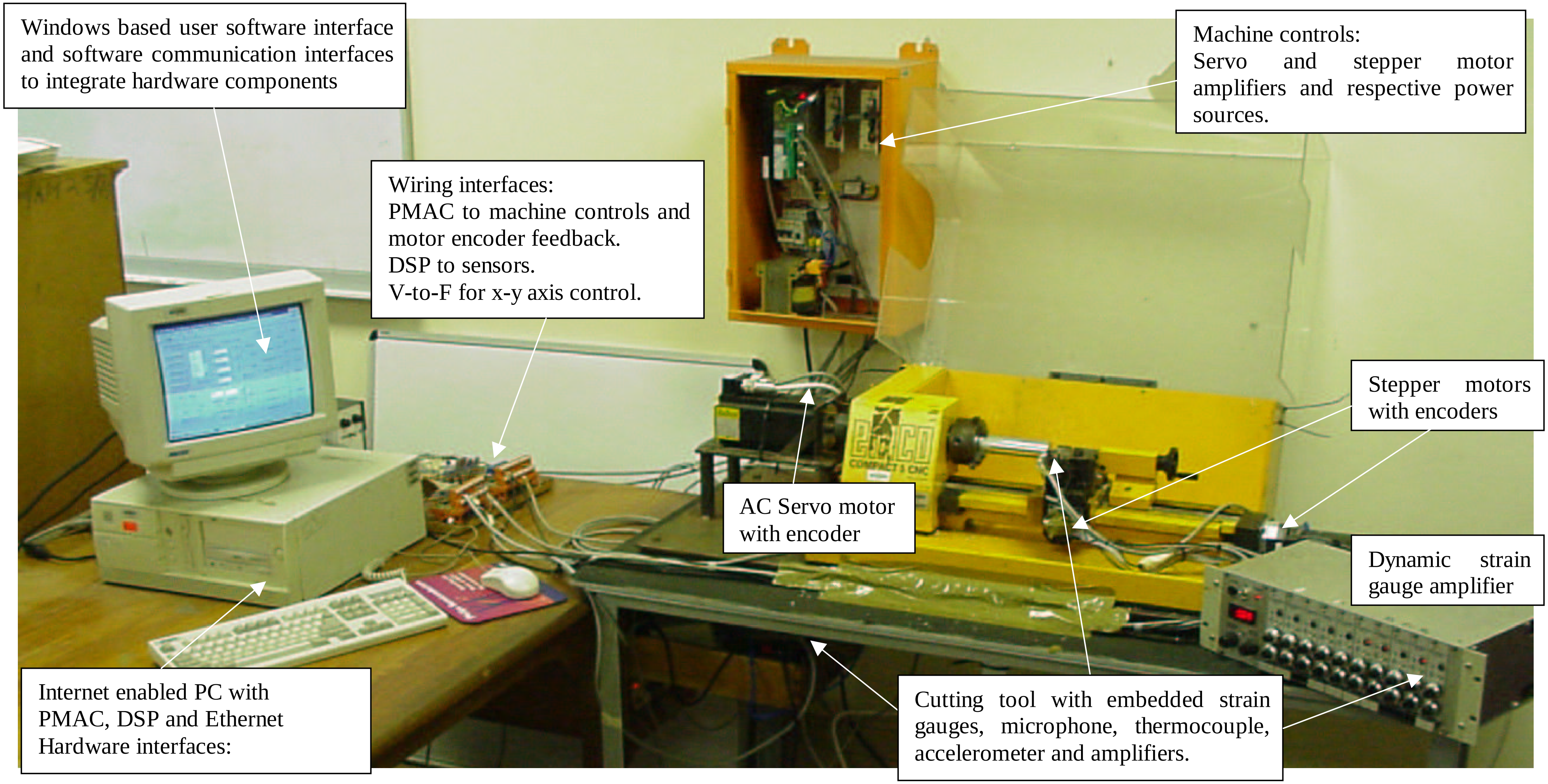
Experimental setup:

Machine Controls, Sensors and Software Components

Intelligent machining systems with in-process quality assurance need to detect and react quickly on measured defects, and then should have the capability to adapt to maintain desired tolerances. The purpose of the experimental set-up is to implement and integrate the sensors, pc-based hardware, software components and machine controls indicated in the proposed framework for intelligent machining, described in Section 2.8. Each software function developed to support hardware operation and overall integration constituted a module with appropriate interfaces so that the reconfiguration of the system may be realized in terms of the modular structure.

3.1 Experimental setup

Figure 3.1 shows the completed experimental set-up consisting of an EMCO Compact 5 training lathe under the control of a 1.5 kW Baldor ac servo motor [76] for spindle rotation and two Powermax hybrid stepper motors [77] for driving the x-y coordinate system. The machine controls include servo and stepper motor amplifiers and respective power sources. The encoder feedback for the servo and stepper motors is returned to the PMAC. The drives are thereby directly controlled from the PMAC enabling instantaneous reaction from software-controlled commands. This arrangement was selected so as to minimize the amount of controls hardware. If a new drive/motor combination is selected the system would only be required to undergo a software reconfiguration.



Windows based user software interface and software communication interfaces to integrate hardware components

Wiring interfaces:
PMAC to machine controls and motor encoder feedback.
DSP to sensors.
V-to-F for x-y axis control.

Machine controls:
Servo and stepper motor amplifiers and respective power sources.

Stepper motors with encoders

AC Servo motor with encoder

Dynamic strain gauge amplifier

Internet enabled PC with PMAC, DSP and Ethernet Hardware interfaces:

Cutting tool with embedded strain gauges, microphone, thermocouple, accelerometer and amplifiers.

11.1.1.1.1 Figure 3.1: Experimental setup: machining process, PC-based control and sensor measurement.

3.1.1 Machine Controls

Figure 3.2 show the layout of the control panel for the servo and stepper motor drives.

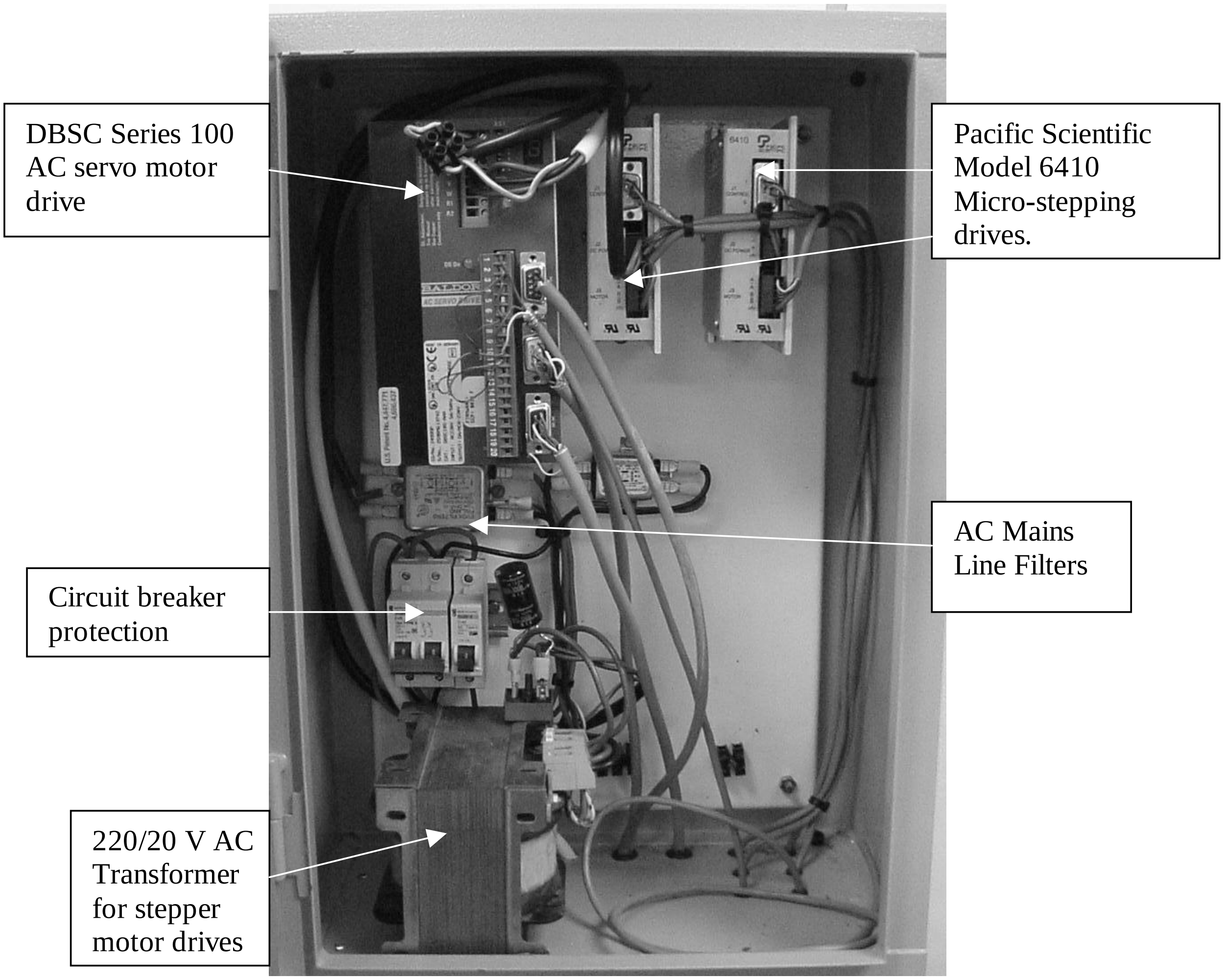

DBSC Series 100 AC servo motor drive
Pacific Scientific Model 6410 Micro-stepping drives.
AC Mains Line Filters
Circuit breaker protection
220/20 V AC Transformer for stepper motor drives

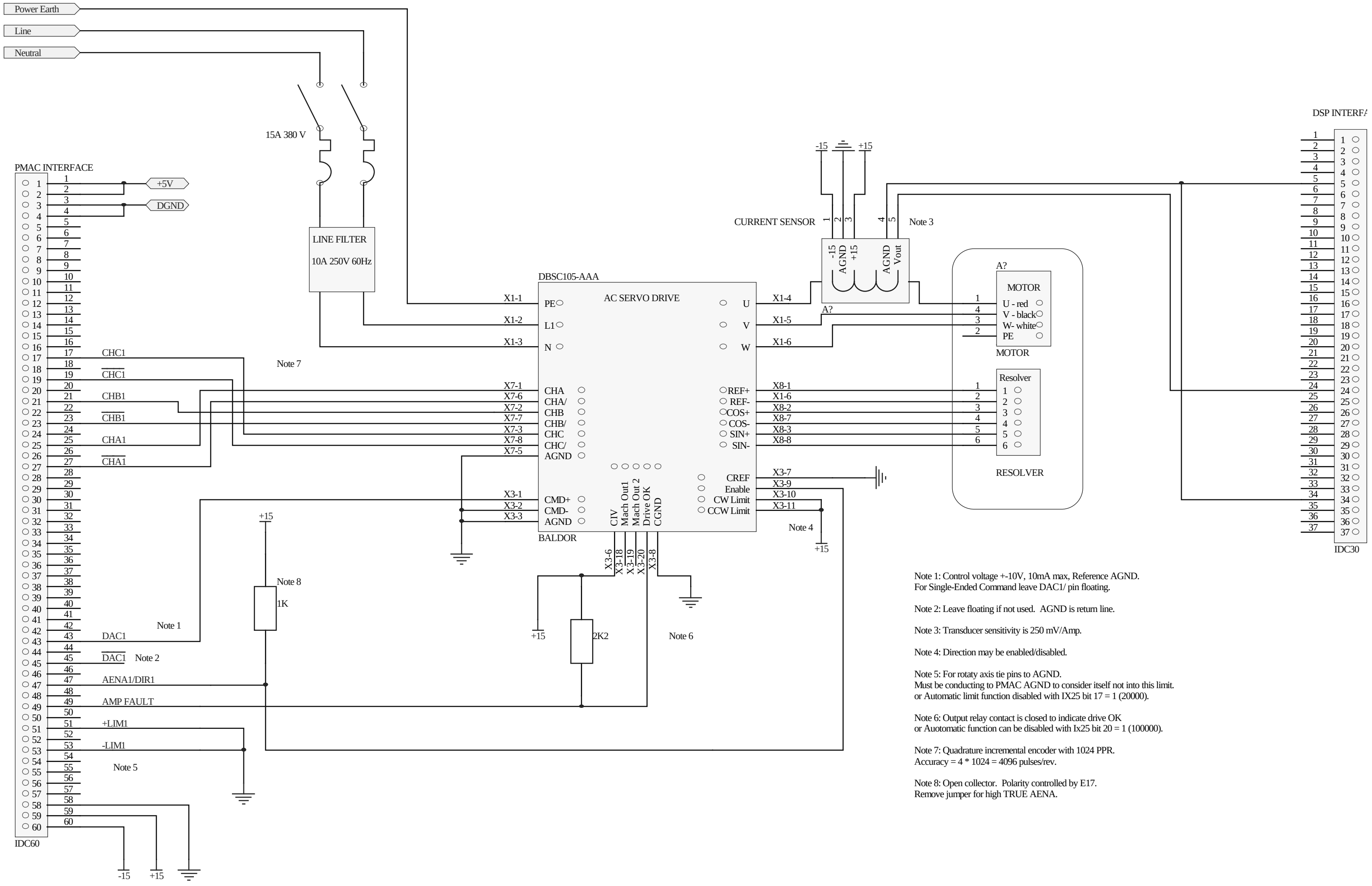
Figure 3.2: Panel layout for the spindle and stepper motor drives.

Spindle Motor Control Circuit Description

Figure 3.3 shows the wiring diagram for the spindle motor control circuit. Power to the servo drive is from 230V ac mains. A line filter is incorporated to eliminate as much of the electrical noise on the power line as possible. A circuit breaker is also incorporated into the supply line to the drive for over current protection. The live wire is connected to terminal X1:2, the neutral connection is made to terminal X1:3 and terminal X1:1 is tied to Earth.

The control signal from the PMAC is a single-ended +-10V analogue magnitude voltage generated on pin 43 (DAC1). Pin 45 (DAC1/) is left floating because the return path for the control signal is analogue ground. Pin 47 (AENA1/DIR1) is used as an amplifier enable signal. This pin is an open collector output and requires the use of a 1 kilo-ohm pull-up resistor. It is configured to give an active LOW output. Pin 49 (AMP FAULT) takes an input that tells the PMAC whether the amplifier is operating correctly. This input is supplied by the amplifier itself and is based upon the result of the amplifier self-test routines. Pins 17, 19, 21, 23, 25 and 27 (CHC1, CHC1/, CHB1, CHB1/, CHA1 and CHA1/ respectively) are the PMAC encoder inputs for quadrature decoding.

Terminals X3:1 (CMD+) and X3:2 (CMD-) are the velocity control inputs of the servo controller. CMD- is tied to the PMAC's analogue ground allowing single-ended control by the PMAC. Pin X3:6 (CIV) and pin X3:8 (CGND) are the inputs that define the voltage level at which the servo drive communicates with its controller (in this case a PMAC-PC). All outputs to the PMAC (e.g. encoder outputs and Drive OK output) use the voltage present on pin X3:6 and are relative to pin X3:8. Pin X3:9 (Enable) is connected to the PMAC amplifier enable output allowing the PMAC to "disconnect" the amplifier when control of the servomotor is not required. Pins X3:10 (CW Limit) and X3:11 (CCW Limit) are tied to PMAC's +15V supply to enable motion in both the clockwise and counterclockwise directions.



Note 1: Control voltage +-10V, 10mA max, Reference AGND.
For Single-Ended Command leave DAC1/ pin floating.

Note 2: Leave floating if not used. AGND is return line.

Note 3: Transducer sensitivity is 250 mV/Amp.

Note 4: Direction may be enabled/disabled.

Note 5: For rotaty axis tie pins to AGND.
Must be conducting to PMAC AGND to consider itself not into this limit.
or Automatic limit function disabled with IX25 bit 17 = 1 (20000).

Note 6: Output relay contact is closed to indicate drive OK
or Auotomatic function can be disabled with Ix25 bit 20 = 1 (100000).

Note 7: Quadrature incremental encoder with 1024 PPR.
Accuracy = 4 * 1024 = 4096 pulses/rev.

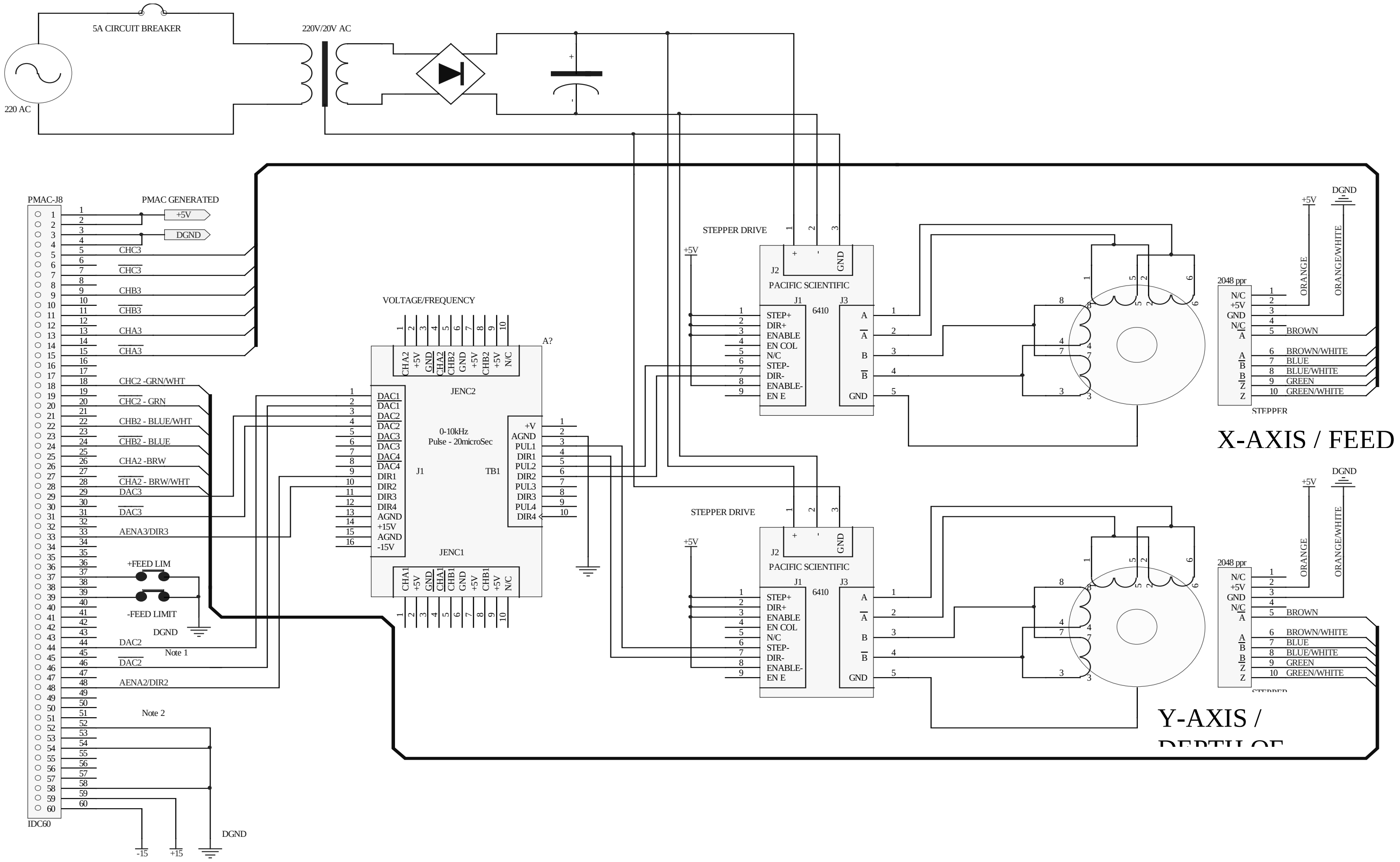
Note 8: Open collector. Polarity controlled by E17.
Remove jumper for high TRUE AENA.

Figure 3.3: Wiring diagram for the spindle motor control circuit and current transducer.

X-Y Axis Control Circuit Description

Figure 3.4 shows the connection-wiring diagram for the x-y axis control. The velocity control voltage is developed between pin 44 (DAC2) and pin 46 (DAC2/) of connector PMAC-J8. This control signal is then fed to connector pins J1:1 (DAC2) and J1:2 (DAC2/) of the voltage-to-frequency converter (V-to-F). The V-to-F then converts this unsigned magnitude voltage signal into a pulse width modulated signal that is compatible with the stepper motor drive input. The frequency of this signal is proportional to the magnitude of the velocity control voltage.

The amplifier enable bit (AENA2/DIR2) on pin PMAC-J8: 48 is used as a digital direction bit and supplies the signal to the V-to-F direction input (DIR1) on connector J1:9. The outputs for the V-to-F are a digital direction bit and a pulse width modulated pulse output, both at TTL levels. For this channel the output on connector TB1: 3 (PUL1) carries the pulse output and the direction signal is on connector TB1: 4 (DIR1). The frequency of the output pulses is directly proportional to the magnitude of the control voltage present between the DAC1 (J1:1) and DAC1/ (J1:2) input pins.
The pulse output from PUL1 is the input to the STEP-input (connector J1: 6) of a Pacific Scientific stepper motor drive. The STEP+ (J1: 6) input is tied to +5V. When logic 0 is present on the STEP-input, the opto-isolator goes ON. Every transition of the opto-isolator from OFF to ON results in the execution of a single step of the stepper motor. DIR1 output of V-to-F is the input to DIR- input (connector J1: 7) of the Pacific Scientific stepper drive. The outputs of the stepper drive are a pair of differential signals that provide excitation voltage to the windings of a stepper motor. Feedback from the stepper motors is by means of an incremental encoder that is attached to the stepper motor shaft at its rear end. This device outputs 4 pulse signals in quadrature mode. A resolution of 2048 pulses per revolution is realized in this application. Output from this device is fed directly to the PMAC quadrature encoder inputs. The PMAC decodes these signals to determine position and velocity of the stepper motor.



11.1.2 Figure 3.4: Wiring diagram for the x-y axis control circuit.

3.1.2 Sensory System

One of the basic requirements for intelligent machining is the need for sensors and measurement systems to obtain signal features that characterize the process. In this project the sensor type and positioning thereof is focused on the measurement of signals that are "close" to the machined product. These signals may then be used as input to an intelligent diagnosis system to ensure the in-process reliability and quality of the machined product (Chapter 5). Figure 3.5 shows the cutting tool (insert and holder) with embedded sensors for: cutting force and feed force, cutting sound, cutting tool temperature and cutting tool-workpiece vibration measurement.

3.1.2.1 Tool-Workpiece Vibration

Vibration may be measured using either a dynamometer, accelerometer or displacement sensor. Lin and Hu [78] found that, of the three, accelerometer performed better. The accelerometer may be placed on the tailstock or on the tool as shown in Figure 3.5. A model 3140 instrumentation grade fully signal conditioned accelerometer is used to measure the tool-workpiece interaction [79]. The accelerometer has a range of 0 to 10g, a build-in amplifier with an output sensitivity of 200.7 mV/g and a frequency response of 0,7Hz to1200 Hz.

3.1.2.2 Cutting Tool Temperature

A J-type thermocouple wire with a braided tip inserted close to the tip of the cutting insert is used to measure temperature close to the cutting zone. The thermocouple wire is fed to a type 1100 signal converter used to linearize and amplify the temperature signal [80]. The amplifier system's sensitivity is 33 mV/^{o}C with a range of 0-300 ^{o}C.

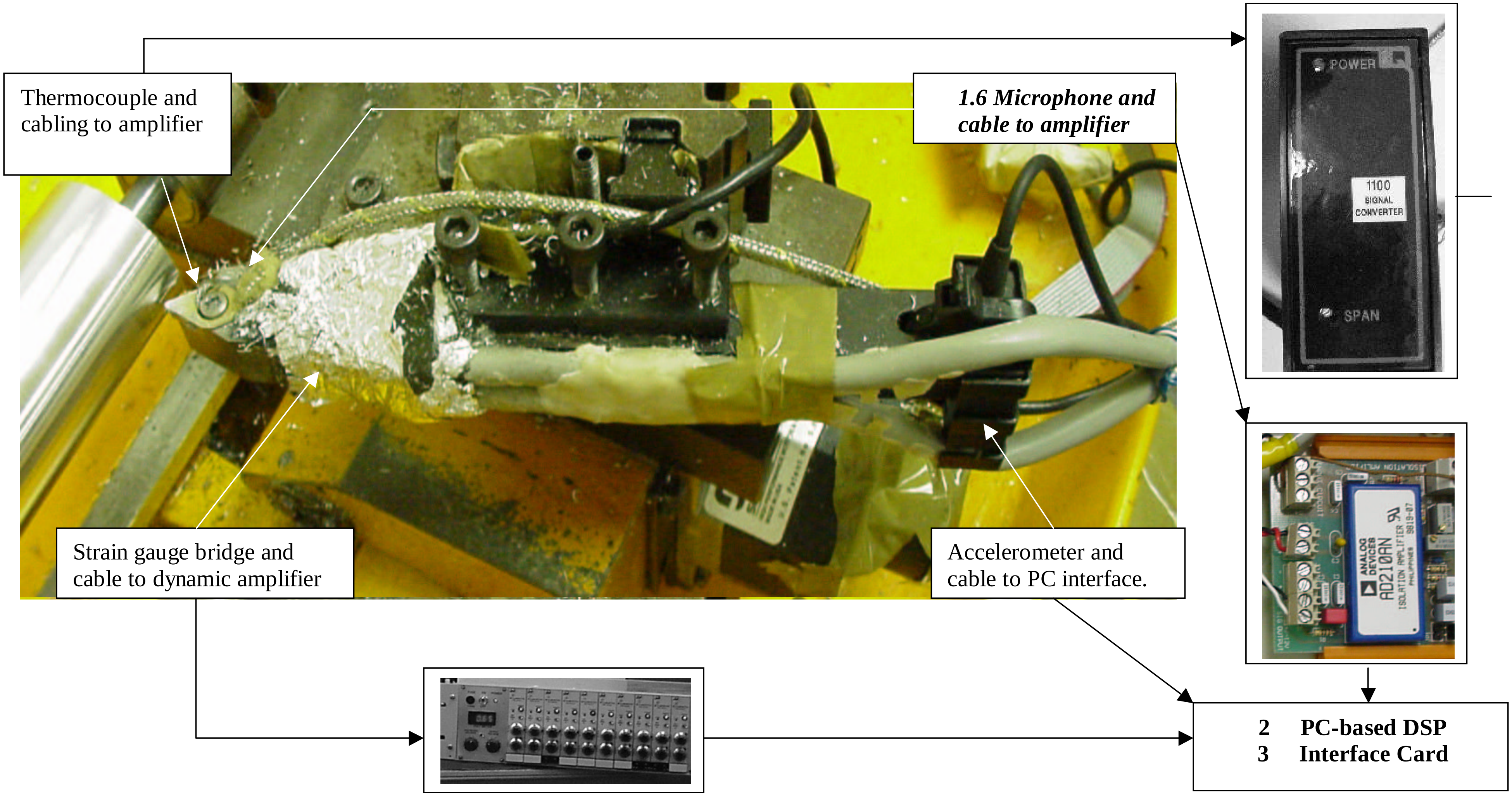
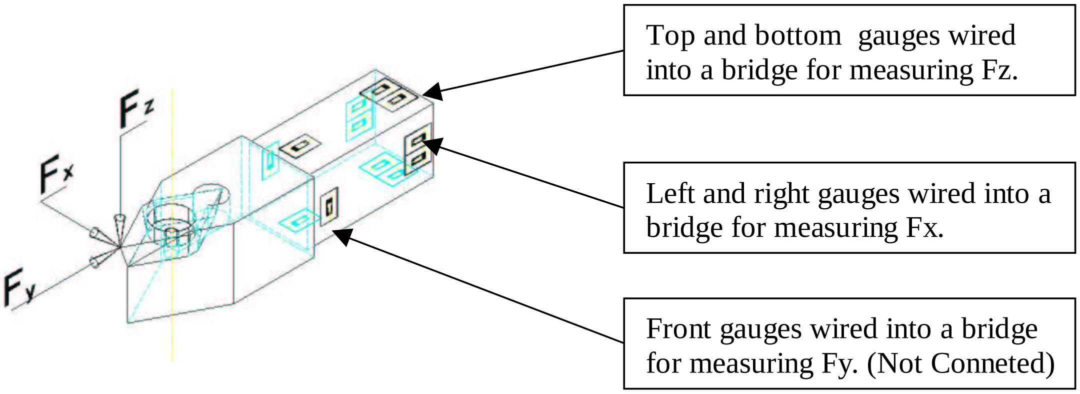

Thermocouple and cabling to amplifier
1.6 Microphone and cable to amplifier
POWER
1100 SIGNAL CONVERTER
SPAN
ANALOG DEVICES
AD210AN
ISOLATION AMPLIFIER
Strain gauge bridge and cable to dynamic amplifier
Accelerometer and cable to PC interface.
2 PC-based DSP
3 Interface Card

Figure 3.5: Cutting tool, embedded microphone, thermocouple, strain gauges, accelerometer sensors and amplifiers.

3.1.2.3 Orthogonal Forces

The objective of strain gauging a cutting tool is to measure the orthogonal cutting forces [81] acting on the tool and workpiece whilst machining. To be able to measure the cutting force Fz and feed force Fx, eight 120-ohm strain gauges [82, 83] are bonded [84] to the shank of the cutting tool as shown in Figure 3.6. The front set of gauges may be used to measure the radial force Fy. However, it was not used in this project. The two sets (top-bottom, left-right) of four gauges are wired into a bridge configuration [85, 86] and connected to a dynamic amplifier with a 3 kHz frequency response range.

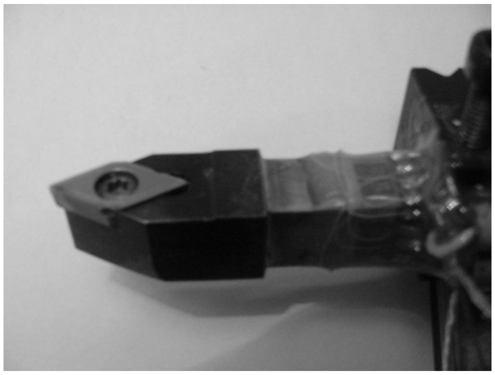


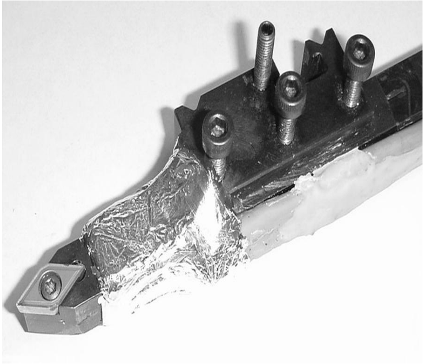
Top and bottom gauges wired into a bridge for measuring Fz.

Left and right gauges wired into a bridge for measuring Fx.

Front gauges wired into a bridge for measuring Fy. (Not Conneted)

(a) Strain gauge positions to measure Fx, Fy and Fz.



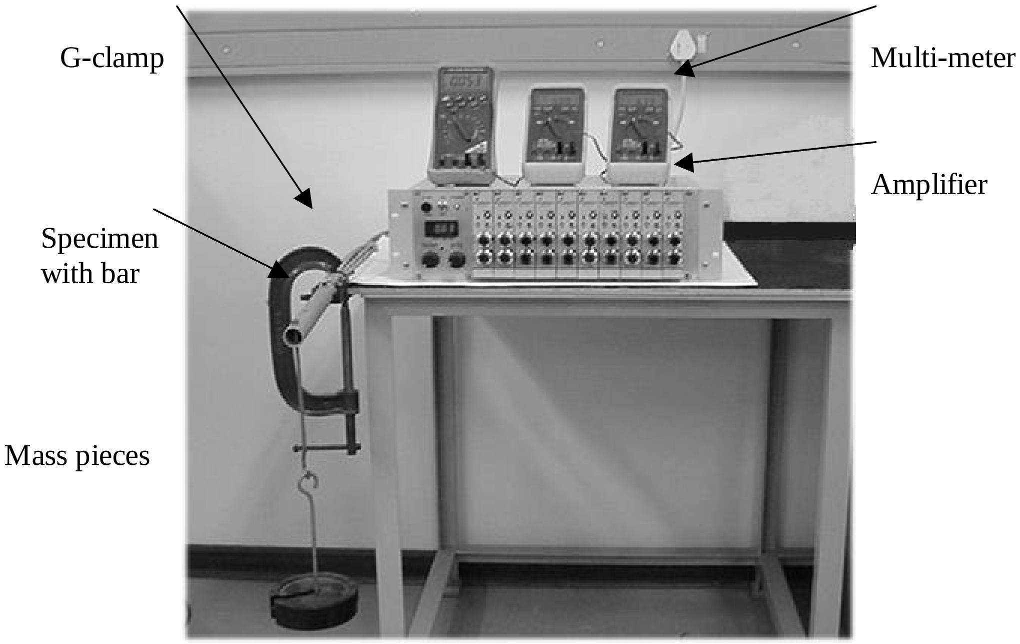


(b) Strain gauges bonded to shank.

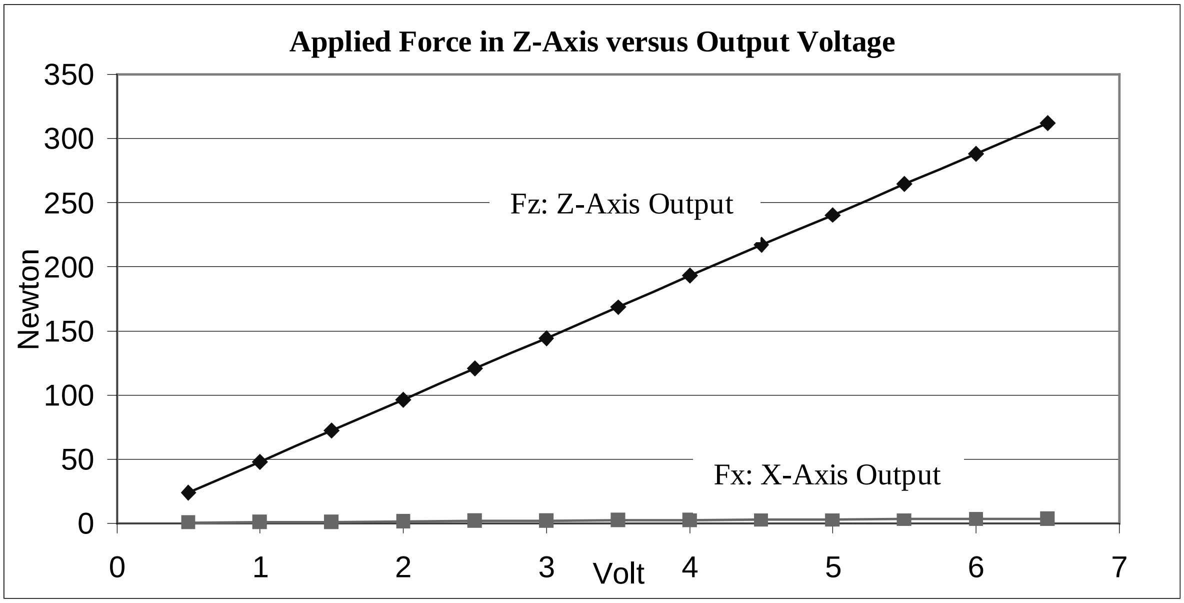
(c) Stain Gauges covered with foil.

Figure 3.6: Cutting tool with bonded strain gauges for orthogonal force measurement.

Figure 3.7 (a) shows the procedure followed in calibrating the equipment with the purpose of determining the ratio a relationship between the output voltages from the dynamic amplifier, and the forces applied at the tip of the cutting tool (sensitivity). A beam was attached to the tool holder and weights suspended from the beam. This was done to overcome the problem of attempting to attach large weights to the tip of the tool. Figure 3.7 (b,c) shows the resulting calibration curves.

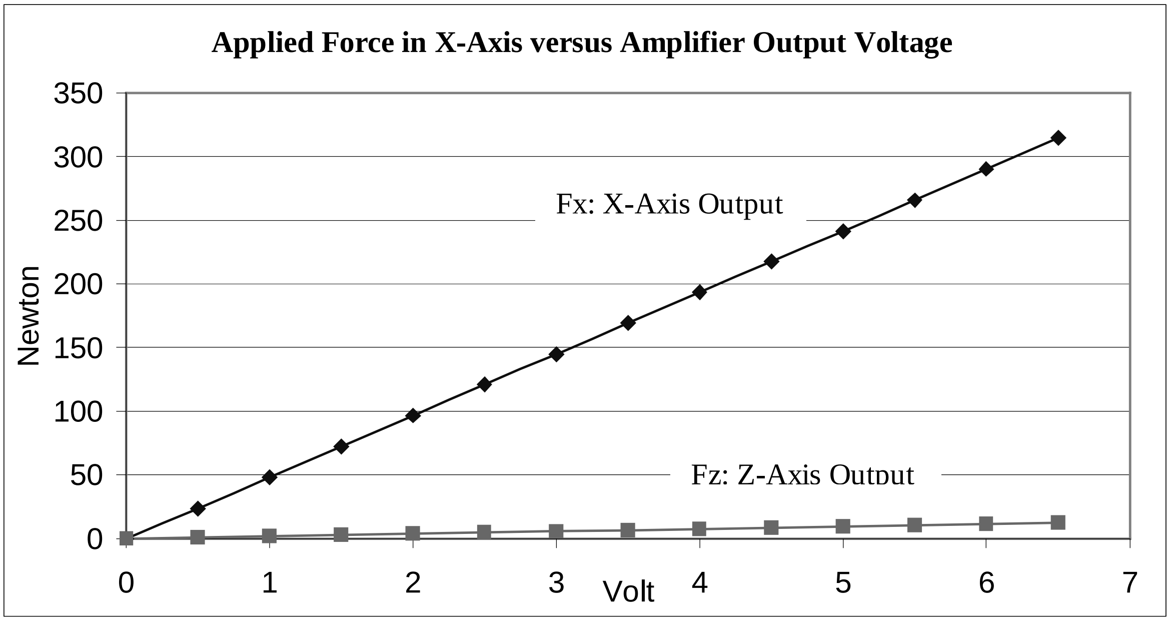


(a) Orthogonal force calibration procedure.



(b) Fz: Applied force at tool tip in the z-axis versus amplifier output voltage

(Sensitivity – 23.67 mV/N).



(c) Fz: Applied force at tool tip in the x-axis versus amplifier output voltage

(Sensitivity – 23.62 mV/N).

Figure 3.7: Calibration procedure and sensitivity for orthogonal cutting force measurement.

Whilst loading the x and z-axis respectively there is an output recorder on the unloaded axis. This output is less than 5% of the value recorded at the loaded axis and may be the result of a small misalignment in the strain gauges position on application.

3.1.2.4 Cutting Sound

Sound may be defined as any pressure variation that the human ear can detect. The number of pressure variations per second is called the frequency of sound and is measured in Hertz. The frequency of sound produces its distinctive tone. These pressure variations travel through any elastic medium (such as air) from the source to the listener's ears. For acoustic and sound measurement purposes, the speed is expressed as 344 ms^{-1} at room temperature, and from this the wavelength may be calculated as speed of sound / frequency. Most industrial noise consists of a wide mixture of frequencies known as broadband noise.

A second main quantity used to describe a sound is the size or amplitude of the pressure fluctuations. The decibel is not an absolute unit of measurement. It is a ratio between a measured quantity and an agreed reference level. The dB scale is logarithmic and uses the hearing threshold of 20 μPa as the reference level [87].

A microphone converts the sound signal into an equivalent electrical signal. The most suitable type of microphone for sound level meters is the condenser microphone, which combines precision with stability and reliability. A LSM900 condenser microphone, with a 20-20kHz frequency response, is small in size and free of "bass boosting proximity" effect found when using most microphones close to a sound source [88]. This makes the microphone ideal for measuring cutting sound close to the cutting zone. The electrical signal produced by the microphones is quite small, and a preamplifier amplifies it before connected to the DSP. Figure 3.8 shows the microphone connected [89] to AD210 [90], a general-purpose amplifier, whose output is connected to one of the channels of the PC-based DSP interface card.

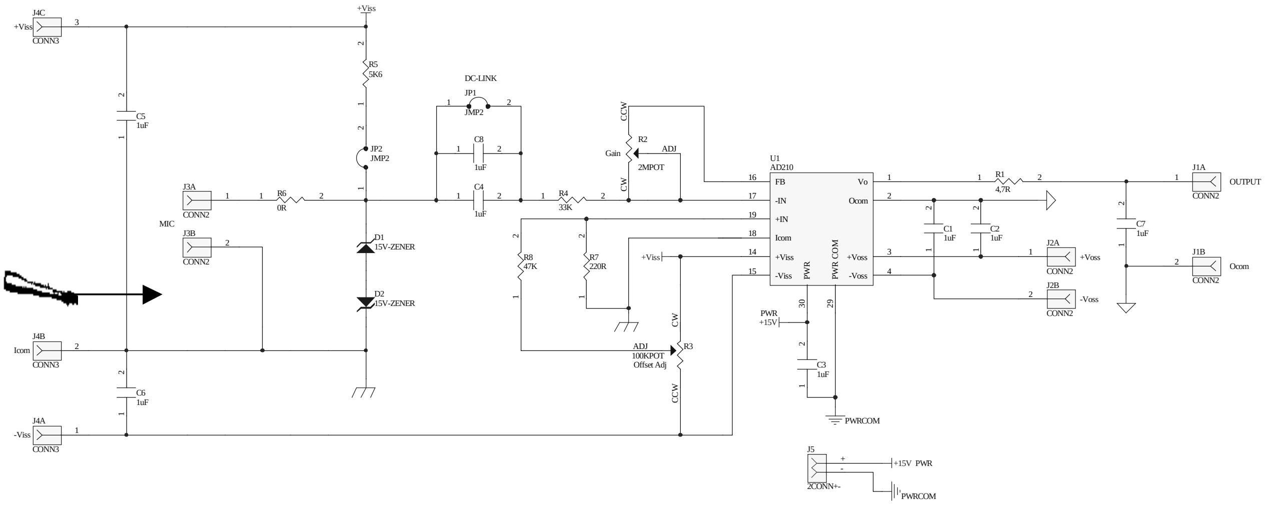


Figure 3.8: Microphone amplifier circuit diagram.

Capacitors C3 (1μF), C1, C2, C5 anC6 (100nF) are connected to filter and eliminate any unwanted high frequency noise ripples. Zener diodes D1 and D2 provide protection against high voltage and incorrect polarity by limiting the input AC and DC voltage amplitudes. Resistor R6 provides current limiting. When the input signal is of DC type, jumper JP1 must be connected to provide a DC-link. An unconnected jumper links JP1 for an AC input signal, which now passes through capacitors C4 and C8 connected in parallel. The parallel-connected capacitors, together with resistor R4, determine the circuit's low frequency response limit as:

$$f_{LOW} = \frac{1}{2\pi(C4+C8)R4} \quad 3.1$$

The signal is taken to the inverting input of the AD210 through resistor R4 with the non-inverting input being connected via R7. A feedback path is provided from the output, via resistor R2. This gives a maximum voltage gain for the amplifier as:

$$VoltageGain = \frac{R2}{R4} \quad 3.2$$

The output signal from the AD210 passes through a low pass filter R1 and C7, which filter any high frequencies. The cut-off frequency may be determined by:

$$f_{HIGH} = \frac{1}{2\pi R1C7} Hz \tag{3.3}$$

In order to reduce background noise measurement the amplifier’s gain was altered using R2 (392k□) until the output was close to zero. Using this value for R2 and equation 3.2 the final gain is calculated at 70. The LSM900 microphone has a sensitivity of 0.02238 mW/Pascal, which results into 0.25 V/Pascal as it is feeding into a 2800□ (R5//R6) impedance. With a gain of 70 the resulting sensitivity for sound measurement is 17.5 V/Pascal. Although the microphone can measure sound of up to 120dB_{SPL} (20 Pascal), which is equivalent to a jet taking off, sound from the cutting process used in this project does not exceed 0.05 of a Pascal (about 70 dB_{SPL}).

3.1.2.5 Spindle Current

The on-line measurement of spindle current is important as it may be related to cutting force as an alternate measurement method and used to manage the supply of torque and power available for cutting. Advantages of current sensors include: low in cost, easy to install, robust and have a fast response time. Furthermore, due to lower maintenance and higher performance, modern machining centers make use of AC or DC Servo Brushless motors. It was therefore decided to make use of a 3kW AC Brushless servomotor and connect it directly to the spindle.

The spindle current is measured using an F.W.Bell current transformer Model IHA-100 [91]. It is able to measure current in a range of 0 – 100 amps with a sensitivity of 50 mV/ampere and a +5/-5 V maximum output. However, with N turns, the current range is reduced by a factor of N divided by the full-scale current. For the model IHA-100, 10 turns through the aperture will change the current range from 0 – 100 ampere to 0 – 10 ampere, thereby increasing sensitivity to 500 mV/ampere. The current sensor is connected in one of the phases of the servomotor [92], as shown in Figure 3.9, so as to measure load changes instantaneously.

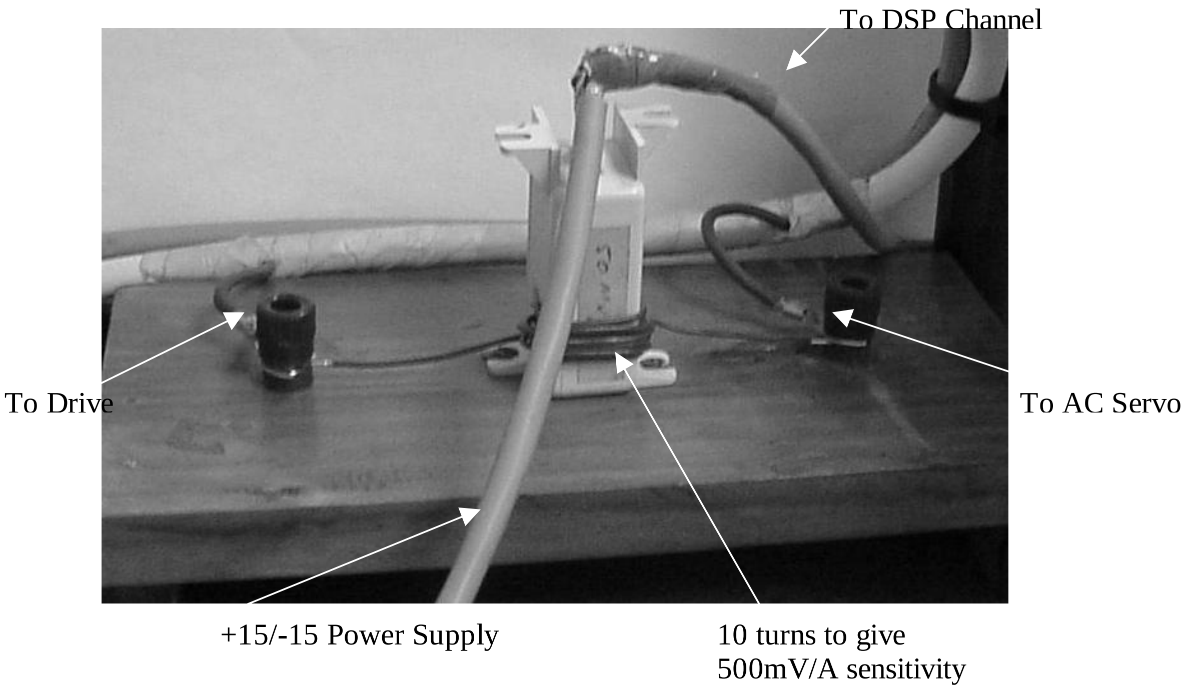


Figure 3.9: IHA-100 current sensor connected into one of the phases of the ac servomotor.

Figure 3.3 shows the wiring diagram for the spindle motor control circuit and indicates how the current transducer is connected in one of the motor's phases and interfaced to the DSP interface card. The –15 V supply goes to pin 1 of the current sensor, +15 V supply to pin 3 and analogue ground to pin 2. The use of a dual rail supply allows the sensor output to swing either negative or positive to indicate negative or positive current flow. Pin4 (AGND) of the current transducer is fed to pins 5 (INPUT B-) and 34 (AGND) of the DSP interface. This means that the signal received by the DSP card will be relative to the card's own analogue ground. Output of the transducer on pin5 (VOut) is fed to pin 24(INPUT B+) of the DSP card. The analog to digital converters on the DSP card then converts this value to a sixteen bit binary value, which can be accessed by the host computer for analysis purposes.

Figure 3.10 shows the relationship between the spindle current Is and the cutting force Fz for all the measurements performed during experimental data acquisition in Chapter 4. The experimental data is shown in Appendix B.

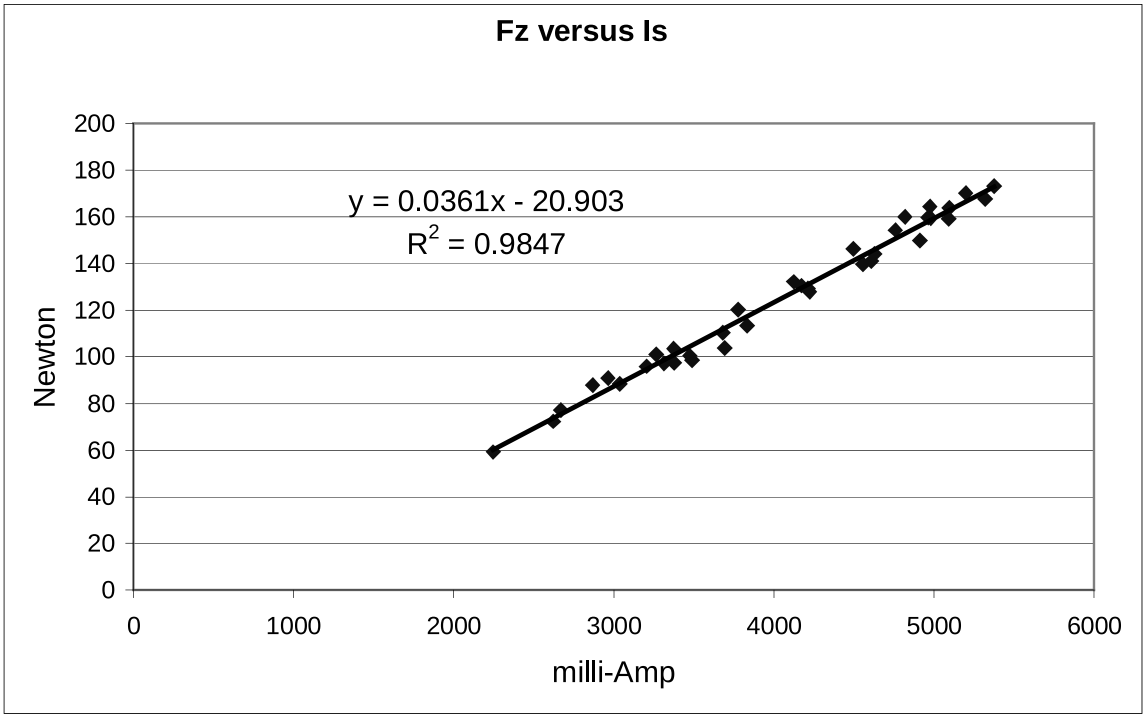
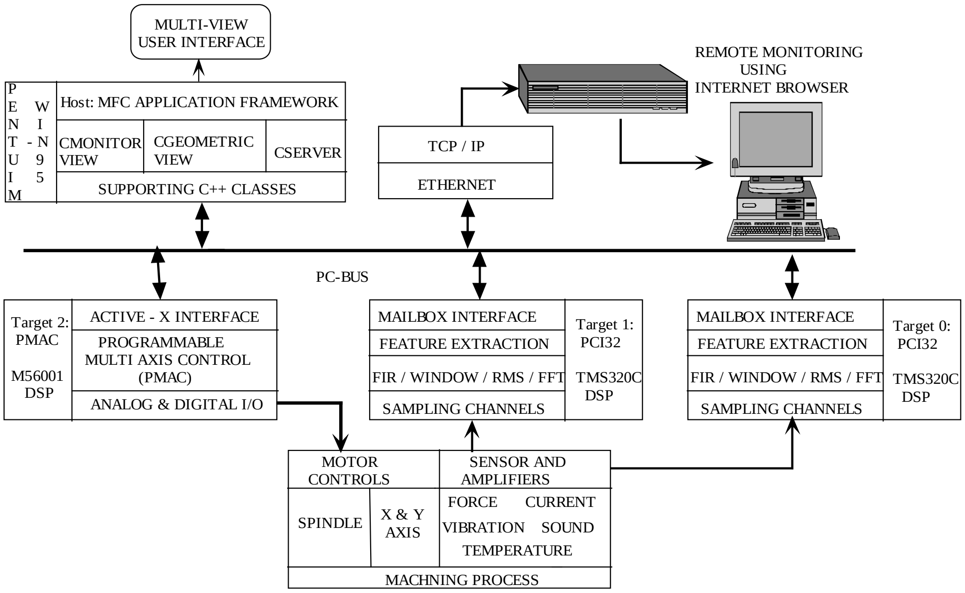


Figure 3.10: The experimental relationship between the spindle current Is and the cutting force Fz.

The values for Is and Fz is obtained after sampling and calculating their rms values. The correlation between the spindle current and cutting force is 98.47%. There is no complex sensor calibration required and the relationship is a simple linear equation.

3.2 Software Components for Experimental Set-Up

Decisions made by the intelligent controller must be made within a relatively short period of time. An adequate response to changing system conditions and events, such as tool wear, must be made within seconds in order to guarantee the reliability of the process [1]. Figure 3.11 shows the hardware architecture with software components implemented to realize the intelligent controller. The two PCI32 DSP modules, target 0 and target 1, continually sample sensor signals and perform real-time signal processing on each. The PMAC target module performs machine control by executing motion control commands. Target 0, 1 and 2 operate independently. The PC-based host module executes a windows based MFC software application framework. It in turn instantiates objects CMonitorView, CGeometricView and CServer each with an appropriate user interface. CMonitorView uses the mailbox interfaces to request on-line process data features from target modules 0 and 1. It may then use the data to execute advanced monitoring and intelligent diagnostic algorithms. CGeometricView uses an Active X [69] to send motion control commands to the PMAC. Its user interface is used to interact directly with the various motion controls, create motion control programs, download motion control programs and start the execution thereof.



11.1.3 Figure 3.11: Hardware architecture and software components for intelligent machining process controller.

The host operating system is Internet enabled and therefore CServer enables remote monitoring of machining process parameters. Figure 3.12 shows the implemented object-oriented software framework for the intelligent machining process controller software components. It shows interfaces to allow communication between modules, basic data structures required, events generated and software functions implemented. The following subsections will briefly describe implementation aspects of the software framework.

3.2.1 Host Module CMonitorView for Machining Process Monitoring

CMonitorView object, created at runtime, by CWinApp, declares a channel_features structure that is used to declare variables to maintain on-line data features from each sampled channel.

```
typedef struct
   {float FREQ;
     float RMS;
     float MEAN;
     float FFTBuf[256];
   } channel_features;

channel_features t0chan0, t0chan1, t0chan2, t0chan3;//t0 – Target 0; t1 – Target 1
channel_features t1chan0, t1chan1, t1chan2, t1chan3;
```

On Windows WM_CREATE event the object constructor opens a device driver to enable communications, using DLL function calls, with target 0 and target 1 [93]. It performs a communication test with both targets.

```
//Open Target 0 and Test Target 0 Responce
target0 = 0;
target_open(target0)
download(0);
do  {    count++;
         read_value = 0;
         read_mb_terminate(target0, TERMINAL_MBOX, &read_value, 0);
         Sleep(100);
     }   while(count<50 && read_value!=0xa5a5);
if  (count==50)
     {   MessageBox(NULL, "Target 0 Application did not respond",
                        MB_ICONINFORMATION);}
```

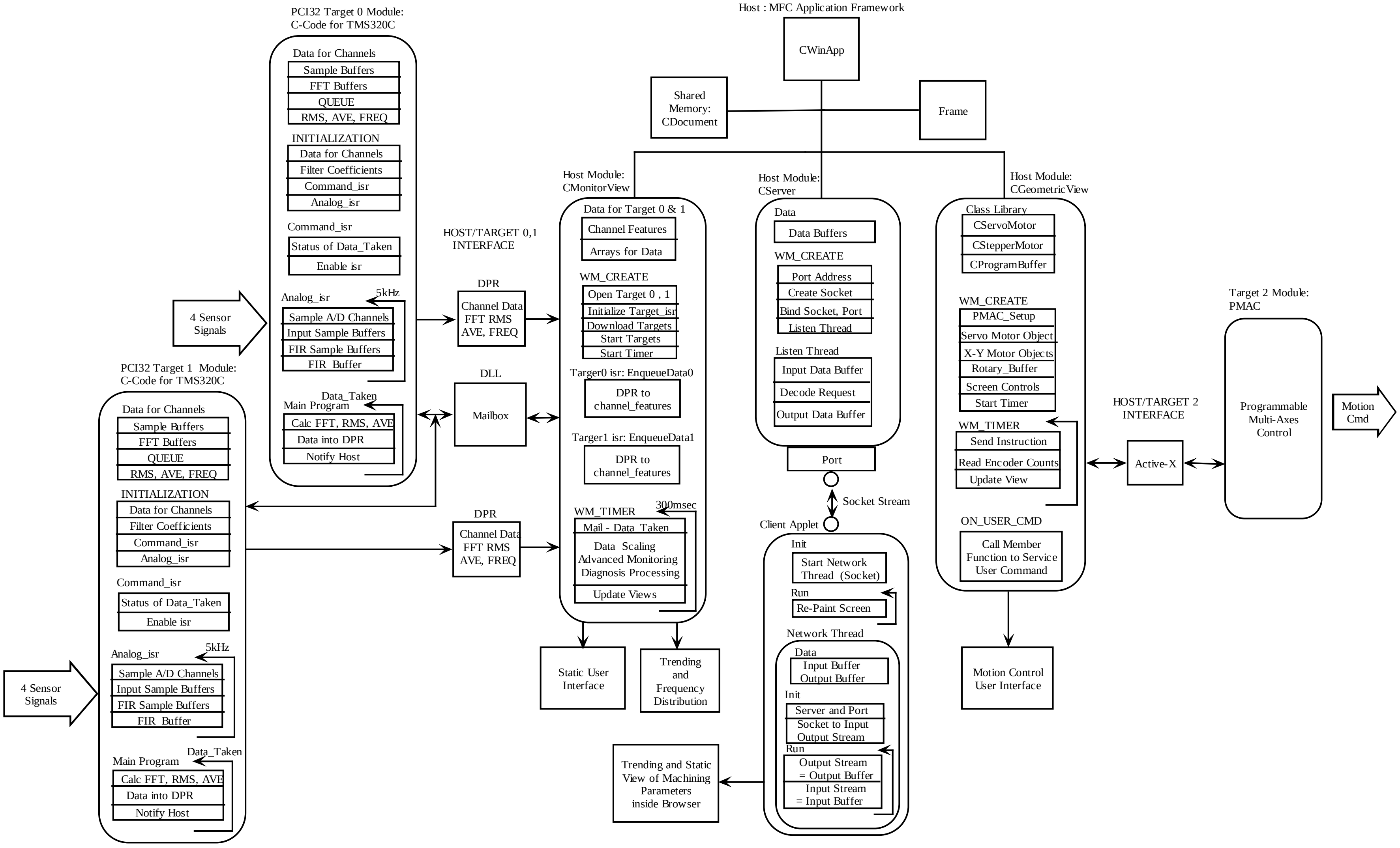


Figure 3.12: Object-oriented software framework for a PC-based intelligent machining process controller.

When the target modules have successfully sampled and processed sensor data, they place the data features into dual port ram used for the host-target interface.   Interrupt service routines, EnqueueData0 and EnqueueData1, transfer data features from the dual port ram, when it receives a hardware interrupt from the target 0 or target 1, into the channel_features data structures. The following code shows how the constructor initializes the interrupt service routines. Code is repeated for target 1.

```
//  Set up the Virtual ISR Enqueue0
host_interrupt_install(target0, EnqueueData0, (PVOID)target0);
host_interrupt_enable(target0);
```

```
void EnqueueData0(void * target0)
{unsigned int i,j,k;
CARDINFO * dsp;
dsp = (CARDINFO* )target_cardinfo((int)target0);
float * dpram =(float *)dsp->BusMaster.Addr;
//Read Target 0 – Channel 0 Data Features
t0chan1.FREQ = dpram[256];
t0chan1.RMS  = dpram[257];
t0chan1.MEAN = dpram[258];
for (i=0;i<256;i++)    t0chan1.FFTBuf[i] = dpram[i];
//Read Target 0 – Channel 1 Data Features
t0chan1.FREQ = dpram[515];
t0chan1.RMS  = dpram[516];
t0chan1.MEAN = dpram[517];
for (i=0, j=259;i<256;i++,j++)  t0chan1.FFTBuf[i] = dpram[j];
//Read Target 0 – Channel 2 Data Features
T0chan2. FREQ = dpram [774];
t0chan2.RMS  = dpram[775];
t0chan2.MEAN = dpram[776];
for (k=0, j=518; k<256;k++,j=j+1)  t0chan2.FFTBuf[k] =dpram[j];
//Read Target 0 – Channel 3 Data Features
t0chan3.FREQ = dpram[1033];
t0chan3.RMS  = dpram[1034];
t0chan3.MEAN = dpram[1035];
for (i=0, j=777;i<256;i++,j++)  t0chan3.FFTBuf[i] = dpram[j];
}
```

The DLL automatically calls the interrupt service routines, EnqueueData0 (shown below) or EnqueueData1, on receiving a hardware interrupt from target 0 or target 1.

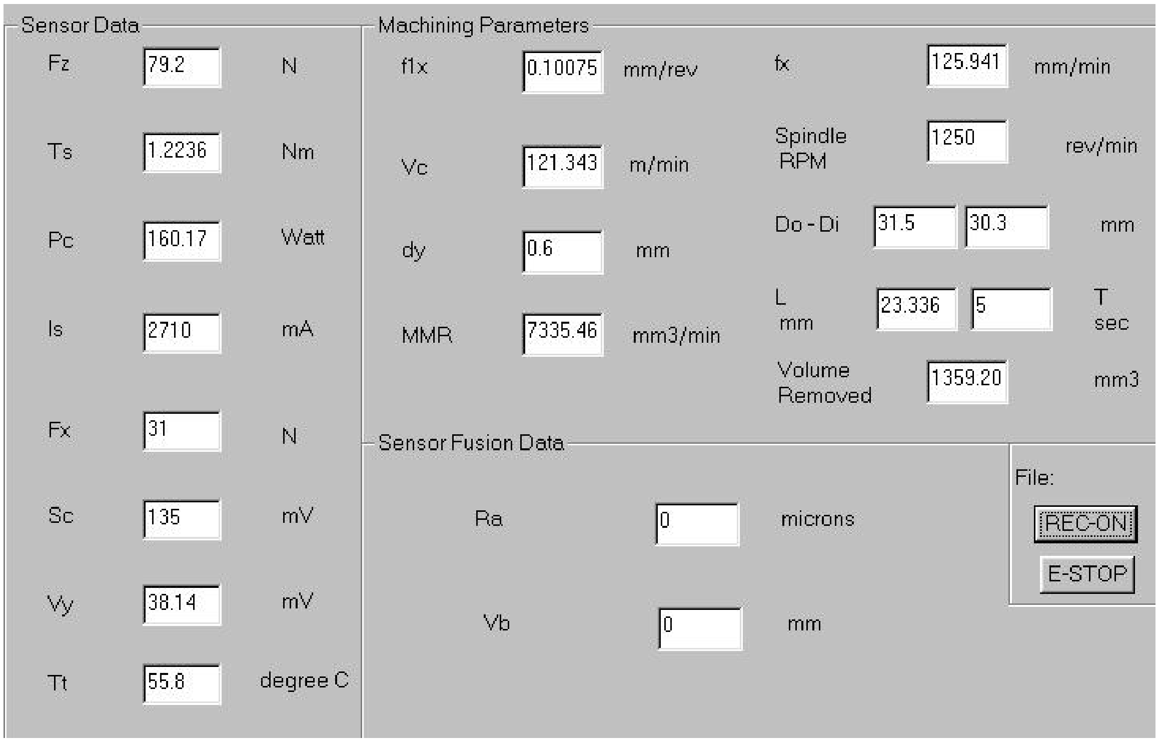
The constructor then downloads the target application file dsptarget.out (source code discussed in Section 3.2.2) into both targets and starts the target applications as shown below. The source code is repeated for target1. Once started the targets operate independently from the host object module CMonitorView.

```
BOOL CMonitorView::download(int tar)
{
//Resets the target, starts talker, then performs a target download
char msg[200];

 if (!(tar))
 {/* reset target0 */
  target_reset(target0);
  clear_mailboxes(target0);
  target_run(target0);
  /* wake up talker */
    if(!start_talker(target0))
    {  MessageBox(NULL, "Target not responding:\ncheck installation and\nmake   sure target
is\nnot held by JTAG", MB_ICONERROR);
    return FALSE;
    }
    else
    {  strcpy(msg, getenv("ii_board")); //c:\pci32cc....
        strcat(msg, "\\examples\\monview\\dsptarget.out");
        if( !iicoffld(msg, target0, NULL) )
            { /* start application */
             start_app(target0);
             return TRUE;
             }
        else
             {sprintf(msg, "COFF load failed. Check that the file to be loaded"
                          " exists and is a COFF file");
              MessageBox(hwnd, msg, szAppName, MB_ICONINFORMATION);
              return FALSE;
             }
    }
 }
}
```

```
void CMonitorView::OnTimer(UINT nIDEvent)
{//Acknowldge to the PCI32 Target Modules – Data Taken
mailbox_interrupt(target0, 1);
mailbox_interrupt(target1, 1);
//Process Parameters
m_DSP_Fx_RMS = float(int((t0chan3.MEAN )*423.3))/10;//Newton
m_DSP_Fz_RMS =  float(int((t0chan2.MEAN)*422.5))/10;//Newton
m_DSP_Is_RMS= (int((t0chan1.RMS*2000))) ;//mA
m_DSP_Sc_RMS = float(int(t1chan0.RMS*1000));//mV
m_DSP_Vy_RMS = float (int(t1chan1.RMS*1000)) ; //mV
m_DSP_Tt = float(int((t1chan2.RMS) *3000))/100 ; //degreeC
//Process Control
//rpm
m_PTalk1.GetResponse(&response, "#1v");
USES_CONVERSION;
strcpy(buf,OLE2T(response));
float Zcount;
m_Speed_RPM = int(-32.9*atof(buf));
//Feedrate mm/min
m_PTalk1.GetResponse(&response, "#3f");
strcpy(buf,OLE2T(response));
Zcount = atof(buf);
m_FeedRate = Zcount * 0.2585 ;
//Feed mm/rev
m_FeedRate_mmrev= m_FeedRate / m_Speed_RPM ;
//Inner & Outer Diameter
m_Diameter_Outer = Part_Outer_Diameter;
m_Depth =  Part_Depth;
m_Diameter_Inner = m_Diameter_Outer - (2*m_Depth);
//Average cutting speed  m/min
m_Speed_Cut=(3.14159*((m_Diameter_Outer + m_Diameter_Inner)/2)*
                                m_Speed_RPM)/1000;
//Metal removal rate  mm3/min
m_Mrr_RMS = 3.14159 * (m_Diameter_Outer + m_Diameter_Inner)/2 * m_FeedRate_mmrev *
m_Depth * m_Speed_RPM ;
//Torque Nm
m_Torque = m_DSP_Fz_RMS * (m_Diameter_Outer+ m_Diameter_Inner)/4000;
//Power in cut - Nm/sec
m_Power_RMS = m_DSP_Fz_RMS  * m_Speed_Cut/60 ;
//PROPOSED POSITION OF Ra AND Vb MONITORING FUNCTIONS
//m_Ra = FTWINRTE(“RA.FTL”,m_Fx,m_Pc,m_Vy );
//m_Wear = FTWINRTE(“WEAR.FTL”,m_Depth, m_FeedRate_mmrev,m_Vy_RMS,
                                                                 m_Is_RMS );

//PROPOSED POSITION OF INTELLIGENT DECISION MAKING SOFTWARE
//SEND DECISION TO PMAC TARGET MODULE
```

Finally the constructor executes the timer (300) function. Every 300 milliseconds a WM_TIMER event initiates a call to the CMonitorView::OnTimer() function. The ::OnTimer() function sends a mailbox message to the target applications indicating that the channel_data have been read. It then uses the sensor sensitivity factors, obtained in Section 3.1.2.1-3.1.2.5, to convert the voltages to standard values. These values are placed into variables linked to Windows dialog controls. Machine control parameters are obtained using the PTalk ActiveX discussed in Section 3.2.3. Finally the timer function cycle ends by executing advanced monitoring and intelligent diagnosis functions (proposed position indicated).

Chapter 4 describes the FTWINRTE () [94] function used to call the fuzzy logic models WEAR.FTL and RA.FTL, whilst Chapter 5 describes intelligent diagnostic scheme. Figure 3.13 shows the user interfaces for on-line viewing of sensor data and machining parameters. The views are updated at the end of ::OnTimer(). The variables shown in Figure 3.13 (a) are linked to the screen's dialog controls and updated by calling the UpdateData(FALSE) function. DLL software functions for implementing real-time scientific graphing and trending [95] are used to display sensor rms and normalized FFT values, as shown in Figure 3.14 (b). To verify these views a function generator was connected to the input of the Fz and Fx channels. Figure 3.14 (b) show the values for a signal generator output voltage set at 3.188 volt (134.94 N) and frequency set at 142 Hz. CMonitorView has on-line recording capability, which will be used in Chapter 4 for experimental analysis.



Sensor Data

Fz	79.2	N
Ts	1.2236	Nm
Pc	160.17	Watt
Is	2710	mA
Fx	31	N
Sc	135	mV
Vy	38.14	mV
Tt	55.8	degree C

Machining Parameters

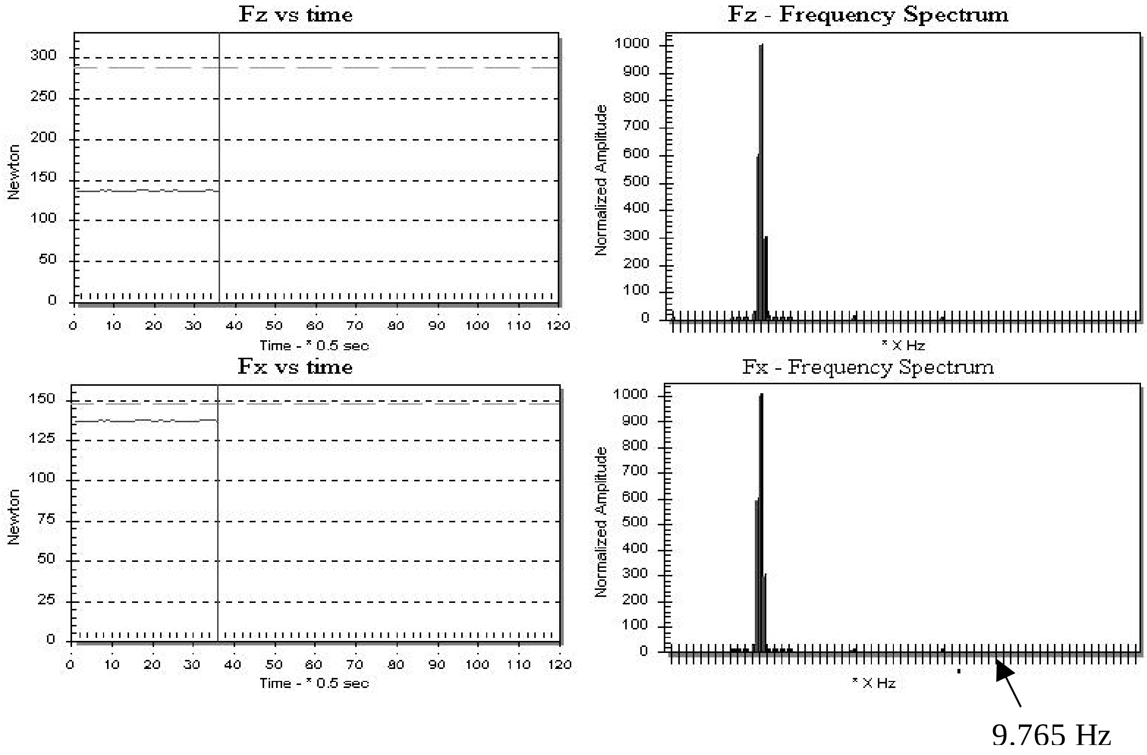
f1x	0.10075	mm/rev
Vc	121.343	m/min
dy	0.6	mm
MMR	7335.46	mm3/min
fx	125.941	mm/min
Spindle RPM	1250	rev/min
Do - Di	31.5 30.3	mm
L mm	23.336	
T sec	5	
Volume Removed	1359.20	mm3

Sensor Fusion Data

Ra	0	microns
Vb	0	mm

File:

(a) Static user interface for on-line viewing of machining process parameters.



(b) On-line trending and frequency distribution of sensor signals.

Figure 3.13: Static and dynamic views of machining process parameters.

3.2.2 PCI32 Target Modules for Signal Sampling and Processing

CMonitorView downloads dsptarget.out into both targets and starts the execution thereof. The executable file is created by compiling dsptarget.c, commented source code given in Appendix C, using the Texas Instruments Floating Point C Compiler [93]. On initialization the target application creates a queue that maintains 2048 filtered sensor samples, 512 from each of the 4 channels, and creates sample buffers to maintain values before filtering and a software timer generated analog service routine.

```
/* ISR data queue */
QUEUE queue;
/* analog sample buffers */
extern volatile float     sample_buffer0[SAMPLE_BUF_SIZE];
extern volatile float     sample_buffer1[SAMPLE_BUF_SIZE];
extern volatile float     sample_buffer2[SAMPLE_BUF_SIZE];
extern volatile float     sample_buffer3[SAMPLE_BUF_SIZE];
volatile int sample_buf_write; /* sample buffer head pointer */
#define analog_isr c_int99
void analog_isr(void);
```

Once started the analog service routine is interrupted every 200 microseconds. It samples the 4 analog channels, filters each channel using fir() function [93] and enqueues the filtered data.

```
void analog_isr(void)
{         int CH0_sample = read_adc(BASEBOARD, 0);
          int CH1_sample = read_adc(BASEBOARD, 1) ;
          int CH2_sample = read_adc(BASEBOARD, 2) ;
          int CH3_sample = read_adc(BASEBOARD, 3);
/* Get sample results, store to circular sample buffers. */
          sample_buffer0[sample_buf_write] = (float)CH0_sample;
          sample_buffer1[sample_buf_write] = (float)CH1_sample;
          sample_buffer2[sample_buf_write] = (float)CH2_sample;
          sample_buffer3[sample_buf_write] = (float)CH3_sample;
if(++sample_buf_write == SAMPLE_BUF_SIZE) /* modulo for rollover */
                     sample_buf_write = 0;              /* correction */
/*call filter routine from library. Arguments are the filter coefficient array (pointer points to the
h(n-1) term), the sample buffer pointer (points to the least recent data point sampled, i.e. the tail
of the sample circular buffer),and the filter order + 1 */
CH0_sample = (float)(fir(&filter_coeff[0], &sample_buffer0[sample_buf_write],
FILTER_ORDER + 1));
CH1_sample = (float)(fir(&filter_coeff[0], &sample_buffer1[sample_buf_write],
FILTER_ORDER + 1));
CH2_sample = (float)(fir(&filter_coeff[0], &sample_buffer2[sample_buf_write],
FILTER_ORDER + 1));
CH3_sample = (float)(fir(&filter_coeff[0], &sample_buffer3[sample_buf_write],
FILTER_ORDER + 1));
/* Place the filtered output samples into the queue */
*((int*)enqueue_ptr(&queue)) = CH0_sample ;
*((int*)enqueue_ptr(&queue)) = CH1_sample;
*((int*)enqueue_ptr(&queue)) = CH2_sample;
*((int*)enqueue_ptr(&queue)) = CH3_sample;
}
```

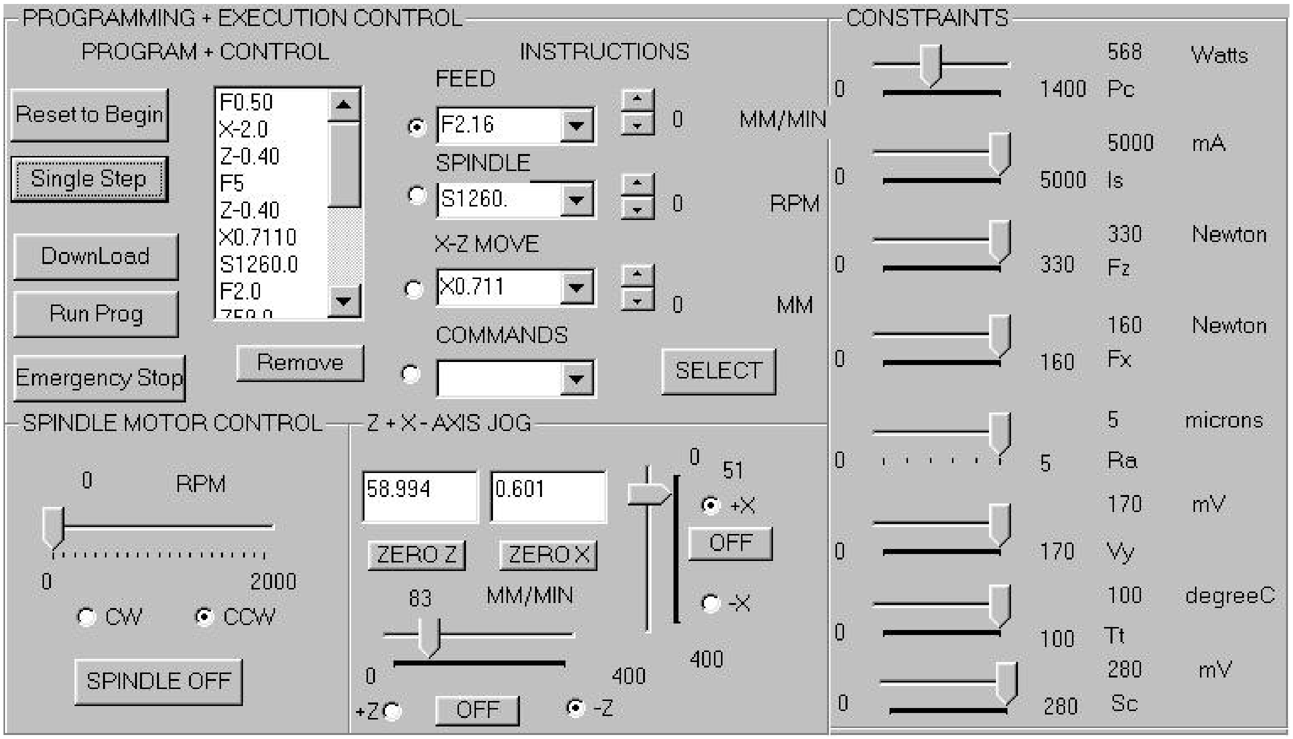
The TMS320C30 DSP used in this project is capable of performing a 16-bit multiplication plus a 32-bit addition in one 60-nanosecond machine cycle. For a realistic filter of 40 taps the TMS320C30 performs the mathematics in around 2.460 microseconds, meaning that it can accept a new input sample every 2.460 microsecond [41].

When a total of 512 samples for each channel (i.e. 2048 samples) is sampled into the filtered queue, it is further processed in the main() function of the program. Processing of each sensor channel includes functions for calculating the rms value, mean value and frequency distribution. The resultant outputs are placed in the dual port ram from where the host application can access it.

```
void main (void)
{
……
//Intialization of data
……
timer(0, 5000);          /* Generates a 5kHz timebase for A/D */
for( ;;)
{/*Wait for Analog_ISR to fill a frame of data */
if (enqueued(&queue) >= FFT_SIZE * 4)
          {/* Place data into FFT input buffer */
          for (i = 0; i < FFT_SIZE; i++)
                    {FFTBufferIn0[i] =   *(volatile int*)dequeue_ptr(&queue);
                    FFTBufferIn1[i] =  *(volatile int*)dequeue_ptr(&queue);
                    FFTBufferIn2[i] =  *(volatile int*)dequeue_ptr(&queue);
                    FFTBufferIn3[i] =  *(volatile int*)dequeue_ptr(&queue);
                    }
          if(data_taken == 1)
                    {
                    /* Process channel 0 */
                    CH0_RMS = CalcRMS(FFTBufferIn0, FFT_SIZE);
                    CH0_AVE = CalcAVE(FFTBufferIn0, FFT_SIZE);
                    CalcFFT(FFTBufferIn0, FFTBufferOut, window, SinTable);
                    CH0_FREQ = CalcFREQ(FFTBufferOut, HALF_FFT_SIZE, 2500.0);
                    for(i = 0; i<256; i++)
                              {dpram[i] = to_ieee(FFTBufferOut[i]);
                              }
                              dpram[256]    = to_ieee(CH0_FREQ);
                              dpram[257]    = to_ieee(CH0_RMS);
                              dpram[258]    = to_ieee(CH0_AVE);
                    ….
                    ….
                    /* Process channel 1 , 2, 3
                    ….
                    …
                    /*Notify host that data is ready to be read*/
                    mailbox_interrupt(1);
                    data_taken =0;
                    /* data_taken will be set to 0 by the host after reading
                    the data*/
                    }//end  if
}//end for
}//end main
```

3.2.3 Host Module CGeometricView Module for Multi-Axis Control

Target 2 module is the PC-based PMAC interface card [68]. The PMAC executes a sequence of motion control commands given to manufacture a part. Execution of the command includes performing all the calculations required to prepare for actual execution of the move [96]. Delta Tau developed PTalkDT, an ActiveX control that is used with 32-bit versions of Visual C++, to serve as a communications link between a Windows application and the PMAC [69]. CGeometricView uses PTalkDT to send motion control commands or a series if commands to the PMAC for execution. Figure 3.14 shows the user interface for CGeometricView.



F0.50
X-2.0
Z-0.40
F5
Z-0.40
X0.7110
S1260.0
F2.0

● FEED F2.16 0 MM/MIN
○ SPINDLE S1260. 0 RPM
○ X-Z MOVE X0.711 0 MM
○ COMMANDS

SPINDLE MOTOR CONTROL: 0 RPM (0 – 2000) ○ CW ● CCW

Z + X - AXIS JOG: 58.994 0.601
83 MM/MIN (0 – 400) ○ +Z ● -Z
0 51 (400) ● +X ○ -X

CONSTRAINTS:
568 Watts, 0 – 1400 Pc
5000 mA, 0 – 5000 Is
330 Newton, 0 – 330 Fz
160 Newton, 0 – 160 Fx
5 microns, 0 – 5 Ra
170 mV, 0 – 170 Vy
100 degreeC, 0 – 100 Tt
280 mV, 0 – 280 Sc

Figure 3.14: User interface for machine control.

Programming and execution controls (Windows) are used to manage the execution of motion control programs. It includes controls to prepare a set of motion control instructions for download to the PMAC. Controls are included to jog motors whilst setting up the tool zero position. Process constraints are set to ensure the on-line integrity, accuracy and quality of the machined part. If a constraint is exceeded the diagnostic system must decide intelligently on a control action that will ensure that the process returns to a reliable state of machining. CGeometricView provides the software interface through which to send the control action. The commands that the PMAC executes are enclosed within classes created to facilitate the execution of on-line multi-axis control commands from within CGeometricView. These classes include: CServoMotor for spindle motor control, CStepperMotor for stepper motor movements within a coordinate system and CProgramBuffer for maintaining a buffer within the PMAC for execution of host programming instructions. The source code for the classes and their member functions is presented in Appendix D. CGeometricView module created at runtime by CWinApp issues a WM_CREATE event, which in turn calls a constructor to perform initialization of PMAC interface card, set up of servo control loops and a coordinate systems. It also instantiates the following machine control objects, classes in Appendix D.

```
CServoMotor Spindle(0,0,0);
CStepperMotor X_Axis(0,0,0, "#2");
CStepperMotor Z_Axis(0,0,0, "#3");
CProgramBuffer RotaryBuff(1,100);
```

It creates a rotary programming buffer in the target to hold motion control instructions for execution.

```
void CGeometricView::OpenRotaryBuffer()
{CHAR buf[255];
BSTR response = SysAllocString(L"");
USES_CONVERSION;
m_PTalk1.SetDeviceNumber(0);
m_PTalk1.SetEnabled(TRUE);
//Create Rotary Buffer - "&1 define rot 100"
m_PTalk1.GetResponse(&response, RotaryBuff.Create());
strcpy(buf,OLE2T(response));
m_PTalk1.GetResponse(&response, "#1j/#2j/#3j/");//close loops
strcpy(buf,OLE2T(response));
//Open Rotary Buffer - "OPEN ROT"
m_PTalk1.GetResponse(&response, RotaryBuff.Open());
strcpy(buf,OLE2T(response));
//Clear Rotary Buffer - "CLEAR"
m_PTalk1.GetResponse(&response, RotaryBuff.Clear());
strcpy(buf,OLE2T(response));
m_PTalk1.GetResponse(&response, RotaryBuff.Execute());//Run Program  - "B0R"
strcpy(buf,OLE2T(response));}
```

The ActiveX function m_PTalk1.GetResponse() is used to communicate commands to the PMAC target [69]. RotaryBuff object member functions return instructions to create, open, and clear a buffer area within the PMAC. Motion control instructions sent to this buffer area will be executed by the PMAC. When a user enters/clicks on a Windows control, as shown in Figure 3.14, an event (ON_USER_CMD) is generated that will direct program control to an appropriate member function within CGeometricView. These member functions make use of the machine control objects Spindle, X_Axis and Z_Axis to control the machining process cutting speed, feed and depth of cut. For example to jog the x-axis stepper motor the following member function is called.

```
void CGeometricView::OnButtonX()
{TCHAR buf[255], buf1[20];
BSTR response = SysAllocString(L"");
USES_CONVERSION;
if (RotaryBuffer == 1) CloseRotaryBuffer() ;
if(X_Axis.GetOffOn() == 0)
{
SetDlgItemText(IDC_BUTTON_X, "X - ON");
if (X_Axis.GetPosNeg() == 1)
        {X_Axis.SetPositive();
        m_PTalk1.GetResponse(&response, X_Axis.GetJogCommand());
        strcpy(buf,OLE2T(response));
        }
if      (X_Axis.GetPosNeg() == -1){ X_Axis.SetNegative();
        m_PTalk1.GetResponse(&response, X_Axis.GetJogCommand());
        strcpy(buf,OLE2T(response));
        }
X_Axis.SetOn();
}
else
{m_PTalk1.GetResponse(&response, X_Axis.KillCMD());
strcpy(buf,OLE2T(response));
SetDlgItemText(IDC_BUTTON_X, "OFF");
X_Axis.SetOff();
m_PTalk1.GetResponse(&response, X_Axis.CloseLoopCMD());
strcpy(buf,OLE2T(response));
}
}
```

On WM_TIMER, the host module sends any new instructions to the target's circular buffer for execution as well as reads back encoder value used to determine and display motor speeds and positions.

3.2.4 CServer to View Process Parameters from within a Remote Browser

Manufacturing companies are looking for ways to assess the performance of their manufacturing equipment and plants from remote sites. World Wide Web (WWW or Web) technologies are a viable vehicle in achieving this objective. Research has focused on multi-media interaction, Virtual Reality modeling and reducing data for file transfer [97]. Ports and sockets are levels of connection supported by both MFC and Java. A port is an abstraction of a physical place through which communication can take place between a server and a client [75]. The server provides the port and the client links to it. The PC used in this project is Web-enabled and on start-up executes Personal Web Server (PWS). PWS listens on the Transport

```
char *bufferin = new char[RECIEVEMAXBUFF], *bufferout = new char [SENDMAXBUFF];
CSockAddr saServer;
ChttpBlockingSocket sConnect;
SaServer = CSockAddr(INADDR_ANY, 8192);//INADDR_ANY use local IP
CBockingSocket g_sListen;  //Global socket for listening, derived from CSocket
CServer::Start()
{g_sListen.Create();
g_Listen.Bind(saServer);
g_sListen.Listen();//Start Listening
AfxBeginThread(ServerThreadProc, , );
..}
UINT ServerThreadProc(LPVOID pParam)
{sConnect.ReadSimpleMsg(bufferin, RECIEVEMAXBUFF, 10 );
..//Decode message stream and place process data into bufferout
..sConnect.Write(bufferout, strlen(bufferout), 10);}
```

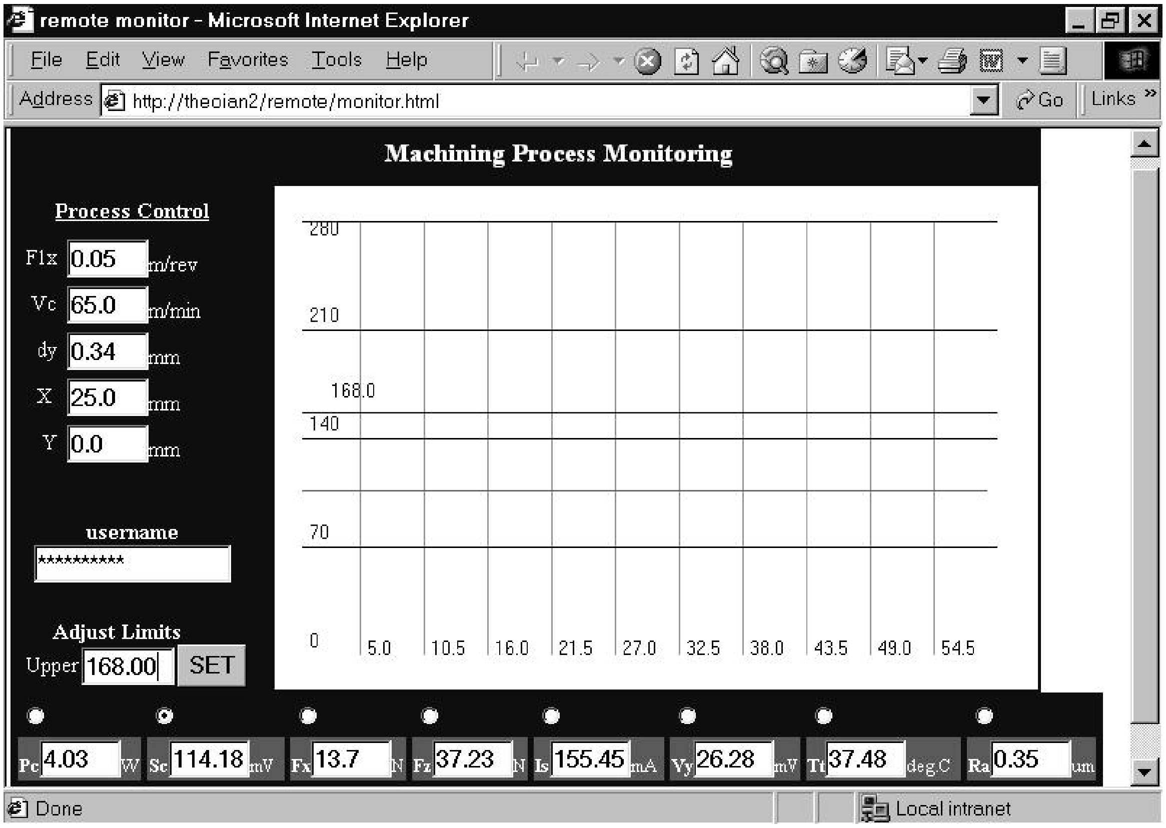
```
Socket socket;
DataInputStream in;
DataOutputStream out;
public void net_start(String ip, int port, JSObject rF)
{..
//open socket to server
socket = new Socket (ip, port);
//create input and output io steams
out = new DataOutputStream (socket.getOutputStream());
in = new DataInputStream (socket.getInputStream());
..
}
```

Control Protocol (TCP) port 80 for a connection from the client side web browser. The browser downloads the Hypertext Markup Language (html) page, which contain a Java applet. The Java applet connects to MFC CServer object through sockets. A socket is an abstraction of the network software that enables communication in and out of a program [75]. Once a socket has been created, the Java client and CServer may communicate any whatever way arranged. A data buffer, that contains on-line machining process performance and limits, is streamed to the Java applet for remote monitoring. The Java applet enables the expert to adjust process performance limits remotely. On WM_CREATE CServer makes use of the MFC CSocket base class to create a socket, bind a socket to a port and listen on the port address as shown below [98]. It then creates a server thread to continually read an input stream, decode a client's data request and send data buffer via output stream.

To establish a simple client in Java using streams requires a socket to connect to the server. The socket methods getInputStream and getOutputstream are used to reference the socket's associated InputStream (in) and OutputStream (out). InputStream read() method is used to input sets of bytes from the server, whilst OutputStream write() method

is used to output sets of bytes to the server.

Figure 3.15 shows the resulting Java client executed inside an Internet browser to monitor machining process parameters. The user has the ability to change process limits, which are updated by CServer. The main applet calls paint () function to refresh the screen every second.


remote monitor - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://theoian2/remote/monitor.html
Go
Links »
Machining Process Monitoring
Process Control
Adjust Limits
SET
280
210
168.0
140
70
0
5.0
10.5
16.0
21.5
27.0
32.5
38.0
43.5
49.0
54.5
Done
Local intranet

F1x 0.05 m/rev
Vc 65.0 m/min
dy 0.34 mm
X 25.0 mm
Y 0.0 mm

username
****

Adjust Limits
Upper 168.00

○ ◉ ○ ○ ○ ○ ○ ○

Pc 4.03 W | Sc 114.18 mV | Fx 13.7 N | Fz 37.23 N | Is 155.45 mA | Vy 26.28 mV | Tt 37.48 deg.C | Ra 0.35 um

Figure 3.15: Java client inside an Internet browser for remote monitoring of machining process.

3.3 Conclusion

A PC-based intelligent machining controller, with in-process quality assurance that is able to detect exceeded tolerances, and adapt quickly (less than a second) to maintain a reliable machining process, has been implemented. The system hardware and software architecture is based on open system philosophy. The Pentium-based PC includes two PCI32 DSP interface cards for signal acquisition and processing, a PMAC for multi-axis control and an Ethernet interface card for remote monitoring and control. An object-oriented software framework for the controller is implemented. The framework includes an MFC application framework to integrate machining process monitoring, diagnosis and machine control. The application framework includes user interfaces to enable visualization of process performance. C++ classes were developed and used to support communication with PMAC interface card.

Machine controls are connected directly to the PMAC interface card, enabling instantaneous reaction from software commands. Sensor and measurement systems that characterize the machining process have been embedded close to the machined product. Signals are connected to PCI32 DSP interface cards. Software to sample and filter sensor signals, determine rms values as well as obtain a signal's frequency distribution, was developed and tested.

The implemented hardware architecture, as shown in Figure 3.11, provide a platform for a generic monitoring, diagnosis and control system to realize an intelligent machining process. The implemented object oriented software framework, as shown in Figure 3.12, enable system re-configurability according to machining process requirements. The successful integration of embedded sensors and machine controls, for a cutting process, with the generic hardware and software contribute to the knowledge in the field of intelligent machining.

Chapter 4 makes use of the machining controller to obtain experimental data that is used to determine sensor signal's sensitivity to tool wear and surface roughness. These sensor signals are used in a multi-sensor fusion model to measure surface roughness and tool wear indirectly. Chapter 5 uses the experimental data to determine the influence of machining input parameter on sensor signals.

Chapter 4

Multi-Sensor Fusion Models for

Tool Wear Classification and Surface Roughness Measurement

To realize advanced automation in machining sensors which will perform reliable on-line, measurement of tool wear and surface roughness is required [1,3]. In this chapter sensor fusion modeling, as shown in Figure 4.1, is used to indirectly measure surface finish and to classify tool wear. Signals that characterize machining process performance (tool-workpiece vibration, tool temperature, cutting forces etc) are processed using DSP technology to extract data features. The data features and cutting parameters may be used as inputs to the FL model.

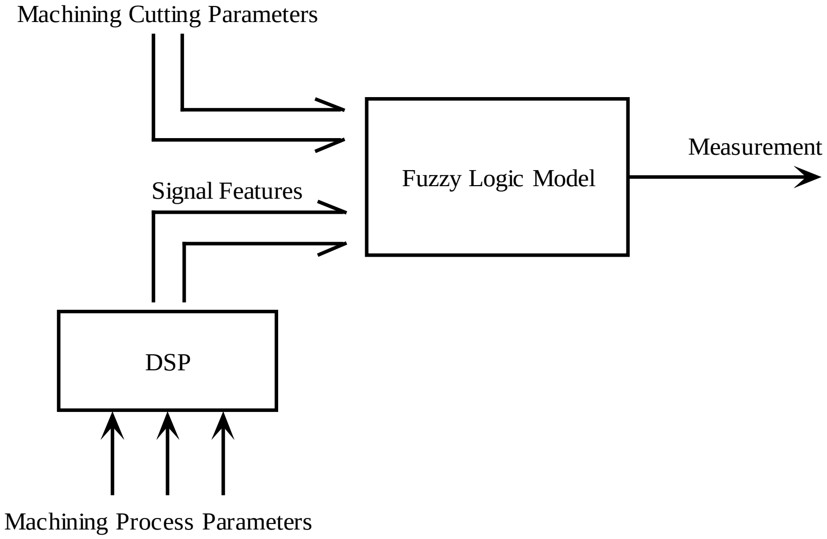


Figure 4.1: Sensor fusion model for tool wear classification and surface roughness measurement.

Section 4.1 describes the methods used to date for indirect measurement of surface roughness and tool wear classification. Section 4.2 describes the process followed in obtaining the FL models from experimental data. In Section 4.3 experimental data is derived by using the experimental set-up explained in Chapter 3. In Section 4.4 and 4.5 the input signals to the multi-sensor models is determined using statistical processing of experimental data. Once the input signal has been identified the experimental data is used to create FL models using FuzzyTech [94] NF-module. The effectiveness of the FL-based sensor fusion model for tool wear and surface roughness measurement is illustrated with numerical examples.

4.1 Introduction

Direct measuring methods for tool wear may include touch trigger probes, optical, radioactive, proximity sensors and electrical resistance measurement techniques [99]. It is difficult to achieve the direct measurements for on-line tool wear monitoring practically due to continuous measurement conducted on a small wear zone. Indirect measurement senses other factors that indicate the cause of tool wear. Indirect sensing methods that have been utilized include cutting forces, acoustic emission, temperature, vibration, spindle motor current, torque, strain and snapshot images of the cutting tool [99]. Most applications use only one or two sensors to detect tool conditions. Tansel and McLaughlin [100] used the force signals for detection of tool breakage for milling process. Force signals were also applied in turning for tool wear monitoring [101]. In the work done by Ko and Cho [102] force and vibration signals were combined for cutting state monitoring in milling with respect to tool wear conditions. Dornfeld [15] used three signals: acoustic emission, cutting forces and spindle motor current for tool wear monitoring. Each signal has sensitivity to tool wear in a certain range and to a certain extent. In order to identify tool conditions and control the process of monitoring in machining, several strategies and techniques have been proposed. Techniques may be summarized as statistical methods, fuzzy technology and neural networks. In statistical methods, the time series analysis approach was applied by Tansel and McLaughlin [100] to detecting tool breakage by monitoring a cutting force or torque signal in any direction. Du and Li [103] proposed a methodology, which uses fuzzy set theory to build a linear fuzzy equation in terms of experimental data for description of the relations between sensing features (monitoring indices) and tool conditions. Since they possess learning capabilities, neural networks have widely been applied to tool condition monitoring, including supervised and unsupervised networks.

Sensing surface roughness may also consist of direct or indirect methods. Direct measurement of surface roughness implies assessing the conditions of the workpiece just behind the cutting edge of the tool. A stylus can be used, but it results in destruction of the sensor head due to high surface speeds of the workpiece. Optical reflection methods have been restricted to measurement of relatively smooth surfaces, but due to limitations are not applicable for use on production floor. A laser measuring system, which employs a linear charge-coupled device sensor and a neural network to process captured light patterns scattered from the workpiece surface, was developed to predict the maximum peak to-valley roughness [104]. However, rather than using direct measurement, several researchers have derived surface roughness indirectly using vibration signals between tool and workpiece generated during the turning process [105].

It is clear that superior performance via neural networks may be achieved if information from multiple sensors is fused [99]. With sensor fusion (concept explained in Section 2.5.5) an individual sensor only senses partially and will contribute to classify the tool wear and surface roughness. However, a combination of sensors or sensor fusion data may classify it with greater accuracy. Research in sensor fusion has a relatively short history in machining. However, recently more attention has been directed using and improving sensor fusion techniques [58]. Two major difficulties are encountered when applying the fusion of sensors. These are the adequate selection of input sensors and the establishment of effective fusion modeling. NN architecture (explained in Section 2.5.3) may be used to learn from data sets whilst FL solutions (architecture and concepts explained in Section 2.5.2) are easy to verify and optimize. Combining the explicit knowledge representation of FL with the learning power of NN result in NF systems. The combination of the NN and FL architectures are described in Section 2.5.4.

4.2 Process for Implementing a Multi-Sensor Model

Figure 4.2 shows the process followed for implementing FL-based model from experimental data. Statistical processing uses experimental data to determine parameters that may be used to classify the tool wear (ANOVA) and for indirect measurement of surface roughness (SPEARMAN). Given the parameters, an NF module (FuzzyTech module) is used to create an FL model in order to classify tool wear using MoM defuzzification. Another FL model is used to measure surface roughness indirectly using CoM defuzzification. The FL model may be edited to enhance the model.

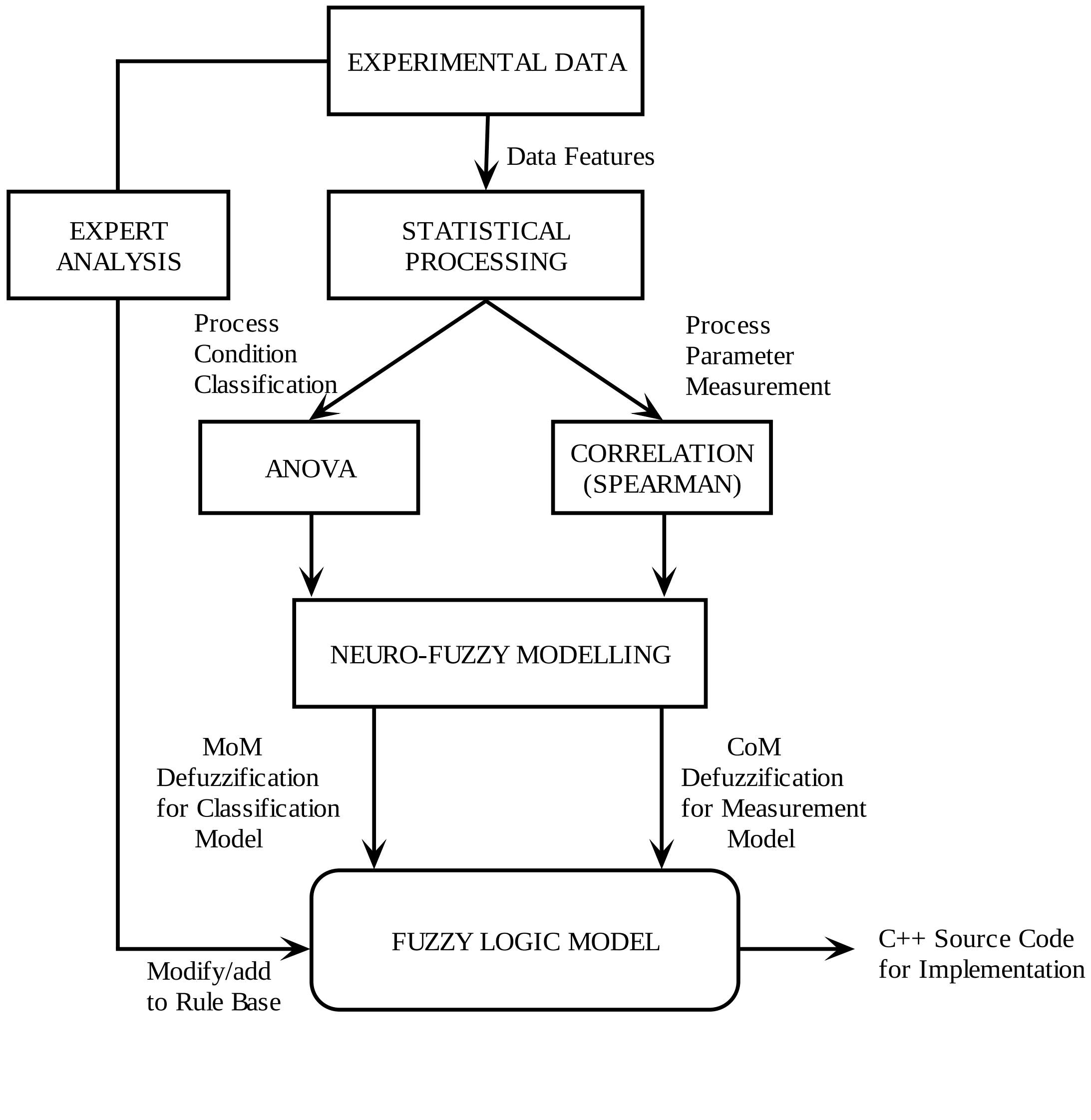

EXPERIMENTAL DATA
Data Features
EXPERT ANALYSIS
STATISTICAL PROCESSING
Process Condition Classification
Process Parameter Measurement
ANOVA
CORRELATION (SPEARMAN)
NEURO-FUZZY MODELLING
MoM Defuzzification for Classification Model
CoM Defuzzification for Measurement Model
FUZZY LOGIC MODEL
Modify/add to Rule Base
C++ Source Code for Implementation

Figure 4.2: Process for implementing FL-based classification / measurement models from experimental data.

4.3Experimental Data Acquisition

Numerous factors influence surface finish during turning operations. Accordingly, as shown by the cause-effect diagram in Figure 4.3, this study will be restricted to cutting parameters, namely: feed, speed and depth of cut and tool wear process condition. The effect of cutting parameters and tool wear on machining process parameters measured, include the following: two cutting forces (Fx and Fz), tool-workpiece vibration (Vy), cutting sound (Sc), spindle current (Is), cutting tool temperature (Tt) and power in the cut (Pc) which is calculated from Fz and Vc.

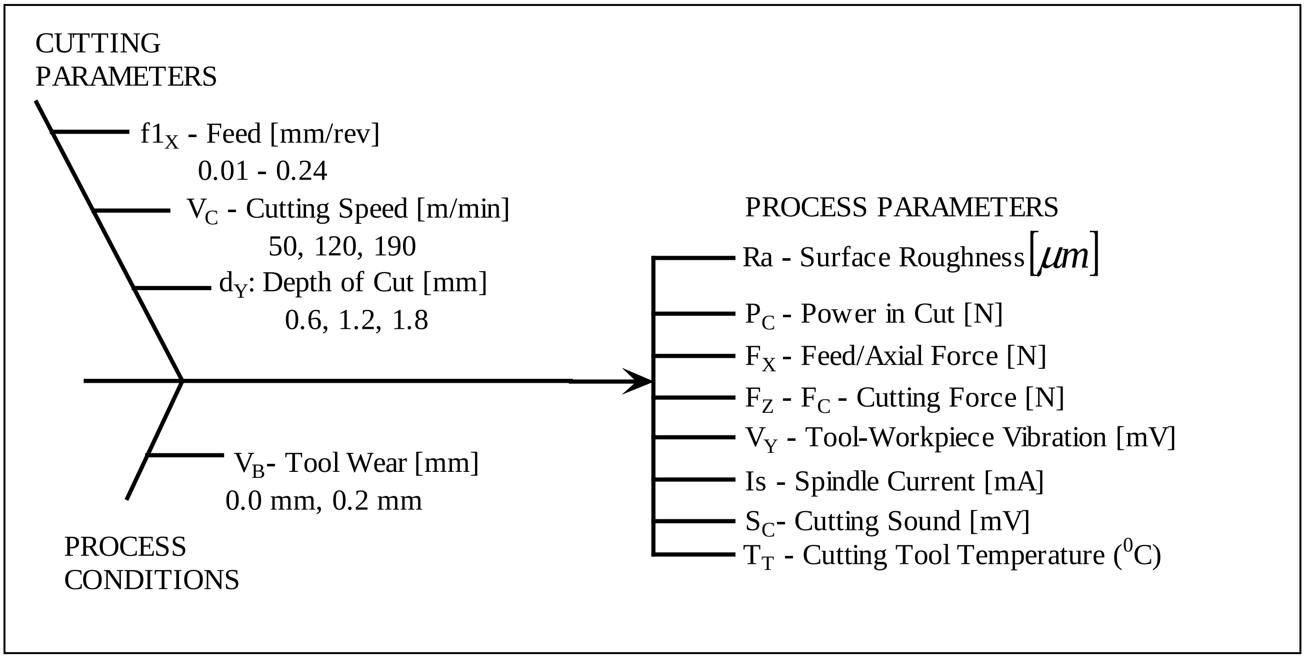
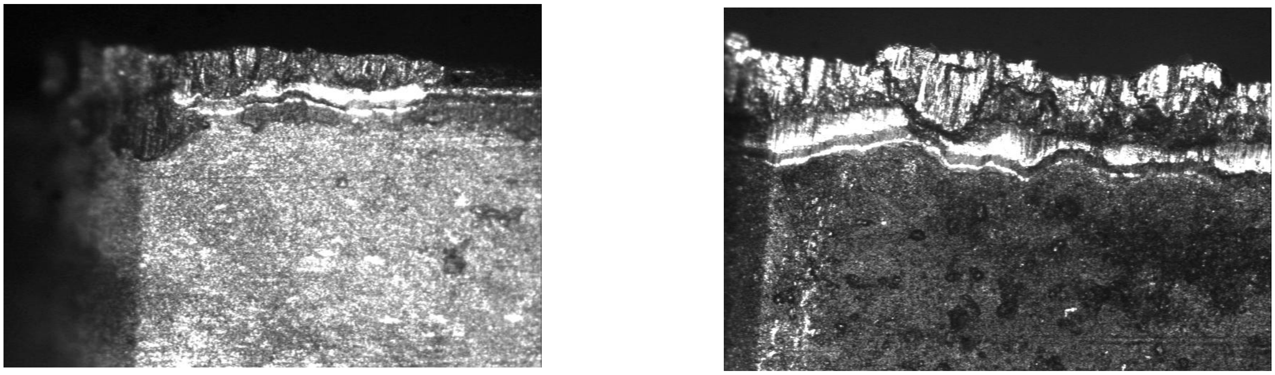

CUTTING PARAMETERS
$f1_X$ - Feed [mm/rev]
0.01 - 0.24
V_C - Cutting Speed [m/min]
50, 120, 190
d_Y: Depth of Cut [mm]
0.6, 1.2, 1.8
V_B- Tool Wear [mm]
0.0 mm, 0.2 mm
PROCESS CONDITIONS
PROCESS PARAMETERS
Ra - Surface Roughness $[\mu m]$
P_C - Power in Cut [N]
F_X - Feed/Axial Force [N]
F_Z - F_C - Cutting Force [N]
V_Y - Tool-Workpiece Vibration [mV]
Is - Spindle Current [mA]
S_C- Cutting Sound [mV]
T_T - Cutting Tool Temperature (^{0}C)

Figure 4.3: Machining process cause - effect diagram.

Machining cutting parameters Vc, dy and f1x were assigned different levels, varying from 50 to 190 [m/min], 0.6-1.8 mm and 0.01 – 0.24 mm/rev, respectively in machining pure aluminum extrusions with a Vickers hardness of 106 (10 kg load). Process conditions were fixed at two levels only, V_B = 0.0 mm and W_B = 0.2 mm, using TP200 a versatile cutting insert. Typical recommended cutting parameters for the TP200 include: Vc = 200 m/min, f1x = 0.3 mm/rev and dy= 0.8 – 3.0 mm [16].

Flank wear land is used as a measure of tool wear. Two TP200 inserts were machined with mild steel at the following fixed machine settings: average cutting speed of 130 m/min, feed of 0.2 mm/rev and a 2 mm depth of cut. The inserts flank wear was monitored using a scanning electron microscope. Figure 4.4 shows the respective wear land, V_B = 0.095 mm and V_B = 0.202 mm, for the two TP200 insets prepared for this project. The tool with V_B = 0.202 mm is used as part of the experimental analysis as shown in Figure 4.2, whilst the tool with wear V_B = 0.096 mm will be used for verification purposes.



(a) Wear land – V_B = 0.095 mm

(b) Wear land – V_B = 0.202 mm

Figure 4.4: Cutting inserts with V_B = 0.095 mm and V_B = 0.202 mm.

The experimental tests have been carried out using the open architecture machine controller, described in Chapter 3, to control the EMCO turning center, equipped with a 1.5kW brushless AC servo spindle motor and two Powermax stepper motors configured into a x-y coordinate system. The multi-axis software control module, as explained in Section 3.2, was used to coordinate the x-y and spindle axis. F_X, $F_{Z,}$ I_S, S_C and Vy signals were sampled at a frequency of 5000 Hz, with the respective RMS and frequency spectrum of each obtained using the DSP monitoring software module as explained in Section 3.2.1. The surface roughness (Ra) had been measured after the cutting operations using a portable Mitutoyo Surftest profilometer. The results of the test for tool wear 0 mm and 0.202 mm is shown in Table B.1 and B.2 in Appendix B respectively.

4.4 Sensor Fusion Model for Tool Wear Classification

To develop a model one first needs to decide which process parameters to use as the input. With statistical analysis one is able to perform an analysis of variance (ANOVA) [28] that indicates which parameter is sensitive to tool wear. After the input-output parameters are in place, a mechanism to train the model is required, whereafter the model is able to operate independently.

4.4.1 Statistical Analysis

The analysis of variance (ANOVA) module of Statistica V6.0 [106] is used to determine which process parameters from Table B.1 (Tool Wear 0.0 mm) and Table B.2 (Tool Wear 0.2 mm) are influenced by tool wear. ANOVA is often used as a screening technique to determine whether there is any probable qualitative relationship between variables before the additional effort and resources are spent in an attempt to develop a quantitative relationship [30]. Statistical hypotheses testing are used to indicate if the long term average values of each data set will be equal, and hence used to reach a decision, if a dependent variable is influenced by tool wear or not. The hypothesis for the ANOVA test is as follows:

Ho: $\mu_{0mm} = \mu_{0.2mm}$, indicates that the long term averages are equal and

H_A: $\mu_{0mm} \neq \mu_{0.2mm}$ indicates the alternate hypothesis.

The p-value, shown in Table 4.1, indicates the truth of the null hypothesis. Therefore a p-value of less than 0.05 indicates that tool wear significantly influences the dependent variable.

Table 4.1:

Resultant p-value of ANOVA hypothesis test between sensor data and tool wear

Dependent variable	p-value
Tt	0.9032
Is	0.4248
Fx	**0.0035**
Sc	0.8924
Fz	0.4248
Vy	**0.0007**

Table 4.1 indicates that tool wear influences feed force (Fx) and tool-workpiece vibration (Vy) significantly, and that these signals may be used in an advanced multi-sensor tool wear monitoring system.

4.4.2 Analysis of Fuzzy Logic Model

The identified signal features influenced by tool wear may serve as inputs to a FL model as shown in Figure 4.5. Pc is added as it contains information on all three the cutting parameters.

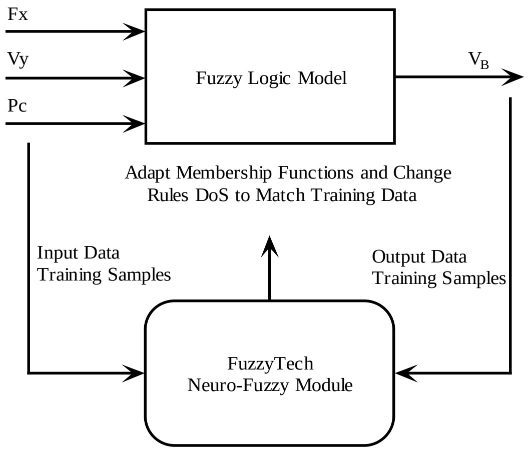

Fx
Vy
Pc
Fuzzy Logic Model
V_B
Adapt Membership Functions and Change Rules DoS to Match Training Data
Input Data Training Samples
Output Data Training Samples
FuzzyTech Neuro-Fuzzy Module

Figure 4.5: Process parameters influenced by tool wear.

FuzzyTech' NF software module [94] is a full graphical development environment that supports all design steps for creating fuzzy logic systems from experimental data. The NF module is used to develop an FL-based sensor fusion model for tool wear classification. Due to the completeness of the data sets it was decided to use the RMS values for Vy, Fx and Pc from experimental data contained in Table B1 and B2, for tool wear 0.0 mm and 0.2 mm respectively as inputs, and tool wear as the output. The power in the cut was added as an input as it contains information regarding all three the machining parameters, illustrated in equations 2.1 – 2.3. The learning process and knowledge representation of the actual data is based on NF modeling described in Section 2.5.4. Table 4.2 shows the values of Vy, Fx and Pc for an additional two cuts taken with the cutting tool, shown in Figure 4.3 (a), having a tool wear land of 0.098 mm. The data is added to the data set for learning and verification purposes.

Table 4.2: Fx, Pc and Vy for V_B = 0.098 mm.

Machining Parameters	Feed Force	Tool-Workpiece Vibration	Power in the Cut

Vc = 122 [m/min]
f1x = 0.2089 [mm/rev]
dy = 0.6 [mm]
	51.6 [N]	81.95 [mV]	282.4 [Watt]

Vc = 119.96 [m/min]
f1x = 0.1041 [mm/rev]
dy = 1.2 [mm]
	83.22 [N]	71.75 [mV]	291.81 [Watt]

Figure 4.6 shows a 3D plot of Vy and Fx versus tool wear for Pc in the region of 300 watts after the FL system was created.

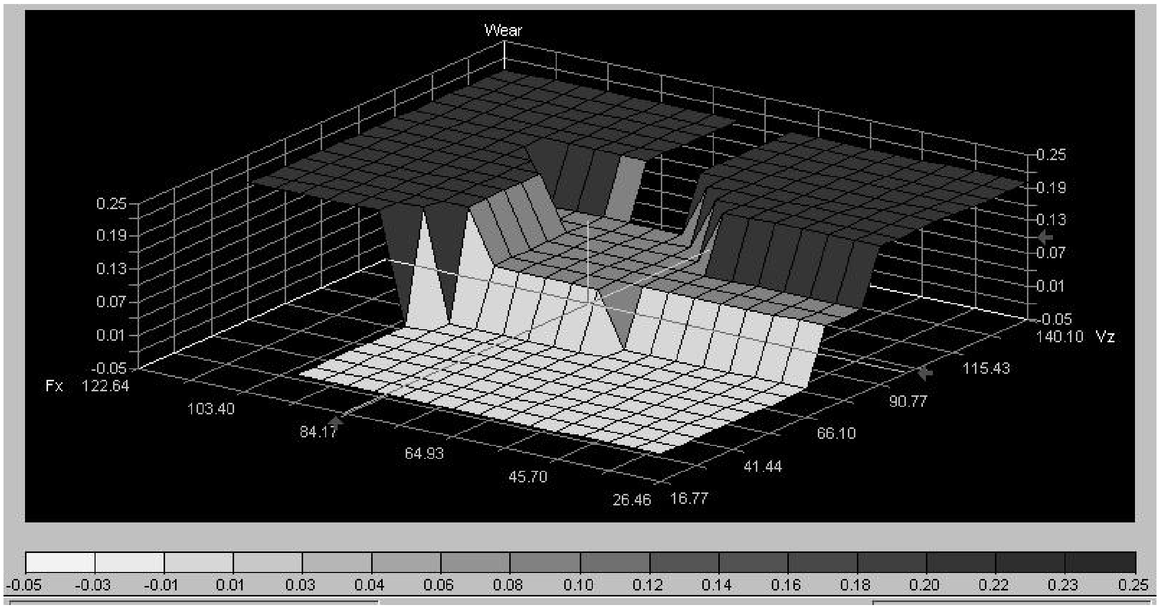
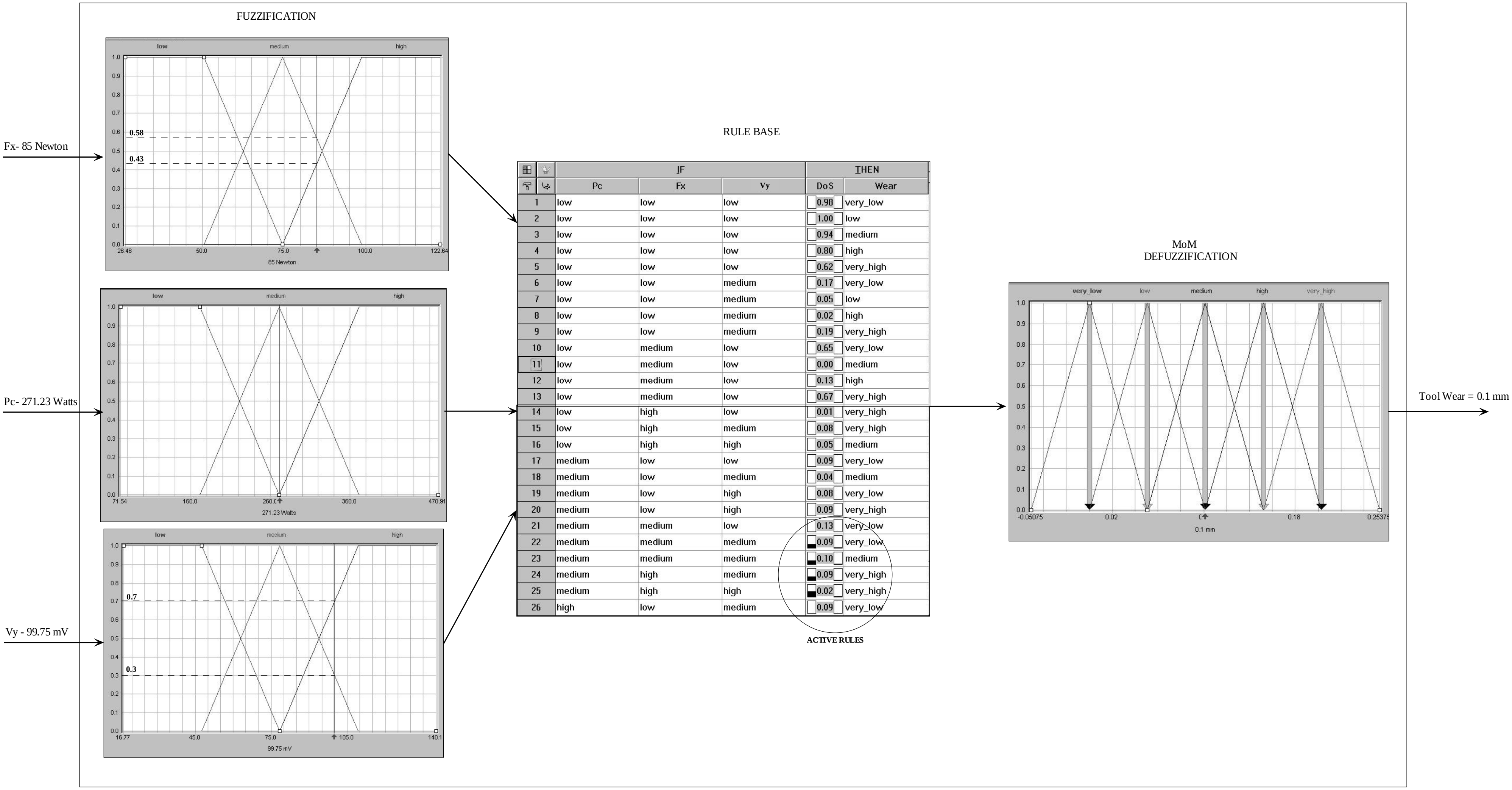


Figure 4.6: 3D plot of Fx and Vy versus V_B for Pc = 280 Watts.

The 3D plot shows how, at a fixed cutting power, the vibration and feed force tool wear increase as the tool wear increases. All the input values of Fx, Vy and Pc from Table B1, B2 (Appendix B) and 4.2 were re-applied to the FL model after training, and the tool wear correctly identified in 90 % of the cases. Figure 4.7 shows the generated FL model for tool wear classification, consisting of input membership functions, a rule base showing the DoS for each rule, as well as an output function specifically configured to produce the MoM defuzzification. After training, the system is a pure FL system which, unlike in the case of a trained NN, allows the addition and /or modification of the rule based knowledge base. To verify the model, input values for Vy = 99.75 mV, Fx = 85 N and Pc = 271.23 Watt were selected using the surface plot shown in Figure 4.6. For these inputs the model should indicate 0.098 mm wear. The fuzzification, inferencing and defuzzification process is shown for analysis purposes.



	IF			THEN	
	Pc	Fx	Vy	DoS	Wear
1	low	low	low	0.98	very_low
2	low	low	low	1.00	low
3	low	low	low	0.94	medium
4	low	low	low	0.80	high
5	low	low	low	0.62	very_high
6	low	low	medium	0.17	very_low
7	low	low	medium	0.05	low
8	low	low	medium	0.02	high
9	low	low	medium	0.19	very_high
10	low	medium	low	0.65	very_low
11	low	medium	low	0.00	medium
12	low	medium	low	0.13	high
13	low	medium	low	0.67	very_high
14	low	high	low	0.01	very_high
15	low	high	medium	0.08	very_high
16	low	high	high	0.05	medium
17	medium	low	low	0.09	very_low
18	medium	low	medium	0.04	medium
19	medium	low	high	0.08	very_low
20	medium	low	high	0.09	very_high
21	medium	medium	low	0.13	very_low
22	medium	medium	medium	0.09	very_low
23	medium	medium	medium	0.10	medium
24	medium	high	medium	0.09	very_high
25	medium	high	high	0.02	very_high
26	high	low	medium	0.09	very_low

Figure 4.7: Fuzzy logic model for tool wear classification.

During fuzzification the following facts, as explained in Equation 2.34, is established:

Fx	=	high	CNF	0.43
Fx	=	medium	CNF	0.58
Pc	=	medium	CNF	1.0
Vy	=	medium	CNF	0.3
Vy	=	high	CNF	0.7

The abovementioned facts activate rules 22, 23, 24, 25 of the fuzzy rule base and may be expressed in a max-min rule structure, from Equation 2.44, as follows:

RULE 22:

IF	Pc	=	medium	CNF		1.0	AND
	Fx	=	medium	CNF		0.58	AND
	Vy	=	medium	CNF		0.3	
THEN	Wear	=	very_low	DoS	=	0.09	

RULE 23:

IF	Pc	=	medium	CNF		1.0	AND
	Fx	=	medium	CNF		0.58	AND
	Vy	=	medium	CNF		0.3	
THEN	Wear	=	medium	DoS	=	0.10	

RULE 24:

IF	Pc	=	medium	CNF		1.0	AND
	Fx	=	high	CNF		0.43	AND
	Vy	=	medium	CNF		0.3	
THEN	Wear	=	very_high	DoS	=	0.09	

RULE 25:

IF	Pc	=	medium	CNF		1.0	AND

	Fx	=	high	CNF		0.43	AND
	Vy	=	high	CNF		0.7	
THEN	Wear	=	very_high	DoS	=	0.02	

During the inference process the If-part of the abovementioned rules are combined using Equation 2.45:

RULE 22: min(1.0, 0.58, 0.3) = 0.3

RULE 23: min(1.0, 0.58, 0.3) = 0.3

RULE 24: min(1.0, 0.43, 0.3) = 0.3

RULE 25: min(1.0, 0.43, 0.7) = 0.43

With FAM inference, the then part of the rule is modified by the DoS-factor as shown in Equation 2.46:

RULE 22:	0.3 * 0.09 =	0.027	RULE 23:	0.3 * 0.10 =	0.03
RULE 24:	0.3*0.09 =	0.027	**RULE 25**:	0.43*0.02=	0.0086

Because Rule 24 and Rule 25 have the same conclusion, they are combined applying Equation 2.47, hence max(0.027, 0.0086) = 0.027. Using MoM defuzzification (best for method of classification applications), Equation 2.49, RULE 23 has the largest final consequence and tool wear is correctly classified as medium (0.1 mm) wear.

4.4.3 Additional Signal Analysis for Data Features Sensitive to Tool Wear

Additional data features that are sensitive to tool wear were identified from signal analysis and include:

- **Fx/Fz Ratio**

Figure 4.8 show the ratio of the feed force to cutting force component for tool wear of 0 mm and 0.2 mm.

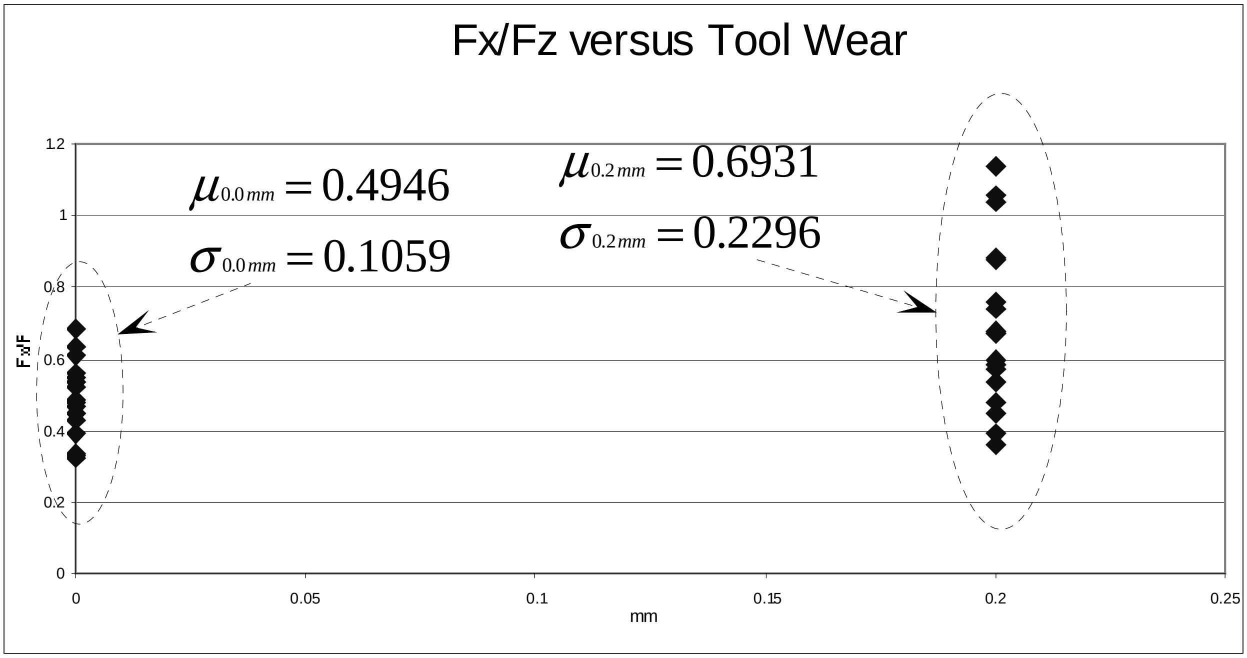
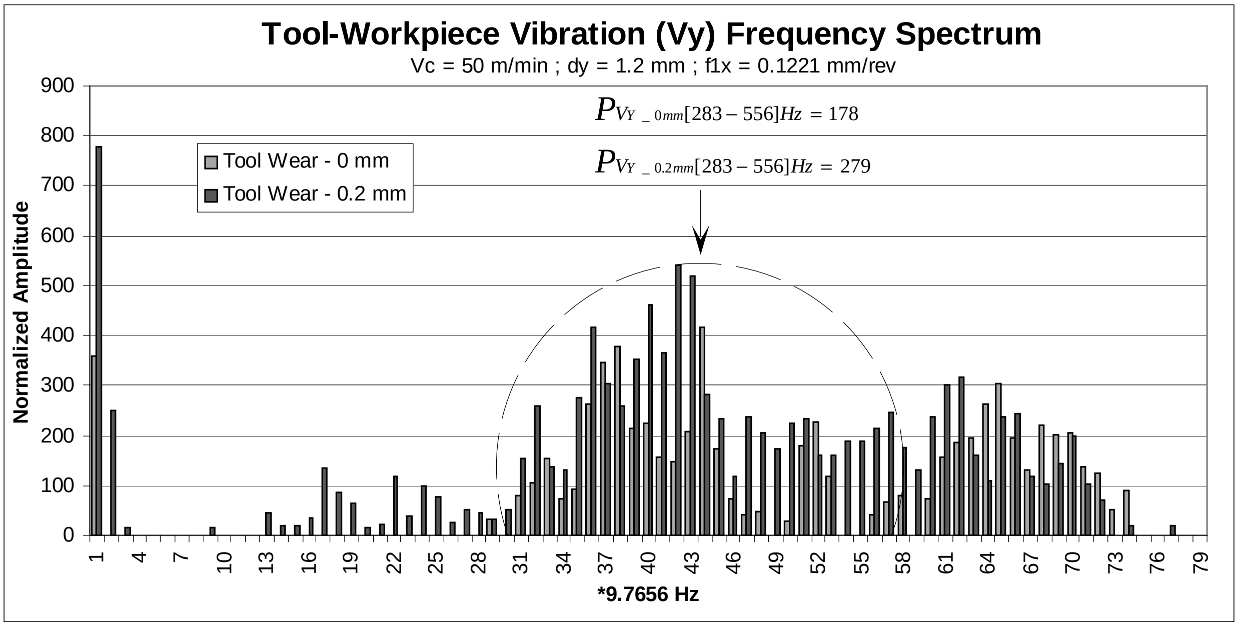


Figure 4.8: Fx/Fz versus tool wear.

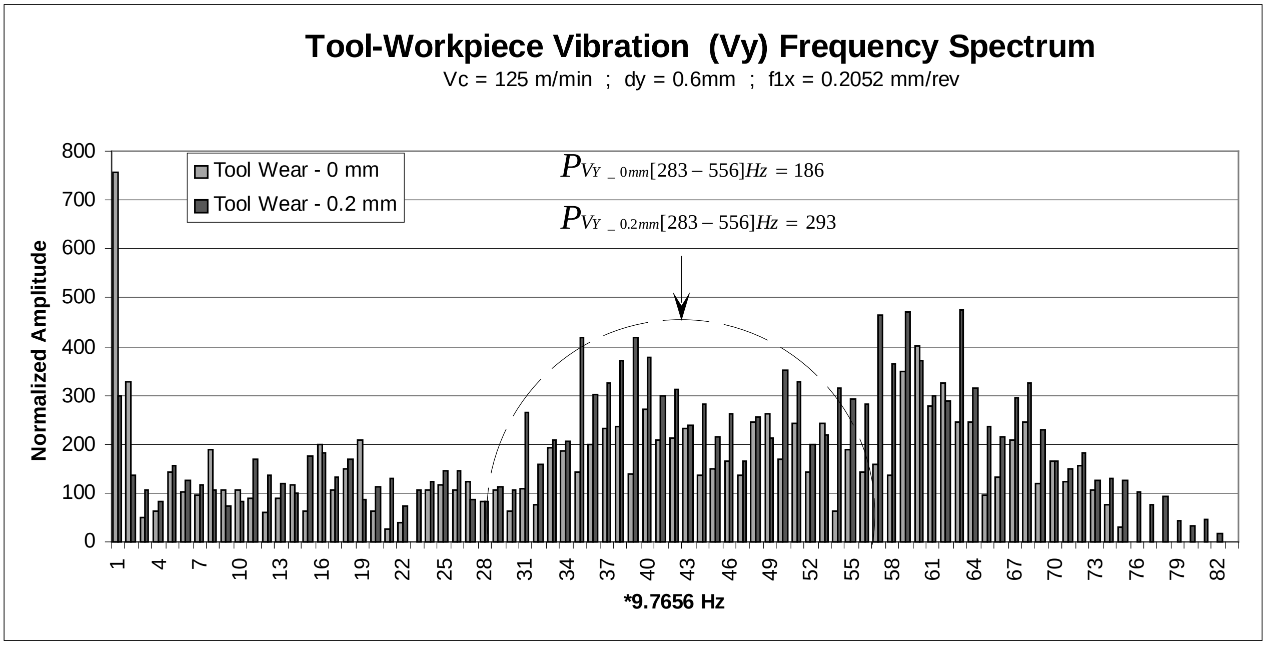
Standard deviation for tool wear 0.2 mm (□$_{0.2mm}$) is more than double that for tool wear 0.0 mm and the ratio of Fx/Fz may therefore serve as a good index in tool wear measurement.

- **Tool-Workpiece Vibration Frequency Spectrum**

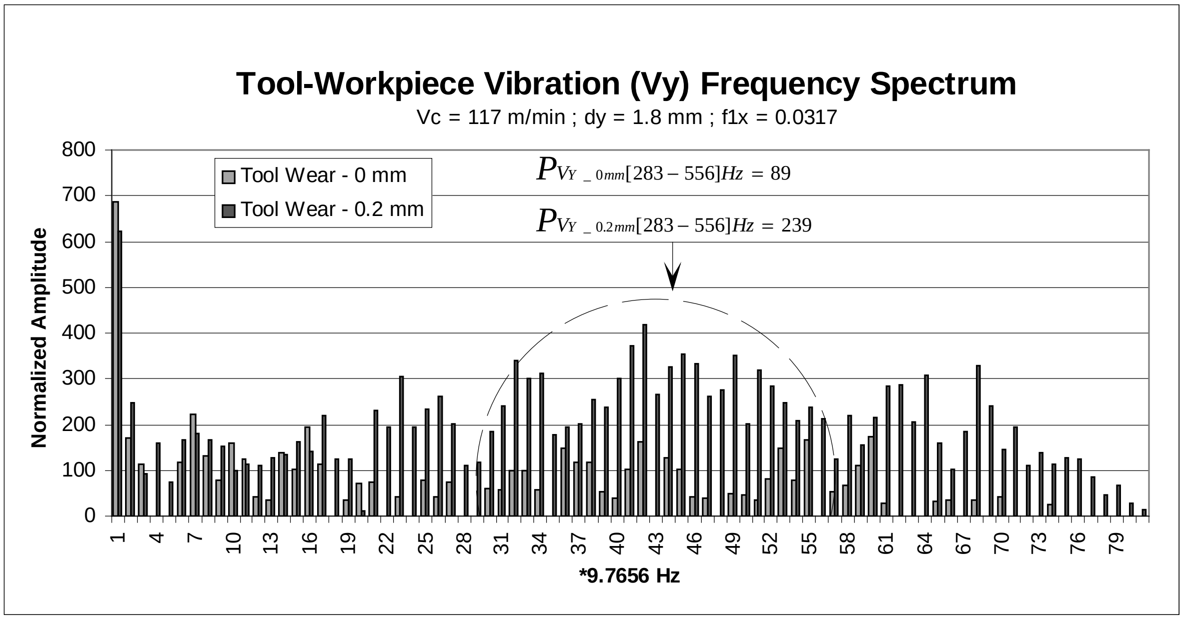
Figure 4.9 shows the normalized amplitude of the vibration signals frequency components for zero tool wear and 0.2 mm for various machining parameters (Vc, f1x, dy).


Tool-Workpiece Vibration (Vy) Frequency Spectrum
Vc = 50 m/min ; dy = 1.2 mm ; f1x = 0.1221 mm/rev
$P_{VY\_0mm}[283-556]Hz = 178$
$P_{VY\_0.2mm}[283-556]Hz = 279$
Tool Wear - 0 mm
Tool Wear - 0.2 mm
Normalized Amplitude
0
100
200
300
400
500
600
700
800
900
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
49
52
55
58
61
64
67
70
73
76
79
*9.7656 Hz

(a)


Tool-Workpiece Vibration (Vy) Frequency Spectrum
Vc = 125 m/min ; dy = 0.6mm ; f1x = 0.2052 mm/rev
Tool Wear - 0 mm
Tool Wear - 0.2 mm
$P_{VY\_0mm}[283-556]Hz = 186$
$P_{VY\_0.2mm}[283-556]Hz = 293$
Normalized Amplitude
0
100
200
300
400
500
600
700
800
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
49
52
55
58
61
64
67
70
73
76
79
82
*9.7656 Hz

(b)



(c)

Figure 4.9: Vibration frequency spectrum for tool wear 0.0 mm and 0.2 mm.

The power of the normalized amplitude in the frequency range 283-556 Hz was calculated and is indicated in Figure 4.9. The values indicate a significance difference between the power in the spectrum for a new tool to that of a tool with wear 0.2 mm. There is also a correlation between the actual sizes of the values for different cutting conditions. Figure 4.10 shows the 283 – 556 frequency spectrum for the vibration signal for tool wear at 0 mm, 0.1 mm and 0.2 mm for verification purposes.

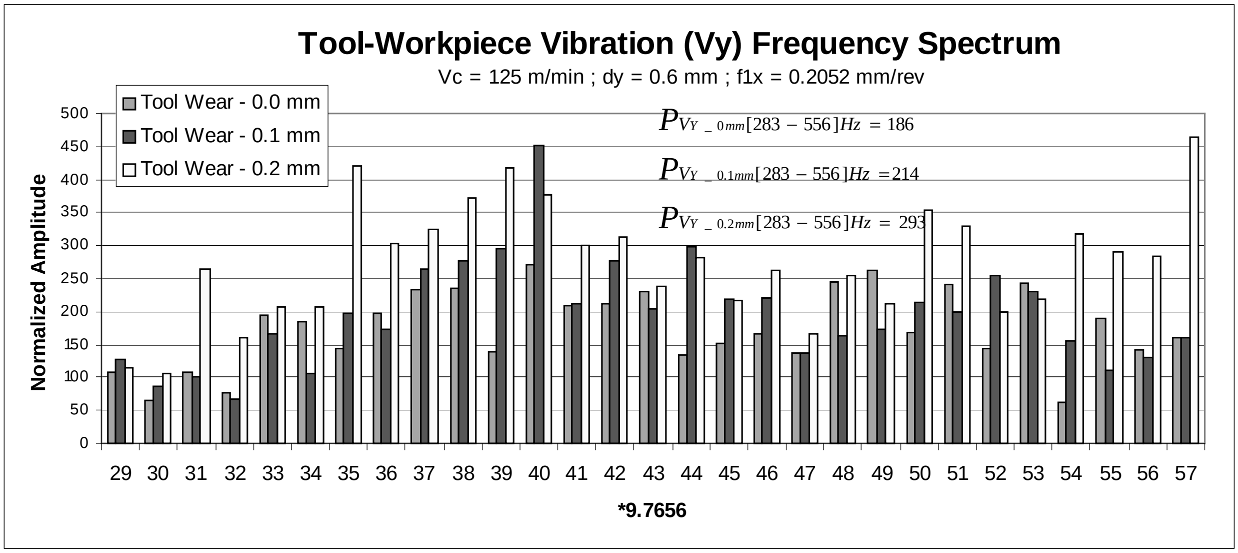
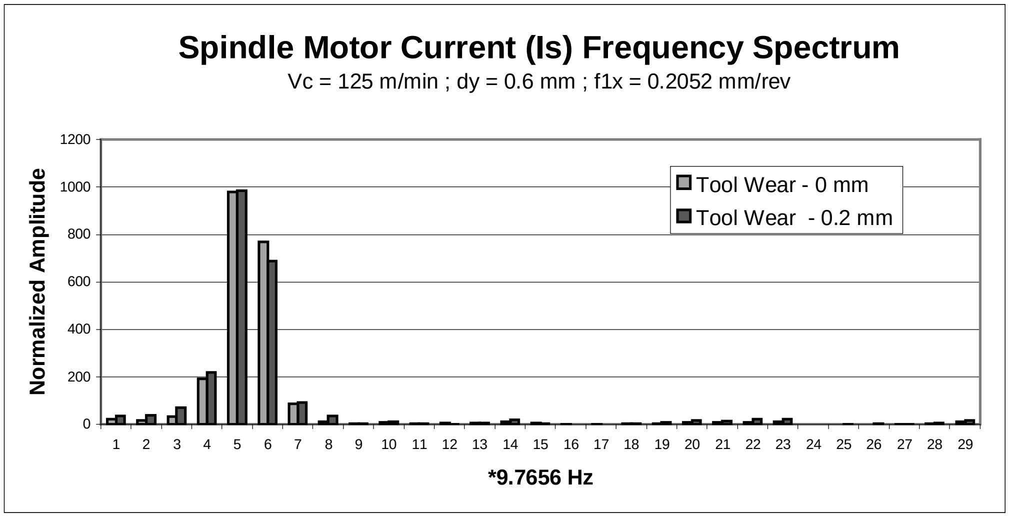


Figure 4.10: Vibration frequency spectrum for tool wear 0.0 mm, 0.1 mm and 0.2 mm.

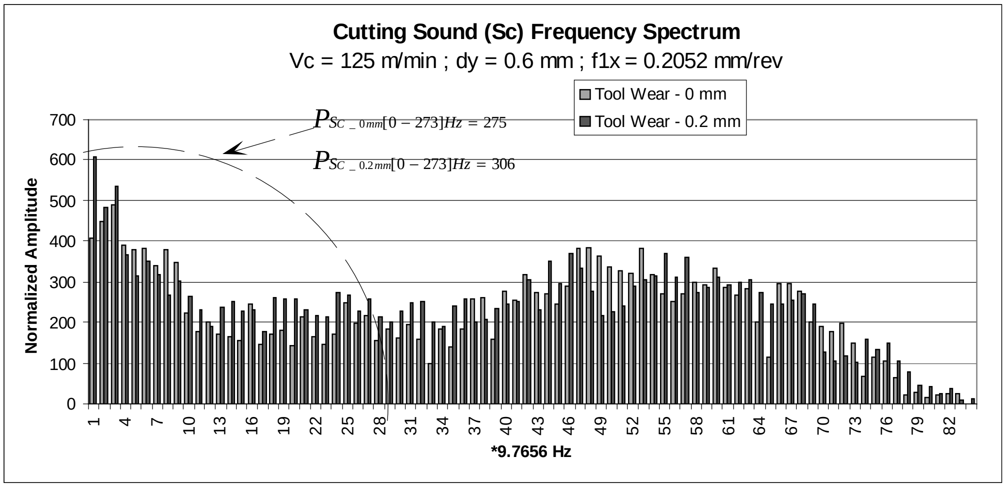
The power for the 0.1 mm worn tool gives a value in between that of the 0 mm and 0.2 mm tools, indicating that this particular frequency spectrum of the vibration is sensitive to tool wear.

- **Spindle current and Cutting Sound Frequency spectrum**

The ANOVA test indicates, for this particular workpiece-cutting tool material combination, that the values of the spindle current and cutting sound signals are not significantly affected by tool wear. However, for completeness of analysis, Figure 4.11 shows the normalized amplitude of the spindle motor current and cutting sound signals frequency components for tool wear 0.0 mm and 0.2 mm.



(a) Spindle current frequency spectrum.



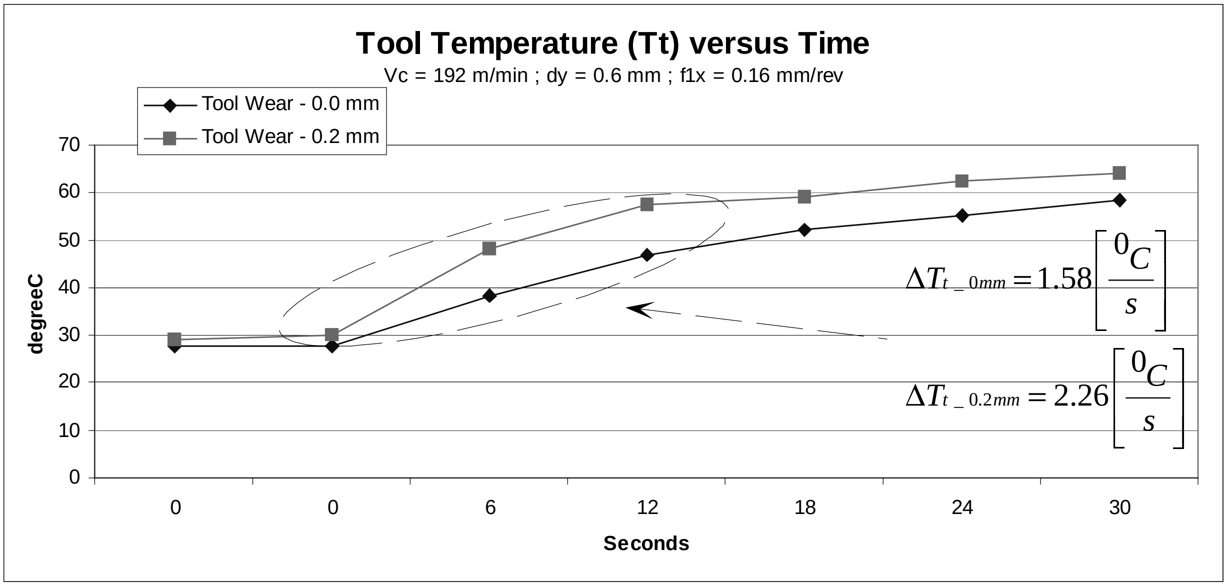
(b) Cutting sound frequency spectrum.

Figure 4.11: Spindle current and cutting sound frequency spectrums.

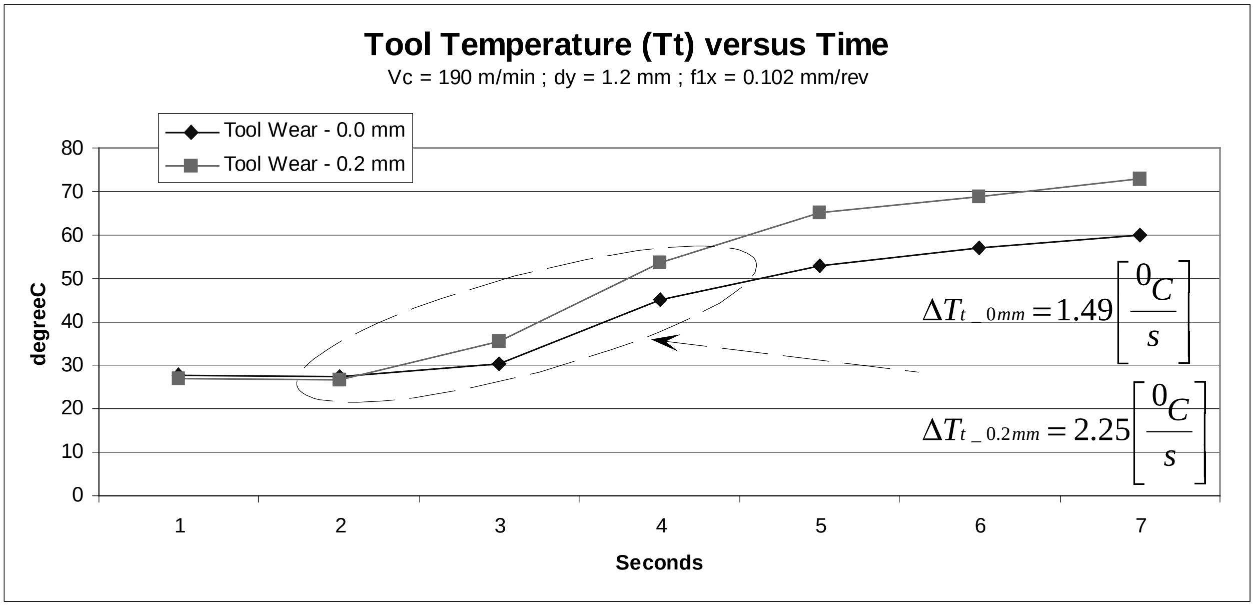
The amplitude of the spindle current frequency components for tool wear 0.2 mm is slightly larger, while the amplitude of the lower frequency components of the cutting sound spectrum indicates an increase.

- ### Cutting Tool Temperature

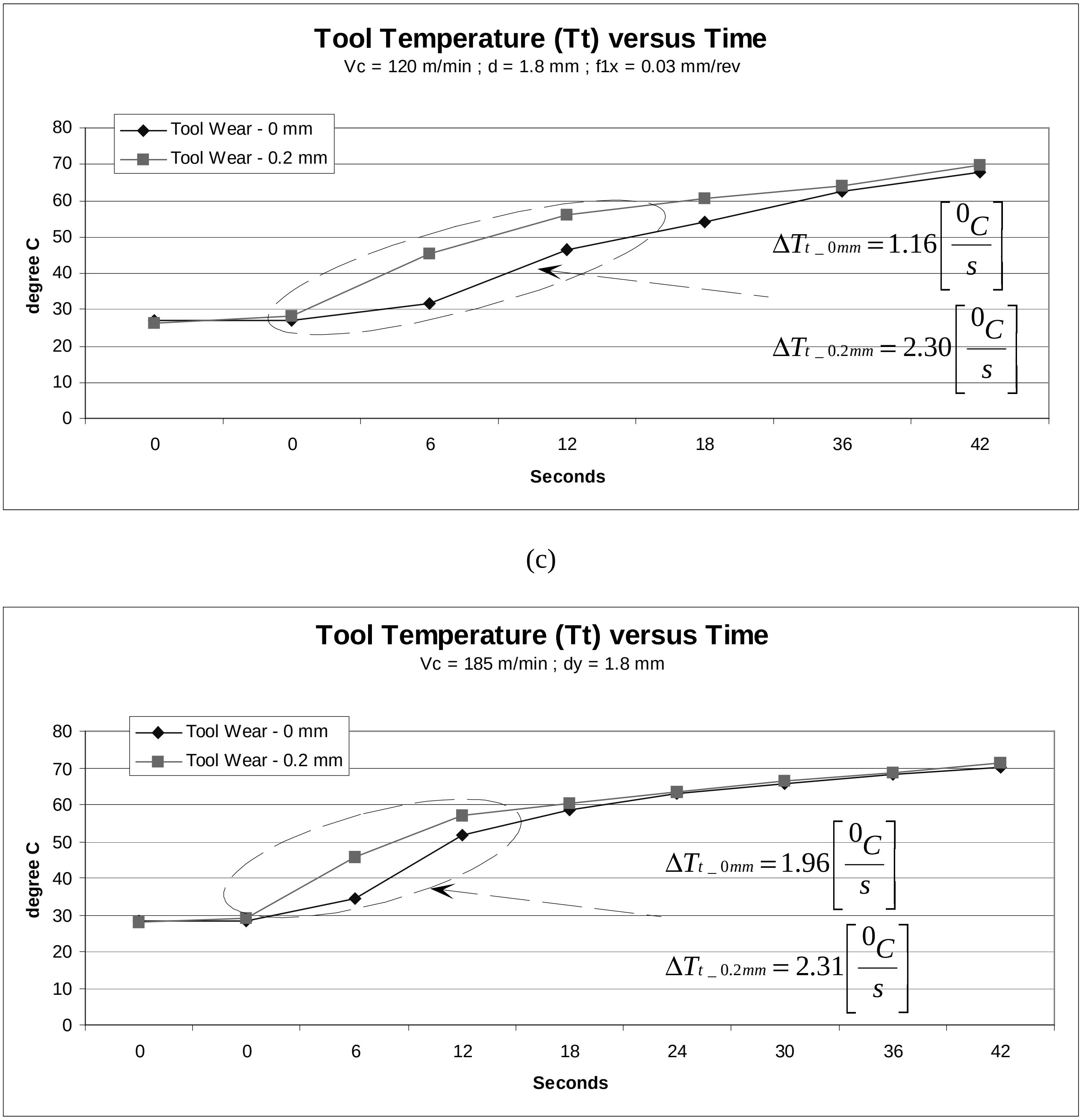
Figure 4.12 shows time domain signals for the cutting tool temperature for tool wear 0.0 mm and 0.2 mm, for various machining parameters (Vc, f1x, dy).



(a)



(b)



(c)

(d)

Figure 4.12: Cutting tool temperature for tool wear 0.0 mm and 0.2 mm.

At high cutting speed there is a difference between the steady state cutting tool temperature for different tool wear levels. More significant is the rate of change in cutting tool temperature, shown in Figure 4.12 (a) to (d), for various machining parameters. The rate of change in the cutting tool temperature may be used as a data feature in a tool wear identification system.

A further advantage of the generated fuzzy logic based system is that it allows for the addition of rules. Additional knowledge regarding data features as discussed may lead to the addition of rules, for example:

RULE 27: If f1x = High and $P_{Sc[0-273]Hz}$ = High then tool wear = High

RULE 28: If Vc = High and Tt = High then tool wear = High

RULE 29: If $\delta Tt / \delta t$ = High then Tool wear = High

Due to the inherent complex and closed nature of a neural network based system, an expert would not be able to add this type of knowledge to the system.

4.5 Sensor Fusion Model for Surface Roughness Measurement

The Spearman's rank correlation [107] module of Statistica V6.0 is used to find the measure of association between surface finish and the machining cutting and process parameters, using the experimental data in Table B.1 (Tool Wear 0.0 mm) and Table B.2 (Tool Wear 0.2 mm). Statistical hypotheses testing are used to indicate if there exists a long-term relationship. The hypothesis for the Spearman's Rank test:

Ho: $R_{Ra\text{-}Sensor\_data} = 0$, indicates that the correlation equals zero and

H_A: $R_{Ra\text{-}Sensor\_data} \neq 0$ indicates the alternate hypothesis.

The p-value, shown in Table 4.3, indicates the truth of the null hypothesis. Therefore a p-value of less than 0.05 indicates correlation between surface roughness and the machining parameter, i.e. alternative hypothesis. It shows that Ra is correlated with Vz (0.3946), Is (0.5043), Fz (0.4999), dy (-0.6478) and strongly correlates to f1x (0.91770). The parameters are used, excluding Fz as it carries the same information as Is, as inputs to an FL based multi-sensor surface roughness monitoring system.

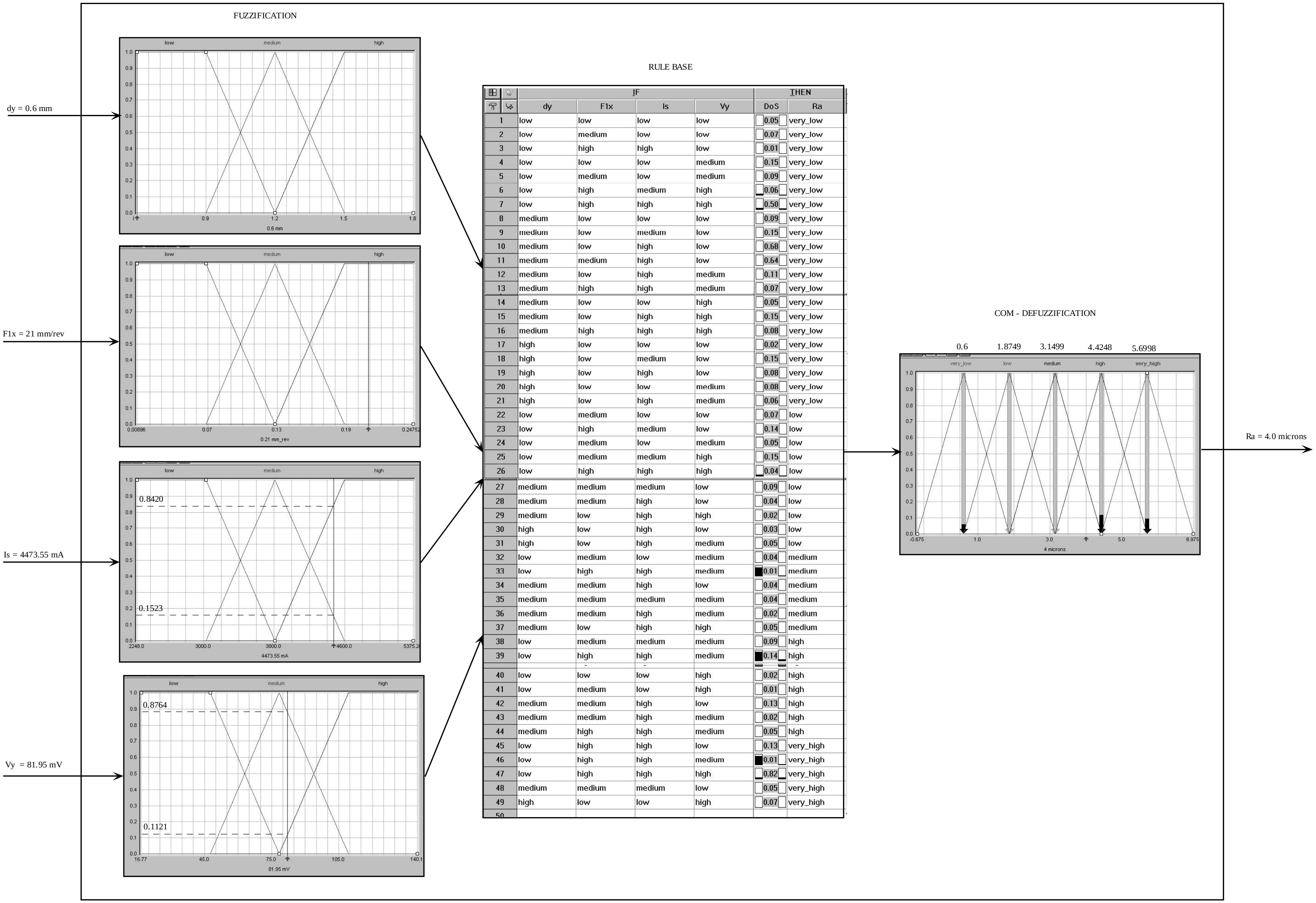
Table 4.3:

Resultant p-value of Spearman's rank hypothesis correlation test between surface roughness and machining parameters

Sensor Machining Data	Spearman's R-value	p-value
Tt	-0.114	0.507947
Pc	0.086435	0.610983
Vz	**0.394604**	**0.015654**
Is	**0.504253**	**0.001461**
Fz	**0.499943**	**0.001629**
Fx	-0.30282	0.068486
Sc	-0.28023	0.102987
dy	**-0.6478**	**1.46E-05**
f1x	**0.917731**	**1.34E-15**
Vc	-0.00565	0.973535

FuzzyTech NF module is used to generate the FL model for surface roughness measurement as shown in Figure 4.13. It shows input membership functions, a rule base with the DoS for each rule as well as an output function specifically configured to produce CoM defuzzification. To verify the fuzzy logic model, input values for Vy = 81.95 mV, f1x = 0.21 mm/rev, Is = 4473.55 mA Watt and dy = 0.6 mm are selected. Fuzzification, inferencing and defuzzification are shown below for analysis purposes. During the fuzzification process the following facts, explained in Equation 2.34, are established:

dy	=	low	CNF	1.0
f1x	=	high	CNF	1.0
Is	=	high	CNF	0.842
Is	=	medium	CNF	0.1523
Vy	=	high	CNF	0.1121
Vy	=	medium	CNF	0.8764



	IF				THEN	
	dy	F1x	Is	Vy	DoS	Ra
1	low	low	low	low	0.05	very_low
2	low	medium	low	low	0.07	very_low
3	low	high	high	low	0.01	very_low
4	low	low	low	medium	0.15	very_low
5	low	medium	low	medium	0.09	very_low
6	low	high	medium	high	0.06	very_low
7	low	high	high	high	0.50	very_low
8	medium	low	low	low	0.09	very_low
9	medium	low	medium	low	0.15	very_low
10	medium	low	high	low	0.68	very_low
11	medium	medium	high	low	0.64	very_low
12	medium	low	high	medium	0.11	very_low
13	medium	high	high	medium	0.07	very_low
14	medium	low	low	high	0.05	very_low
15	medium	low	high	high	0.15	very_low
16	medium	high	high	high	0.08	very_low
17	high	low	low	low	0.02	very_low
18	high	low	medium	low	0.15	very_low
19	high	low	high	low	0.08	very_low
20	high	low	low	medium	0.08	very_low
21	high	low	high	medium	0.06	very_low
22	low	medium	low	low	0.07	low
23	low	high	medium	low	0.14	low
24	low	medium	low	medium	0.05	low
25	low	medium	medium	high	0.15	low
26	low	high	high	high	0.04	low
27	medium	medium	medium	low	0.09	low
28	medium	medium	high	low	0.04	low
29	medium	low	high	high	0.02	low
30	high	low	high	low	0.03	low
31	high	low	high	medium	0.05	low
32	low	medium	low	medium	0.04	medium
33	low	high	high	medium	0.01	medium
34	medium	medium	high	low	0.04	medium
35	medium	medium	medium	medium	0.04	medium
36	medium	medium	high	medium	0.02	medium
37	medium	low	high	high	0.05	medium
38	low	medium	medium	medium	0.09	high
39	low	high	high	medium	0.14	high
		-	-			-
40	low	low	low	high	0.02	high
41	low	medium	low	high	0.01	high
42	medium	medium	high	low	0.13	high
43	medium	medium	high	medium	0.02	high
44	medium	high	high	medium	0.05	high
45	low	high	high	low	0.13	very_high
46	low	high	high	medium	0.01	very_high
47	low	high	high	high	0.82	very_high
48	medium	medium	medium	low	0.05	very_high
49	high	low	low	high	0.07	very_high
50						

Figure 4.13: Fuzzy logic model for surface roughness measurement.

The abovementioned facts activate rules 6, 7, 25, 26, 33, 38, 39, 46, 47 of the fuzzy rule base and may be expressed as a max-min rule structure from Equation 2.44, as follows:

RULE 6:

IF	dy	=	low	cnf	1.0
	f1x	=	high	CNF	1.0
	Is	=	medium	CNF	0.1523
	Vy	=	high	CNF	0.1121
THEN	Ra	=	very_low	DoS	0.06

RULE 7:

IF	dy	=	low	CNF	1.0
	f1x	=	high	CNF	1.0
	Is	=	high	CNF	0.842
	Vy	=	high	CNF	0.1121
THEN	Ra	=	very_low	DoS	0.5

RULE 26:

IF	dy	=	low	cnf	1.0
	f1x	=	high	cnf	1.0
	Is	=	high	cnf	0.842
	Vy	=	high	cnf	0.1121
THEN	Ra	=	low	DoS	0.04

RULE 33:

IF	dy	=	low	cnf	1.0
	f1x	=	high	cnf	1.0
	Is	=	high	cnf	0.842
	Vy	=	medium	cnf	0.8764
THEN	Ra	=	meduim	DoS	0.01

RULE 39:

IF	dy	=	low	cnf	1.0
	f1x	=	high	cnf	1.0
	Is	=	high	cnf	0.842

	Vy	=	medium	cnf	0.8764
THEN	Ra	=	high	DoS	0.14
RULE 46:					
IF	dy	=	low	cnf	1.0
	f1x	=	high	cnf	1.0
	Is	=	high	cnf	0.842
	Vy	=	medium	cnf	0.8764
THEN	Ra	=	very_high	DoS	0.01
RULE 47:					
IF	dy	=	low	cnf	1.0
	f1x	=	high	cnf	1.0
	Is	=	high	cnf	0.842
	Vy	=	high	cnf	0.1121
THEN	Ra	=	very_high	DoS	0.82

During the inference process the If-part of the abovementioned rules is combined using Equation 2.45:

RULE 6: min(1.0, 1.0, 0.1523, 0.1121) = 0.1121
RULE 7: min(1.0, 1.0, 0.842, 0.1121) = 0.1121
RULE 26: min(1.0, 1.0, 0.842, 0.1121) = 0.1121
RULE 33: min(1.0, 1.0, 0.842, 0.8764) = 0.842
RULE 39: min(1.0, 1.0, 0.842, 0.8764) = 0.842
RULE 46: min(1.0, 1.0, 0.842, 0.8764) = 0.842
RULE 47: min(1.0, 1.0, 0.842, 0.1121) = 0.1121

With FAM inference, the then part of the rule is modified by the DoS-factor as shown in Equation 2.46:

RULE 6:	0.1121* 0.06=0.0076	RULE 7:	0.1121*0.5=0.0561
RULE 26:	0.1121*0.04 = 0.0045		
RULE 33:	0.842*.01= 0.0842		
RULE 39:	0.842*0.14 = 0.1179		
RULE 46:	0.842*.01=0.00842	RULE 47:	0.1121*0.82=0.0919

RULE 6 and Rule 7 are combined applying Equation 2.47: max(0.0076, 0.0561) = 0.0561.

RULE 46 and Rule 47 are combined applying Equation 2.47: max(0.00842, 0.0919) = 0.0919

Applying COM defuzzification (best for method of control applications) as shown in Equation 2.48 result in:

$$Ra = \frac{((5.6998 * .0919) + (4.4248 * 0.1179) + (3.1499 * .0842) + (1.8749 * 0.0045) + (0.6 * .0561))}{(0.0919 + 0.1179 + 0.0842 + 0.0045 + 0.0561)}$$

$$Ra = 3.815 \ \textit{microns}$$

Figure 4.14 compares the surface roughness obtained experimentally with the predicted values determined using the FL model and a commonly used theoretical model given by Equation 2.8.

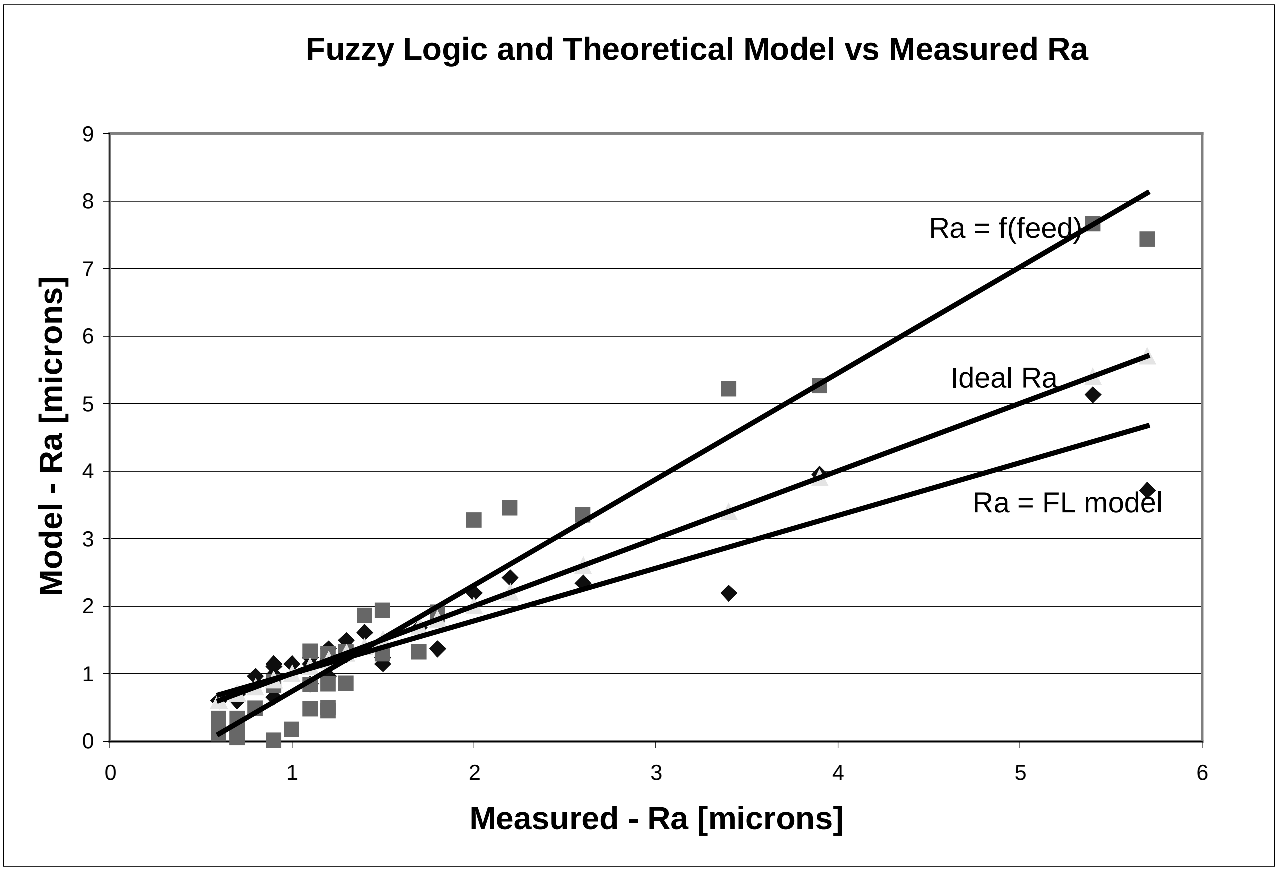


Figure 4.14: Comparison between fuzzy logic and theoretical model vs measured surface finish.

The comparative results indicate that the FL model (13.56%) represented an average error of at least three times lower than the theoretical model (43.31 %). In both cases the error increased as the feed increased.

4.6 Conclusion

A process to implement FL-based classification and measurement models from experimental data has been identified.

ANOVA is successfully used to identify signals, Fx and Vy, used as inputs to an FL model for the classification of tool wear. Further analysis of signals indicate that Fx/Fz, $P_{Vy[283\text{-}556]Hz}$, $P_{Sc[0\text{-}273]Hz}$, δTt/δt are sensitive to tool wear.

Spearman’s correlation is successfully applied to identify signals, dy, f1x, Is and Vy, that correlate with Ra, used as inputs to an FL model to measure surface roughness.

FuzzyTech’s NF module is used to successfully create pure FL models. The models are able to measure Ra with an accuracy of 86.44% and to classify tool wear with a 90% success rate. With the addition of expert or sensor data the rule-based knowledge bases can be enhanced and improved, which is not the case with a pure neural network model.

NF modeling tools, like FuzzyTech, allow for the generation of fuzzy logic systems into C code. The resulting code can be integrated within other C source code and compiled to form a standalone application. The proposed FL models can therefore be integrated into the experimental set-up, explained in Chapter 3, for on-line monitoring of tool wear and surface roughness. The online measurement of Ra now serves as an input to an intelligent diagnostic system, explained in Chapter 5, that will ensure that the quality of the machined product is maintained. The tool wear sensing system feeds the controller with on-line estimates of tool wear. Based on these estimates, the controller may adjust the depth of cut to maintain on-line dimensional accuracy.

Chapter 5

Diagnosis for Intelligent Machining Process Control

Turning processes, like any single-point tool machining process, are automatically controlled via three independent variables, namely cutting speed (Vc [m/min]), feed (fx [mm/rev]) and depth of cut (dy [mm]). These variables modulate the process's performance parameters (dependent variables), such as, workpiece surface roughness (Ra [microns]), workpiece-tool vibration (Vy [mV]), cutting power (Pc [Watts]), tool temperature (Tt [^{0}C]) , cutting forces (Fx [N] as well as Fz [N]), spindle current [Is] and cutting sound (Sc [mV]). Appendix B contains experimental data for varying independent variables and the effect it has on the dependent variables.

Low-level adaptive force control has been successfully applied within machining [53, 54, 55, 56, 57]. The machining process is complex, and to maintain several output parameters at variable set points has proven unattainable since it implies a tremendously complicated, multi-input-multi-output control algorithm [108]. In Section 5.1 an advanced strategy to compliment adaptive control and to respond to changing system conditions, such as tool wear, in order to guarantee the reliability of machining process parameters, is proposed. The response includes a diagnostic scheme to decide intelligently which machine control action to perform. Typical machining situations include: if the tool wears and causes the allowable cutting power to exceed its limit, should the cutting speed, the depth of cut or the feed be changed in order to return to a reliable state of machining? If the surface finish of the workpiece is poor and unacceptable, which of the independent variables should be changed? The feed, speed or depth? If the movement between the cutting tool and workpiece vibrate excessively, what should be done to eliminate it? The goal of the strategy is to return the process to the best reliable state of machining. The execution of the strategy is formulated in a manner similar to one in which a human being would proceed [1]. Statistica [106], a statistical software tool, was used to perform multiple regression analysis on the experimental data (Table B1 (tool wear 0 mm) and B1 (tool wear 0.2 mm) in Appendix B) in order to obtain empirical relations relating input (independent) and output (dependent) parameters for a machining process model. Section 5.2 shows the non-linear equations used to model the machining process. Section 5.3 introduces a fuzzy relation used to represent the "knowledge base" of the diagnostic scheme. Section 5.4 shows a software simulation with graphical trending of the machining processes control and performance parameters, used to test the intelligent decision making component of the diagnostic strategy.

5.1 Basic Structure for Intelligent Diagnosis

Figure 5.1 shows a block diagram of the diagnostic scheme, and indicates how it is connected to the machining process simulation, with user interface and graphical display for testing purposes. The knowledge base of the intelligent diagnosis scheme is a fuzzy relation (concept explained in Section 2.5.2.2). The relation is in the form of a matrix that indicates the strength of connection ($\mu_{performence\_control}$), obtained from the experimental data, between performance of the process and control parameters. If there is no connection then $\mu_{performance\_control} = 0$, whereas, a strong connection indicate $\mu_{features\_control} = 1$. The execution of the decision-making process follows the following four steps.

(i) Determine control alternatives

The limit monitor determines when an on-line machining process parameter exceeds a machining process constraint. The machining constraint is set for a specific part being manufactured. If a constraint, say for example, Vy is exceeded, the limit monitor makes use of the fuzzy relation to determine the control alternatives.

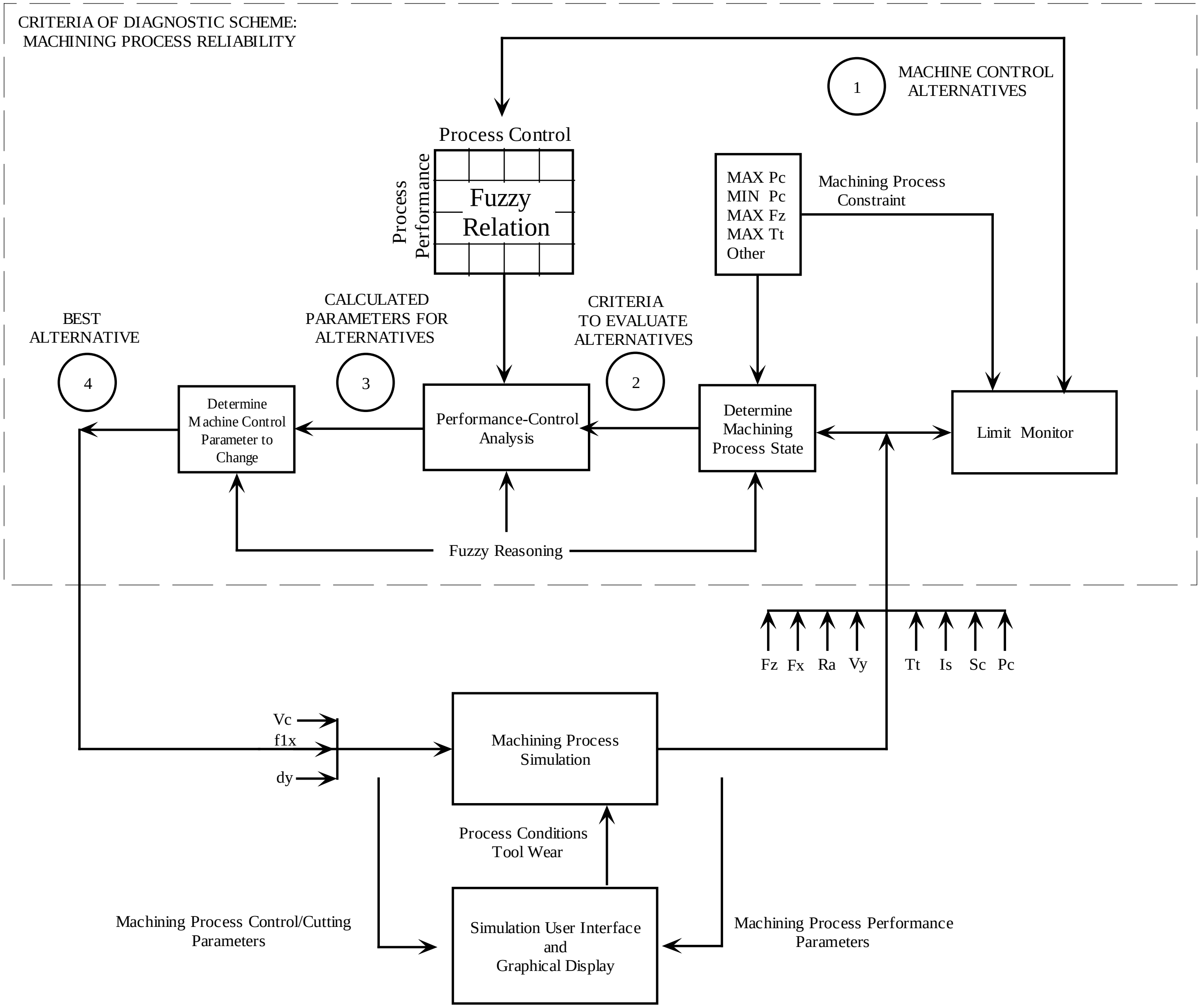

CRITERIA OF DIAGNOSTIC SCHEME:
MACHINING PROCESS RELIABILITY
1
MACHINE CONTROL ALTERNATIVES
Process Control
Process Performance
Fuzzy Relation
MAX Pc
MIN Pc
MAX Fz
MAX Tt
Other
Machining Process Constraint
BEST ALTERNATIVE
CALCULATED PARAMETERS FOR ALTERNATIVES
CRITERIA TO EVALUATE ALTERNATIVES
4
3
2
Determine Machine Control Parameter to Change
Performance-Control Analysis
Determine Machining Process State
Limit Monitor
Fuzzy Reasoning
Fz
Fx
Ra
Vy
Tt
Is
Sc
Pc
Vc
f1x
dy
Machining Process Simulation
Process Conditions
Tool Wear
Machining Process Control/Cutting Parameters
Simulation User Interface and Graphical Display
Machining Process Performance Parameters

Figure 5.1: A block diagram of the diagnostic scheme.

This is achieved by searching for a match of the exceeded machining parameter, “Vy” , within the row indexes of the fuzzy relation.

Control Alternatives
“f1x” and “Vc”

“Vy”

$$\begin{array}{c|ccc|}
 & dy & f1x & Vc \\
Fz & \mu_{Fz,dy} & \mu_{Fz,f1x} & \mu_{Fz,Vc} \\
Fx & \mu_{Fx,dy} & \mu_{Fx,f1x} & \mu_{Fx,Vc} \\
Ra & \mu_{Ra,dy} & \mu_{Ra,f1x} & \mu_{Ra,Vc} \\
Vy & \mu_{Vy,dy} & \mu_{Vy,f1x} & \mu_{Vy,Vc} \\
Tt & \mu_{Tt,dy} & \mu_{Tt,f1x} & \mu_{Tt,Vc} \\
Is & \mu_{Is,dy} & \mu_{Is,f1x} & \mu_{Is,Vc} \\
Sc & \mu_{Sc,dy} & \mu_{Sc,f1x} & \mu_{Sc,Vc} \\
Pc & \mu_{Pc,dy} & \mu_{Pc,f1x} & \mu_{Pc,Vc}
\end{array} \quad = \quad FR[i][j]$$

5.1

The connection strength ($\mu_{\text{performance\_control}}$) for each of the process control parameters is evaluated. If $\mu_{\text{performance\_control}} \neq 0$ the control parameter is selected as an alternative.

(ii) Obtain criteria to evaluate control alternatives

The state of the machining process, which is also the criteria that will be used to evaluate the alternatives, is determined by dividing the on-line process performance parameters with the process constraints.

SMP[i] = [Fz/Fc_C, Fx/Fx_C, Ra/Ra_C, Vz/Vz_C, Tt/Tt_C, Is /Is_C, Sc/Sc_C, Pc/Pc_C] 5.2

(iii) Calculate parameters for alternatives

To make a decision as to which process control parameter to change, one needs to perform process performance – control analysis. This is done by calculating the contribution each process state has on the control alternatives. The diagnosis is performed by:

Diagnostic[i][j]=SMP[i]*FR[i][j]=

$$\begin{array}{c} Fz \\ Fx \\ Ra \\ Vy \\ Tt \\ Is \\ Sc \\ Pc \end{array} \begin{bmatrix} \mu_{Fz,dy} * Fz/Fz\_C & \mu_{Fz,f1x} * Fz/Fz\_C & \mu_{Fz,Vc} * Fz/Fz\_C \\ \mu_{Fx,dy} * Fx/Fx\_C & \mu_{Fx,f1x} * Fx/Fx\_C & \mu_{Fx,Vc} * Fx/Fx\_C \\ \mu_{Ra,dy} * Ra/Ra\_C & \mu_{Ra,f1x} * Ra/Ra\_C & \mu_{Ra,Vc} * Ra/Ra\_C \\ \mu_{Vy,dy} * Vy/Vy\_C & \mu_{Vy,f1x} * Vy/Vy\_C & \mu_{Vy,Vc} * Vy/Vy\_C \\ \mu_{Tt,dy} * Tt/Tt\_C & \mu_{Tt,f1x} * Tt/Tt\_C & \mu_{Tt,Vc} * Tt/Tt\_C \\ \mu_{Is,dy} * Is/Is\_C & \mu_{Is,f1x} * Is/Is\_C & \mu_{Is,Vc} * Is/Is\_C \\ \mu_{Sc,dy} * Sc/Sc\_C & \mu_{Sc,f1x} * Sc/Sc\_C & \mu_{Sc,Vc} * Sc/Sc\_C \\ \mu_{Pc,dy} * Pc/Pc\_C & \mu_{Pc,f1x} * Pc/Pc\_C & \mu_{Pc,Vc} * Pc/Pc\_C \end{bmatrix} \quad 5.3$$

The elements within diagnostic[i][j] are certainty factors. Diagnostic[i][j], containing certainty factors, is used in an algorithm to determine which one of the independent variables should be changed, hence an intelligent decision.

(iv) Obtain best alternative

The certainty factor Control_Parameter_CNF_j, to decide which one of the control parameters to change, is found by averaging the sum of the maximum certainty factors.

$$Control\_Parameter\_CNF\_j = \frac{\sum_{i=0}^{N-1}\left(\coprod_{N}^{\max} S[i]*R[i][j]\right)}{N}$$

5.4

N is determined by finding the column with the most zeros, hence N = no of rows – no of zeros.

5.2 Machining Process Model

Statistica's multiple regression analysis module was used to obtain the non-linear equations 5.5 – 5.11 (machining process model). These equations relate the dependent with the independent variables. Appendix B contains the experimental data used in the regression analysis. A surface plot, contour plot and graph (to indicate measured versus model values for each of the non-linear equations) are shown in Figures 5.2 – 5.8.

$$S_C = 41.9171 d_y^{0.2648} V_C^{0.3085}$$

5.5

$$I_S = 11496.2272 \mathrm{d}_{\mathrm{y}}^{0.6239} f_x^{0.4444}$$

5.6

$$F_Z = (406.51 + 66.44 * \mathrm{Vb})\ f_X^{0.5305}\ d_Y^{0.7465}$$

5.7

$$T_t = 24.1969\, d_Y^{0.2389}\ f_X^{0.1095} V_C^{0.2450}$$

5.8

$$F_{X-W0} = 140.6187\ f_X^{0.4029}\ d_Y^{1.04256}$$

$$F_{X-W2mm} = 113.086\ f_X^{0.1525}\ d_Y^{0.8609}$$

5.9

$$V_{Y-W0} = 8.3239\ f_X^{0.4544} V_C^{0.6007}$$

$$V_{Y-W2mm} = 17.8941\ f_X^{0.2883} V_C^{0.4762}$$

5.10

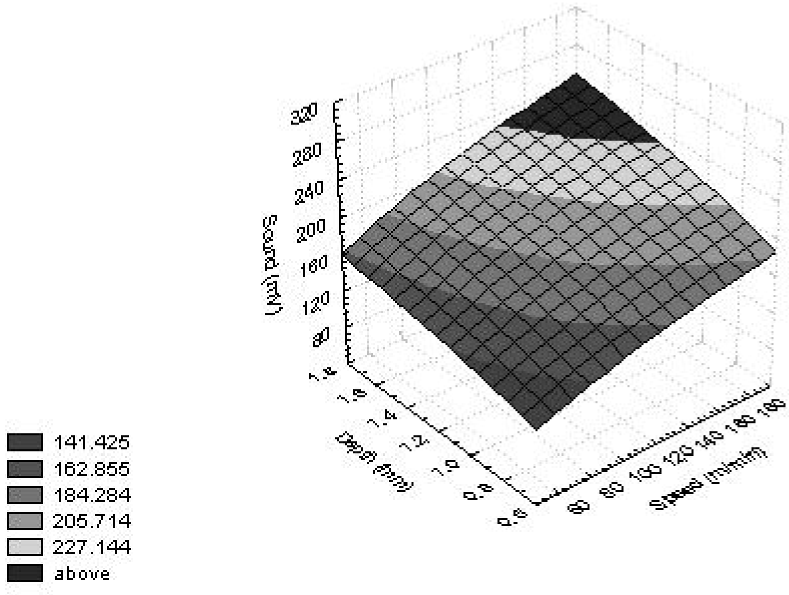
$$R_a = 53.0571\ f_X^{1.5866}\ d_y^{0.2568}$$

5.11

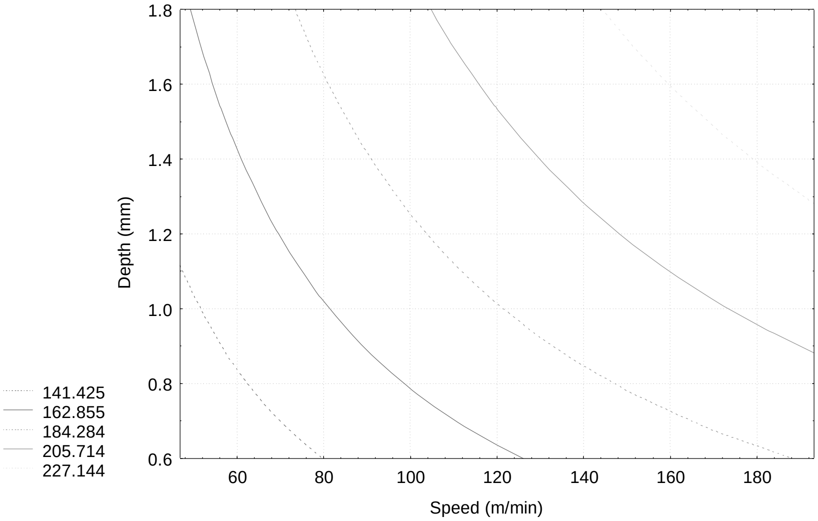
$$P_C = F_Z * V_C$$

5.12

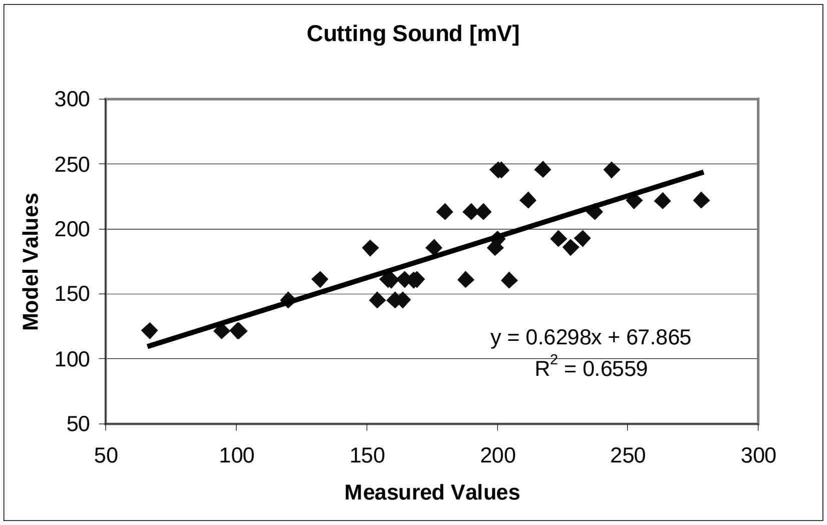
The equations include the process condition, tool wear, and are used in modeling the machining process as shown in Figure 5.1 for testing the intelligent diagnostic scheme.



(a) Surface plot.

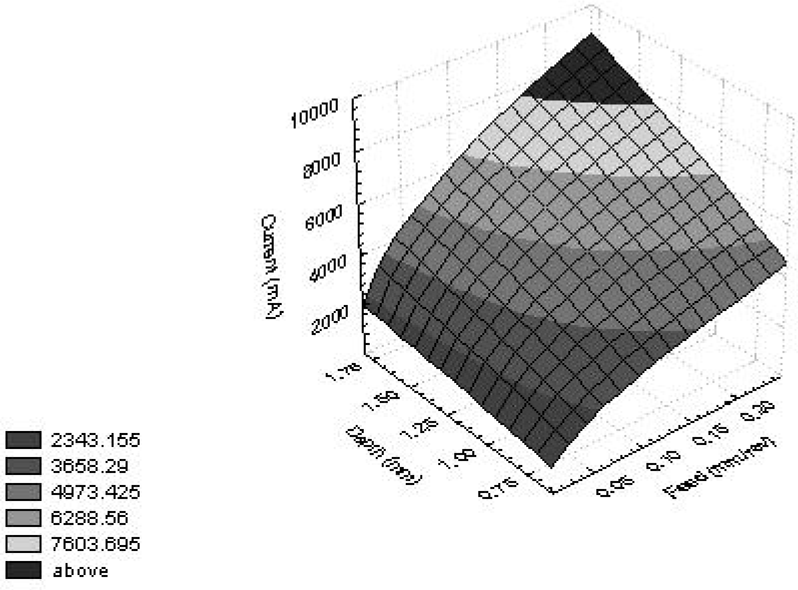


(b) Contour plot.

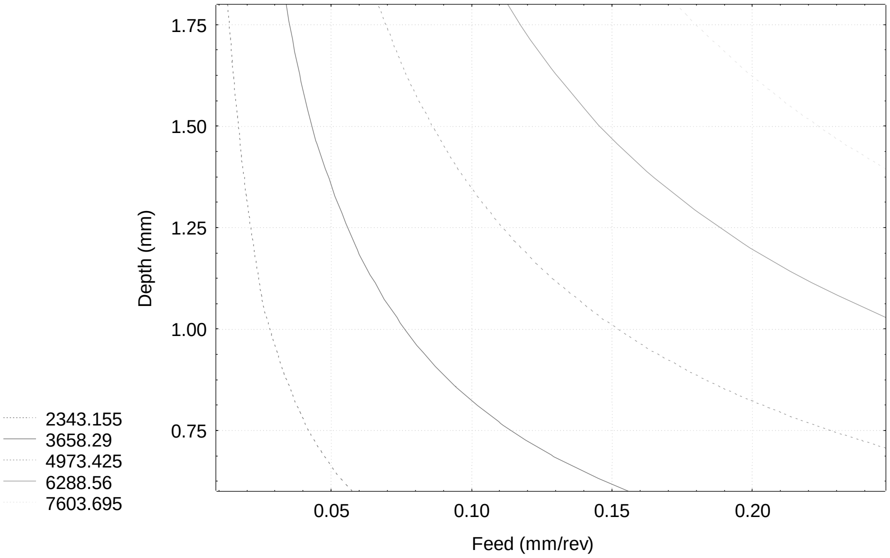


(c) Model versus measured values.

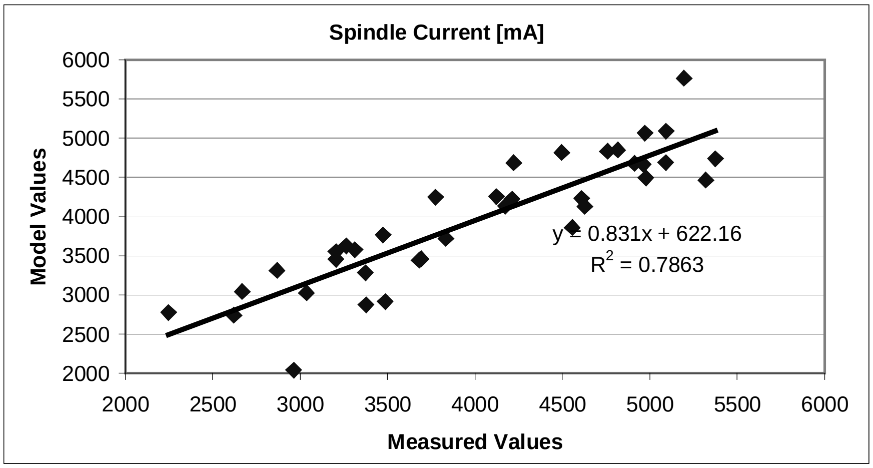
Figure 5.2: Surface plot, contour plot and a graph of model versus measured values for cutting sound.



(a) Surface plot.

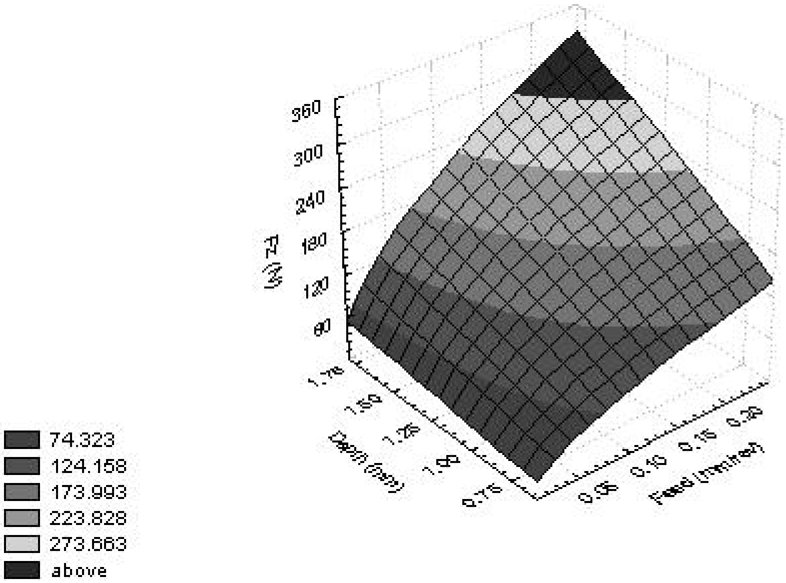


(b) Contour plot.

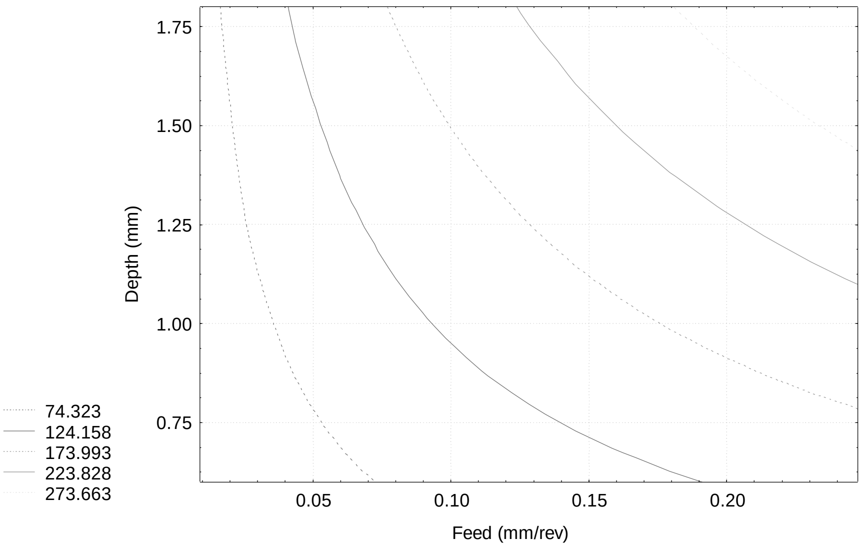


(c) Model versus measured values.

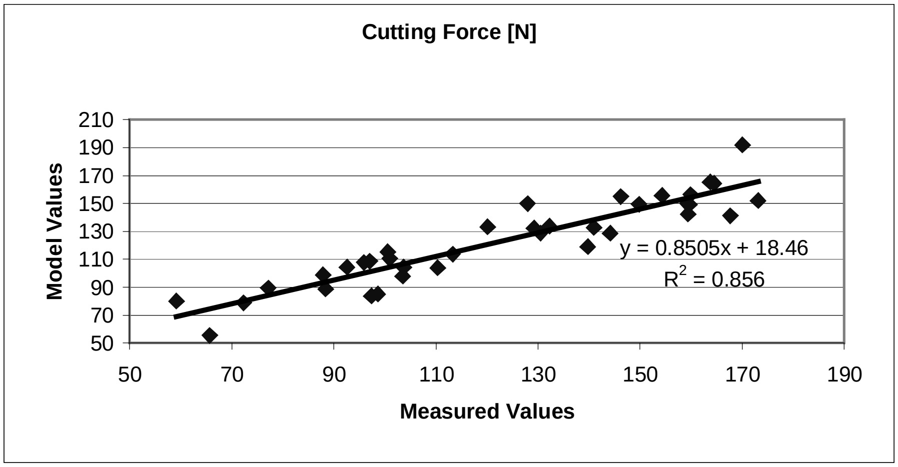
Figure 5.3: Surface plot, contour plot and a graph of model versus measured values for spindle current.



(a) Surface plot.

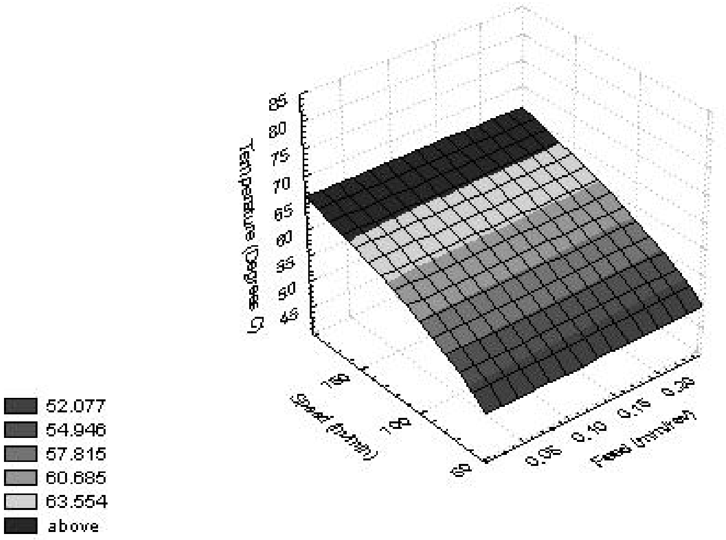


(b) Contour plot.

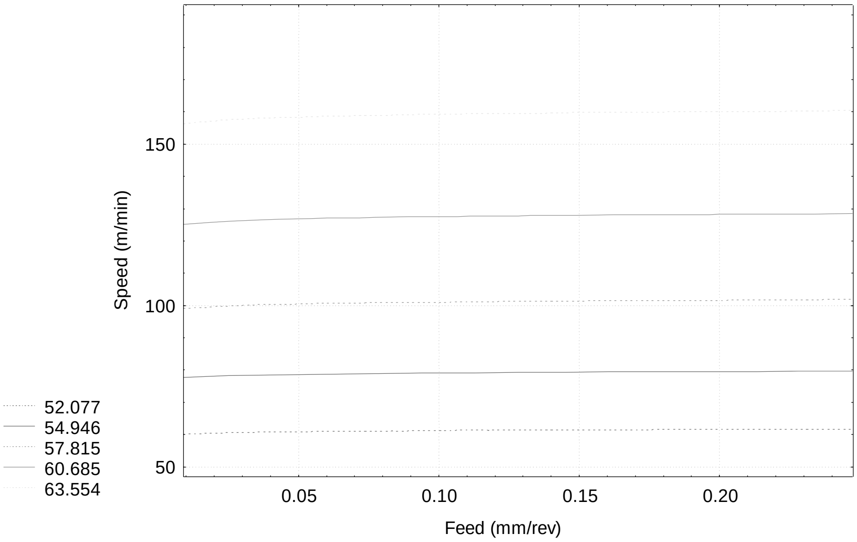


(c) Model versus measured values.

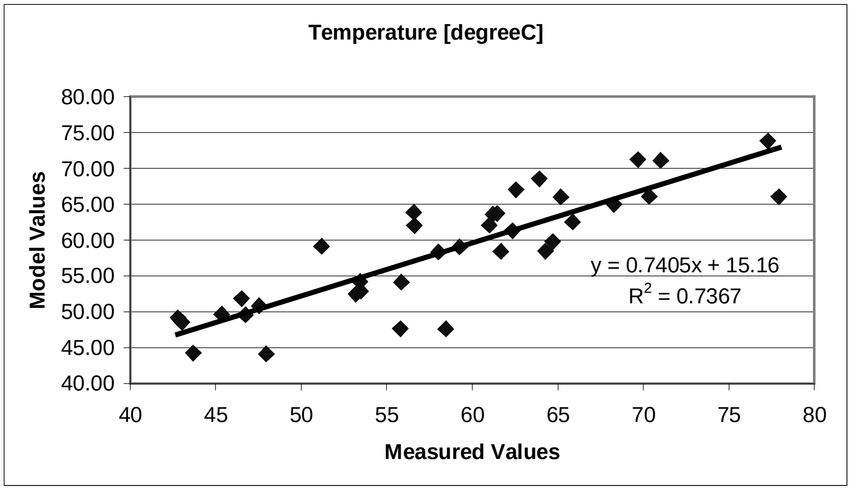
Figure 5.4: Surface plot, contour plot and a graph of model versus measured values for cutting force.



(a) Surface plot.

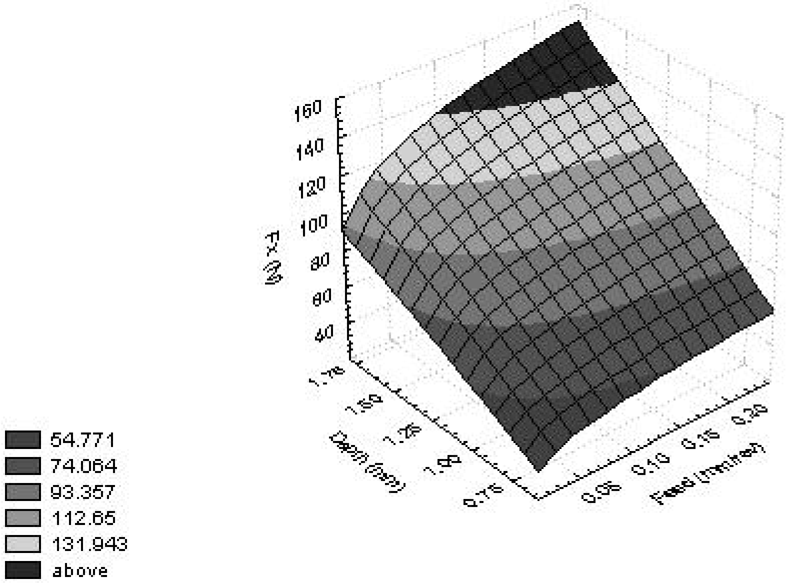


(b) Contour plot.

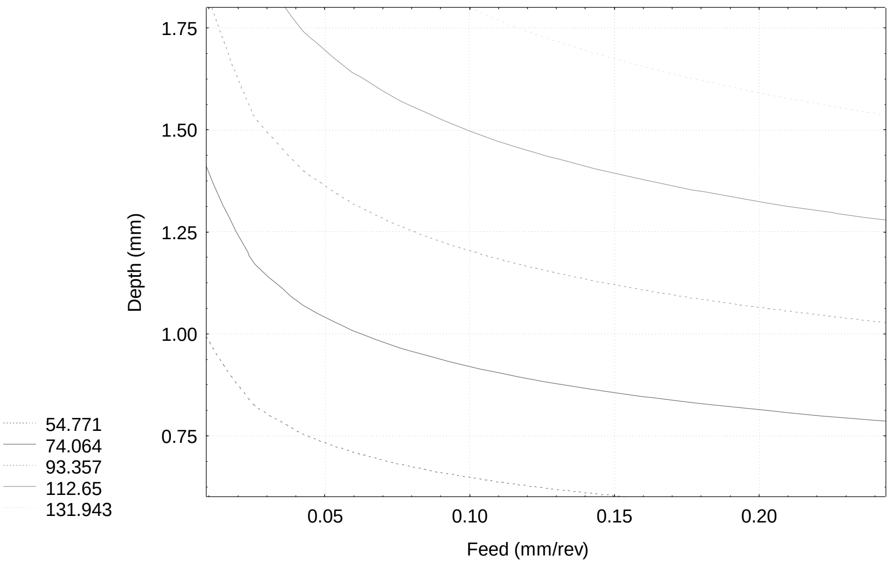


(c) Model versus measured values.

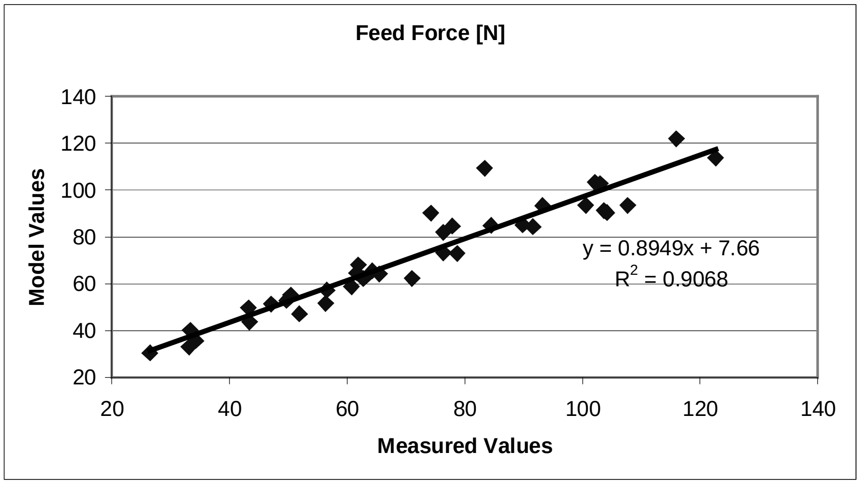
Figure 5.5: Surface plot, contour plot and a graph of model versus measured values for cutting temperature.



(a) Surface plot.

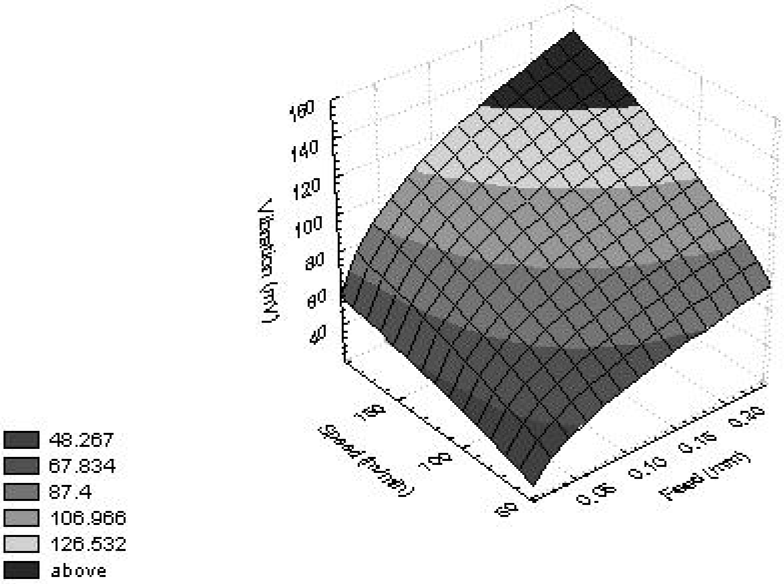


(b) Contour plot.

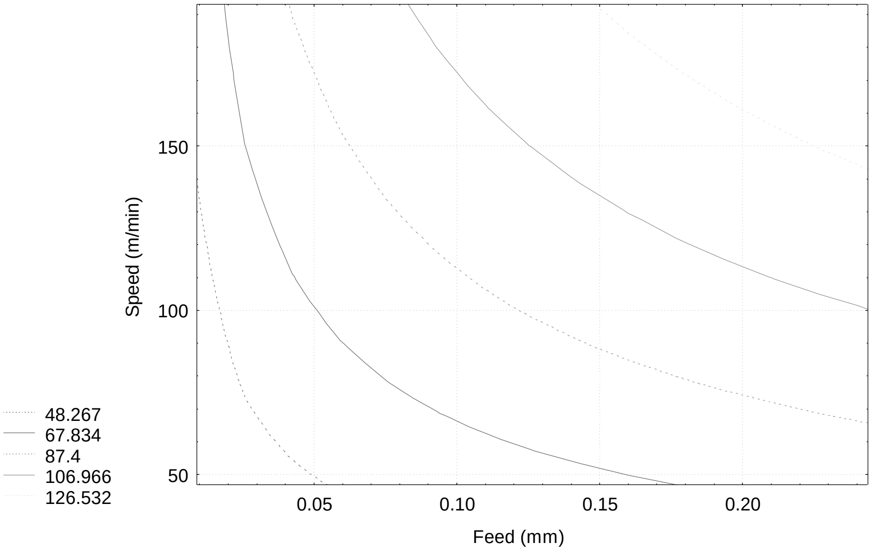


(c) Model versus measured values.

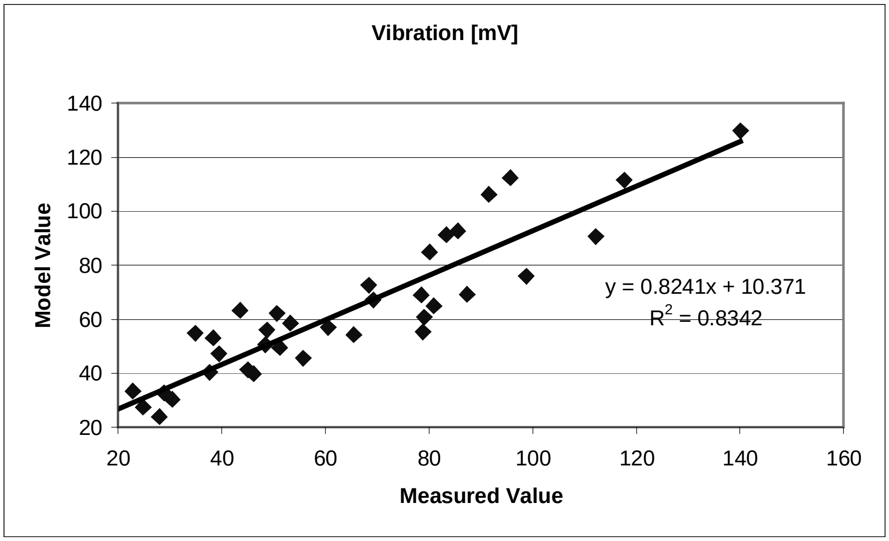
Figure 5.6: Surface plot, contour plot and a graph of model versus measured values for feed force.



(a) Surface plot.

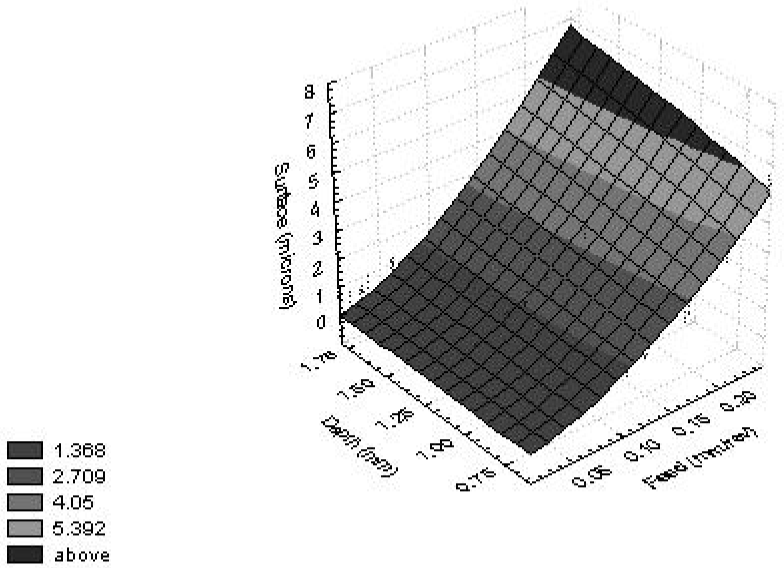


(b) Contour plot.

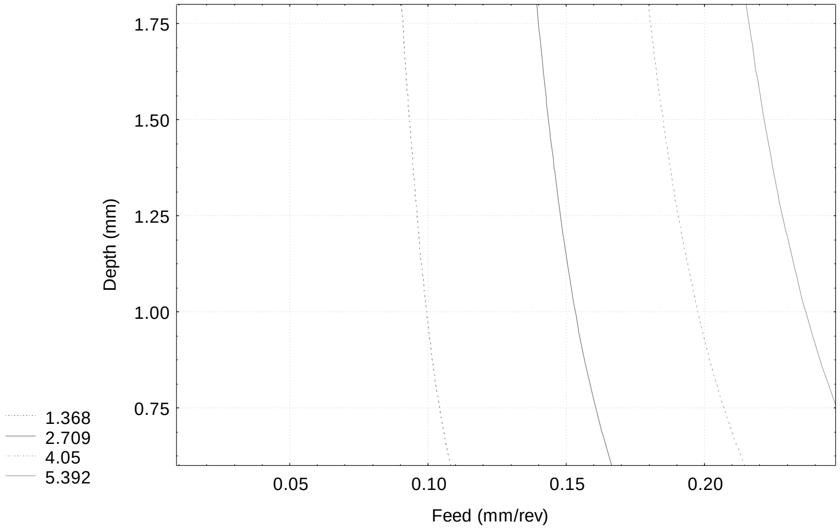


(c) Model versus measured values.

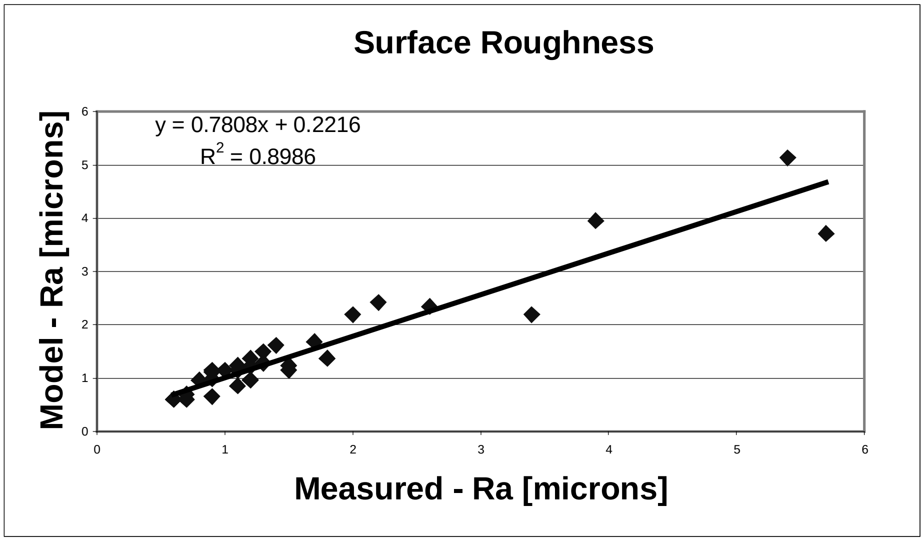
Figure 5.7: Surface plot, contour plot and a graph of model versus measured values for vibration.



(a) Surface plot.



11.1.3.1 (b) Contour plot.



(c) Model versus measured values.

Figure 5.8: Surface and contour plots and measured versus model values for surface roughness.

5.3 Regression Analysis for Fuzzy Relation

Statistica's multiple regression analysis module was used to obtain BETA coefficients, shown in Table 5.1, for linear equations written in the form (equation 2.19):

$$\frac{Y}{s_Y} = \alpha + \beta_1 \frac{X_1}{s_{X_1}} + \beta_2 \frac{X_2}{s_{X_2}}$$

The equation relate the dependent (Y) with the independent (X_1, X_2) variables. Appendix B contains the experimental data used in the linear regression analysis.

Table 5.1: Linear regression summary of BETA coefficients used to relate independent and dependent variables.

Dependent Variable	R^2	11.1.3.2 BETA	p-value

Tool Temperature	0.73910088	DEPTH	0.796169	**7.3E-06**
		FEED	0.701622	**5.91E-05**
		SPEED	0.868919	**2.52E-10**
Power in Cut	0.81068678	DEPTH	0.629604	**1.53E-05**
		FEED	0.691316	**4.73E-06**
		SPEED	0.946443	**2.46E-13**
Tool-Workpiece Vibration	0.53125349	DEPTH	0.177927	0.369661
		FEED	0.648731	**0.002611**
		SPEED	0.652823	**1.18E-05**
Spindle Current	0.64264508	DEPTH	1.011078	**1.22E-06**
		FEED	1.134478	**2.09E-07**
		SPEED	-0.15833	0.161893
Cutting Force	0.68728082	DEPTH	1.025118	**2.85E-07**
		FEED	1.154627	**3.99E-08**
		SPEED	-0.18772	0.078883
Feed Force	0.55165043	DEPTH	0.884785	**5.55E-05**
		FEED	0.202535	0.306088
		SPEED	-0.02736	0.826679
Cutting Sound	0.68512988	DEPTH	0.702251	**0.000293**
		FEED	0.283591	0.116261
		SPEED	0.734558	**1.29E-07**
Surface Finish	0.88982711	DEPTH	0.370669	**0.000436041**
		FEED	1.198262	**5.63796E-14**
		SPEED	-0.00445	0.942749918

From Equation 2.20 the ß$_i$ coefficients are equal to [32]:

$$\beta_i = b_i \frac{s_{X_i}}{s_Y}$$

ß$_i$ measures the number of standard deviations for changes in Y, with each change of one standard deviation in X$_i$. The advantage of using BETA coefficients is that it allows one to compare the relative contribution of each independent variable in predicting the dependent variable. Furthermore, to determine the significance of each independent variable, the hypothesis tested:

Ho: $\beta_i = 0$

H_A: $\beta_i \neq 0$ (alternate hypothesis)

A p-value indicates the truth of the hypothesis. A p-value of less than 0.05 gives the probability of Ho to be correct, else accept the alternative. Table 5.1 indicates the R^2 value for each equation. It is known as the coefficient of determination, a ratio of the explained variation to the total variation (equation 2.16 and 2.17). In other words, of the total variation measured in the dependent variable, R^2 indicate the percentage attributed to the independent variables contained in the equation. This uncertainty is included when calculating the connection strength coefficients for the fuzzy relation, FR[i][j]: $\mu_{performance\_control} = R^2$ * BETA. For example: the connection strength for μ_{Tt_dy} = 0.7391 * 0.7961 = 0.5884. The completed fuzzy relation:

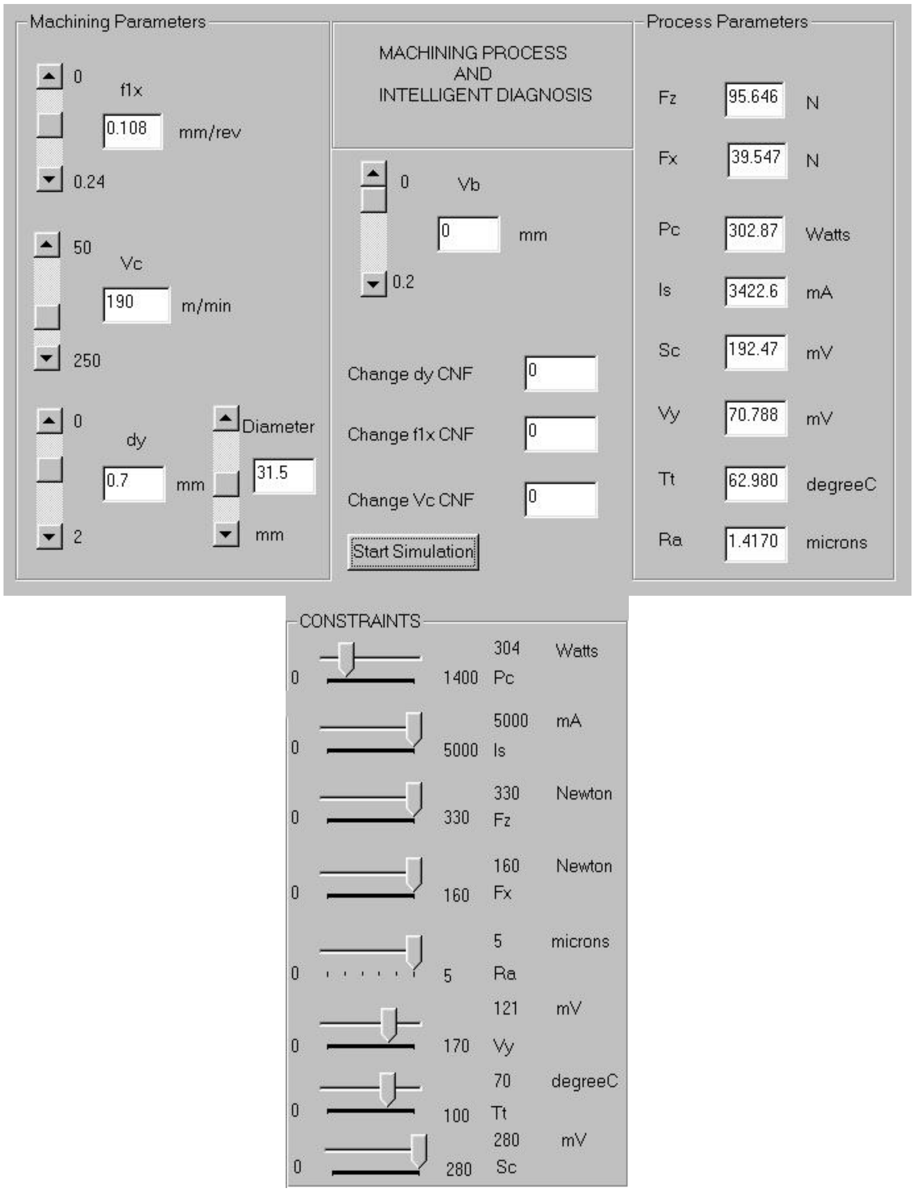
$$FR_{[i][j]} = \begin{array}{c|ccc|} & dy & f1x & Vc \\ Fz & 0.7045 & 0.7936 & 0.0 \\ Fx & 0.4881 & 0.0 & 0.0 \\ Ra & 0.3298 & 1.0662 & 0.0 \\ Vy & 0.0 & 0.3446 & 0.3468 \\ Tt & 0.5884 & 0.5186 & 0.6422 \\ Is & 0.6498 & 0.7290 & 0.0 \\ Sc & 0.4811 & 0.0 & 0.5033 \\ Pc & 0.5104 & 0.5604 & 0.7673 \end{array}$$

5.13

The elements in the relation FR[i][j] represent the influence which the independent variables have on the dependent variables. The elements represent process knowledge based on statistical modeling and may be used to decide, in collaboration with the on-line sensor values, which one of the independent variables dy, fx, Vc to change in order to maintain system constraints.

5.4 Process Simulation

Figure 5.9 shows the machining process simulation user interface. The machining control parameters (independent) are set, whereupon Equations 5.5 to 5.12 calculate the process performance parameters (dependent). Process constraints for the part being manufactured, are set. Tool wear, Vb, may now be increased, which may result in a process performance parameter exceeding a set constraint. The diagnostic scheme will decide intelligently, which one of the three machining control parameters to change. This is indicated by the highest certainty factor.



MACHINING PROCESS
AND
INTELLIGENT DIAGNOSIS

Machining Parameters

f1x 0.108 mm/rev (0 – 0.24)

Vc 190 m/min (50 – 250)

dy 0.7 mm (0 – 2)

Diameter 31.5 mm

Vb 0 mm (0 – 0.2)

Change dy CNF 0

Change f1x CNF 0

Change Vc CNF 0

Process Parameters

Fz 95.646 N

Fx 39.547 N

Pc 302.87 Watts

Is 3422.6 mA

Sc 192.47 mV

Vy 70.788 mV

Tt 62.980 degreeC

Ra 1.4170 microns

CONSTRAINTS

Pc 304 Watts (0 – 1400)

Is 5000 mA (0 – 5000)

Fz 330 Newton (0 – 330)

Fx 160 Newton (0 – 160)

Ra 5 microns (0 – 5)

Vy 121 mV (0 – 170)

Tt 70 degreeC (0 – 100)

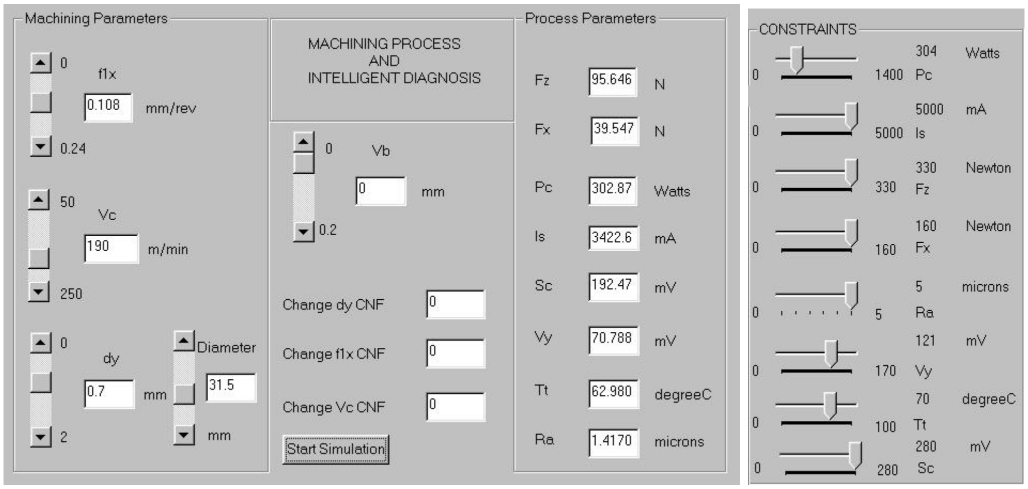
Sc 280 mV (0 – 280)

Figure 5.9: Machining process simulation user interface.

Test cases in the following subsections make use of different input cutting parameters and impose varying process constraints. In the first six test cases the tool wear is increased and causes Pc or Fz to exceed its limit. In the final test case the Pc lower limit is reached when decreasing the depth of cut.

5.4.1 Test Case 1: Pc exceeded with Tt and Vy constraints

Figure 5.10 shows set process parameter constraints imposed whilst machining a part. The power in the cut is limited to 304 Watts, the tool temperature to 70^{o}C and vibration level to 121 mV. The latter constraints are set to ensure accurate part tolerance.



Machining Parameters

f1x: 0.108 mm/rev (0 – 0.24)

Vc: 190 m/min (50 – 250)

dy: 0.7 mm (0 – 2)

Diameter: 31.5 mm

MACHINING PROCESS AND INTELLIGENT DIAGNOSIS

Vb: 0 mm (0 – 0.2)

Change dy CNF: 0

Change f1x CNF: 0

Change Vc CNF: 0

Start Simulation

Process Parameters

Fz	95.646	N
Fx	39.547	N
Pc	302.87	Watts
Is	3422.6	mA
Sc	192.47	mV
Vy	70.788	mV
Tt	62.980	degreeC
Ra	1.4170	microns

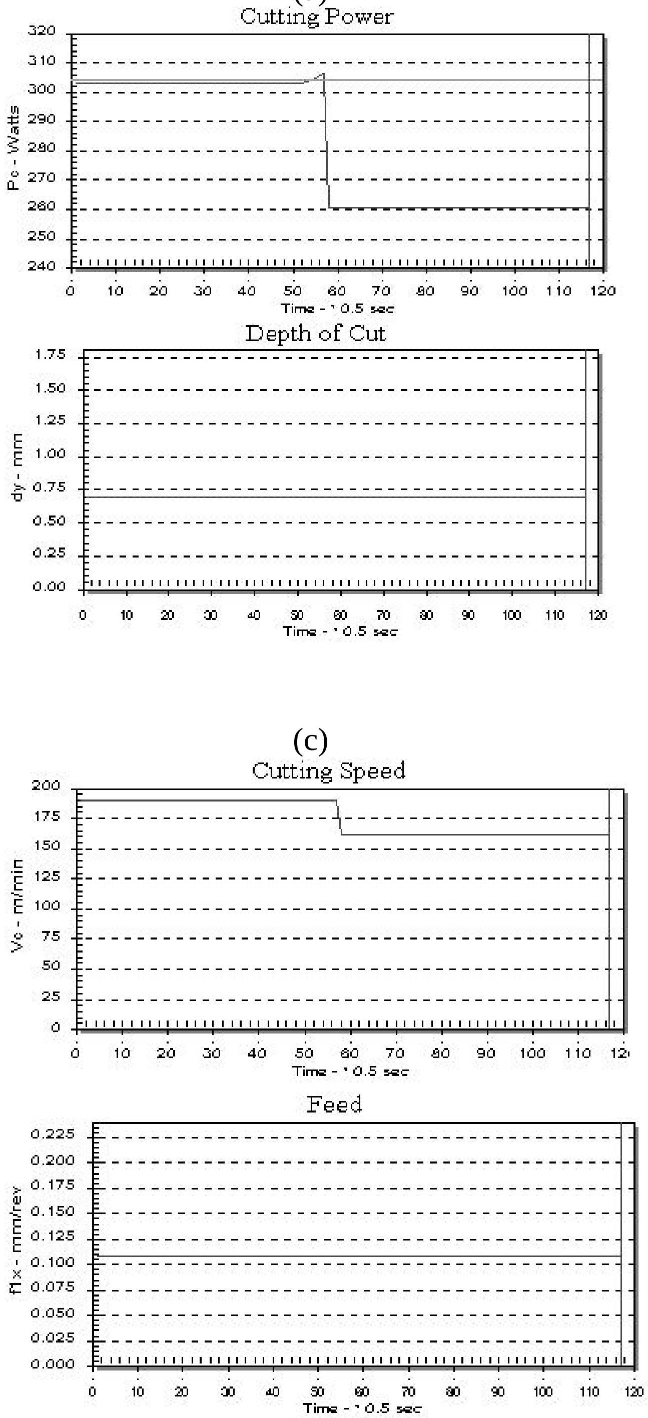
CONSTRAINTS

Pc	304	Watts	0	1400
Is	5000	mA	0	5000
Fz	330	Newton	0	330
Fx	160	Newton	0	160
Ra	5	microns	0	5
Vy	121	mV	0	170
Tt	70	degreeC	0	100
Sc	280	mV	0	280

Figure 5.10: Simulation user interface with Pc, Tt and Vy constraints.

Machining control parameters are set so that Pc = 302.87 Watts, close to its limit. If the tool wears it will increase Fz, which in turn causes Pc to exceed its power limit. The intelligent diagnostic scheme needs to decide which one of Vc, f1x or dy to change. Figure 5.11 shows the graphical simulation of this process. From inspection we find that Vc is already high (c), the cutting tool temperature is close to its limit (e) and the tool-workpiece vibration is not too far from its limit (f). Therefore, knowing that Vc influences Pc, Tt and Vy, (more that dy and fx1) a machining expert may suggest that Vc should be decreased (intelligent decision).

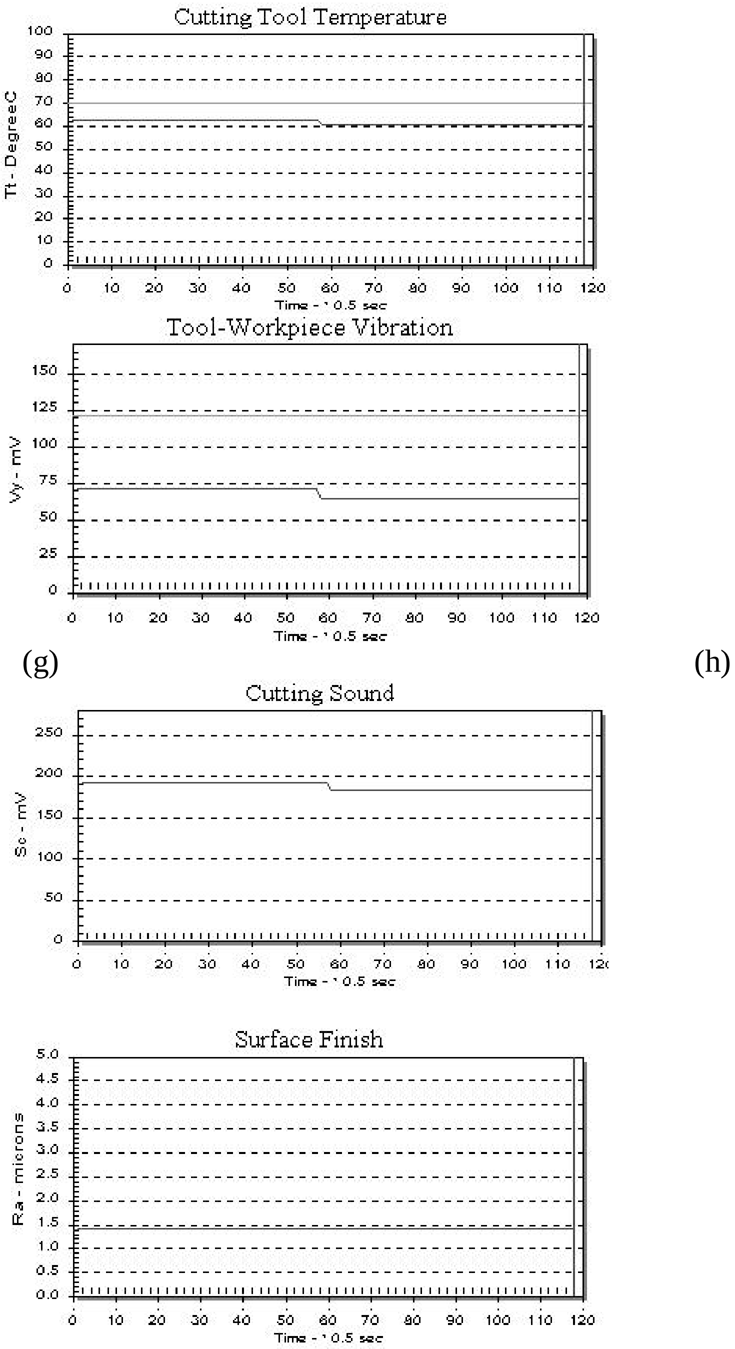
(a) (b)



Cutting Power
Pc - Watts
320
310
300
290
280
270
260
250
240
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec
Depth of Cut
dy - mm
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec
Cutting Speed
Vc - m/min
200
175
150
125
100
75
50
25
0
0 10 20 30 40 50 60 70 80 90 100 110 12
Time - * 0.5 sec
Feed
f1x - mm/rev
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec

(c) (d)

(e) (f)


Cutting Tool Temperature
Tt - DegreeC
0
10
20
30
40
50
60
70
80
90
100
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec
Tool-Workpiece Vibration
Vy - mV
0
25
50
75
100
125
150
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec
(g)
(h)
Cutting Sound
Sc - mV
0
50
100
150
200
250
0 10 20 30 40 50 60 70 80 90 100 110 12
Time - * 0.5 sec
Surface Finish
Ra - microns
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec

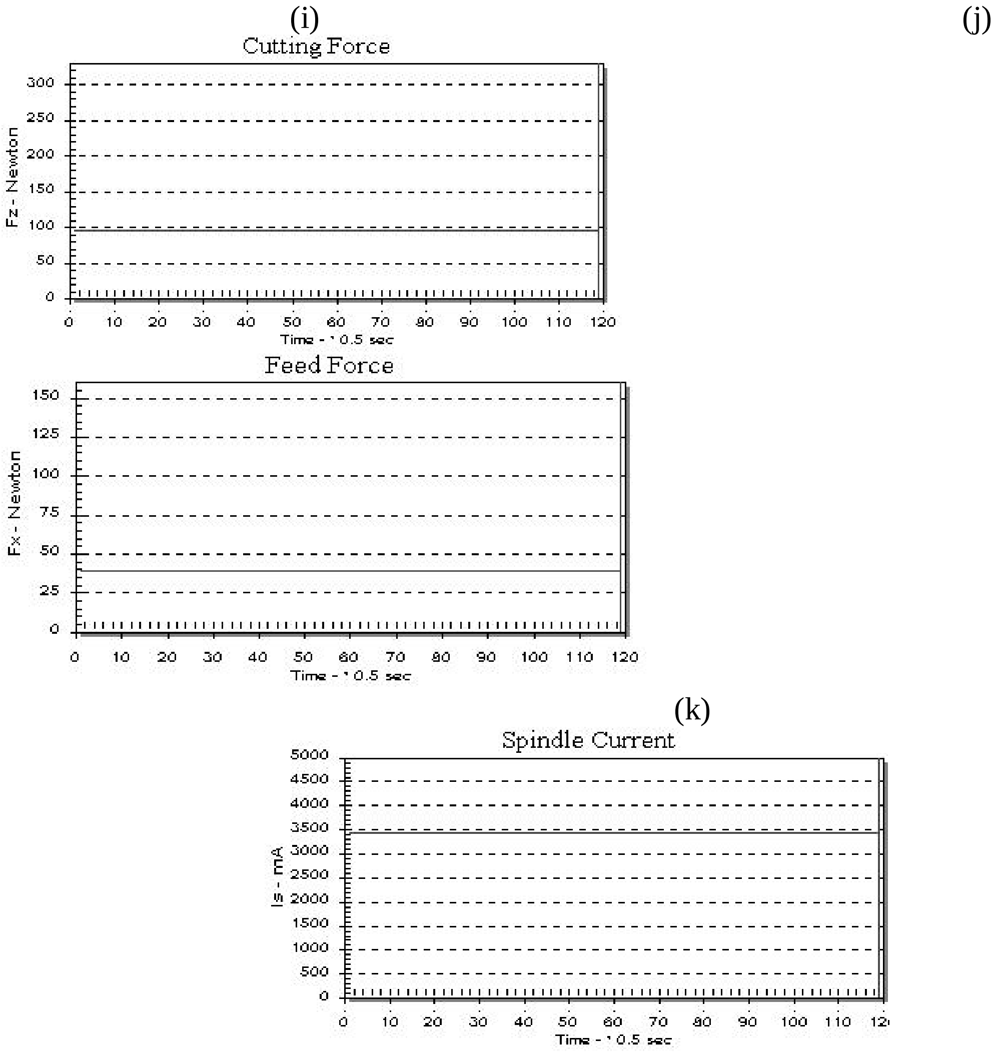


Figure 5.11: Graphical simulation with Pc, Tt and Vy constraints.

Applying Equation 5.2 the machining state is determined from the sampled signals:

SMP[8]={95.646/330, 39.547/160, 1.417/5, 70.788/121, 62.98/70, 3422.6/5000, 192.47/280, 304/304 }

Multiplying SMP[8] with Equation 5.13, result in Diagnostic[8][3]:

$$\text{Diagnostic[8][3] = SMP[8]* FR[8][3] =} \begin{array}{c} Fz \\ Fx \\ Ra \\ Vy \\ Tt \\ Is \\ Sc \\ Pc \end{array} \begin{bmatrix} 0.2041 & 0.2299 & 0.0 \\ 0.1207 & 0.0 & 0.0 \\ 0.0935 & 0.3022 & 0.0 \\ 0.0 & 0.2016 & 0.2029 \\ 0.5293 & 0.4667 & 0.578 \\ 0.4448 & 0.499 & 0.0 \\ 0.3308 & 0.0 & 0.3459 \\ 0.5103 & 0.5604 & 0.7674 \end{bmatrix}$$

Equation 5.4 is applied to determine the certainty factors:

CNF_1 = Change_dy_CNF = (0.5293 +0.4448+0.3308+0.5103)/4 = 0.4538

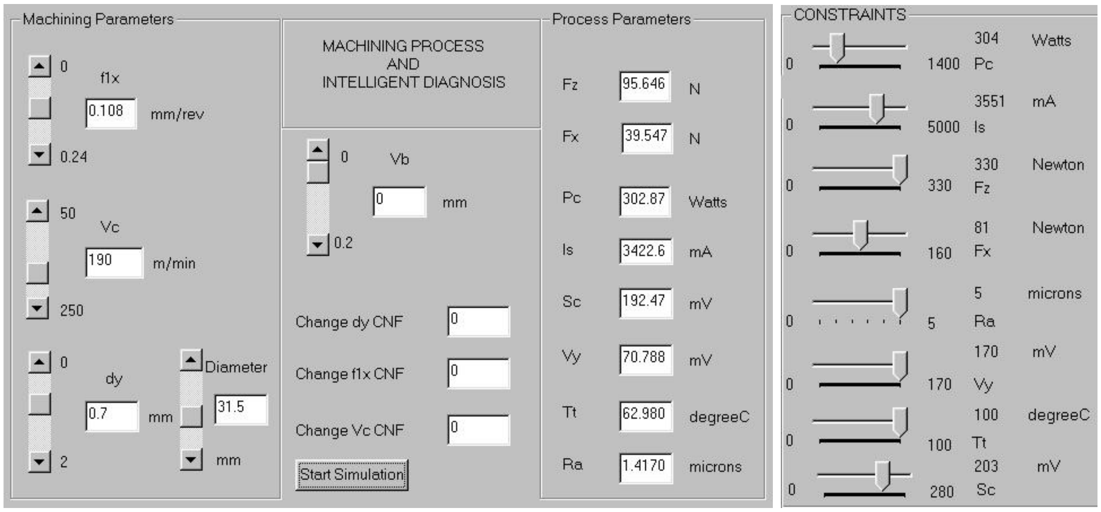
CNF_2 = Change_f1x_CNF = (0.3022+0.4667+0.499+0.5604)/4 = 0.4570

CNF_3 = Change_Vc_CNF = (0.2029+0.578+0.3459+0.7673)/4=0.4735

The strategy concludes that the cutting speed should decrease. The amount it decreases depend on the size of the certainty factor (low), therefore, Vc = Vc*0.85 = 161.5 m/min. The value of Vc is adjusted as shown (c) and results in a decrease in Pc (a), Tt (e) and Vy (f). The decision leaves the system in a more reliable state, as envisaged by the human expertise.

5.4.2 Test Case 2: Pc exceeded with Is, Fx and Sc constraints

Figure 5.12 show set process parameter constraints imposed whilst machining a part. The power in the cut is limited to 304 Watts, the spindle current to 3551 mA, the cutting sound to 203 mV and the feed force to 81 Newton. The latter constraints are set to ensure part surface integrity.



Machining Parameters

f1x: 0.108 mm/rev (0 – 0.24)

Vc: 190 m/min (50 – 250)

dy: 0.7 mm (0 – 2)

Diameter: 31.5 mm

MACHINING PROCESS
AND
INTELLIGENT DIAGNOSIS

Vb: 0 mm (0 – 0.2)

Change dy CNF: 0

Change f1x CNF: 0

Change Vc CNF: 0

Start Simulation

Process Parameters

Fz	95.646	N
Fx	39.547	N
Pc	302.87	Watts
Is	3422.6	mA
Sc	192.47	mV
Vy	70.788	mV
Tt	62.980	degreeC
Ra	1.4170	microns

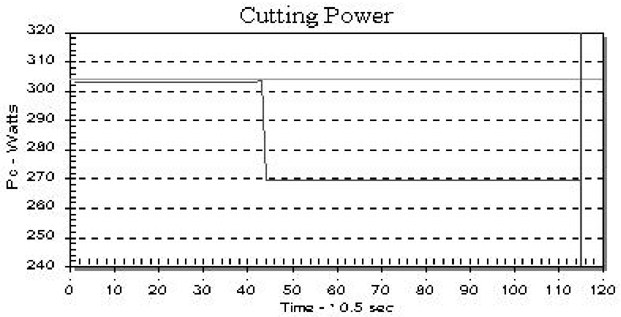
CONSTRAINTS

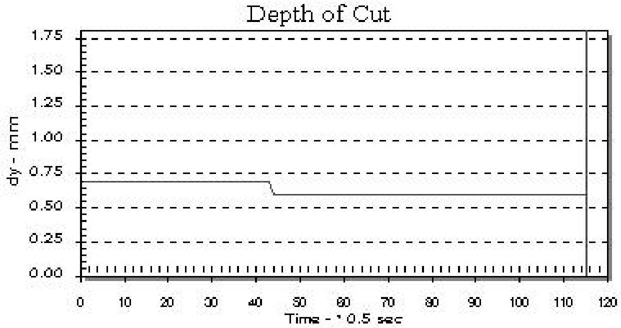
0	1400	Pc	304	Watts
0	5000	Is	3551	mA
0	330	Fz	330	Newton
0	160	Fx	81	Newton
0	5	Ra	5	microns
0	170	Vy	170	mV
0	100	Tt	100	degreeC
0	280	Sc	203	mV

Figure 5.12: Simulation user interface with Is, Fx and Sc constraints.

Machining control parameters are set (same as in test case no 1) so that Pc = 302.87 Watts, close to its limit. If the tool wears it will increase Fz, which in turn causes Pc to exceed its power limit. The intelligent diagnostic scheme needs to decide which one of Vc, f1x or dy to change. Figure 5.13 shows the graphical simulation of this process. From inspection we find that the cutting sound is close to its limit (g), the feed force is not far from its limit (j) and the spindle current is close to its limit (k). Therefore, knowing that dy influences Sc, Is and Fx a machining expert may suggest that dy should be decreased.

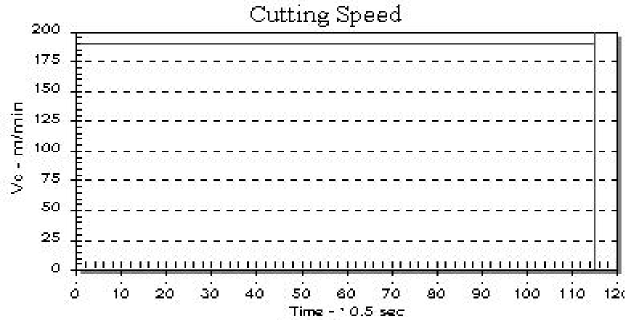
(a) (b)

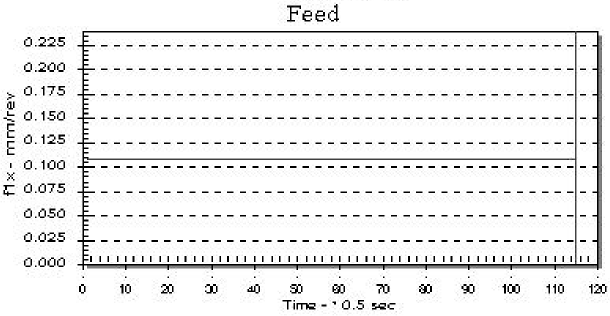

Cutting Power
Pc - Watts
320
310
300
290
280
270
260
250
240
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec


Depth of Cut
dy - mm
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

(c)

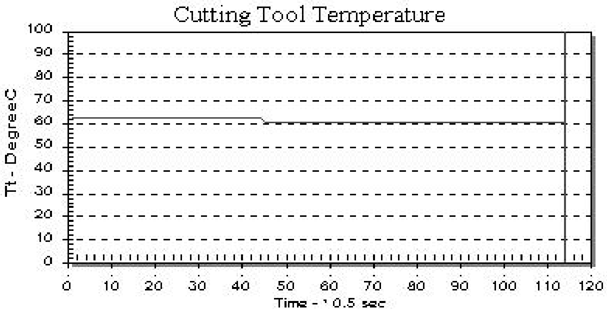
(d)

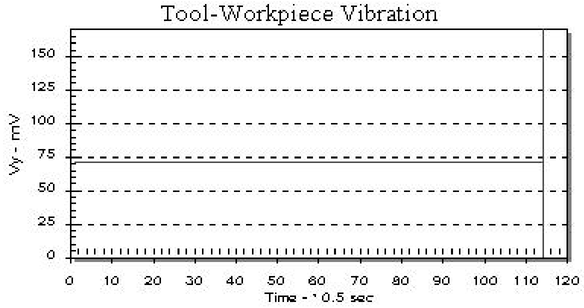

Cutting Speed
Vc - m/min
200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec


Feed
f1x - mm/rev
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

(e)

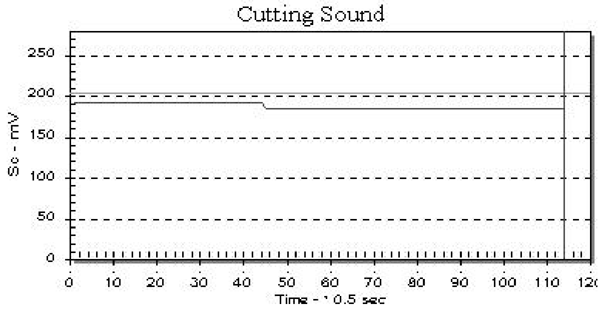
(f)

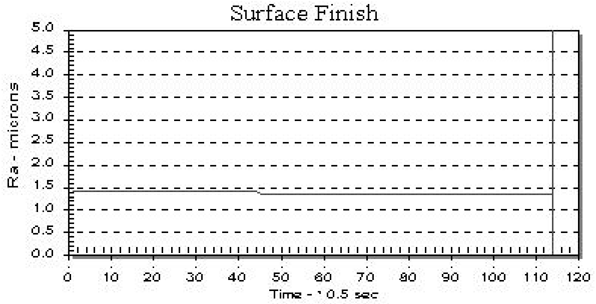

Cutting Tool Temperature
Tt - DegreeC
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec


Tool-Workpiece Vibration
Vy - mV
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

(g)

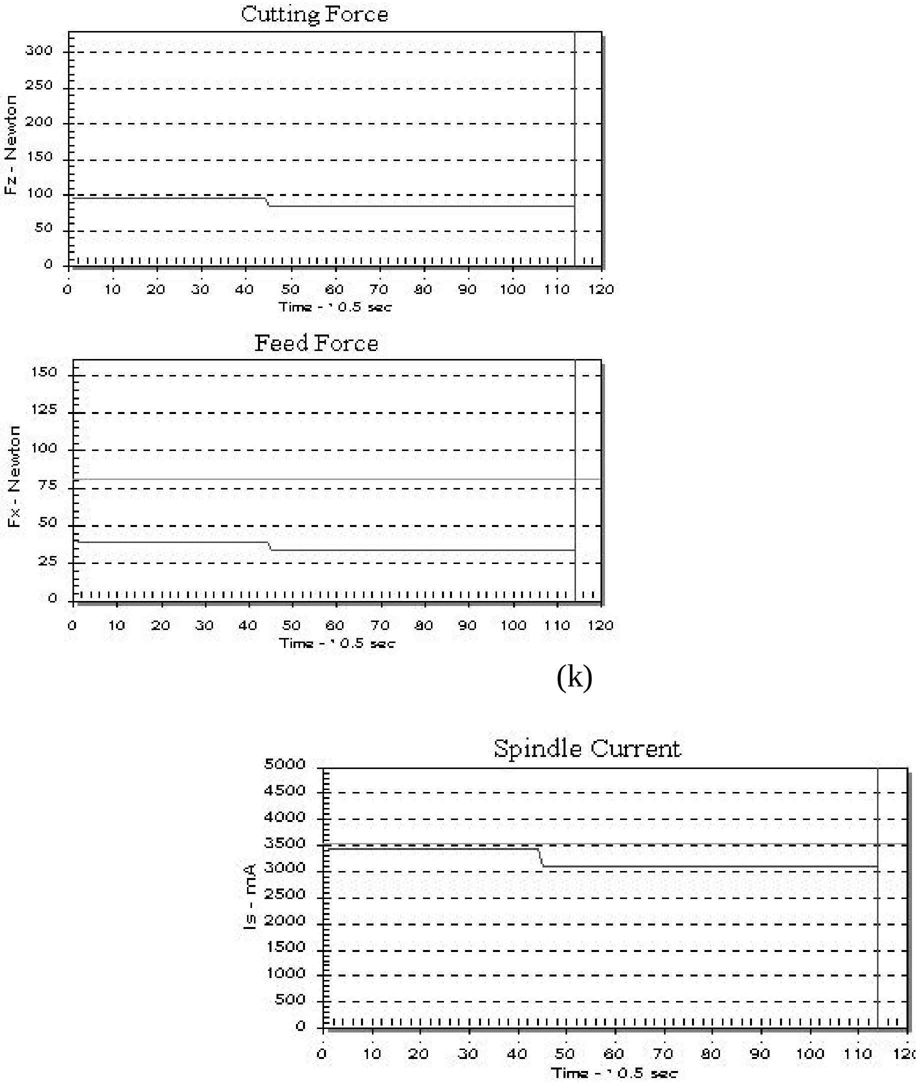
(h)


Cutting Sound
Sc - mV
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80
90
100
110
12(
Time - * 0.5 sec


Surface Finish
Ra - microns
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

(i)

(j)



(k)

Figure 5.13: Graphical simulation with Is, Fx and Sc constraints.

Applying Equation 5.2 machining state is determined from the sampled signals:

SMP[8] = { 95.646/330, 39.547/81, 1.417/5, 70.788/170, 62.98/100, 3422.6/3551, 192.47/203, 304/304}

Multiplying SMP[8] with Equation 5.13, result in Diagnostic[8][3]:

$$\text{Diagnostic[8][3]} = \text{SMP[8]} * \ \text{FR[8][3]} = \begin{array}{c} Fz \\ Fx \\ Ra \\ Vy \\ Tt \\ Is \\ Sc \\ Pc \end{array} \begin{bmatrix} 0.2041 & 0.2299 & 0.0 \\ 0.2383 & 0.0 & 0.0 \\ 0.0935 & 0.3022 & 0.0 \\ 0.0 & 0.1435 & 0.1444 \\ 0.3705 & 0.3266 & 0.4046 \\ 0.6263 & 0.7026 & 0.0 \\ 0.4561 & 0.0 & 0.4772 \\ 0.5103 & 0.5604 & 0.7674 \end{bmatrix}$$

Equation 5.4 is applied to determine the certainty factors:

CNF_1 = Change_dy_CNF = (0.6263+0.3705+0.4561+0.5103)/4 = 0.4908

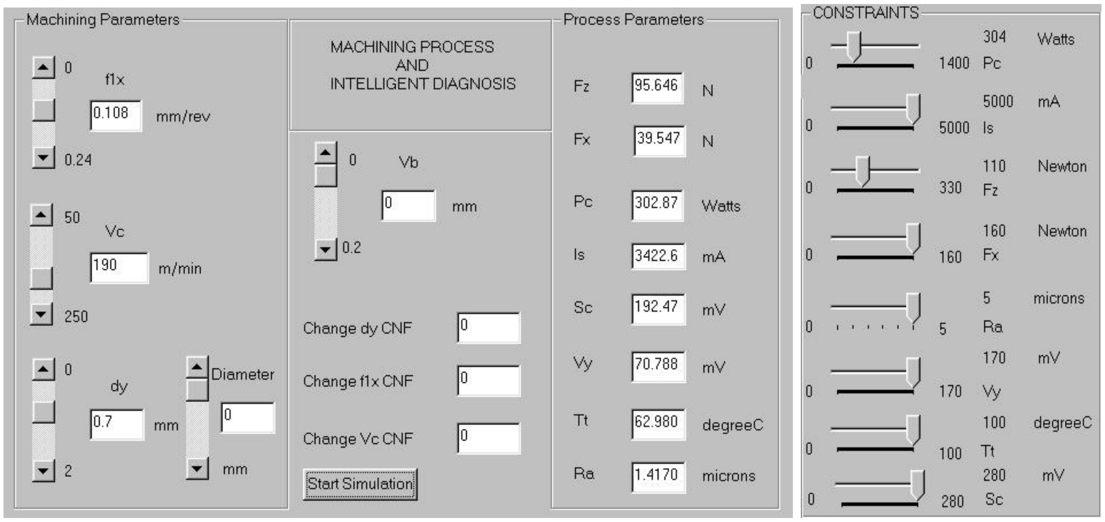
CNF_2 = Change_f1x_CNF = (0.7025+0.3266+0.3022+0.5604)/4=0.4729

CNF_3 = Change_Vc_CNF = (0.7674+0.4772+0.4046+0.1444)= 0.4484

The inference strategy concludes that the depth of cut should be decreased. The amount it decreases, depends on the size of the certainty factor (low), dy = dy * 0.85 = 0.595 mm. Once the value is determined the value of dy is adjusted as shown in Figure 5.13 (b) and result in a decrease in Pc (a), Is (k) and Sc (g). Again the decision leaves the system in a more reliable state.

5.4.3 Test Case 3: Pc exceeded with Fz constraint

Figure 5.14 show set process parameter constraints imposed whilst machining a part.



Machining Parameters

f1x: 0.108 mm/rev (0 – 0.24)

Vc: 190 m/min (50 – 250)

dy: 0.7 mm (0 – 2)

Diameter: 0 mm

MACHINING PROCESS
AND
INTELLIGENT DIAGNOSIS

Vb: 0 mm (0 – 0.2)

Change dy CNF: 0

Change f1x CNF: 0

Change Vc CNF: 0

Process Parameters

Fz: 95.646 N

Fx: 39.547 N

Pc: 302.87 Watts

Is: 3422.6 mA

Sc: 192.47 mV

Vy: 70.788 mV

Tt: 62.980 degreeC

Ra: 1.4170 microns

CONSTRAINTS

Pc: 304 Watts (0 – 1400)

Is: 5000 mA (0 – 5000)

Fz: 110 Newton (0 – 330)

Fx: 160 Newton (0 – 160)

Ra: 5 microns (0 – 5)

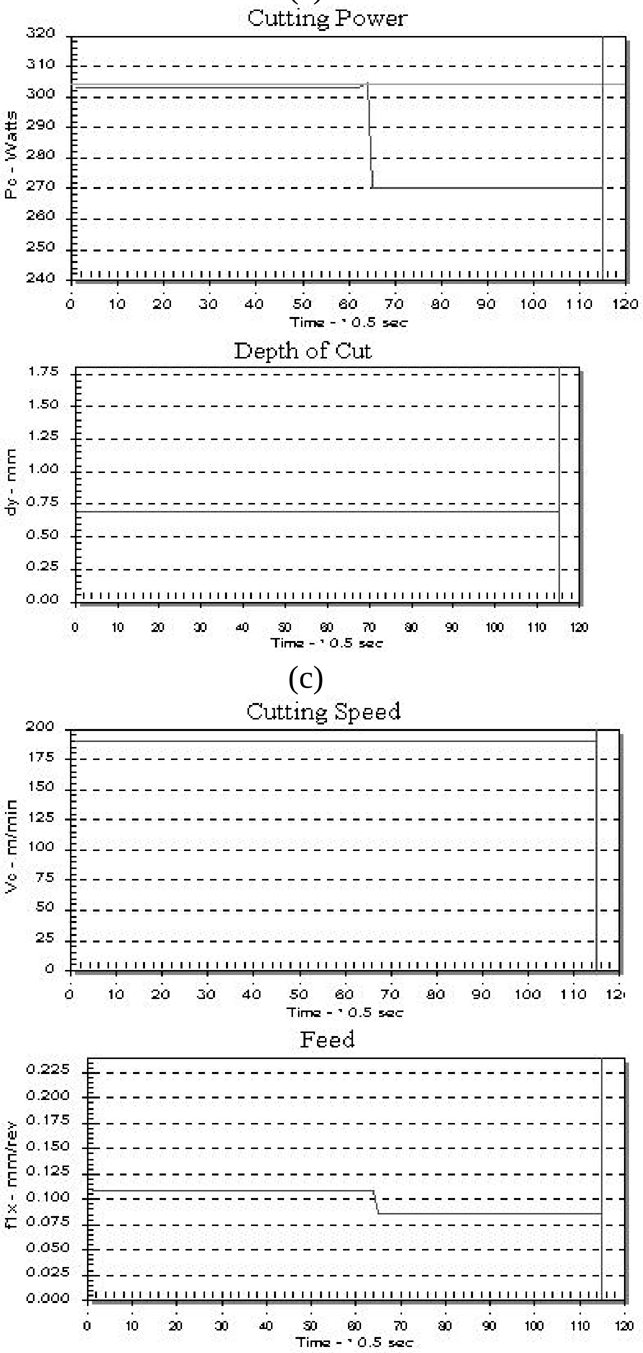
Vy: 170 mV (0 – 170)

Tt: 100 degreeC (0 – 100)

Sc: 280 mV (0 – 280)

Figure 5.14: Simulation user interface with Pc and Fz constraint.

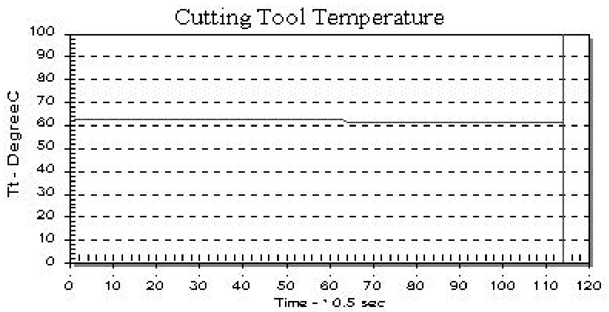
The power in the cut is limited to 304 Watts and the cutting force to 110 Newton. The latter constraint is set to ensure part surface integrity. Machining control parameters are set (same as in test cases no 1 and 2) so that Pc = 302.87 Watts, close to its limit. If the tool wears it will increase Fz, which in turn causes Pc to exceed its power limit. The intelligent diagnostic scheme needs to decide which one of Vc, f1x or dy to change. Figure 5.15 shows the graphical simulation of this process. From inspection we find that the cutting force is close to its limit (i). Therefore, knowing that f1x influences Fz (little more than dy and much more than Vc) a machining expert may suggest that f1x should be decreased.

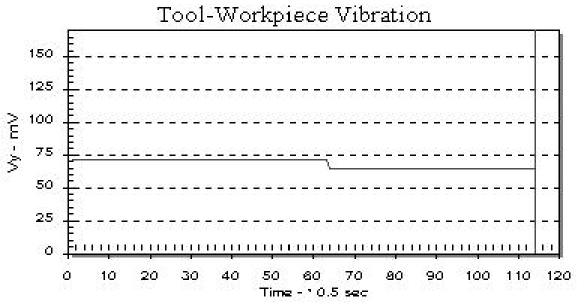

Cutting Power
Pc - Watts
320
310
300
290
280
270
260
250
240
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec
Depth of Cut
dy - mm
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec
Cutting Speed
Vc - m/min
200
175
150
125
100
75
50
25
0
0 10 20 30 40 50 60 70 80 90 100 110 12
Time - * 0.5 sec
Feed
f1x - mm/rev
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec

(a) (b)

(c) (d)

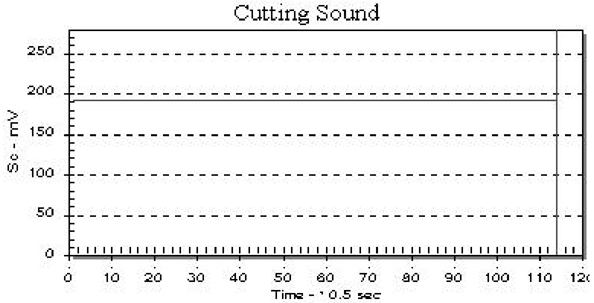
(e) (f)

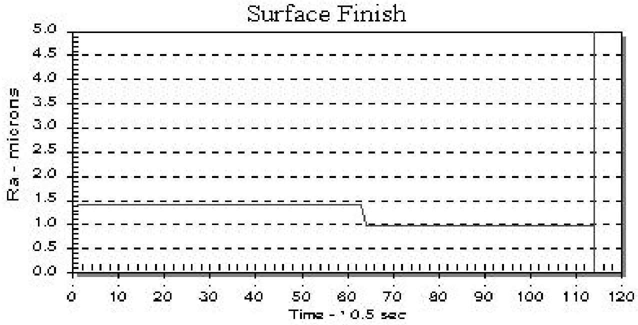

Cutting Tool Temperature
Tt - DegreeC
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec


Tool-Workpiece Vibration
Vy - mV
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

(g)

(h)


Cutting Sound
So - mV
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec


Surface Finish
Ra - microns
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

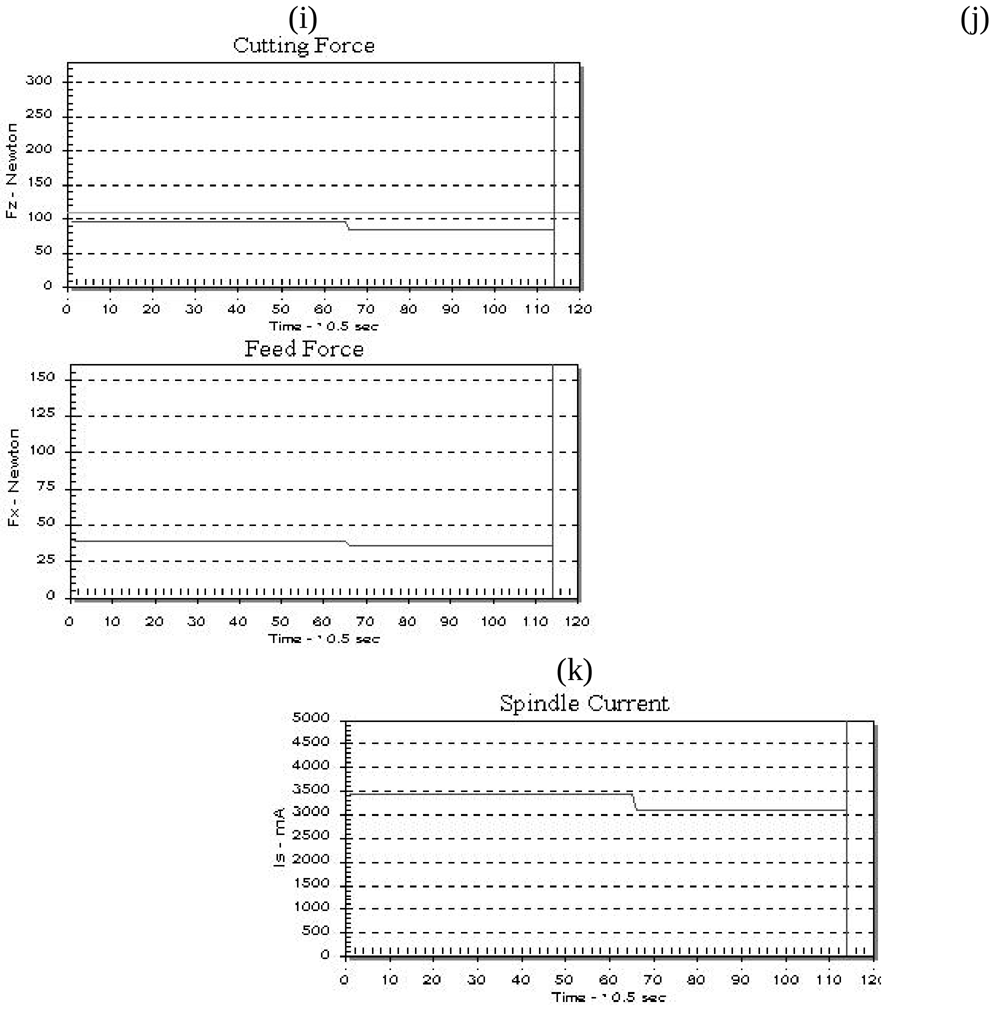


Figure 5.15: Graphical simulation with Pc and Fz constraint.

Applying Equation 5.2 machining state is determined from the sampled signals:

SMP[8]={95.646/110, 39.547/160, 1.417/5, 70.788/170, 62.98/100, 3422.6/5000, 192.47/280, 304/304}

Multiplying SMP[8] with Equation 5.13, result in Diagnostic[8][3]:

$$\text{Diagnostic[8][3] = SMP[8] * FR[8][3]} = \begin{array}{c}Fz\\Fx\\Ra\\Vy\\Tt\\Is\\Sc\\Pc\end{array}\begin{bmatrix}0.6126 & 0.6900 & 0.0\\0.1206 & 0.0 & 0.0\\0.0935 & 0.3022 & 0.0\\0.0 & 0.1435 & 0.1444\\0.3706 & 0.3266 & 0.4045\\0.4448 & 0.499 & 0.0\\0.3308 & 0.0 & 0.3459\\0.5103 & 0.5604 & 0.7674\end{bmatrix}$$

Equation 5.4 is applied to determine the certainty factors:

CNF_1 = Change_dy_CNF = (0.6126+0.3706+0.4448+0.5103)/4 = 0.4845

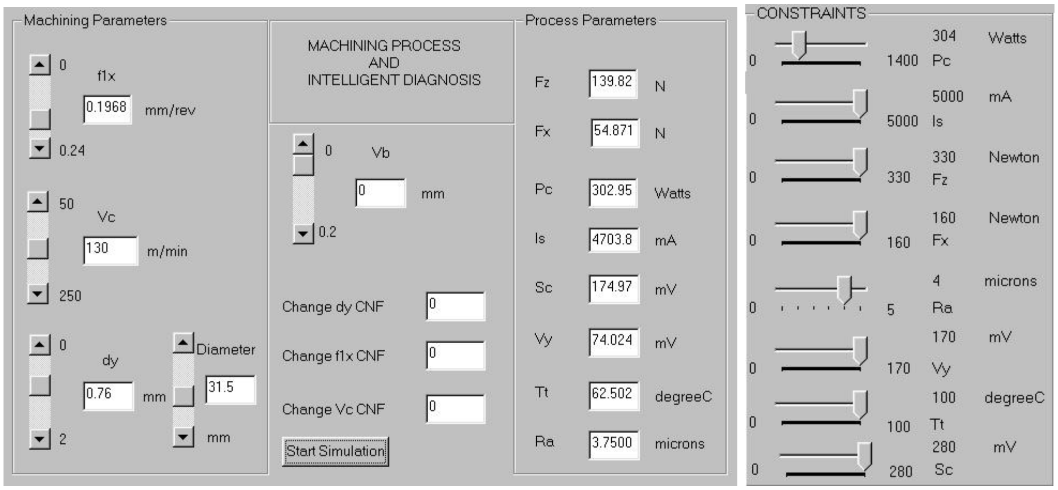
CNF_2 = Change_f1x_CNF = (0.69 + 0.3266 + 0.499 + 0.5604)/4 = 0.519

CNF_3 = Change_Vc_CNF = (0.1444+0.4045+0.3459+0.7674)/4=0.4155

The inference strategy concludes that the feed should be decreased. The amount it decrease, depends on the size of the certainty factor (medium), f1x = f1x * 0.8 = 0.0864 mm/rev. Once the value is determined the value of f1x is adjusted as shown in Figure 5.15 (d) and result in a decrease in Pc (a). Again the decision leaves the system in a more reliable state, as envisaged by the human expertise.

5.4.4 Test Case 4: Pc exceeded with Ra constraint

Figure 5.16 show set process parameter constraints imposed whilst machining a part. The power in the cut is limited to 304 Watts and the surface roughness to 4 microns. The latter constraint is set to maintain part surface quality.



f1x 0.1968 mm/rev

Vc 130 m/min

dy 0.76 mm

Diameter 31.5 mm

Vb 0 mm

Change dy CNF 0

Change f1x CNF 0

Change Vc CNF 0

Fz 139.82 N

Fx 54.871 N

Pc 302.95 Watts

Is 4703.8 mA

Sc 174.97 mV

Vy 74.024 mV

Tt 62.502 degreeC

Ra 3.7500 microns

304 Watts Pc

5000 mA Is

330 Newton Fz

160 Newton Fx

4 microns Ra

170 mV Vy

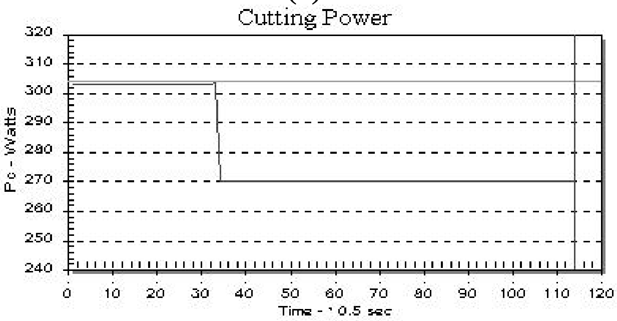
100 degreeC Tt

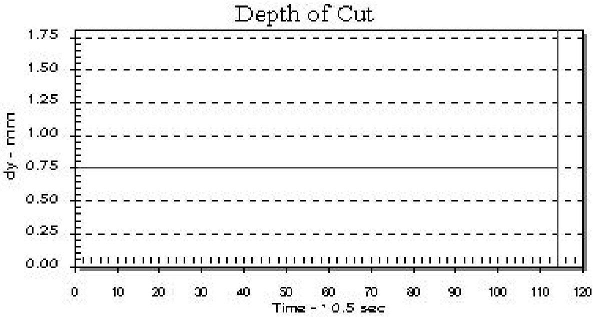
280 mV Sc

Figure 5.16: Simulation user interface with Pc and Ra constraint.

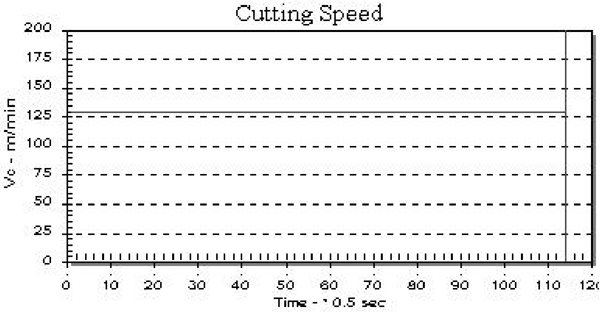
Machining control parameters are set so that Pc = 302.95 Watts, close to its limit. If the tool wears it will increase Fz, which in turn causes Pc to exceed its power limit. The intelligent diagnostic scheme needs to decide which one of Vc, f1x or dy to change. Figure 5.17 shows the graphical simulation of this process. From inspection we find that the surface roughness is close to its limit. Therefore, knowing that f1x influences Ra (more than dy and much more than Vc) a machining expert may suggest that f1x should be decreased. .

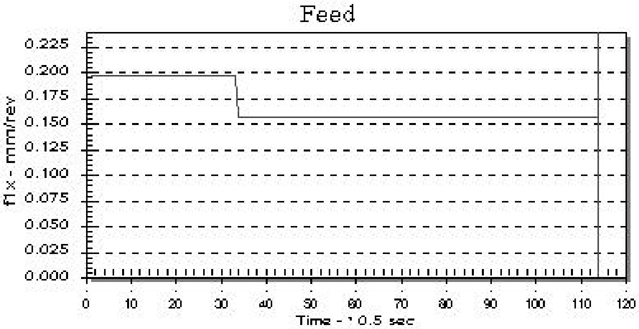
(a) (b)


Cutting Power
Pc - Watts
320
310
300
290
280
270
260
250
240
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec

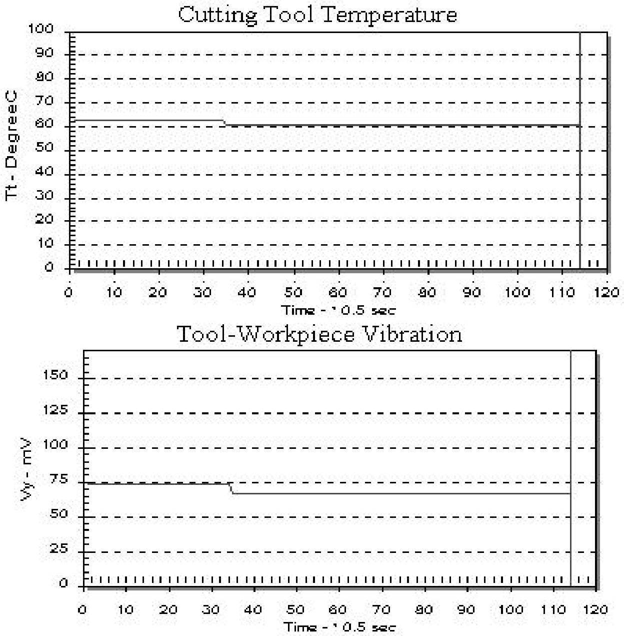

Depth of Cut
dy - mm
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec

(c) (d)

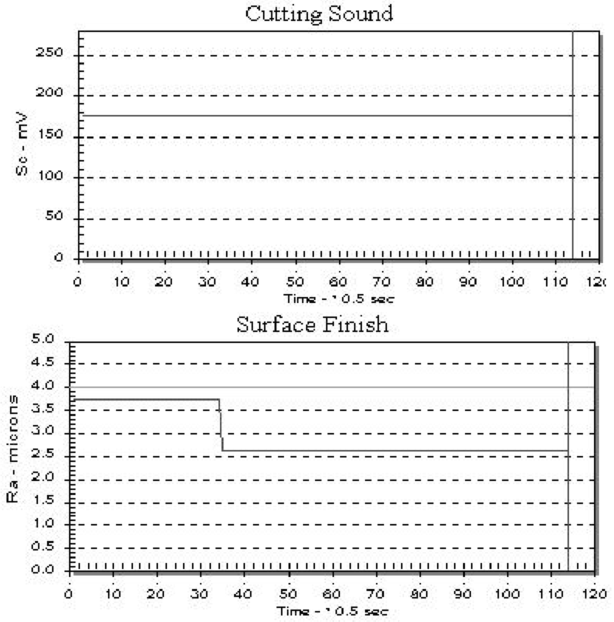

Cutting Speed
Vc - m/min
200
175
150
125
100
75
50
25
0
0 10 20 30 40 50 60 70 80 90 100 110 12(
Time - * 0.5 sec


Feed
f1x - mm/rev
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec

(e) (f)


Cutting Tool Temperature
Tt - DegreeC
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec
Tool-Workpiece Vibration
Vy - mV
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

(g) (h)


Cutting Sound
Sc - mV
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80
90
100
110
12(
Time - * 0.5 sec
Surface Finish
Ra - microns
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

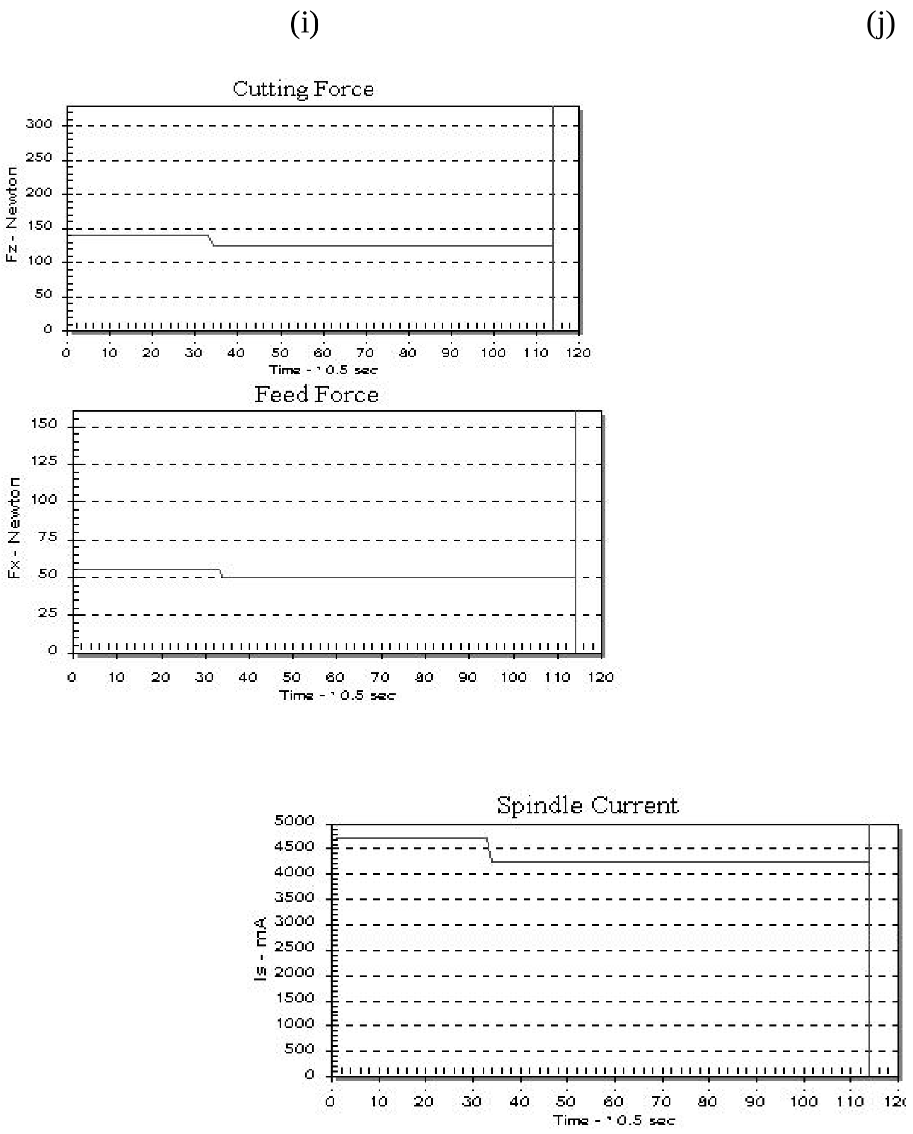


Figure 5.17: Graphical simulation with Pc and Ra constraint.

Applying Equation 5.2 machining state is determined from the sampled signals:

SMP[8] = { 139.82/330, 54.871/160, 3.75/4, 74.024/170, 62.502/100, 3703.8/5000, 174.97/280, 304/304 }

Multiplying SMP[8] with Equation 5.13, result in Diagnostic[8][3]:

$$\text{Diagnostic[8][3]} = \text{SMP[8]} * \text{ FR[8][3]} = \begin{array}{c} Fz \\ Fx \\ Ra \\ Vy \\ Tt \\ Is \\ Sc \\ Pc \end{array} \begin{bmatrix} 0.2984 & 0.3362 & 0.0 \\ 0.1674 & 0.0 & 0.0 \\ 0.3092 & 1.1373 & 0.0 \\ 0.0 & 0.1501 & 0.151 \\ 0.3678 & 0.3241 & 0.4014 \\ 0.4813 & 0.54 & 0.0 \\ 0.3006 & 0.0 & 0.3145 \\ 0.5103 & 0.5604 & 0.7674 \end{bmatrix}$$

Equation 5.4 is applied to determine the certainty factors:

CNF_1 = Change_dy_CNF = (0.5103+0.4813+0.3678+0.3092)/4 = 0.4172

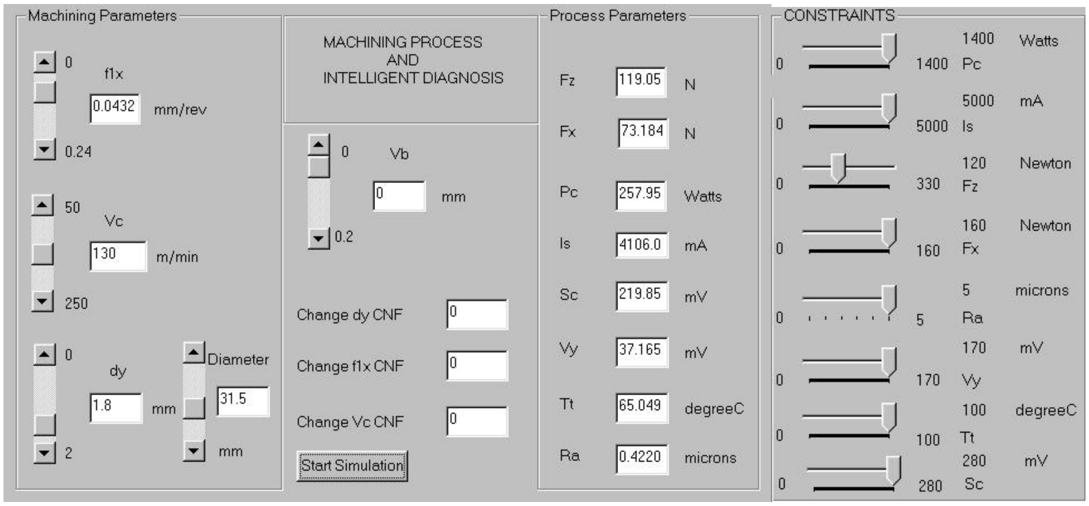
CNF_2 = Change_f1x_CNF = (1.1373+0.5604+0.3363+0.3241)/4 = 0.5895

CNF_3 = Change_Vc_CNF = (0.7674+0.3145+0.4014+0.151)/4=0.4086

The inference strategy concludes that the feed should be decreased. The amount it decrease, depends on the size of the certainty factor (medium), f1x = f1x * 0.8 = 0.1574 mm/rev. Once the value is determined the value of f1x is adjusted as shown in (d) and result in a decrease in Pc (a). Again the decision leaves the system in a more reliable state, as envisaged by the human expertise.

5.4.5 Test Case 5:Fz Exceeded

Figure 5.18 show set process parameter constraints imposed whilst machining a part. The cutting force is limited to 120 Newton. The constraint is set to ensure part surface integrity.



Machining Parameters

f1x 0.0432 mm/rev

Vc 130 m/min

dy 1.8 mm

Diameter 31.5 mm

Vb 0 mm

Change dy CNF 0

Change f1x CNF 0

Change Vc CNF 0

Process Parameters

Fz 119.05 N

Fx 73.184 N

Pc 257.95 Watts

Is 4106.0 mA

Sc 219.85 mV

Vy 37.165 mV

Tt 65.049 degreeC

Ra 0.4220 microns

CONSTRAINTS

0 1400 Pc 1400 Watts

0 5000 Is 5000 mA

0 330 Fz 120 Newton

0 160 Fx 160 Newton

0 5 Ra 5 microns

0 170 Vy 170 mV

0 100 Tt 100 degreeC

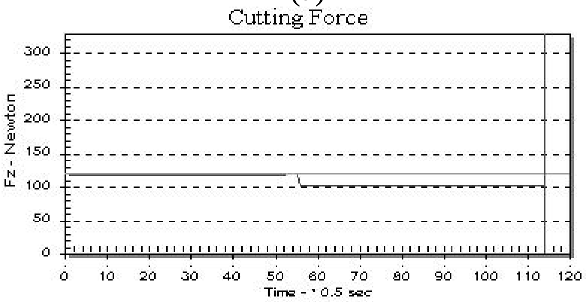
0 280 Sc 280 mV

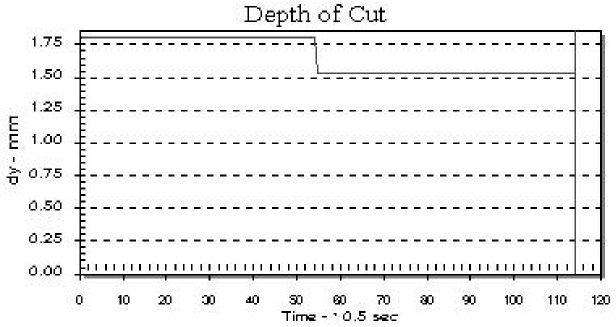
Figure 5.18: Simulation user interface with Fz constraint and a high dy.

If the tool wears it will cause Fz to exceed its limit. The intelligent diagnostic scheme needs to decide which one of Vc, f1x or dy to change. Figure 5.19 shows the graphical simulation of this process. From inspection we find that the cutting sound is close to its limit, cutting temperature is not too far from its limit and spindle current is close to its limit. These signals are mostly influenced by the depth of cut.

(a)

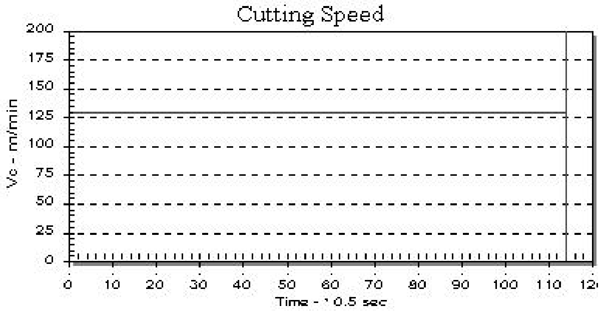
(b)

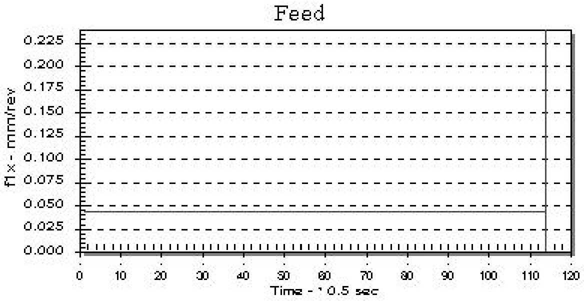

Cutting Force
Fz - Newton
0
50
100
150
200
250
300
Time - * 0.5 sec
0
10
20
30
40
50
60
70
80
90
100
110
120


Depth of Cut
dy - mm
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Time - * 0.5 sec
0
10
20
30
40
50
60
70
80
90
100
110
120

(c)

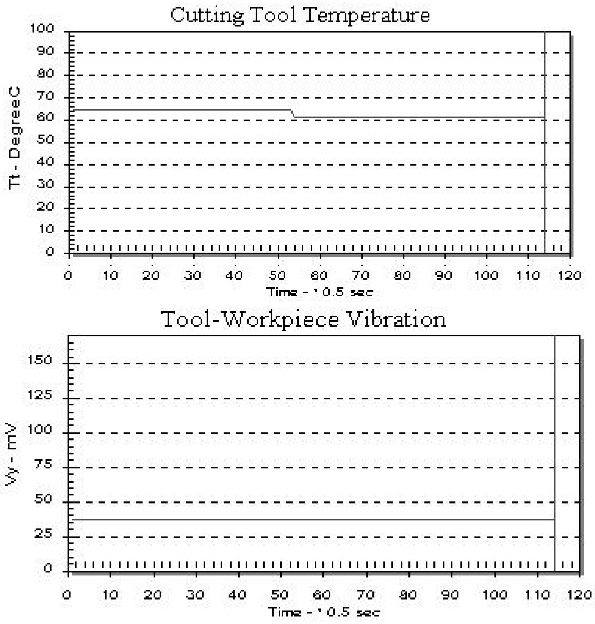
(d)


Cutting Speed
Vc - m/min
0
25
50
75
100
125
150
175
200
Time - * 0.5 sec
0
10
20
30
40
50
60
70
80
90
100
110
120

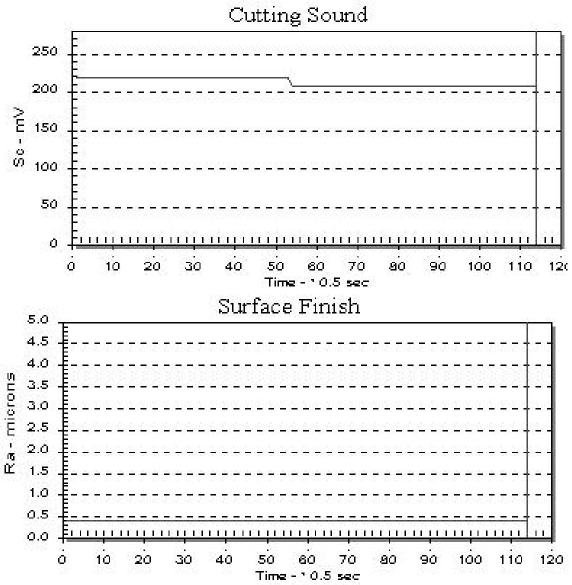

Feed
f1x - mm/rev
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
0.225
Time - * 0.5 sec
0
10
20
30
40
50
60
70
80
90
100
110
120

(e)

(f)


Cutting Tool Temperature
Tt - DegreeC
100
90
80
70
60
50
40
30
20
10
0
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec
Tool-Workpiece Vibration
Vy - mV
150
125
100
75
50
25
0
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec

(g) (h)


Cutting Sound
Sc - mV
250
200
150
100
50
0
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec
Surface Finish
Ra - microns
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0 10 20 30 40 50 60 70 80 90 100 110 120
Time - * 0.5 sec

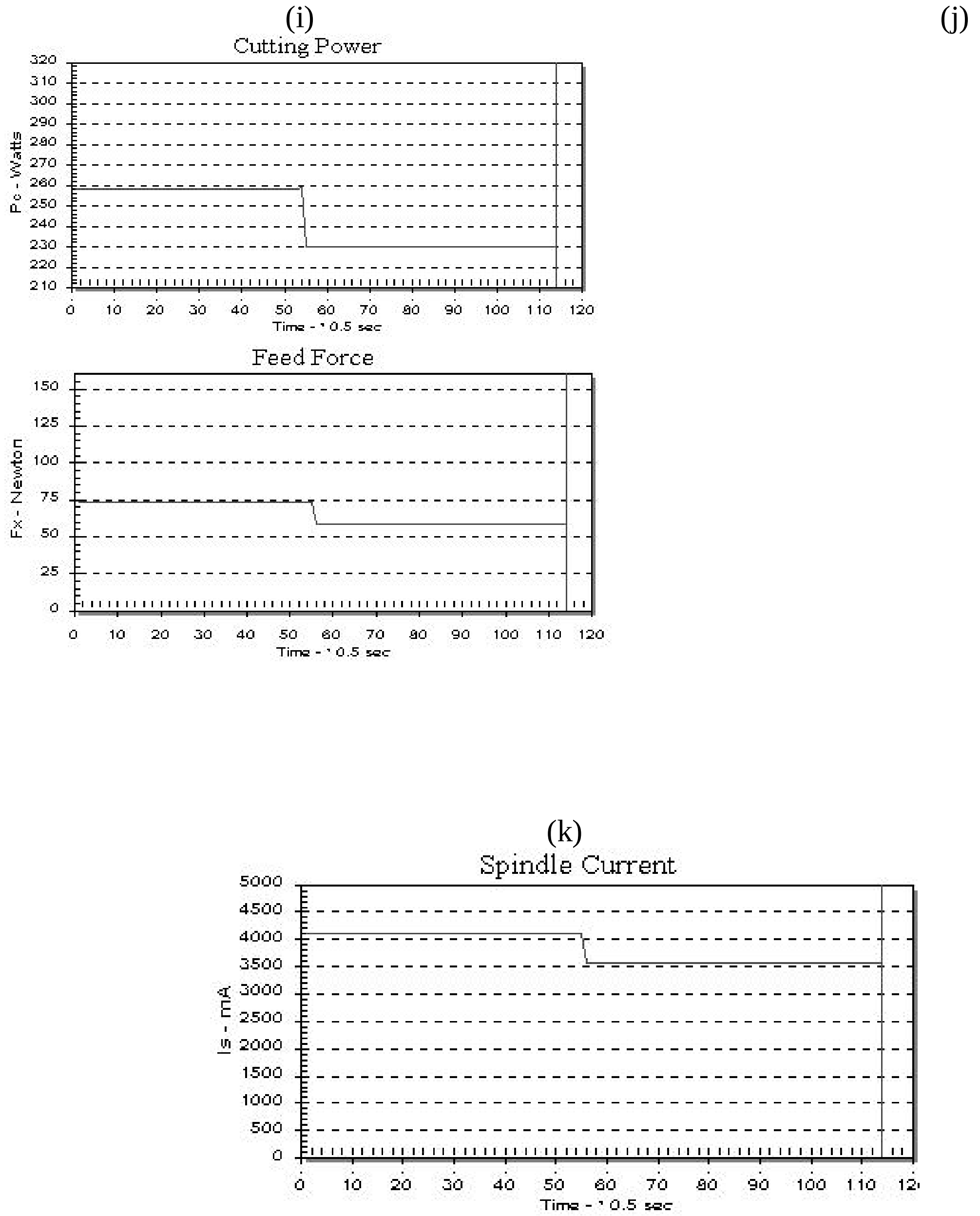


Figure 5.19: Graphical simulation with Fz constraint and a high dy.

Applying Equation 5.2 machining state is determined from the sampled signals:

SMP[8] = { 120/120, 73.184/160, 0.422/5, 37.165/170, 65.049/100, 4106.0/5000, 219.85/280, 257.95/1400}

Multiplying SMP[8] with Equation 5.13, result in Diagnostic[8][3]:

$$\text{Diagnostic[8][3]} = \text{SMP[8]} * \ \text{FR[8][3]} = \begin{array}{r}Fz\\Fx\\Ra\\Vy\\Tt\\Is\\Sc\\Pc\end{array}\begin{bmatrix} 0.7045 & 0.7936 & 0.0 \\ 0.2233 & 0.0 & 0.0 \\ 0.0278 & 0.09 & 0.0 \\ 0.0 & 0.0753 & 0.0758 \\ 0.3827 & 0.3373 & 0.4177 \\ 0.5336 & 0.5987 & 0.0 \\ 0.3777 & 0.0 & 0.3952 \\ 0.094 & 0.1033 & 0.1414 \end{bmatrix}$$

Equation 5.4 is applied to determine the certainty factors

CNF_1 = Change_dy_CNF = (0.7045+0.3827+0.5336+0.3777)/4 = 0.4996

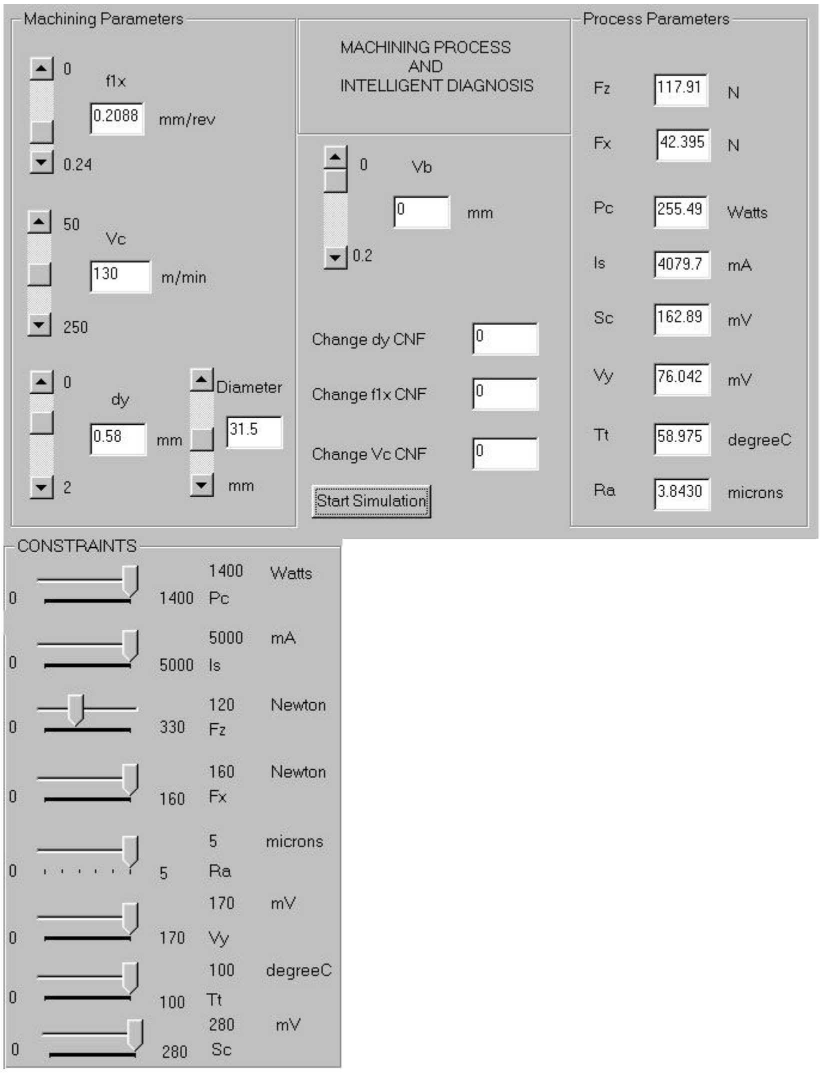
CNF_2 = Change_f1x_CNF = (0.7936+0.3373+0.5987+0.1033)/4 =0.4582

CNF_3 = Change_Vc_CNF = (0.0758+0.4177+0.3952+0.1414)/4=0.2575

The inference strategy conclude that the depth of cut should be decreased. The amount it decrease, depends on the size of the certainty factor (low), dy = dy * 0.85 = 1.53 mm. Once the value is determined the value of dy is adjusted as shown (b) and result in a decrease in Fz (a). Again the decision leaves the system in a more reliable state, as envisaged by the human expertise.

5.4.6 Test case 6: Fz Exceeded

Figure 5.20 show set process parameter constraints imposed whilst machining a part. The cutting force is limited to 120 Newton. The constraint is set to ensure part surface integrity.



Machining Parameters

0
f1x 0.2088 mm/rev
0.24

50
Vc 130 m/min
250

0
dy 0.58 mm
2

Diameter 31.5 mm

0
Vb 0 mm
0.2

Change dy CNF 0

Change f1x CNF 0

Change Vc CNF 0

Process Parameters

Fz 117.91 N

Fx 42.395 N

Pc 255.49 Watts

Is 4079.7 mA

Sc 162.89 mV

Vy 76.042 mV

Tt 58.975 degreeC

Ra 3.8430 microns

CONSTRAINTS

0 1400 Pc 1400 Watts

0 5000 Is 5000 mA

0 330 Fz 120 Newton

0 160 Fx 160 Newton

0 5 Ra 5 microns

0 170 Vy 170 mV

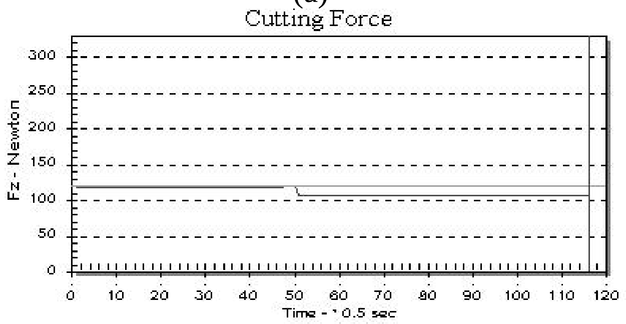
0 100 Tt 100 degreeC

0 280 Sc 280 mV

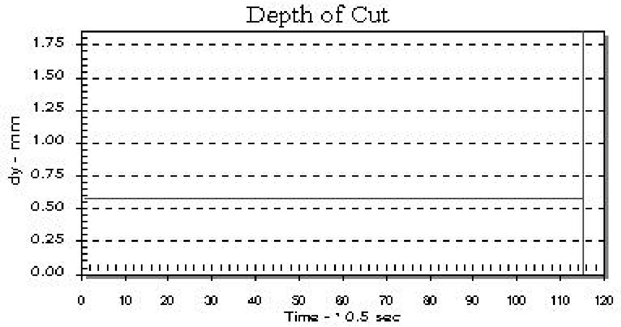
Figure 5.20: Simulation user interface with Fz constraint and a high f1x.

If the tool wears it will cause Fz to exceed its limit. The intelligent diagnostic scheme needs to decide which one of Vc, f1x or dy to change. Figure 5.21 shows the graphical simulation of this process. From inspection we find that the surface roughness is close to its limit, The signal is mostly influenced by feed.

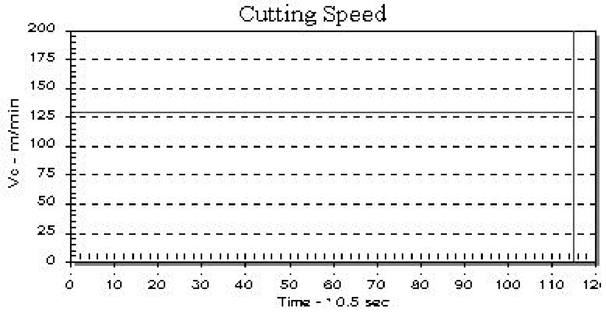
(a)


Cutting Force
Fz - Newton
300
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

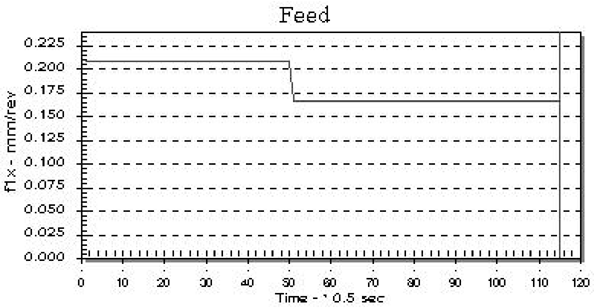
(b)

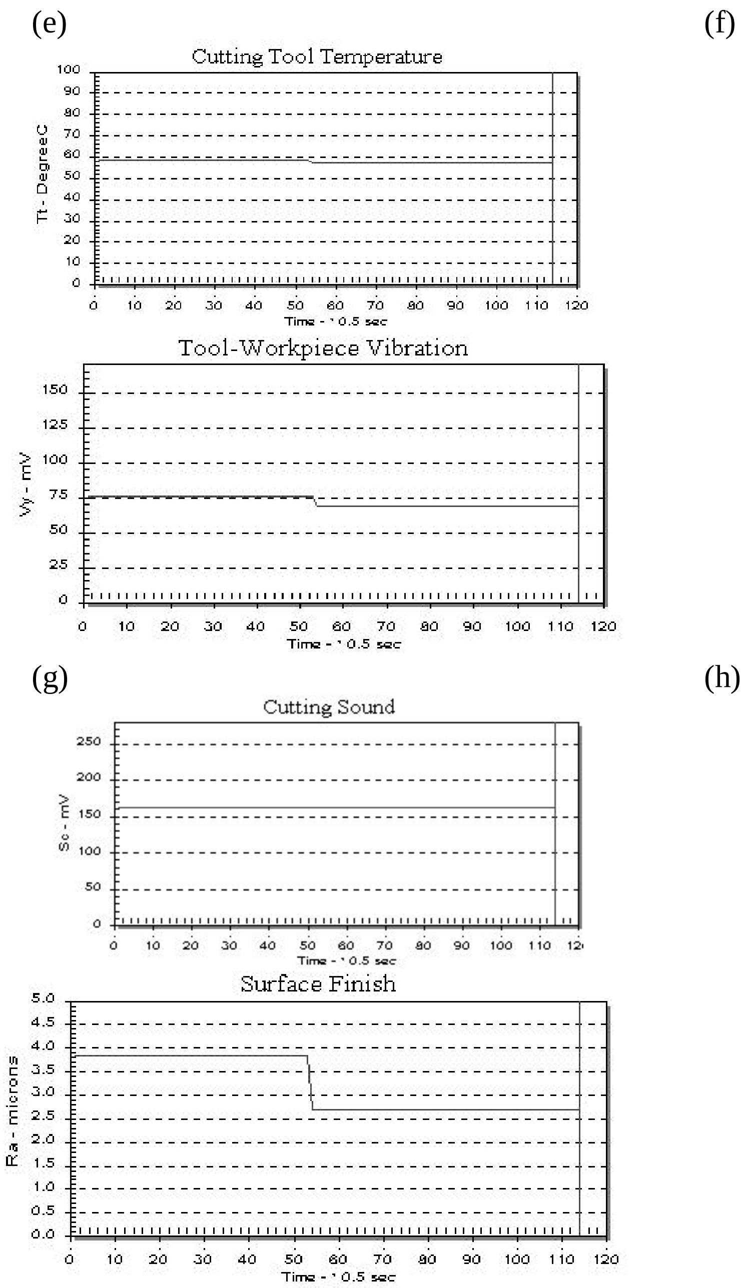

Depth of Cut
dy - mm
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

(c)


Cutting Speed
Vc - m/min
200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
90
100
110
12
Time - * 0.5 sec

(d)


Feed
f1x - mm/rev
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec


(e)
(f)
Cutting Tool Temperature
Tt - DegreeC
0
10
20
30
40
50
60
70
80
90
100
Time - * 0.5 sec
110
120
Tool-Workpiece Vibration
Vy - mV
25
75
125
150
(g)
(h)
Cutting Sound
Sc - mV
200
250
Surface Finish
Ra - microns
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

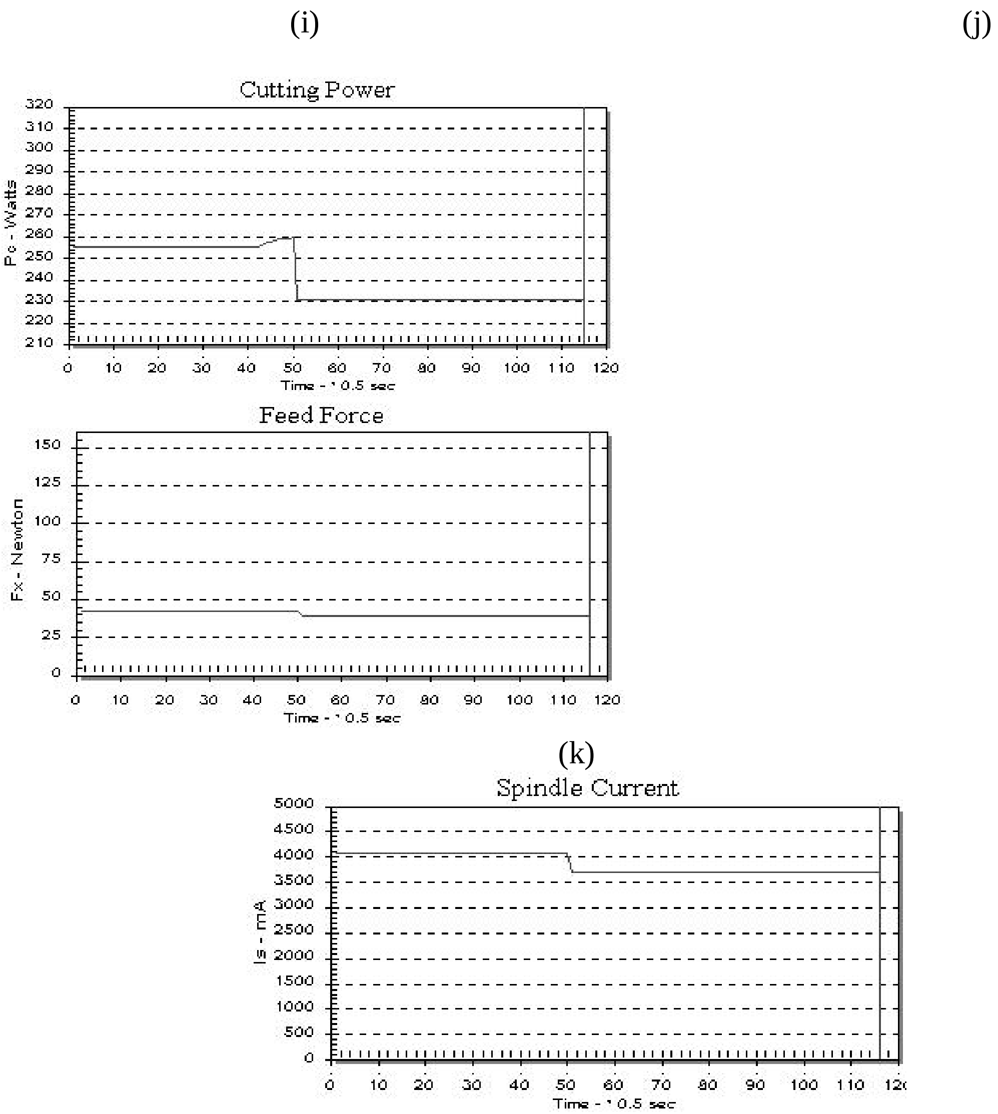


Figure 5.21: Graphical simulation with Fz constraint with a high f1x.

Applying Equation 5.2 machining state is determined from the sampled signals:

SMP[8] = { 120/120, 42.395/160, 3.843/5, 76.042/170, 58.975/100, 4079.7/5000, 162.89/280, 255.49/1400}

Multiplying SMP[8] with Equation 5.13, result in Diagnostic[8][3]:

$$\text{Diagnostic}[8][3] = \text{SMP}[8] * \ \text{FR}[8][3] = \begin{array}{c} Fz \\ Fx \\ Ra \\ Vy \\ Tt \\ Is \\ Sc \\ Pc \end{array} \begin{bmatrix} 0.7045 & 0.7936 & 0.0 \\ 0.1293 & 0.0 & 0.0 \\ 0.2535 & 0.8195 & 0.0 \\ 0.0 & 0.1541 & 0.1551 \\ 0.347 & 0.3058 & 0.3787 \\ 0.5302 & 0.5948 & 0.0 \\ 0.2799 & 0.0 & 0.2928 \\ 0.0931 & 0.1023 & 0.14 \end{bmatrix}$$

Equation 5.4 is applied to determine the certainty factors:

CNF_1 = Change_dy_CNF = (0.7045+0.347+0.5302+0.2799)/4 = 0.4654

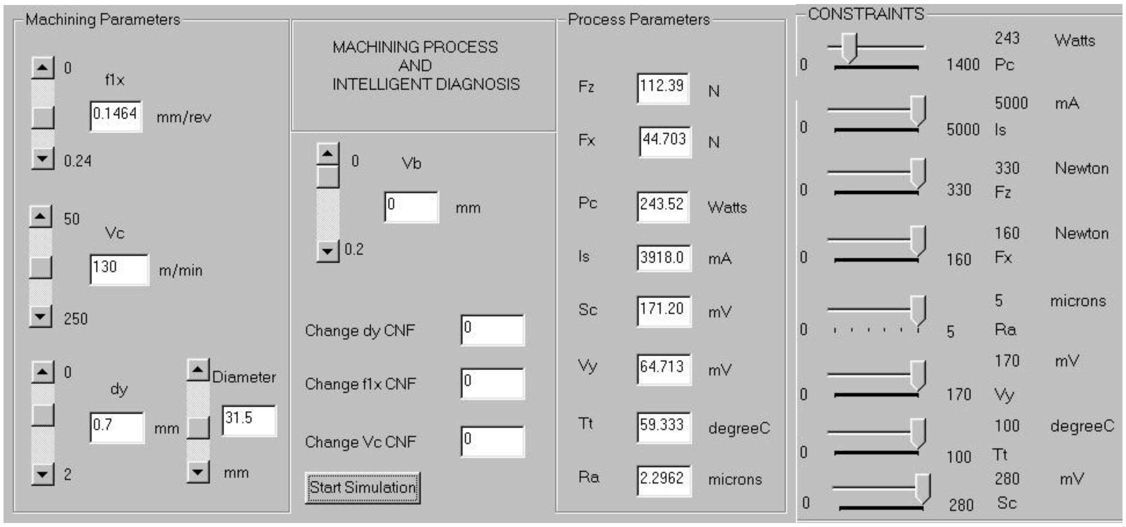
CNF_2 = Change_f1x_CNF = (0.7936+0.8195+0.3058+0.5948)/4 =0.6284

CNF_3 = Change_Vc_CNF = (0.1551+0.3787+0.2928+0.14)/4=0.2416

The inference strategy concludes that the feed should be decreased. The amount it decrease, depends on the size of the certainty factor (medium), f1x = f1x * 0.85 = 0.167 mm/rev. Once the value is determined the value of f1x is adjusted as shown in (d) it results in a decrease in Fz (a). Again the decision leaves the system in a more reliable state, as envisaged by the human expertise.

5.4.6 Test case 7: Pc lower Limit

Figure 5.22 show set process parameter constraints imposed whilst machining a part. The lower limit for cutting power is set at 243 Watts.



f1x 0.1464 mm/rev

Vc 130 m/min

dy 0.7 mm

Diameter 31.5 mm

Vb 0 mm

Change dy CNF 0

Change f1x CNF 0

Change Vc CNF 0

Fz 112.39 N

Fx 44.703 N

Pc 243.52 Watts

Is 3918.0 mA

Sc 171.20 mV

Vy 64.713 mV

Tt 59.333 degreeC

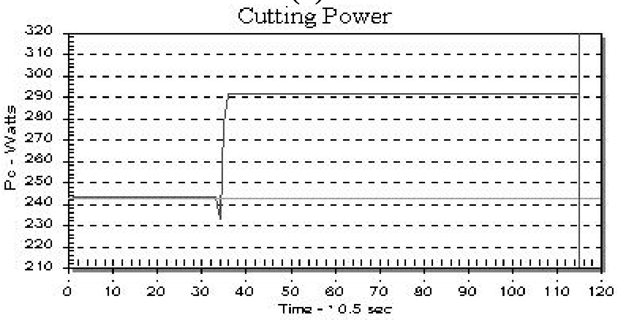
Ra 2.2962 microns

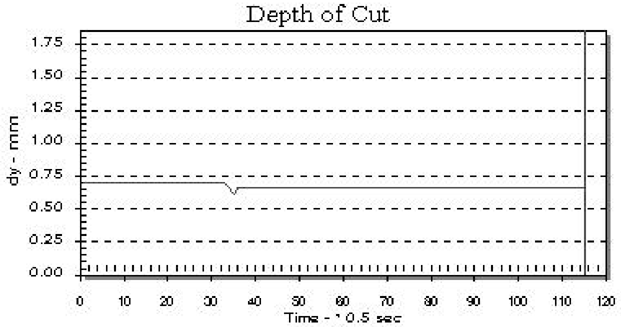
0	1400 Pc	243	Watts
0	5000 Is	5000	mA
0	330 Fz	330	Newton
0	160 Fx	160	Newton
0	5 Ra	5	microns
0	170 Vy	170	mV
0	100 Tt	100	degreeC
0	280 Sc	280	mV

Figure 5.22: Simulation user interface with Pc lower limit .

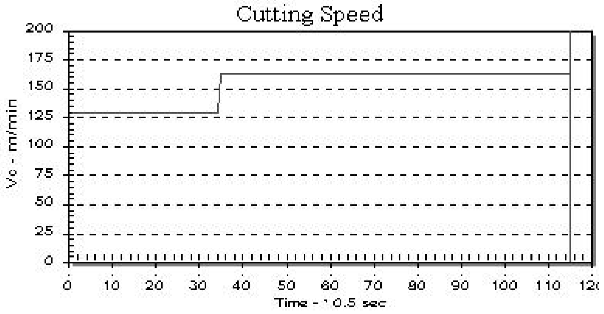
Whilst machining the depth of cut is reduced and this causes the Pc to go below its lower limit. The intelligent diagnostic scheme needs to decide which one of Vc, f1x or dy to change. Figure 5.23 shows the graphical simulation of this process. From inspection we find that the spindle current is close to its limit, therefore to increase the speed is the best choice.

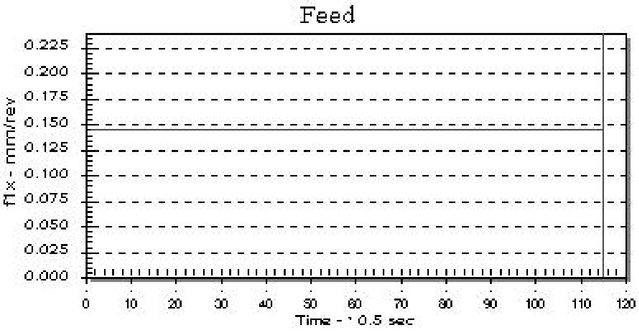
(a) (b)

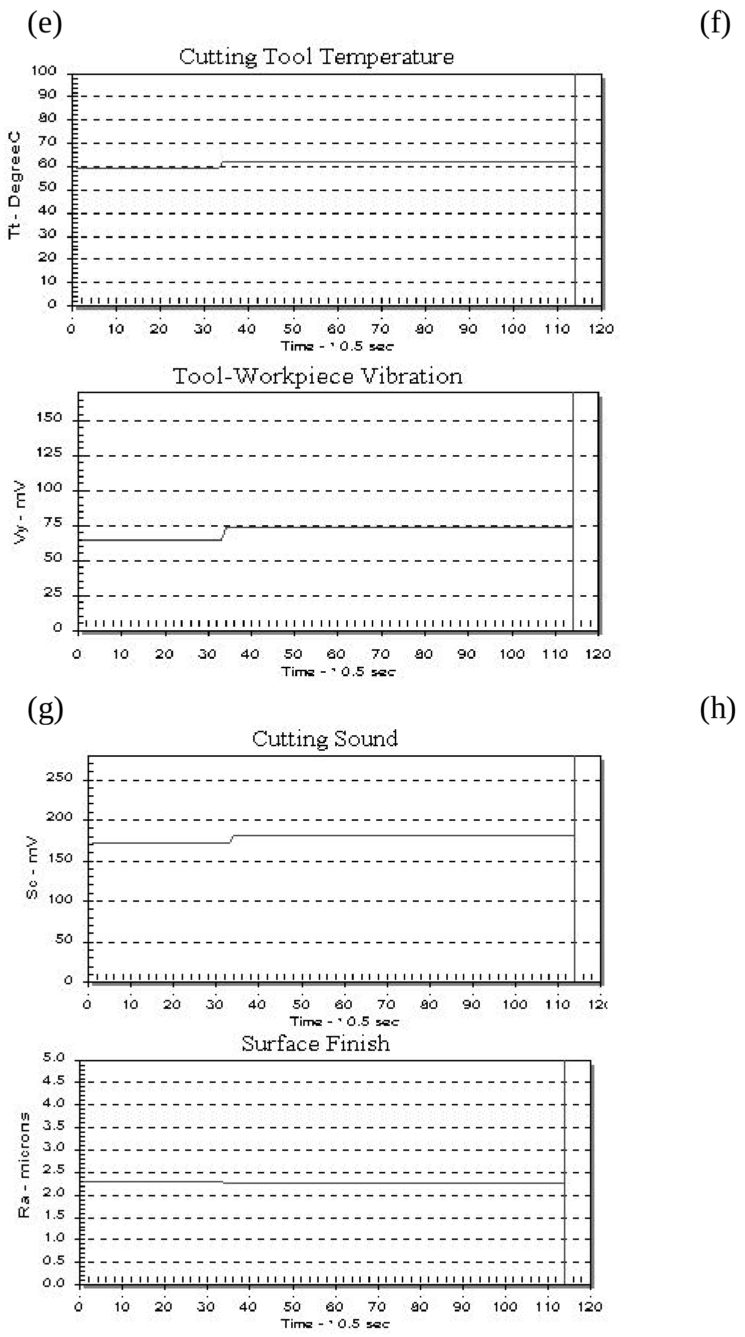

Cutting Power
Pc - Watts
320
310
300
290
280
270
260
250
240
230
220
210
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec


Depth of Cut
dy - mm
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec

(c) (d)


Cutting Speed
Vc - m/min
200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec


Feed
f1x - mm/rev
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec


(e)
(f)
Cutting Tool Temperature
Tt - DegreeC
0
10
20
30
40
50
60
70
80
90
100
110
120
Time - * 0.5 sec
Tool-Workpiece Vibration
Vy - mV
0
25
50
75
100
125
150
(g)
(h)
Cutting Sound
Sc - mV
0
50
100
150
200
250
Surface Finish
Ra - microns
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

(ii) (j)

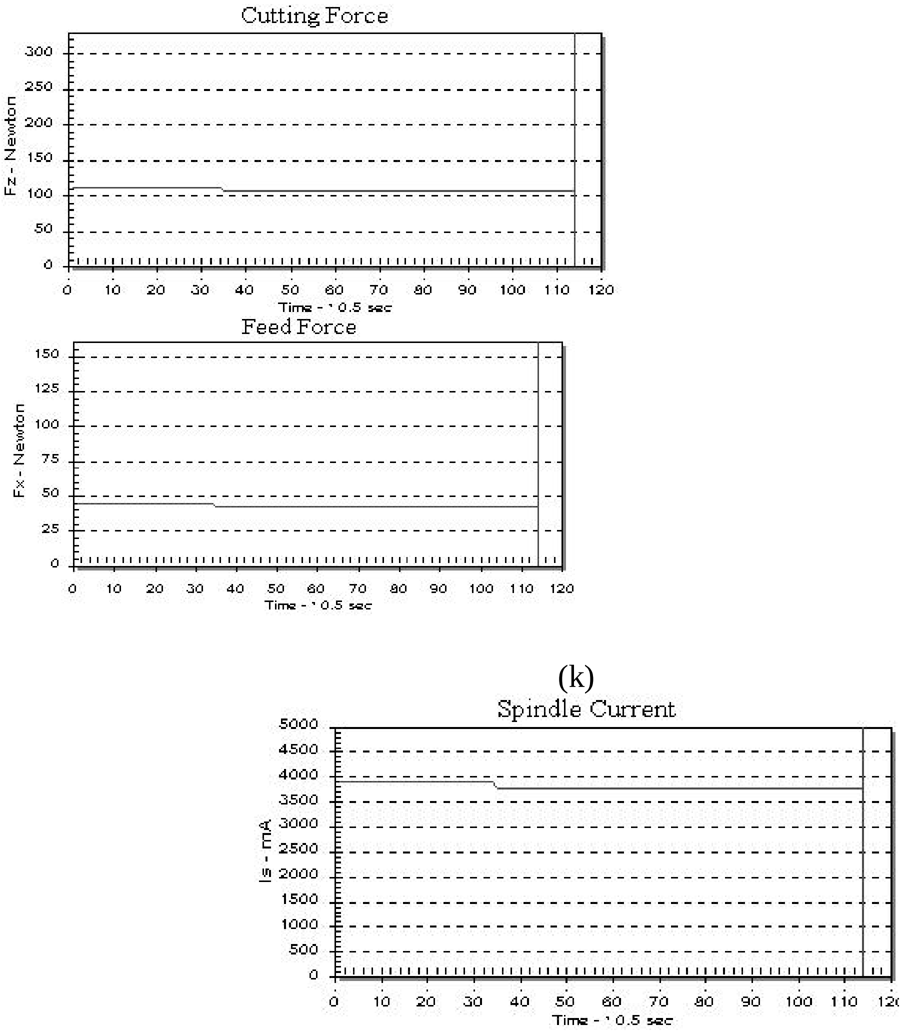


Figure 5.23: Graphical simulation with Pc lower limit.

Applying Equation 5.2 machining state is determined from the sampled signals:

SMP[8] = { 112.39/330, 44.703/160, 2.2962/5, 64.713/170, 59.333/100, 3918/5000, 171.20/280, 243/1400}

Multiplying SMP[8] with Equation 5.13, result in Diagnostic[8][3]:

$$\text{Diagnostic}[8][3] = \text{SMP}[8] * \ \text{FR}[8][3] = \begin{array}{c} Fz \\ Fx \\ Ra \\ Vy \\ Tt \\ Is \\ Sc \\ Pc \end{array} \begin{bmatrix} 0.2399 & 0.2703 & 0.0 \\ 0.1364 & 0.0 & 0.0 \\ 0.1515 & 0.4896 & 0.0 \\ 0.0 & 0.1312 & 0.1320 \\ 0.3491 & 0.3077 & 0.381 \\ 0.5092 & 0.5712 & 0.0 \\ 0.2941 & 0.0 & 0.3077 \\ 0.0886 & 0.0973 & 0.1332 \end{bmatrix}$$

Equation 5.4 is applied to determine the certainty factors:

CNF_1 = 1-Change_dy_CNF = 1- (0.2399+0.3491+0.5092+0.2941)/4 = 0.6519

CNF_2 = 1-Change_f1x_CNF =1- (0.2703+0.4896+0.3077+0.5712)/4 =0.5903

CNF_3 = 1-Change_Vc_CNF = 1- (0.1320+0.381+0.3077+0.1332)/4=0.7615

The inference strategy concludes that the cutting speed should be increased. The amount it increases, depends on the size of the certainty factor (medium), Vc = Vc * 1.25 = 162.5 m/min. Once the value is determined the value of Vc is adjusted as shown in (c) and result in an increase in Pc (a). The decision leaves the system in the most reliable state, as envisaged by the human expertise.

5.5 Conclusion

An advanced diagnostic scheme to complement low-level adaptive control has been proposed and implemented. The knowledge base of the diagnostic scheme consists of a fuzzy relation. The fuzzy relation indicates the strength of connection between the control and process parameters. It was derived from statistical processing of experimental data. A process model based on experimental data was implemented within a simulation, created to test the diagnostic scheme. Within the simulation tool wear process condition causes a process parameters to exceed its limit. The diagnostic scheme is able to reason and decide intelligently which control parameter to change to return the machining process to its most reliable state.

Chapter 6

Conclusion and Future Development

Conventional CNC machines have limitations because of their closed architecture [59, 60]. In order to deal with machining complexity an intelligent machining controller should have a suitable architecture. Open architecture is a philosophy in design and implementation of machine tool, production processes and control. It creates an open environment for manufacturing and enables manufacturing systems to change and reconfiguration of system hardware and software. An open architecture in the design and implementation of intelligent machine tools is an on-going process and need to embrace sensor integration, software and hardware integration, flexibility, openness and knowledge based characteristics [61, 62, 63, 64].

Intelligent machining is an advanced approach in manufacturing, strongly related to the efforts in developing re-configurable manufacturing equipment. This research project introduces all the relevant components and concepts required in the monitoring, diagnosis and control for intelligent machining. These include: identifying sensors to characterize the machining process, digital signal processing for signal measurement, intelligent systems for monitoring and intelligent diagnosis, and multi-axis control technology for machine control. Intelligent machining systems with in-process quality assurance need to detect and react quickly on measured defects, and have the capability to adapt to maintain desired tolerances. The PC-based system implemented for this purpose is one of the major accomplishments of this project and can be summarized as follows:

- The integration of hardware architecture: DSP, PMAC and Ethernet interface cards.
- The implementation of an embedded sensory system that characterize the machining process.
- The implementation of software components, executed on the two PCI32 DSP interface cards, for on-line signal acquisition, filtering and advanced processing (including FFT).
- The implementation and interfacing of machine controls connected to a PMAC interface card to realize multi-axis control.
- Implementation of software for remote monitoring and setting of machining process constraints.
- The implementation of an MFC software application framework (object oriented) to integrate all the modules, including: CMonitorView to request data from the DSP targets, CGeometricView to send motion control commands to the PMAC and CServer for remote monitoring. The application framework includes user interfaces to enable the visualization of the process‘s performance. C++ classes were developed and used to support communication with PMAC interface card.

To realize advanced automation in machining sensors that perform reliable on-line measurement of tool condition and surface roughness, are required [1,3]. In this research project sensors that characterize the machining process were used in multi-sensor fusion models to indirectly measure surface finish and to classify tool wear. An experimental procedure was completed, with the findings and accomplishments summarized as follows:

- The implementation of a procedure to implement FL-based classification and measurement models. The procedure includes statistical processing and FL defuzzification techniques.
- Successful statistical processing of experimental data to identify machining signals and parameters influenced by tool wear and that correlate with surface roughness.
- Successful use of the experimental data and FuzzyTech's NF module to implement FL models.
- The FL models are able to measure Ra with an accuracy of 86.44% and to classify tool wear with a 90% success rate.
- Additional signal analysis found that Fx/Fz, $P_{Vy[283\text{-}556]Hz}$, $P_{Sc[0\text{-}273]Hz}$, $\delta Tt/\delta t$ are sensitive to tool wear.

The monitoring of tool status and surface roughness by means of intelligent systems will enhance automated machining. However, the primary difference between automated machining and intelligent machining is that an intelligent system (applied in the latter) is capable of making decisions based on significant information from the machining process. An advanced diagnostic scheme to complement low-level adaptive control has been proposed and implemented. The findings and accomplishments of the scheme can be summarized as follows:

- A knowledge base for the diagnostic scheme. It consists of a fuzzy relation and is derived from the statistical processing of experimental data.
- Implementing a machining process model (based on experimental data) and executing it within a software simulation. Within the simulation, tool wear process condition causes a process parameters to exceed its limit, the diagnostic scheme is able to reason and decide intelligently which control parameter to change to return the machining process to its most reliable state.

The main knowledge contribution to the field of intelligent machining is the PC-based intelligent machining process controller with artificial intelligent system components to: classify tool wear and measure surface roughness indirectly, and a diagnostic scheme with intelligent decision-making capability. This intelligent machining process controller is sensor based, modular, flexible and include all the components (hardware and software) to perform in-process quality assurance on the machined product.

By using the same principles and components, as used in this project, the system can be extended to include all aspects of advanced machine monitoring. The object oriented software application framework can be enhanced to accommodate these extensions. The fuzzy relation may be viewed as an "intelligent cell" and the principle may be duplicated into various areas of intelligent diagnosis. The following suggestions about the future development of this specific project can be summarized as follows:

- Standardized application software framework with monitoring, diagnosis and machine control objects.
- To increase user interaction one need to develop a standardized user interfaces for the application framework, an area normally neglected by engineers.
- DSP to extract features that relate the monitoring of machining states and conditions to machine control parameters. This will expand the intelligence of the system.
- To expand the diagnostic scheme for implementation and test the interaction and performance with adaptive control.

The continuation of this project is strongly recommended, as it will contribute to the implementation of re-configurable intelligent machining systems.

References

[1] Tonshoff H. K., Wulfsberg J. P., Kals H. J. J., Konig W., Aachen R. W., van Luttervelt C. A., 1988, Developments and Trends in Monitoring and Control of Machining Processes, Annals of the CIRP, Vol. 37, No. 2, pp. 611-619.

[2] Koelsch J. R., 1995, NAMRC XXIII Report: A Deeper Understanding of Machining, Manufacturing Engineering, July, Vol. 115, No.1, pp. 57-61.

[3] Haber R. E., Peres C. R., Alique A., Ros S., Gonzalez C., Alique J. R., 1998, Towards Intelligent Machining: Hierarchical Fuzzy Control for End Milling Process, IEEE Transactions on Control Systems Technology, Vol. 6, No. 2.

[4] Passino K. M., 1995, Intelligent Control for Autonomous Systems, IEEE Spectrum, Vol. 36, No. 6, pp. 55-62.

[5] Von Altrock C., 1997, Fuzzy Logic in Automotive Engineering, Circuit Cellar, November Issue.

[6] Frenzel E. F., 1987, Crash Course in Artificial Intelligence and Expert Systems, Howard W. Sams & Co.

[7] Mirzai A. R., 1990, Artificial Intelligence: Concepts and Applications in Engineering, Chapman and Hall Computing.

[8] Doyle L. E., Keyser C. A., Leach J. L., Schrader G. F., Singer M. B., 1985, Manufacturing Processes and Material for Engineers, Prentice-Hall.

[9] Kalpakjian C., 1995, Manufacturing Engineering and Technology, Addison Wesley, USA.

[10] Tlusty J. Andrews G. C., 1983, A Critical Review of Sensors for Unmanned Machining, Annals of the CIRP, Vol. 32, No. 2, pp. 563-573.

[11] Jung C., Oh J., 1990, Improvement of Surface Waviness by Cutting Force Control in Milling, International Journal for Machine Tool Manufacture, Vol. 31, No.1, pp. 9-21, Great Britain.

[12] Cakir M. C., Gurarda A., 1998, Optimization and Graphical representation of Machining conditions in Multi-Pass Turning Operation, Computer Integrated Manufacturing Systems, Vol. 11, No. 3, pp.157-170, Elsevier.

[13] Ay H., Yang W., Yang J., 1995, Dynamics of Cutting Tool Temperatures During Cutting Process, Experimental Heat Transfer, July-September, Vol. 7, No. 3, pp. 203-216.

[14] Young H., 1996, Cutting Temperature Responses to Flank Wear, Wear, Vol. 201, No. 1, pp.117-120, Elsevier Science.

[15] Dornfeld D. A., 1990, Neural Network Sensor Fusion for Tool Condition Monitoring, Annals of the CIRP, Vol. 39, No. 1, pp. 101-105.

[16] SECO TOOLS, 1996, Secolor Guide Turning Inserts.

[17] Yang K., 1998, Empirical Surface Roughness Monitoring and Cutting Tool Change Procedure, Key Engineering Materials, Vol. 138-140, pp. 593-608, Trans Tech Publications, Switzerland.

[18] Capello E., Davoli P., Bisi G., 1999, Residual Stresses and Surface Roughness in Turning, Transactions of the ASME, Journal of Engineering Materials, Vol. 121, pp.346-351.

[19] Shiraishi M., Yamanaka K., Fujita H., 1990, Optimal Control of Chatter in Turning, International Journal of Machine Tools Manufacture, Vol. 31, No. 1, pp. 31-43, Great Britain.

[20] Lee A., Liu C., Chiang S., 1991, Analysis of Chatter Vibration in a Cutter-Workpiece System, International Journal of Machine Tools Manufacture, Vol. 31, No. 2, pp. 221-234, Great Britain.

[21] Kim J., Lee E., Hyun D., 1994, A Study on the Modeling of Tool Motion and High-Accuracy Surface Generation by the use of Cutting-Force Signals, Journal of Materials Processing Technology, Vol. 47, No. 2, pp. 45-62.

[22] Wiercigroch M., 1997, Chaotic Vibration of a Simple Model of the Machine Tool-Cutting Process System, Transactions of the ASME, July, Vol. 119, Part 3., pp. 468-475.

[23] Tarng Y. S., Young H. T., Lee B. Y., 1992, An Analytical Model of Chatter Vibration in Metal Cutting, International Journal of Machine Tool Manufacture, Vol. 34, No. 2, pp. 183-197, Printed in Great Britain.

[24] Stone B. J., 1990 Suppression of Self-Excited Chatter, Invitation Lecture to Japan Machine Tool Builders Association, Australia.

[25] Fuller C. R., von Flotow A. H., 1995, Active Control of Sound and Vibration, IEEE Control Systems, December, pp. 9-19.

[26] Saini D. P., Park Y. J., 1996, A Quantitative Model of Acoustic Emissions in Orthogonal Cutting Operations, Journal of Materials Processing Technology, April, Vol. 58, No. 4., pp.343 –350.

[27] Smith S. , Tlusty J., 1992, Stabilizing Chatter by Automatic Spindle Speed Regulation, Annals of the CIRP, Vol. 41, No. 1, pp. 433-436.

[28] Spiegel M. R., 1972, Theory and Problems of Statistics in SI Units, McGraw-Hill International, pp. 217-284.

[29] Wylie C. R., Barrett L. C., 1985, Advanced Engineering Mathematics, McGraw-Hill, pp.179-204.

[30] Bethea R. M., Rhinehart R., 1991, Applied Engineering Statistics, Marcel Dekker.

[31] Weiss N. A., Hassett M. J., 1991, Introductory Statistics, Addison-Wesley.

[32] Hamburg M., Young P., 1994, Statistical Analysis for Decision Making, Duxbury Press.

[33] Fang X. D., 1995, Expert System-Supported Fuzzy Diagnosis of Finish-Turning Process States, International Journal of Machine Tools Manufacturing, Vol. 35, No. 6, pp. 913-924, Elsevier Science.

[34] Bellanger M., 2000, Digital Processing of Signals: Theory and Practice, 3rd Edition, p.20, John Wiley & Sons Ltd.

[35] Haykin S., 1988, Digital Communications, p.136, John Wiley and Sons.

[36] McClellan J. H., Schafer R. W., Yoder M. A., DSP First: A Multimedia Approach, pp.119-152, Prentice-Hall.

[37] Kaiser J. F., Mitra S. K., 1993, Handbook for Digital Signal Processing, p.157, 2nd Edition, John Wiley and Sons.

[38] Ifeachor E. C., Jervis B. W., 1993, Digital Signal Processing: A Practical Approach, p.287, Addison-Wesley.

[39] Thede L., 1996, Analog and Digital Filter Design Using C, pp.261-268, Prentice Hall.

[40] Lam H. Y., 1979, Analog and Digital Filters: Design and Realization, p.615-618, Prentice-Hall.

[41] Chassaing R., 1992, Digital Signal Processing with C and the TMS320C30, Wiley-Interscience.

[42] Lynn P. A., Fuerst W., 1997, Introductory Digital Signal Processing with Computer Applications, p.66, 2nd Edition, John Wiley and Sons.

[43] Marven C., Ewers G., 1996, A Simple Approach to Digital Signal Processing, pp.135-139, Wiley Interscience.

[44] Etter D. M., Ingber J. A., Engineering Problem Solving with C, pp. 234-245, 2nd Edition, Prentice Hall.

[45] Marshall G., 1990, Student's Guide To Expert Systems, pp. 143-156, Heinemann Newnes Professional Publishing Ltd.

[46] Mendel J. M., 1995, Fuzzy Logic Systems for Engineering: A Tutorial, Proceedings of the IEEE, Vol. 83, No. 3, March, pp.345-377.

[47] Shaw I. S., 1997, The Theory and Application of Fuzzy Logic to Industrial Process Control, p. 45, RAU Press.

[48] Landau L. J., Taylor J. G., 1998, Concepts for Neural Networks: A Survey, pp.3-11, Springer.

[49] Medsker L. R., 1994, Hybrid Neural Network and Expert Systems, Kluwer Academic Publishers, London.

[50] Monostori L., Egresits C., 1997, On Hybrid Learning and its Application in Intelligent Manufacturing, Computers in Industry, Vol. 33, pp. 111-117, Elsevier Science.

[51] Von Altrock C., 1997, Fuzzy Logic and NeuroFuzzy Applications in Business and Finance, pp. 141-171, Prentice Hall.

[52] Isermann R., 1998, On Fuzzy Logic Applications for Automatic Control, Supervision, and Fault Diagnosis, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 28, No.2.

[53] Koren Y., 1997, Control of Machine Tools, Journal of Manufacturing Science and Engineering, November, Vol. 119, pp. 749-755.

[54] Hsu P., Fann W., 1996, Fuzzy Adaptive Control of Machining Processes with Self-Learning Algorithm, Transactions of the ASME, Vol. 118.

[55] Trang Y., Yen Z., Nian C., 1996, Genetic Synthesis of Fuzzy Logic Controllers in Turning, Fuzzy Sets and Systems, pp. 301-310, Elsevier Science.

[56] Masory O., Koren Y., Adaptive Control System for Turning, Annals of the CIRP, Vol. 29, No. 1, 1980.

[57] Elbestawi M. A., Mohamed Y., Liu L., 1990, Application of Some Parameter Adaptive Control Algorithms in Machining, Journal of Dynamic Systems, Measurement and Control, Vol. 112, pp. 611-617.

[58] Azouzi R., Guillot M., 1997, On-Line Prediction of Surface Finish and Dimensional Deviation in Turning using Neural Network Based Sensor Fusion, Vol. 37, No. 9, pp.1201-1217.

[59] Altintas Y., Newell N., Ito M., 1996, Modular CNC Design for Intelligent Machining, Part 1: Design of a Hierarchical Motion Control Module for CNC Machine Tools, ASME Journal of Manufacturing Science and Engineering, Vol. 118, No. 4, pp. 506-513.

[60] Altintas, Y. and Munasinghe, W. K., 1996, Modular CNC Design for Intelligent Machining, Part 2: Modular Integration of Sensor Based Milling Process monitoring and Control Tasks, ASME Journal of Manufacturing Science and Engineering, Vol. 118, No. 4, pp. 514-521.

[61] Schofield, S. and Wright, P., 1998, Open Architecture Controllers for Machine Tools, Part 1: Design Principles, ASME Journal of Manufacturing Science and Engineering, Vol. 120, No. 2, pp. 417-424.

[62] Altintas W., 1994, A Hierarchical Open-Architecture CNC System for Machine Tools, Annals of the CIRP, Vol. 43, No. 1, pp. 349-354.

[63] Yellowley I., Pottier P. R., 1992, The Integration of Process and Geometry within an Open Architecture Machine Tool Controller, International Journal of Machine Tool Manufacture, pp. 277-293.

[64] Lundholm T., 1991, A Flexible Real-Time Adaptive Control System for Turning, Annals of the CIRP, Vol. 40, No. 1, pp. 441-444.

[65] Wright P. K., 1995, Principles of Open-Architecture Manufacturing, Journal of Manufacturing Systems, Vol. 14, No. 3, pp. 187-201.

[66] Owen J. V., 1995, Opening up Controls Architecture: Plug-and-Play systems are in, and high-priced boxes filled with secret ingredients are out, Manufacturing Engineering, November, pp. 53-60

[67] Innovation Integration, Products specifications available from Internet URL http://www.innovative-dsp.com.

[68] Olsen W.R., Dimitri D.S., 1989, Control for Eight Axes by DSP, Intelligent Motion, Delta Tau Data Systems, Home Page available from Internet URL http://www.deltatau.com.

[69] Delta Tau, 1998, User's Guide for PTalkDT ActiveX, Delta Tau Data Systems.

[70] Hunting B., 1996, Object-Oriented Abstraction through Polymorphism and Virtual Functions, Embedded Systems Programming, July, pp. 49-62.

[71] Deitel H. M., Deitel P. J., 1994, C++ How to Program, Prentice Hall.

[72] Chagnot G., 1996, Flexibility by Design, Embedded Systems Programming, April, pp. 32-54.

[73] Kruglinski D. J., 1996, Inside Visual C++: The Standard Reference for Programming with Microsoft Visual C++, Microsoft Press, Washington.

[74] CD, 1997, Mastering MFC Development, Microsoft training, Microsoft Press, Washington.

[75] Bishop J., 2001, Java Gently 3rd Edition, Addison Wesley.

[76] Installation and Instruction Manual, Baldor Motors and Drives, Baldor Germany.

[77] Data sheet, Pacific Scientific Motor and Controls: Powermax hybrid step motor.

[78] Lin S. C., Hu M. R., 1992, Low Vibration Control System in Turning, International Journal of Machine Tools Manufacture, Vol. 32, No. 5, pp.629-640, Pergamon Press Ltd, Great Britain.

[79] Datasheet, Model 3140 accelerometer, ICSensors, California.

[80] Data sheet, Model 1100 Signal Converter, Measurement Control and Instrumentation, Port Elizabeth, South Africa.

[81] Hoffmann K., 1986, Measuring Elementary Load Cases with Strain Gauges, Hottinger Baldwin Messtechnik GmbH, Darmstadt, West Germany.

[82] Data sheet, Strain Gauge 0.6/120LY11, Hottinger Baldwin Messtechnik GmbH, Darmstadt, West Germany.

[83] Hottinger Baldwin Messtechnik GmbH, Darmstadt, West Germany, Articles available from Internet URL http://www.hbmwt.com.

[84] Hoffmann K., 1984, Practical Hints for the Application of Strain Gauges, Hottinger Baldwin Messtechnik GmbH, Darmstadt, West Germany.

[85] Hoffmann K., 1986, Applying the Wheatstone Bridge Circuit, Hottinger Baldwin Messtechnik GmbH, Darmstadt, West Germany.

[86] Bolton W., 2000, Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering 2nd Edition, Addison Wesley Longman, New York.

[87] Measuring Sound, Bruel & Kjaer, North America, Application note available from URL www.bkhome.com.

[88] LSM900 Professional Omni-directional Instrument Microphone, Leon Audio, Application note available from URL, http://www.LeonAudio.com.au

[89] Engdahl T., 2000, Powering Microphones, Application note available from URL http://www.hut.fi/Misc/Electronics/Circuits/microphone_powering.html.

[90] Data sheet, AD210 Precision Wide Bandwidth 3-Port Isolation Amplifier, Analog Devices.

[91] Models IHA-25/IHA-100, F.W. Bell, USA, Data sheet available from URL http://fwbell.com/html/cs_intro.html.

[92] Drafts B., 1998, A tutorial on Current Measurement Methods, Application note available from URL http://fwbell.com/html/cs_intro.html.

[93] PCI32 Development Package Software Manual, 1997, Innovative Integration, California.

[94] FuzzyTech 4.2 Reference Manual, Inform Software Corporation, Germany.

[95] ProEssentials, 1997, GigaSoft Inc., Software application note available from URL http://www.gigasoft.com

[96] PMAC User's Manual, 1998, Delta Tau Data Systems, California.

[97] Goncharenko I. A., Kimura F., Mori K., 1998, Web-Based User Interfaces for Machine Tool Monitoring and Control, Proceedings of the 31st CIR International Seminar on Networked: Integrated Design Prototyping and Rapid Prototyping, Berkeley, USA, pp. 448-453.

[98] Kruglinski D. J., 1997, Inside Visual C++: The Standard Reference for Programming with Microsoft Visual C++ Version 5, Microsoft Press, Washington.

[99] Dimla D. E., Lister P. M., Lighton N. J., 1997, Neural Network Solutions to the Tool Condition Monitoring Problem in Metal Cutting, Int. J. Mach. Tools Manufacture, Vol. 37, No. 9, pp. 1219-1241.

[100] Tansel, I. N., Mclaughlin, C., 1993, Detection of Tool Breakage in Milling Operation. The Time Series Analysis Approach, International Journal for Machine Tools Manufacture, Vol. 33, No. 4, pp. 531-544.

[101] Purushothaman S., Srinivasa Y. G., 1994, A Back-Propagation Algorithm Applied to Tool Wear Monitoring, International Journal for Machine Tools Manufacture, Vol. 34, No. 5, pp. 625-631.

[102] Ko, T. J., Cho D. W., 1994, Cutting State Monitoring in Milling By a Neural Network, International Journal for Machine Tools Manufacture. Vol. 34, No. 5, pp. 659-676.

[103] Du, R. X., Li, S., 1992, Tool Condition Monitoring in Turning Using Fuzzy Set Technology, International Journal for Machine Tools Manufacture. Vol. 32, No. 6, pp.781-796.

[104] Yan D., Cheng M., Popplewell N., Balakrishnan S., 1995, Application of neural networks for surface roughness measurement in finish turning, International Journal of Production Research, Vol. 83, No. 12, pp.3425-3438.

[105] Jang D. Y., Choi Y., Kim H., Hsiao A., 1996, Study of the Correlation between Surface Roughness and Cutting Vibrations to Develop an On-Line Roughness Measuring Technique in Hard Turning, International Journal of Machine Tools Manufacturing, Vol. 36, No. 4, pp. 453-464, Elsevier Science Ltd, Great Britain.

[106] Statistica V6.0, 1994, User Manual, Copyright StatSoft.

[107] Wegner T., 1993, Applied Business Statistics: Methods and Applications, Juta & Co.

[108] Suliman S.M.A., Hassan G. A., 1992, Turning process model for steady state optimal control. International Journal for Production Research, Vol. 30, No.2, pp.383-394.

Appendix A

Calculation of Filter Coefficients of an FIR Low Pass Filter to meet the Specifications as used in this Project

Specification

Pass band edge frequency: 1.0kHz Transition width: 420Hz

Sampling frequency: 5.0kHz

12 Solution

Normalized □f = 420/5000 = 0.084

Using Equation 2.5, N = 3.3/0.084=40, within hardware memory restriction.

The filter coefficients are obtained from:

$$h[n] = h_D[n]\, w_H[n] \qquad -20 \le n \le 20$$

Because of the smearing effect of the window on the filter response, the cutoff frequency of the resulting filter will be different from that given in the specification. To account for this, we will use an f_c that is centered on the transition band:

$f'_c = f_c + \delta_f$ = (1.0+420/2)= 1.21kHz

Normalized f'_c = 1.21/5 = 0.242

Noting that h[n] is symmetrical; we need only compute values for h[0], h[1],…h[20] and then use the symmetry property to obtain the other coefficients.

n=0:

$$h_D[0] = 2 f_c = 2 * 0.242 = 0.484$$

$$w_H[0] = 0.54 + 0.46\cos(\frac{2\pi 0}{40}) = 1$$

$$h[0] = h_D[0]\, w_H[0] = 0.484$$

n=1:

$$h_D[1] \ = 2*0.242\frac{\sin(2*\pi*1*0.242)}{2*1*\pi*0.242} = 0.317907$$

$$w_H[1] = 0.54 + 0.46\cos(\frac{2*1*\pi}{40}) = 0.994336$$

$$h[1] = h_D[1]\, w_H[1] \ \ = 0.317907*0.994336 = 0.316106 = h[-1]$$

n=2:

$$h_D[2] \ = 2*0.242\frac{\sin(2*\pi*2*0.242)}{2*2*\pi*0.242} = 0.015973$$

$$w_H[2] = 0.54 + 0.46\cos(\frac{2*2*\pi}{40}) = 0.977486$$

$$h[2] = h_D[2]\, w_H[2] \ \ = 0.015973*0.977486 = 0.015613 = h[-2]$$

.

.

.n=20:

$$h_D[20] \ = 2*0.242\frac{\sin(2*\pi*20*0.242)}{2*20*\pi*0.242} = -0.013438$$

$$w_H[20] = 0.54 + 0.46\cos(\frac{2*20*\pi}{40}) = 0.08$$

$$h[20] = h_D[20]\, w_H[20] \ \ = -0.01438*0.08 = -0.00107504 = h[-20]$$

To make the filter causal (necessary for implementation) we add 20 to each index so that the indices start at zero. The filter coefficients, with indices adjusted, are listed in Table A.1.

Table A.1
FIR coefficients for N=41, Hamming window and fc = 1210Hz

H[0]	-0.00107504	H[40]
H[1]	-0.000828893	H[39]
H[2]	0.00142542	H[38]
H[3]	0.0015999	H[37]
H[4]	-0.00240534	H[36]
H[5]	-0.00332171	H[35]
H[6]	0.00396656	H[34]
H[7]	0.00643822	H[33]
H[8]	-0.00598659	H[32]
H[9]	-0.0115257	H[31]
H[10]	0.00828074	H[30]
H[11]	0.0194664	H[29]
H[12]	-0.0106226	H[28]
H[13]	-0.0319655	H[27]
H[14]	0.0127704	H[26]
H[15]	0.0533542	H[25]
H[16]	-0.0144962	H[24]
H[17]	-0.0996399	H[23]
H[18]	0.0156134	H[22]
H[19]	0.316106	H[21]
H[20]	0.484	

5 Appendix B

Machining Process Data for Tool Wear 0 mm and 0.2 mm

Table B.1:
Machining Process Data for Tool Wear: 0.0 mm

12.1Cutting Parameters							***12.2Process Parameters***									
RPM	**Vc**	**f2x**	**f1x**	**dy**	**MRR**	**Diameter**	**T**	**Pc**	**Sc**	**Fx**	**Fz**	**Is**	**Vy**	**Tt**	**Ra**	
[rpm]	[m/min]	[mm/min]	[mm/rev]	[mm]	[mm3/s]	[mm]	[Nm]	[N]	[mV]	[N]	[N]	[mA]	[mV]	[^{o}C]	[microns]	
502.5	48.94	62.53	0.124446	0.6	3654.07	31.6	1.39	71.54	66.76	34.25	87.89	2867.46	22.82	43.69	1.5	
502.15	48.84	124.29	0.247524	0.6	7253.56	31.56	2.51	129.74	94.39	51.85	159.38	4978.7	55.72	55.8	5.4	
1259.94	122.15	129.71	0.10295	0.6	7545.2	31.46	1.21	157.05	132.08	33.09	77.14	2668.12	38.36	53.44	1.1	
1259.55	122.11	258.39	0.20515	0.6	15030.67	31.46	2.05	265.49	158.1	43.4	130.45	4173.53	68.35	64.28	3.9	
1985.38	191.36	166.06	0.08364	0.6	9603.19	31.28	0.93	188.74	151.31	26.46	59.18	2248	43.55	51.19	0.9	
1986.3	192.45	330.04	0.166155	0.6	19185.72	31.44	1.58	322.47	199.125	33.35	100.55	3474.5	79.05	56.59	2.2	
503.04	48.01	41.62	0.08275	1.2	4767.32	31.58	2.09	105.78	154.05	71	132.2	4122.85	24.84	45.36	1.3	
502.09	47.92	61.42	0.122332	1.2	7034.64	31.58	2.6	131.34	160.78	78.68	164.45	4973.42	28.84	46.51	1.8	
1255.14	119	103.37	0.082358	1.2	11760.86	31.38	1.88	238.26	200	62.74	120.13	3775.19	39.49	56.62	1.2	
1265.05	119.55	156.44	0.12366	1.2	17740	31.28	2.46	326.2	-	76.37	163.72	5093.55	60.55	-	1.8	
1270.7	120.32	207.9	0.16361	1.2	23623.07	31.34	2.56	341.04	-	76.38	170.07	5196.8	80.9	62.56	2.6	
1987.48	189.06	102.93	0.051788	1.2	11749.49	31.48	1.46	291.85	252.35	56.36	92.62	1182.66	48.44	65.18	0.6	
1988.13	189.75	204.26	0.10274	1.2	23393.98	31.58	2.02	404.61	278.13	61.87	127.94	4222.4	87.33	69.69	1.7	
501.73	46.97	16.39	0.032668	1.8	2762.24	31.6	1.59	79.01	167.93	64.3	100.93	3263.08	16.77	42.78	0.7	
504.02	47.16	31.67	0.062842	1.8	5333.74	31.58	2.52	125.69	187.84	89.86	159.91	4817.125	27.98	53.47	1.2	
1253.52	117.36	39.7	0.031672	1.8	6689.9	31.6	1.53	189.87	179.91	61.61	97.07	3313.92	30.45	62.34	0.6	
1258.32	117.33	78.44	0.062333	1.8	13164.19	31.48	2.43	301.77	237.32	84.47	154.32	4757.94	45.03	77.93	0.8	
1987.69	186.09	62.05	0.031218	1.8	10456.3	31.6	1.52	297.43	217.47	65.46	95.9	3205.66	46.15	63.92	0.6	
1989.58	185.64	123.12	0.06188	1.8	20677.99	31.5	2.3	452.47	243.86	77.9	146.24	4496.7	65.46	77.28	1.1	

12.2.1 Table B.2

12.3Machining Process Data for Tool Wear: 0.2 mm

Cutting Parameters							***12.4Process Parameters***									
RPM	**Vc**	**f2x**	**f1x**	**dy**	***12.5MRR***	**Diameter**	**T**	**Pc**	**Sc**	**Fx**	**Fz**	**Is**	**Vy**	**Tt**	**Ra**
[rpm]	[m/min]	[mm/min]	[mm/rev]	[mm]	[mm3/s]	[mm]	[Nm]	[N]	[mV]	[N]	[N]	[mA]	[mV]	[^{o}C]	[microns]
502.03	48.89	61.3	0.1221	0.6	3581.98	31.6	1.6	84.64	101.09	49.68	103.49	3374.16	50.66	47.95	1.4
503.49	48.94	122.73	0.2438	0.6	7157.53	31.54	2.59	136.78	100.5	60.72	167.69	5320.55	98.75	58.46	5.7
1256.09	122.09	127.61	0.1016	0.6	7442.44	31.54	1.365	179.87	164.48	47.09	88.4	3036.44	83.33	55.85	1.5
1249.73	122.03	255.4	0.20436	0.6	14962.03	31.68	2.24	293.13	169.15	56.53	144.13	4627.38	117.68	61.67	3.4
1987.4	192.41	161.7	0.08136	0.6	9386.91	31.4	1.113	231.82	175.75	43.25	72.33	2621.2	91.54	59.26	0.9
1989.15	193.22	321.51	0.1618	0.6	18736.66	31.52	1.75	364.44	228.1	50.42	113.29	3832.45	140.1	61.45	2.0
501.11	47.98	40.71	0.08124	1.2	4677.69	31.68	1.97	103.36	119.98	74.21	129.25	4213.32	34.86	46.73	0.9
502.9	48.03	51.18	0.10177	1.2	5865.47	31.6	2.425	127.81	163.75	93.24	159.67	4963.28	53.26	47.54	1.2
1253.7	119.89	102.38	0.08167	1.2	11749.27	31.64	2.145	281.6	232.68	104.17	140.93	4609.06	80.09	61.01	1.1
1250.42	119.5	128.63	0.10287	1.2	14750.97	31.62	2.423	317.19	223.4	107.68	159.26	5092.3	112.18	61.2	1.3
1987.58	189.7	103.15	0.0519	1.2	11813.61	31.58	1.577	327.87	211.83	91.61	103.7	3692.98	85.58	70.35	0.7
1986.8	188.62	203.16	0.10226	1.2	23145.27	31.42	2.263	470.91	263.37	100.55	149.79	4912.1	95.7	71.01	1.3
499.4	47	14.5	0.02903	1.8	2456.05	31.76	1.6525	86.45	159.31	83.37	110.35	3680.63	37.66	43.04	0.6
500.34	46.84	29.83	0.05962	1.8	5026.46	31.6	2.58	135.18	204.58	115.97	173.14	5375.28	51.2	53.2	1.2
1254.07	117.41	25.06	0.02	1.8	4223.47	31.6	1.471	193	189.98	102.14	98.63	3486.97	48.72	58.02	0.7
1252.74	117.44	47.12	0.03762	1.8	7951.38	31.64	2.0838	273.49	194.58	122.64	139.73	4555.89	69.23	65.87	1.0
1986.97	185.9	17.82	0.00896	1.8	3001.17	31.58	0.9783	203.39	200.29	103.65	65.65	2963.18	78.84	64.7	0.9
1985.57	185.27	38.49	0.01939	1.8	6464.48	31.5	1.4475	300.7	201.48	103.03	97.38	3376.64	78.53	68.29	0.7

Appendix C

DSP Target Source Code

/*Source filename : dsptarget.c

Output filename : dsptarget.out

Compiler : C for TMS320C3x/4x

System : PCI32 PC-Based Interface Card

Source Library : Innovative Integration provides libraries that support an extensive set of DSP functions.

The following code is created by adapting example code and applying the DSP functions supplied.

Description: This program reads in analog samples from all four analog channels and then filters them using an FIR routine. The resultant output samples are then stored, in an interleaved fashion, in a queue. This is all done in the **analog_isr** (interrupt service routine). When a total of 512 samples for each channel, i.e. 2048 samples in total, are stored in the queue, the samples are dequeued into four individual buffers from where they are then further processed. This is done in the **main** body of the program. While the current frames of 512 samples of each channel are being processed the **analog_isr** will continue writing new samples into the queue thus no data is lost. Processing is done for each channel and it involves determining the **RMS** and **mean** value of the data samples for each channel. An **FFT** is also performed on each channel. The resultant

```
outputs are then placed in the dual-port RAM from where the host application can then
access them.*/
#include <values.h>
#include <math.h>
#include "c:\pci32cc\include\target\stdio.h"
#include "c:\pci32cc\include\target\dsp.h"
#include "c:\pci32cc\include\target\periph.h"
#define Q_SIZE              0x1000      /* Heap size for queue size */
#define FFT_SIZE                  512
#define HALF_FFT_SIZE       256
#define LOG2_SIZE                  9
#define BITREV                    1      /*1 Bit reversal will be performed */
#define FILTER_ORDER        40    /*40 max number of filter coeffs */
/* Remember to change the buffer sizes in "buffers.asm" when the filter order changes*/
#define SAMPLE_BUF_SIZE           FILTER_ORDER + 1 /*ADC circular buffersize/
/* Function Prototypes */
int count=1;
int busy =0;
#define analog_isr c_int99
void analog_isr(void);
void  CalcFFT(float *BufferIn, float *BufferOut, float *Window, float *TwiddleTable);
float CalcRMS(float *BufferIn, int BUF_SIZE);
float CalcAVE(float *BufferIn, int BUF_SIZE);
float CalcFREQ(float *BufferIn, float   BUF_SIZE, float Max_Freq);

#define command_isr c_int03

void command_isr(void);

/* SEE APPENDIX A FOR FILTER COEFFICIENT CALCULATIONS*/

float filter_coeff[FILTER_ORDER + 1] =

{-0.00107503,
-0.000828893,
0.00142542,
0.0015999,
-0.00240534,
-0.00332171,
0.00396656,
0.00643822,
-0.00598659,
-0.0115257,
0.00828074,
0.0194664,
-0.0106226,
-0.0319655,
0.0127704,
0.0533542,
-0.0144962,
-0.0996399,
0.0156134,
0.316107,
0.484,
0.316107,
0.0156134,
-0.0996399,
-0.0144962,
0.0533542,
0.0127704,
-0.0319655,
-0.0106226,
0.0194664,
0.00828074,
-0.0115257,
-0.00598659,
0.00643822,
0.00396656,
-0.00332171,
-0.00240534,
0.0015999,
0.00142542,
-0.000828893,
-0.00107503};
/* ISR data queue */

QUEUE queue;

/* The following extern variables are defined in "buffer.asm" */

extern float coeff_buffer[SAMPLE_BUF_SIZE];     /*coefficient buffer*/

/* analog sample buffers */

extern volatile float     sample_buffer0[SAMPLE_BUF_SIZE];

extern volatile float     sample_buffer1[SAMPLE_BUF_SIZE];

extern volatile float     sample_buffer2[SAMPLE_BUF_SIZE];

extern volatile float     sample_buffer3[SAMPLE_BUF_SIZE];

volatile int sample_buf_write; /* sample buffer head pointer */

/* Used as flag to show if host has read data from dpram */

volatile int data_taken = 1;

void main()

{int I, k ,z;

float max;

int CH1_Dec_Count = 0;

float CH0_FREQ, CH1_FREQ, CH2_FREQ, CH3_FREQ;

float CH0_RMS, CH1_RMS, CH2_RMS, CH3_RMS;

float CH0_AVE, CH1_AVE, CH2_AVE, CH3_AVE;

float HAR, sum_of_coeffs, theta;
float window[FFT_SIZE];
float SinTable[HALF_FFT_SIZE];
float FFTBufferIn0[FFT_SIZE], FFTBufferIn1[FFT_SIZE];
float FFTBufferIn2[FFT_SIZE], FFTBufferIn3[FFT_SIZE];
extern float FFTBufferOut[FFT_SIZE];        /* This buffer address MUST have at least n
LSB's set to zero (where 2^n = FFT_SIZE) */
volatile int* dpram = (volatile int*)&Periph->Dpram; /* Initialise with starting address of
dualport RAM */
enable_cache();
if (!queue_init(&queue, Q_SIZE))     /* Initialise data queue */
                    while (1);
/* Synchronization - Notify Host that you are ready */
write_mailbox(0xA5A5, TERMINAL_MBOX);
/* Initialize all variables and buffers for the FIR filter */
for(i = 0; i < SAMPLE_BUF_SIZE; i++)
                    {sample_buffer0[i] = 0.0;
                    sample_buffer1[i] = 0.0;
                    sample_buffer2[i] = 0.0;
                    sample_buffer3[i] = 0.0;}
sample_buf_write = 0;
/* Normalize filter coefficients */
sum_of_coeffs = 0.0;
for(i = 0; i < FILTER_ORDER + 1; i++)            /* sum coeff's */
```

```
     sum_of_coeffs += filter_coeff[i];
for(i = 0; i < FILTER_ORDER + 1; i++) /*divide coeff's by sum */
    coeff_buffer[i] = filter_coeff[i]/sum_of_coeffs;
/* Build a table with sine samples - "twiddle factors" */
theta = 2*PI/FFT_SIZE;
for (i=0;i<HALF_FFT_SIZE;i++)   /* fill sin table in memory */
        SinTable[i]=sin(i*theta);
/*      Create the windowing data and place inside buffer "window" */
Hamming(window, FFT_SIZE);
timer(0, 0);
enable_analog(BASEBOARD,0);
install_int_vector(analog_isr, 9);       /* Install analog isr */
enable_interrupts();
mailbox_interrupt_install(command_isr);
mailbox_interrupt_enable();
/*Final sync from host - Wait here until host signals that interruptsare active and ready to
be read */
read_mailbox(TERMINAL_MBOX);
timer(0, 5000);          /* Generates a 5kHz timebase for A/D */

for( ;;)
{/*Wait for Analog_ISR to fill a frame of data */
if (enqueued(&queue) >= FFT_SIZE * 4)
```

```
{/* Place data into FFT input buffer */
for (i = 0; i < FFT_SIZE; i++)
        {FFTBufferIn0[i] =   *(volatile int*)dequeue_ptr(&queue);
        FFTBufferIn1[i] =  *(volatile int*)dequeue_ptr(&queue);
        FFTBufferIn2[i] =  *(volatile int*)dequeue_ptr(&queue);
        FFTBufferIn3[i] =  *(volatile int*)dequeue_ptr(&queue);
        }
if(data_taken == 1)
        {
        /* Process channel 0 */
        CH0_RMS = CalcRMS(FFTBufferIn0, FFT_SIZE);
        CH0_AVE = CalcAVE(FFTBufferIn0, FFT_SIZE);
        CalcFFT(FFTBufferIn0, FFTBufferOut, window, SinTable);
        CH0_FREQ = CalcFREQ(FFTBufferOut, HALF_FFT_SIZE, 2500.0);
        for(i = 0; i<256; i++)
                {dpram[i] = to_ieee(FFTBufferOut[i]);
                dpram[256]    = to_ieee(CH0_FREQ);
                dpram[257]    = to_ieee(CH0_RMS);
                dpram[258]    = to_ieee(CH0_AVE);}
        /* Process channel 1 */
        CH1_RMS = CalcRMS(FFTBufferIn1, FFT_SIZE);
        CH1_AVE = CalcAVE(FFTBufferIn1, FFT_SIZE);
        CalcFFT(FFTBufferIn1, FFTBufferOut, window, SinTable);
```

```
CH1_FREQ = CalcFREQ(FFTBufferOut, HALF_FFT_SIZE, 2500.0);
for (i=0, max =0;i<255;i++)
        {if(FFTBufferOut[i]> max) max = FFTBufferOut[i];
        }
for(i = 259, k=0; i < 515; i++,k++)//515
        {if(FFTBufferOut[k] < 1.0) FFTBufferOut[k] =0;
                FFTBufferOut[k] = ((FFTBufferOut[k])/max)*1000;
        dpram[i] = to_ieee(FFTBufferOut[k]);
        }
        dpram[515]    = to_ieee(CH1_FREQ);
        dram[516]     = to_ieee(CH1_RMS);
        dpram[517]    = to_ieee(CH1_AVE);
/* Process Channel 2 */
CH2_RMS = CalcRMS(FFTBufferIn2, FFT_SIZE);
CH2_AVE = CalcAVE(FFTBufferIn2, FFT_SIZE);
CalcFFT(FFTBufferIn2, FFTBufferOut, window, SinTable);
CH2_FREQ = CalcFREQ(FFTBufferOut, HALF_FFT_SIZE, 2500.0);
for (i=0, max =0;i<255;i++)
        {if(FFTBufferOut[i]> max) max = FFTBufferOut[i];
        }
for(i = 518, k =0; i < 774; i++,k++)//774
        {if (FFTBufferOut[k] < 1.0) FFTBufferOut[k] =0;
        FFTBufferOut[k] = (FFTBufferOut[k]/max)*1000;
```

```
                dpram[i] = to_ieee(FFTBufferOut[k]);
                }
                dpram[774]    = to_ieee(CH2_FREQ);
                dpram[775]    = to_ieee(CH2_RMS);
                dpram[776]    = to_ieee(CH2_AVE);
        /* Process Channel 3 */
        CH3_RMS = CalcRMS(FFTBufferIn3, FFT_SIZE);
        CH3_AVE = CalcAVE(FFTBufferIn3, FFT_SIZE);
        CalcFFT(FFTBufferIn3, FFTBufferOut, window, SinTable);
        CH3_FREQ = CalcFREQ(FFTBufferOut, HALF_FFT_SIZE, 2500.0);
         for (i=0, max =0;i<255;i++)
                {if(FFTBufferOut[i]> max) max =     FFTBufferOut[i];
                }
                for(i = 777, k= 0; i < 1033; i++, k++)//1033
                {FFTBufferOut[k] =  ((FFTBufferOut[k])/max)*1000;
                dpram[i] = to_ieee(FFTBufferOut[k]);
                }
                dpram[1033] = to_ieee(CH3_FREQ);
                dpram[1034] = to_ieee(CH3_RMS);
                dpram[1035] = to_ieee(CH3_AVE);
/* Notify host that data is ready to be read */
mailbox_interrupt(1);
data_taken = 0;
```

```
/* data_taken will be set to 1 by the host after reading the data */
        }//end if
}//end for
}//end main
/*****************************************************************
*                         Define interrupt service routines
*****************************************************************/
void analog_isr(void)
{       int CH0_sample = read_adc(BASEBOARD, 0);
        int CH1_sample = read_adc(BASEBOARD, 1) ;
        int CH2_sample = read_adc(BASEBOARD, 2) ;
        int CH3_sample = read_adc(BASEBOARD, 3);
/* Get sample results, store to circular sample buffers. */
        sample_buffer0[sample_buf_write] = (float)CH0_sample;
        sample_buffer1[sample_buf_write] = (float)CH1_sample;
        sample_buffer2[sample_buf_write] = (float)CH2_sample;
        sample_buffer3[sample_buf_write] = (float)CH3_sample;
if(++sample_buf_write == SAMPLE_BUF_SIZE) /* modulo for rollover */
                sample_buf_write = 0;           /* correction */
/*call filter routine from library. Arguments are the filter coefficient array (pointer points
to the h(n-1) term), the sample buffer pointer (points to the least recent data point
sampled, i.e. the tail of the sample circular buffer),and the filter order + 1 */
```

```
CH0_sample = (float)(fir(&filter_coeff[0], &sample_buffer0[sample_buf_write],
FILTER_ORDER + 1));
CH1_sample = (float)(fir(&filter_coeff[0], &sample_buffer1[sample_buf_write],
FILTER_ORDER + 1));
CH2_sample = (float)(fir(&filter_coeff[0], &sample_buffer2[sample_buf_write],
FILTER_ORDER + 1));
CH3_sample = (float)(fir(&filter_coeff[0], &sample_buffer3[sample_buf_write],
FILTER_ORDER + 1));
/* Place the filtered output samples into the queue */
*((int*)enqueue_ptr(&queue)) = CH0_sample ;
*((int*)enqueue_ptr(&queue)) = CH1_sample;
*((int*)enqueue_ptr(&queue)) = CH2_sample;
*((int*)enqueue_ptr(&queue)) = CH3_sample;
}
void command_isr(void)
{
enable_interrupts();
data_taken = mailbox_interrupt_ack();        /* read data from host */
}
/*****************************************************************
*                    Function definitions                                        *
*****************************************************************/
/*****************************************************************
```

Determine the FFT of the samples in the input buffer. The output buffer must be aligned on an address such that the n least significant bits of the address must be zero (where FFT_SIZE = 2^n). The Window buffer must contain the windowing samples and its size must be the same as that of the input buffer (FFT_SIZE). The twiddle table buffer must contain the twiddle factor samples. The resultant samples in the output buffer will be the magnitude of the FFT, determined from the complex output of the FFT() function. Only the positive half of the frequency spectrum is determined therefore only half of the FFT_SIZE samples will be available in the output buffer.

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



void CalcFFT(float *BufferIn, float *BufferOut, float *Window, float *TwiddleTable)
{int i;
/* Next multiply the input frame of data with the window data */
vmul(BufferIn, 1, Window, 1, BufferIn, FFT_SIZE);
/* Now determine the actual FFT of the windowed data*/
ffft_rl(FFT_SIZE, LOG2_SIZE, BufferIn, BufferOut, TwiddleTable, BITREV);
/* Determine the actual magnitude values from the complex values in FFTBufferOut */
for(i=0;i<256;i++)      //HALF_FFT_SIZE
BufferOut[i] = sqrt(pow((BufferOut[i]) , 2)+ pow((BufferOut[HALF_FFT_SIZE + i]) ,
2)   );
}
/**************************************************************/
```

```
/* Determine the RMS value of the samples in the input buffer */
/***********************************************************/
float CalcRMS(float *BufferIn, int BUF_SIZE)
{       int i;
        float Temp = 0;
        for (i = 0; i < BUF_SIZE; i++)
                Temp += (BufferIn[i] * BufferIn[i]);
        Temp /= BUF_SIZE;
        Temp = sqrt(Temp);
        return(Temp/3276.8);        /* Scaling factor */
}
/*******************************************************/
/* Determine the mean value of the samples in the buffer */
/*******************************************************/
float CalcAVE(float *BufferIn, int BUF_SIZE)
{       int i;
        float Temp = 0;
        for (i = 0; i < BUF_SIZE; i++)
                 Temp += BufferIn[i];
        Temp /= BUF_SIZE;
        return(Temp/3276.8);
}
/***************************************************************/
```

```
/* Determine the frequency of the maximum amplitude bin component */
/*****************************************************************/
float CalcFREQ(float *BufferIn, float BUF_SIZE, float Max_Freq)
{       int i;
        int Max_Index;
        float temp;
        float Max_Amplitude = 0;
        for(i=0;i<128;i++)      //i<BUF_SIZE
                if(Max_Amplitude < BufferIn[i])
                        {
                        Max_Amplitude = BufferIn[i];
                        Max_Index = i;
                        }
                 temp = Max_Index* Max_Freq/BUF_SIZE;
        return( temp);
}
```

Appendix D

Class Library for Motion Control Commands

```
/* The PMAC executes motion control commands.
Traditionally one download programs that control a specific
system or process in full.
The following source code include three class libraries:
CprogramBuffer: To control the buffer area where motion
control commands are placed for execution.
CservoMotor: To control all aspects of the spindle motor.
CstepperMotor: To control all aspects of a stepper motor
within an axis and a coordinate system.
The code is used with PTalkDT ActiveX control, and allows
one to control all aspects of the machine’s controls, from
within a windows based application.
*/
//#define ON  0x01
//#define OFF 0x00
class CProgramBuffer
{
public:
	char * Create(void);
	char * Open(void);
	char * Clear(void);
	char * Execute(void);
	char * Abort(void);
	char * Close(void);
	char * Quit (void);
	char * Step (void);
	char * Halt (void);
	char * Run (void);
	char * Hold (void);
	char * Absolute (void);
	char * Delete(void);
	char * DefineRotaryCMD(void);
	char * CommandCMD (char []);
	CProgramBuffer(int, int);
	int GetOpenOrClosed(void);
	int GetRunOrStop(void);
	int GetBufferEmpty(void);
	void SetOpenOrClosed(int );
	void SetRunOrStop(int);
	void SetBufferEmpty(int);
private:
	char Command[30];
	int OpenClose; //Buffer Condition
```

```
        int RunStop;   // Program execution condition
        int BufferIsEmpty; //Software program
        int BufferSize;
        int Coordinate;
};



CProgramBuffer::CProgramBuffer(int Coord, int BuffSize)
{
Coordinate = Coord;
BufferSize = BuffSize;
}
char * CProgramBuffer::Delete(void)
{char buff1[5] = "&";
char buff2[5];
itoa(Coordinate, buff2, 10);
strcpy(Command,buff1);
strcat(Command,buff2);
strcat(Command, " DEL ROT");
return(Command);
}
//Define a Rotary Motion Program Buffer
char * CProgramBuffer::DefineRotaryCMD(void)
{char buff1[2]="&";
char buff2[15] = "define rot ";
char buff3[5];
char buff4[5];
itoa(Coordinate,buff3,10);
itoa(BufferSize, buff4,10);
strcpy(Command,buff1);
strcat(Command, buff3);
strcat(Command, buff2);
strcat(Command, buff4);
return(Command);
}
void CProgramBuffer::SetOpenOrClosed(int SetReset)
{OpenClose = SetReset;
}
void CProgramBuffer::SetRunOrStop(int SetReset)
{
RunStop = SetReset;
}
void CProgramBuffer::SetBufferEmpty(int SetReset)
{
BufferIsEmpty     = SetReset;
}
int  CProgramBuffer::GetOpenOrClosed()
{
return(OpenClose);
}
int CProgramBuffer::GetRunOrStop()
{
return(RunStop);
}
```

```
int CProgramBuffer::GetBufferEmpty()
{
return(BufferIsEmpty);
}
char * CProgramBuffer::Create(void)
{
char buff[30]="&1 define rot 100";
strcpy(Command,buff);
return(Command);
}


char * CProgramBuffer::Open(void)
{
char buff[30]="OPEN ROT";
strcpy(Command,buff);
return(Command);
}
//Erase currently opened buffer
//Usually start with OPEN, then CLEAR.
char * CProgramBuffer::Clear(void)
{
char buff[30]="CLEAR";
strcpy(Command,buff);
return(Command);
}


char * CProgramBuffer::Execute(void)
{
char buff[30]="B0R";
strcpy(Command,buff);
return(Command);
}




//Abort all programs and moves in the currently
//addressed coordinate system
//Rather use H, Q , / or \ commands
//B1R, A, #1J=#2J=, R
char * CProgramBuffer::Abort(void)
{
char buff[30]="A";
strcpy(Command,buff);
return(Command);
}

//Step Working Motion Programs in all coordinate Systems
// If already running mode (after Run command) then
//"S" command will place the program in a single-step mode.
char * CProgramBuffer::Step(void)
{
char buff[30]="s";
strcpy(Command,buff);
return(Command);
}

//Causes all motion programs running in a any
//coordinate system to stop,
//Program execution may be resumed with R (run)
// or S (step);
char * CProgramBuffer::Quit(void)
{
char buff[5]="Q";
strcpy(Command,buff);
return(Command);
}


//Close Open Rotary Buffer
char * CProgramBuffer::Close(void)
{
char buff[30]="CLOSE";
strcpy(Command,buff);
return(Command);
}

//Halt program execution
//Then apply J= to return
//Then apply R command to resume
char * CProgramBuffer::Halt (void)
{char buff[30]="/";
strcpy(Command,buff);
return(Command);
}
char * CProgramBuffer::CommandCMD (char Cmd[])
{
char buff1[10] = "cmd \"";
char buff3[10] = "\"";
strcpy(Command,buff1);
strcat(Command,Cmd);
strcat(Command,buff3);
return(Command);
}

//Run Motion Program
char * CProgramBuffer::Run (void)
{
char buff[30]="R";
strcpy(Command,buff);
return(Command);
}

//Do a program Hold
//Permitting jog while in hold mode
//Execute J= to return to point prioir to jog
//Execute R to resume prog
char *  CProgramBuffer::Hold (void)
```

```
{
char buff[30];
strcpy(Command,buff);
return(Command);
}

//Select absolute position mode for axes
//in addressed ccoordinate system
```

```
#define OFF 0x00
class CServoMotor
{public:
        int OnOff();
        void Off();
        void On();
        CServoMotor(int , int , int );
        void SetRPM(int );
        char * GetSpeedCommand (void);
        int GetRPM(void);
        char * SetIVarCMD(int IVar, int value);
private:
        int RPM, MaxSpeed, MinSpeed;
        int Position, MaxPosition, MinPosition;
        int MotorOnOff;
        char SpeedCommand[30]  ;
};
CServoMotor::CServoMotor(int Rpm, int MaxSpeed, int MinSpeed)
{RPM = Rpm;
MotorOnOff =0;
}
void CServoMotor::SetRPM (int Rpm)
{RPM = Rpm;
}
char * CServoMotor::SetIVarCMD(int IVar, int value)
{char buff1[10];
char buff2[10];
itoa(IVar, buff1, 10);
sprintf(buff2,"%.2f",float(value * 0.0683));
strcpy(SpeedCommand, "i");
strcat(SpeedCommand,buff1);
strcat(SpeedCommand, "=");
strcat(SpeedCommand, buff2);
return(SpeedCommand);
}
```

```
char *CServoMotor::GetSpeedCommand ()
{char buff1[30] = "i122=";
char buff2[30];
//Use "cmd" when buffer is closed
//char buff3[20] = "cmd\"j+\"";
char buff3[20] = "j+";
itoa(  (int (RPM*0.1334))  ,buff2,20);
strncat(buff1, buff2, 30);
strncat(buff1, buff3, 20);
strcpy(SpeedCommand, buff1 );
return(SpeedCommand);
}
int CServoMotor::GetRPM(void)
{ return(RPM);
}
void CServoMotor::On()
{MotorOnOff = 1;
}
void CServoMotor::Off()
{MotorOnOff = 0;
}
int CServoMotor::OnOff()
{return(MotorOnOff);
}
//Direction Define
#define POSITIVE 1
#define NEGATIVE -1
#define ON 1
#define OFF 0
class CStepperMotor
{
//Motor Initialization Information
public:
        CStepperMotor(int , int , int , char * );

private:
        char MotorNumber[10];

//Jog Related Functions
public:
        void SetJogRate(int );
        char * GetJogCommand (void); //Used to Send to PMAC
        int GetJogRate();
        char * JogCMD(int dir , int speed);
        char * JogStopCMD(void);
        char * JogReltoCommandCMD(int distance);
        char * JogPreJogCMD(void);
        char * JogPosCMD(int position);
        char * JogReltoActualCMD(int distance);
private:
        int  JogRate, MaxJogRate, MinJogRate;
        char JogRateCMD[30];
//Utility Related Functions
public:
        char * CloseLoopCMD(void);
        char * KillCMD (void);
        char * KillAllCMD (void);
        char * JogLastCMD(void);
        char * ZeroCMD(void);
        void SetLoopOC(int OC); // 0 - Open , 1 - Close
        int GetLoopOC(void);
        int GetPosNeg(void);  // 0 - Motor Off , 1 - Motor Positive
                                        //  -1 - Negative
        int GetOffOn(void);
        void SetPositive();   //
        void SetNegative();
        void SetOff();
        void SetOn();
private:
        int OffOn;        //0 - Motor Off, 1 - Motor Positive
                                        // -1 - Motor Negative
        int PosNeg;                     //1 - Positive , -1 Negative
        int LoopOpenClose;  // 0 - Open Loop , 1 - Close Loop
        char Command[10];  // "A" - Abort , "K" - Kill
//Positional Information
public:
        void SetPosition(long );
        long GetPosition(void);
private:
        long Position;
//Online Motor Command While Executing a Program
public:
        void SetFeedRate(int );
        char * GetFeedCommand (void);
        int GetFeedRate(void);
        int GetVelocity(void);
        char * GetVelocityCMD(void);
private:
        //int Position, MaxPosition, MinPosition;
        int Feed, MaxFeed, MinFeed;
        char FeedCommand[30]  ;
        int Velocity; //counts_per_msec
//Open Loop Functions and Commands

public:
        char * OpenLoop(int Percentage, int Direction);
private:
        int   OLoop, MaxOpenLoop, MinOpenLoop;
        char OpenLoopCommand[30];
};

//CLASS DEF - Jog Related Functions - BEGIN
int CStepperMotor::GetJogRate()
{
return(JogRate);
}
void CStepperMotor::SetJogRate(int Jog )
{
JogRate = Jog;
}
```

```
char * CStepperMotor::GetJogCommand (void)
{
char buff1[10] = "i";
char buff2[30] = "2";
strncat(buff1, MotorNumber + 1, 20);
strncat(buff1, "22=", 20);

if ( PosNeg == POSITIVE)
{
int val =  int (JogRate * 0.03435);
itoa(  (int(val))  ,buff2,20);
strncat(buff1, buff2, 30);
strncat(buff1, MotorNumber, 20);
strncat(buff1, "j+", 20);
strcpy(JogRateCMD, buff1 );
return(JogRateCMD);
}
if (PosNeg == NEGATIVE)
{int val =  int (JogRate * 0.03435);
itoa(  (int(val))  ,buff2,20);
strncat(buff1, buff2, 30);
strncat(buff1, MotorNumber, 20);
strncat(buff1, "j-", 20);
strcpy(JogRateCMD, buff1 );
return(JogRateCMD);
}
if (OffPosNeg == OFF)
{
itoa(  (int (0))  ,buff2,20);
strncat(buff1, buff2, 30);
strncat(buff1, MotorNumber, 20);
strncat(buff1, "k", 20);
strcpy(JogRateCMD, buff1 );
return(JogRateCMD);
}*/
}
//This function is called after an abort has occured
//and the PMAC need to complete its last positional
// command befor completing the rest of the program
char * CStepperMotor::JogLastCMD(void )
{char buff1[20];
strcpy(buff1,MotorNumber);
strncat(buff1,"j=",20);
return(buff1);
}
char * CStepperMotor::JogCMD(int dir , int speed)
{//char buff1[20]="#";
//strcat(buff1,MotorNumber);
strcpy(Command,MotorNumber);//buff1);
if (dir == POSITIVE)
{strncat(Command,"j+",20);
return(Command);
}
else
{strncat(Command,"j-",20);
return(Command);
}
}
//This Causes the addressed motor to stop jogging
//Also restores position control if motor's servo loop
//has been opened
char * CStepperMotor::JogStopCMD(void)
{strcpy(Command, MotorNumber);
strncat(Command,"j/",20);
return(Command);
}
//Jog Relative to Commanded Position
//J:2000 -> jog 2000 counts
char *  CStepperMotor::JogReltoCommandCMD(int distance)
{char buff2[10];
strcpy(Command, MotorNumber);
strncat(Command,"j:",20);
itoa(distance, buff2,10);
strcat(Command,buff2);
return(Command);
}
//Jog to PreJog Position
//J= See Ix22 for velocity
char *  CStepperMotor::JogPreJogCMD(void)
{strcpy(Command, MotorNumber);
strncat(Command,"j=",20);
return(Command);
}
//Jog to a Specific Position
//#3J=5000
char *  CStepperMotor::JogPosCMD(int position)
{char buff2[10];
strcpy(Command, MotorNumber);
strncat(Command,"j=",20);
itoa(position, buff2,10);
strcat(Command,buff2); reurn(Command);}
/*char * JogReltoActualCMD(int distance)
{char buff2[10];
char buff1[20]="#";
strcat(buff1,MotorNumber);
strcpy(Command, buff1);
strncat(Command,"j:",20);
itoa(buff2, distance);
strcat(Command,buff2);
return(Command);
}
*/
//CLASS DEF - Jog Related Functions - END
///CLASS DEF - Utility Retated Functions - BEGIN
char * CStepperMotor::CloseLoopCMD(void)
{char buff[10]="j/";
strcpy(Command, MotorNumber);
strcat(Command,buff);
return(Command);
}
```

```
//Make commanded axis positions zero
char * CStepperMotor::ZeroCMD(void)
{strcpy(Command, "z");
return (Command);
}
//Kills all motor outputs by opening the servo loop,
//Commanding Zero Output and Making the AE false.
//All motion programs are automatically aborted.
char *  CStepperMotor::KillCMD (void)
{char buff[10] = "k";
strcpy(Command, MotorNumber);
strcat(Command,buff);
return(Command);
}
char * CStepperMotor::KillAllCMD (void)
{char buff[10] = "k";
strcpy(Command,buff);
return(Command);
}
void CStepperMotor::SetLoopOC(int OC)
        {LoopOpenClose = OC;}
int CStepperMotor::GetLoopOC(void)
        {return(LoopOpenClose);}
void CStepperMotor::SetPositive()
{PosNeg = 1;
}
void CStepperMotor::SetOff()
{OffOn = 0;
}
void CStepperMotor::SetNegative()
{ PosNeg = -1;
}
void CStepperMotor::SetOn()
{OffOn = 1;
}
int CStepperMotor::GetPosNeg()
{return(PosNeg);
}
int CStepperMotor::GetOffOn()
{return(OffOn);}
//CLASS DEF - Utility Retated Functions - END
//CLASS DEF - Motor Initialization Info - BEGIN
//      char MotorNumber[10];
CStepperMotor::CStepperMotor(int Fd, int Jog, int A, char MotorNum[10])
{
Feed = Fd;
OffOn = 0;
strcpy(MotorNumber, MotorNum );
}

//CLASS DEF - Motor Initialization Info - END


//CLASS DEF - Open loop - BEGIN
```

```
//Open loop output
//Output as a % of Ix69

char * CStepperMotor::OpenLoop(int Percentage, int Direction)
{
char buff1[30];
char buff2[3]= "o";
char buff3[4] ="-o";
itoa(Percentage, buff1,10);
if (Direction==POSITIVE)
{
strcpy(Command,MotorNumber);//"#3o10");
strcat(Command,buff2);
strcat(Command,buff1);
return(Command);
}
else
{
strcpy(Command,MotorNumber);//"#3o10");
strcat(Command,buff3);
strcat(Command,buff1);
return(Command);
}

}

//CLASS DEF - Open loop - END

//CLASS DEF - Online Commands - BEGIN

void CStepperMotor::SetFeedRate (int Fd)
{
Feed = Fd;
}

char *CStepperMotor::GetFeedCommand ()
{
        char str1[20] = "F", str2[20];

int val;
val = (int ) Feed / 60;
itoa(  val  ,str2,20);
strcat(str1,str2);
strcpy(FeedCommand, str1);
return(FeedCommand);

/*char buff1[30] = "i122=";
char buff2[30];
char buff3[20] = "cmd\"j+\"";
itoa(  (int ((RPM*10)/145))  ,buff2,20);
strncat(buff1, buff2, 30);
strncat(buff1, buff3, 20);
strcpy(SpeedCommand, buff1 );
return(SpeedCommand);
*/
```

```
}

int CStepperMotor::GetFeedRate(void)
{
 return(Feed);

}

char * CStepperMotor::GetVelocityCMD(void)
{
char buff[10] = "v";
strcpy(Command, MotorNumber);
strcpy(Command,buff);
return(Command);
}

int CStepperMotor::GetVelocity(void)
{
return(Velocity);
}

//CLASS DEF  - Online Commands - END

//CLASS DEF  - Positional Information - BEGIN
void CStepperMotor::SetPosition(long Pos)
{
Position = Pos;
}
long CStepperMotor::GetPosition(void)
{
return (Position);
}

//CLASS DEF - Positional Information - END
```