uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Lei Wang

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Mechanical Engineering)

GRADE / DEGREE

Department of Mechanical Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Chatter Detection and Suppression Using Wavelet and Fuzzy Control Approaches in End Milling

TITRE DE LA THÈSE / TITLE OF THESIS

M. Liang

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

R. Lui

D. Necsulescu

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

Chatter Detection and Suppression Using Wavelet and Fuzzy Control Approaches in End Milling

Lei Wang

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirement for the degree of

MASTER OF APPLIED SCIENCE

in Mechanical Engineering

Ottawa-Carleton Institute for Mechanical and Aerospace Engineering
University of Ottawa
Ottawa, Canada

November 2005

Your file Votre référence
ISBN: 0-494-11448-7
Our file Notre référence
ISBN: 0-494-11448-7

Abstract

In metal cutting processes, chatter has been recognized as one of the main factors that limit machining productivity and affect product quality. Two different categories of chatter were classified by researchers, i.e., regenerative chatter and non-regenerative chatter, and in this thesis the former is mainly studied. Over the past few decades though various chatter detection and suppression methods have been developed, their industrial acceptance is still very limited. This research work presents a new system for on-line chatter detection and suppression. Its detection module implements a statistical index to identify chatters by performing wavelet transform and conducting statistical analysis of positive wavelet transform modulus maxima (WTMM). To suppress chatter, two versions of fuzzy control modules, i.e., plain fuzzy control and self-regulating fuzzy control have been implemented. Unlike the previous chatter suppression systems, the new suppression module features two-way adjustment, i.e., both increasing and decreasing the amount of adjustment. Along with the use of single or multi-output control variables to suppress chatter, productivity is preserved as much as possible.

The proposed system is implemented on a SERVO 2000 milling machine. Extensive tests have been carried out. The experimental results show that the wavelet-based chatter detection index can not only detect the existence of chatters but also distinguish the severity levels. The new chatter suppression module works reasonably well in most tests. However, its performance is adversely affected in the presence of non-regenerative vibrations due to the lack of workpiece or clamping rigidity. Further improvements need to be carried out for industrial applications.

Acknowledgement

I would like to thank Dr. Liang, my supervisor, for having been provided me with scientific guidance and financial support to my thesis work. This thesis subject was part of a project initiated by him and has been developed based on his continuous research over the last eight years at the Computer Integrated Manufacturing lab, University of Ottawa.

The work accomplished by several previous graduate students in this research field was informative and helpful. Also I appreciate the technical assistance of Mr. John Perrins and other technicians from the machine workshop.

And lastly, thanks to my family, especially to my father in another world, for their encouragement and generous help in support of the completion of my Master's study.

Table of Contents

Abstract i
Acknowledgement ii
List of Figures vi
List of Tables ix
Nomenclature x
Chapter 1 Introduction 1
1.1 Overview 1
1.2 Motivation and objective 1
1.3 Organization 3
Chapter 2 Literature Review 4
2.1 Review of chatter detection and prediction approaches 4
2.1.1 Signal processing techniques 4
2.1.2 Stability analysis techniques 7
2.2 Review of chatter suppression and avoidance approaches 8
2.2.1 Active control approaches to chatter suppression 8
2.2.2 Passive control approaches to chatter suppresssion 10
2.3 Summarization of reviewed methods 12
Chapter 3 Development of Chatter Detection Module 14
3.1 Wavelet transforms and wavelet transform modulus maxima 15
3.1.1 Theoretical background of wavelet transforms 15
3.1.2 Wavelet transform modulus maxima (WTMM) 19
3.2 Wavelet-based de-noising algorithm 19
3.2.1 Noise reduction by thresholding wavelet coefficients 22
3.2.2 Procedure of wavelet-based de-noising algorithm 23
3.3 Feature extraction and chatter detection index formulation 26
3.3.1 Feature selection and chatter index formulating approach 27
3.3.2 Random process and statistics review pertinent to chatter detection 28
3.3.3 Statistical distribution analysis of wavelet transform modulus maxima 33

3.3.3.1 Rayleigh distribution of wavelet transform modulus maxima....34
3.3.3.2 Weibull distribution of wavelet transform modulus maxima....38
3.3.3.3 Procedure of statistical chatter detection index formulation....40
3.3.3.4 Reference threshold setting and decision making....43
3.4 Conclusions....43
Chapter 4 Development of Chatter Suppression Module....45
4.1 Fuzzy logic controller for end milling processes....45
4.1.1 Design procedure of fuzzy logic controller for end milling....47
4.1.2 Input and output scaling factors....56
4.1.3 Constraints to the proposed control system....57
4.1.4 Some other constraints....57
4.2 Development of self-regulating algorithm....58
4.2.1 Control output adjustment mechanism....58
4.2.2 Auto-tuning of controller parameters....61
4.3 Implementation procedure of proposed control strategy....63
4.4 Conclusions....65
Chapter 5 Experiments....66
5.1 Experimental apparatus....66
5.1.1 CNC SERVO 2000 milling machine....66
5.1.2 Vibration sensor....68
5.1.3 Data acquisition (DAQ) board....68
5.1.4 Software and development kit....69
5.2 Determination of the experimental parameters....69
5.3 Experimental set-up and implementation....70
5.4 Experimental results and discussions....73
5.4.1 System behaviors in chatter detection....75
5.4.2 Comparison between single and multiple parameter adjustment....77
5.4.3 Comparison between plain fuzzy and fuzzy with self-regulting controls....79
5.4.4 System behavior in cutting workpieces under different conditions....81
Chapter 6 Conclusions and Future Research....135
6.1 Conclusions....135

6.2 Future research 135
References 137
Appendices 149

List of Figures

Figure 3.1 Flow chart of chatter detection module 15
Figure 3.2 a), b) Wavelet transform decomposition steps and tree 18
Figure 3.3 Time domain plot of a given input signal *f* 20
Figure 3.4 Wavelet transform plot (64 scale levels) 20
Figure 3.5 a), b) Plots of wavelet coefficients a) and its first derivative b) at scale u=50 21
Figure 3.6 Plot of customized thresholding function for various ζ's from 0 to 1 25
Figure 3.7 Sample of a stationary random process of wavelet coefficients 35
Figure 3.8 Peaks identification in the range $w_{thr}=a$ to $w_{thr}=a+da_p$ 36
Figure 3.9 Rayleigh distribution of wavelet coefficients with positive peaks for a Gaussian process $w_{thr}(t)$ 38
Figure 3.10 Plot of Weibull probability density 39
Figure 3.11 Plots of Weibull cumulative distribution function 40
Figure 3.12 Flow chart of chatter detection index formulation 41
Figure 4.1 Flow chart of chatter suppression module 46
Figure 4.2 Membership functions of linguistic variables, *ee* and *ce* 48
Figure 4.3 Membership functions of linguistic variable *SI* 48
Figure 4.4 Membership functions of linguistic variable *DU* 48
Figure 4.5 Shifted stability lobe and two-way adjustment of spindle speed 49
Figure 4.6 Stability lobes and two-way adjustment strategy with respect to spindle speed 50
Figure 4.7 a), b), c) Illustrations of input membership degrees associated with three fuzzy inputs *ee*, *ce*, *SI* 53,54
Figure 4.8 Defuzzification using singleton membership functions 55
Figure 5.1 System hardware configuration 67
Figure 5.2 CNC SERVO 2000 milling machine 68
Figure 5.3 Designed LabWindowsTM panel 70
Figure 5.4 Systematic block diagram for experiments 71
Figure 5.5 Setting of sensor, cutter, workpiece and slotting 72
Figure 5.6 Profile of two workpieces 72

Figure 5.7 Dimensional measurement of two workpieces used for tests.....73

Figure 5.8 a), b), c), d), e), f), g) Chatter detection result of test #1 (Spindle speed 700rpm, feed rate 70mm/min, depth of cut 2.54mm, WP2).....83,84,85

Figure 5.9 a), b), c), d), e), f), g), h) Chatter detection result of test #2 (Spindle speed 700rpm, feed rate 70mm/min, depth of cut 3.048mm, WP2).....86,87,88

Figure 5.10 a), b), c), d), e), f), g) Chatter detection result of test #3 (Spindle speed 900rpm, feed rate 90mm/min, depth of cut 2.032mm, WP2).....89,90,91

Figure 5.11 a), b), c), d), e), f), g), h) Chatter detection result of test #4 (Spindle speed 900rpm, feed rate 90mm/min, depth of cut 2.54mm, WP2).....92,93,94

Figure 5.12 a), b), c), d), e), f), g), h) Chatter detection result of test #5 (Spindle speed 900rpm, feed rate 90mm/min, depth of cut 3.048mm, WP2).....95,96,97

Figure 5.13 a), b), c), d), e), f), g), h) Chatter detection result of test #6 (Spindle speed 1000rpm, feed rate 100mm/min, depth of cut 3.048mm, WP2).....98,99,100

Figure 5.14 a), b), c), d) Chatter suppression result of test #7 by adjusting speed and feed (Fuzzy control only, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.048mm, WP2).....101,102

Figure 5.15 a), b), c), d) Chatter suppression result of test #8 by adjusting speed and feed (Fuzzy control with self-regulating, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.048mm, WP2).....103,104

Figure 5.16 a), b), c), d), e), f), g), h) Chatter detection result of test #9 (Spindle speed 1000rpm, feed rate 100mm/min, depth of cut 3.048mm, WP2).....105,106,107

Figure 5.17 a), b), c), d) Chatter suppression result of test #10 by adjusting spindle speed only (Fuzzy control only, initial spindle speed 100rpm, feed rate 100mm/min, depth of cut 3.048mm, WP2).....108,109

Figure 5.18 a), b), c), d) Chatter suppression result of test #11 by adjusting speed only (Fuzzy control with self-regulating, initial spindle speed 1000rpm, feed rate 100mm/min, depth of cut 3.048mm, WP2).....110,111

Figure 5.19 a), b), c), d), e), f), g), h) Chatter detection result of test #12 (Spindle speed 1000rpm, feed rate 90mm/min, depth of cut 4.572mm, WP1).....112,113,114

Figure 5.20 a), b), c), d) Chatter suppression result of test #13 by adjusting speed and feed (Fuzzy control only, initial spindle speed 1000rpm, initial feed rate 90mm/min, depth of cut 4.572mm, WP1) 115,116

Figure 5.21 a), b), c), d) Chatter suppression result of test #14 by adjusting speed and feed (Fuzzy control with self-regulating, initial spindle speed 1000rpm, initial feed rate 90mm/min, depth of cut 4.572mm, WP1) 117,118

Figure 5.22 a), b), c), d) Chatter suppression result of test #15 by adjusting speed only (Fuzzy control only, initial spindle speed 1000rpm, feed rate 90mm/min, depth of cut 4.572mm, WP1) 119,120

Figure 5.23 a), b), c), d) Chatter suppression result of test #16 by adjusting speed only (Fuzzy control with self-regulating, initial spindle speed 1000rpm, feed rate 90mm/min, depth of cut 4.572mm, WP1) 121,122

Figure 5.24 a), b), c), d), g), h) Chatter detection result of test #17 (Spindle speed 1000rpm, feed rate 100mm/min, depth of cut 3.556mm, WP2) 123,124,125

Figure 5.25 a), b), c), d), e) Chatter suppression result of test #18 by adjusting speed and feed (Fuzzy control only, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.556mm, WP2) 126,127

Figure 5.26 a), b), c), d), e), f) Chatter suppression result of test #19 by adjusting speed and feed (Fuzzy control with self-regulating, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.556mm, WP2) 128,129,130

Figure 5.27 a), b), c), d), e), f) Chatter suppression result of test #20 by adjusting speed only (Fuzzy control with self-regulating, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.556mm, WP2) 130,131,132

Figure 5.28 a), b), c), d), e) Chatter suppression result of test #21 by adjusting speed only (Fuzzy control only, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.556mm, WP2) 133,134

List of Tables

Table 3.1 General relations in the MRA theory 17
Table 4.1 Feed rate fuzzy rule base for (*SI=ZE* and *PS*) based on *ee* and *ce* 50
Table 4.2 Feed rate fuzzy rule base for (*SI=PM*) based on *ee* and *ce* 50
Table 4.3 Feed rate fuzzy rule base for (*SI=PL*) based on *ee* and *ce* 51
Table 4.4 Spindle speed fuzzy rule base for (*SI=ZE* and *PS*) based on *ee* and *ce* 51
Table 4.5 Spindle speed fuzzy rule base for (*SI=PM*) based on *ee* and *ce* 51
Table 4.6 Spindle speed fuzzy rule base for (*SI=PL*) based on *ee* and *ce* 52
Table 4.7 Calculations of the coefficients of output fuzzy rules 55
Table 4.8 Some additional constraints to control system 57
Table 4.9 Example of performance measures (Yamazaki 1982) 59
Table 4.10 Data for the above example 60
Table 5.1 Experimental test conditions of end milling process 74
Table 5.2 Comparison of control and adjustment methods 78

Nomenclature

a	Constant level of wavelet transform modulus maxima
a_0	Median peak height
a_p	Wavelet transform modulus maxima with positive magnitude
A^*	Parameter vector of ideal approximator (usually fuzzy system) of a plant or controller
A	Approximation of the A^*
b	Intercept of a straight line
c	Approximation coefficients of wavelet transform
cA_j, cD_j	Approximation and detail coefficients of wavelet decomposition tree at scale level j
c_{xy}	Covariance of sample data x and y
ce_i	Fuzzified value of change of the energy error CE_i
CE_i	Change of the energy error at time i
$CE_{\max}$	Maximal values of CE_i
$CE_{\min}$	Maximal and minimal values of CE_i
C_{jkl}	Location of the peak value associated with rule set (j, k, l)
d	(i) Detail coefficients of wavelet transform (ii) Reinforcement delay
D_4	Daubechies wavelet function
DU	Fuzzy output universe
ee	Fuzzified value of the energy error EE_i
E_i	Vibration energy at time i
E_{max}, E_{min}	Maximal and minimal vibration energy
EE_i	The energy error at time i
E_{ref}	Reference energy level
E_{low}	Minimal allowable limits of E_i
E_{up}	Maximal allowable limits of E_i
$E[x]$	Ensemble average of a random process x

f	Feed rate
$f, f(n), f(t), f(\mathrm{x})$	Signals or function
g_k	Wavelet function coefficients of wavelet transform
G	Set of wavelet function coefficients
GC	Input scaling factor of the energy error EE_i
GE	Input scaling factor of change of the energy error CE_i
GU	Output scaling factor
G_p, G_p^*	Adaptation gain, "optimal" value of G_p
h_k	Dilation function coefficients of wavelet transform
H	Set of wavelet scaling function coefficients
$I(u)$	Wavelet spectrum in terms of scale parameter u
j	Scale parameter of discrete wavelet transforms
j, k, l	Subscripts denoting the rules corresponding to the individual input variable, j is for input of energy error, k is for change of the energy error input and l is for input of SI
J	Total number of wavelet scale levels
J, K, L	The number of fuzzy sets associated with j, k, l respectively
J_{te}	Cost function with respect to tracking error te
J_u	Cost function with respect to control command u
k	Translation parameter of discrete wavelet transforms
K_1, K_2	Coefficients for adjusting output scaling factor GU
L	Total number of dilation function coefficients and wavelet function coefficients
m	Slope of a straight line
m_{X1}, m_{X2}	means of random variables X_1 and X_2
M	Number of wavelet coefficients within frequency band
$n_a^+(T)$	Number of positive slope crossings of level a in time T for a typical sample of stationary random process
N	Number of data samples for each data collection cycle
$N_0(thr)$	Number of noisy wavelet coefficients below the threshold

$N_a^+(T)$	Average number of crossings of the level a with positive slope in time T for a stationary random process
NWP	Normalized wavelet power
p_R	Probability density function of Rayleigh distribution
p_W	Probability density function of Weibull distribution
p_X	Probability density function of Gaussian distribution
$p_{X1,X2}$	Bivariate (joint) probability density function of (X_1, X_2)
$p(a_p)$	Probability density function in terms of a_p
$P(ee, ce)_i$	Performance measurement at time i in terms of ee and ce
ΔP	the increment of $P(ee, ce)$ during sampling period
P_R	Cumulative distribution function of Rayleigh distribution
P_W	Cumulative distribution function of Weibull distribution
Q	Covariance matrix of noise η
RP	Relative wavelet power
$R_X(\tau)$	Autocorrelation function of the random process $X(t)$
s	Scale parameter of continuous wavelet transforms
ss	Spindle speed
SI	Statistical index
$S_X(\omega)$	Spectral density of the random process $X(t)$
thr	Noise reduction threshold value
$te(i)$	Instantaneous tracking error
T_p	Dead time
T	Time period
T_s	Sampling period
u	(i) Translation parameter of continuous wavelet transforms (ii) Control signal
$\Delta u(i)$	Change of control signal at time i
μ_{jkl}	Height of the spike associated with rule set (j, k, l)
Δu	Change of control signal u
$U(i)$	Physical control command to plant
$\Delta U(i)$	Change of $U(i)$

v_a+	Average frequency of positive slope crossings of level y=a
v_0^+	Average frequency of positive slope crossings of level y=0
V^j	D_4 scaling function at scale level j
w	Noisy wavelet coefficients
w_0	Noise-free wavelet coefficients
w_{thr}, w'_{thr}	Wavelet coefficients after thresholding and its derivative
w_{thr_c}	Customized thresholding function
w_{thr_h}	Hard-thresholding function
w_{thr_s}	Soft-thresholding function
$W[f(u, s)]$	Wavelet transforms of signal $f(t)$ in terms of u and s
W^j	D_4 wavelet function at scale level j
x_i	Vibration sensor signal at time i
y	Variable stands for a straight line
α	Scale parameter of Weibull distribution
γ	Shape parameter of Weibull distribution
ζ	Customized thresholding function parameter
η	Independent and identically distributed Gaussian noise
μ_X	Mean of random variable X
ρ	Adaptation gain
ρ_{X1X2}	Covariance coefficient of bivariate random variables X_1 and X_2
ς_{te}	Incremental vector
σ_X, σ_{wthr}	Standard deviation of stationary random variables X and w_{thr}
τ	(i) Target time constant (ii) Time difference
τ^*	"Optimal" value of target time constant
τ_p	Time constant of control system
ϕ	Scaling functions of continuous wavelet transform
ψ	Wavelet functions of continuous wavelet transform
Ψ	Fourier transform of wavelet function ψ
ω	Frequency

Chapter 1 Introduction

1.1 Overview

Maintaining a reliable and productive machining process is the key to success in metal cutting industry. Chatter, namely the violent self-excited vibration between the workpiece and cutting tool, has been considered one of the most important causes of instability in the cutting process. It not only limits productivity of machining processes but also causes poor surface finish, reduced dimensional accuracy, escalated tool wear and noisy workplace. Therefore it is essential for designing an effective system to detect and suppress chatter. Vibration in machining can be classified in terms of two types: forced and self-excited vibration. Forced vibration is caused by cyclic variation in cutting force (as in milling). Self-excited vibration is defined as an instant of dynamic instability in machining which results in the tool to vibrate relative to the workpiece. Self-excited vibration is also separated into two types: regenerative and non-regenerative. Regenerative chatter is caused due to undulation on the surface of the workpiece, produced during previous successive cuts. Non-regenerative chatter is very complex and is inherently related to the dynamics of the cutting process. The major attention of this study is concentrated on regenerative chatter.

1.2 Motivation and objective

Over the past few decades, various approaches have been proposed to address the chatter problem. Two approaches used for chatter detection include signal processing techniques and stability analysis of cutting processes. The methods for chatter suppression are classified as active control and passive control. However, their applications have been very limited due to various drawbacks as summarized in the following.

(1) Limitations of existing chatter detection techniques

- Require the knowledge of machining process dynamics and the characteristics of machine-tool-workpiece structure for stability analysis and model development, which is very time-consuming and has to be re-calibrated for any changes in

machine, tool and workpiece. This is obviously not suitable for a modern manufacturing environment featuring frequent work order and tool changes.

- Often need excessive computing power for calculating detection index and hence are difficult for on-line implementation.
- Rely on pattern recognition techniques through learning processes, which is knowledge-demanding and cannot meet the industrial need for easy and cost-effective application.
- Overlook the non-stationary property of vibration signals and are prone to errors due to noise contamination and external disturbances. For instance, the data collected from machine tool are often considered as stationary signals. However, the transient signals from machining processes can sometimes provide information about machine conditions which cannot be revealed from stationary signals. Therefore, conventional stationary signal analysis approaches cannot be used. Instead, non-stationary signal analysis approaches should be implemented.

(2) Limitations of existing chatter suppression techniques

- Require experimental identification of stability lobe diagram (STD), which is subjected to constant changes due to cutting dynamics and is rarely used in an industrial setting.
- Depend on a properly selected set of nominal spindle speed and frequency of speed modulation, which itself is an open research topic. It should also be pointed out that ill-selected such parameters may cause machine failure and could even propagate chatter.
- Involve modification of machine tool structure with the added dampers and re-design of tool geometry. This is very costly and leads to a long period of downtime for existing machines.
- Have a very limited application potential for state feedback control and optimal control approaches since they are only suitable for time-invariant linear systems.

Although some approaches have shown to be successful in detecting and suppressing chatter under certain machining conditions, there is still a compelling need to further investigate the underlying problem to overcome those drawbacks mentioned above. Consequently, in this study a new system is developed consisting of chatter detection

module and suppression module. It is suitable for on-line implementation since the system program is coded in C language and with polynomial computing complexity. The two modules are briefly presented as follows.

(1) The proposed chatter detection module

A new statistical index (SI) is proposed based on the discrete wavelet coefficients and statistical analysis of positive wavelet transform modulus maxima. In addition, due to the noisy machine environment, a customized thresholding de-noising method featuring a level-dependent universal threshold rule is implemented. The proposed *SI* is independent of cutting conditions and does not require the time-consuming re-calibration in different machining processes. This is because it is dimensionless and varies in the range of [0, 1]. Such a range does not change with variations in machine, process, tool and workpiece.

(2) The proposed chatter suppression module

Both plain fuzzy control and fuzzy control with self-regulating approaches are adopted for chatter suppression in end milling process. The self-regulating algorithm also characterizes on-line regulation of fuzzy control output and auto-tuning of controller parameters based on a cost function. The use of fuzzy control strategy does not need to establish process model. Moreover, it can compensate for our incomplete knowledge of system dynamics and tolerate imprecise representation of the system. Machining processes are complex and non-linear, and the control plant contains unknown parts such as feed and spindle controllers and it is almost impossible to obtain their derails since they are based on proprietary technologies.

1.3 Organization

The remainder of this study is organized as follows. Chapter 2 provides a brief literature review of methods or theories on chatter detection and suppression. Then the proposed chatter detection module is reported in Chapter 3. Chapter 4 presents the chatter suppression module. The experimental results and discussions are given in Chapter 5. Chapter 6 concludes the thesis and suggests some possible future work.

Chapter 2 Literature Review

Automation of machining operations and machine tools at the process level has been a focus of research attention in both academia and industry for several decades. In particular, the automation technology for machine tools plays a critical role in improving productivity and part quality, cost, and relaxing part design constraints. However, the above expectations have long been impeded by the influencing factors such as tool conditions and machine tool chatter. Since chatter is recognized as one of the most significant factors limiting the general performance of machine tools, a significant amount of research work has been devoted to chatter detection and suppression. There is no doubt that the success of manufacturing process automation relies largely on the effectiveness of process monitoring and control systems. Therefore in this chapter the state of the art of machining process monitoring and control approaches for chatter vibration are reviewed. Although in comparison to other manufacturing processes, milling is one of the most widely used cutting processes, some methods and techniques reported in this chapter are also equally applicable to other machining processes.

2.1 Review of chatter detection and prediction approaches

The research efforts to delineate the causes of chatter vibration and its prediction have been on going since as early as 1946. At the early stage, chatter was mostly attributed to negative damping effect (Arnold 1946). This idea of negative damping was later challenged by Tobias (1958), Tlusty and Polacek (1963), and Koenigsberger and Tulsty (1967) who recognized that the most powerful sources of chatter or self-excitation vibration were due to the regenerative and mode coupling effects. These effects stem from the interaction of the structural dynamics of the workpiece and machine tool, and the feedback of subsequent cuts of the tool. Since then, two tendencies of methodology to study regenerative chatter have been evolved.

2.1.1 Signal processing techniques

One trend of study fully implements signal processing and analysis techniques to detect chatter in an efficient and robust manner. Investigation of spectral density of a process

signal along with development of a threshold value to indicate chatter is the most commonly used method. Sound pressure was examined as the process signal by Delio *et al* (1992), and Altintas and Chan (1992). Tarng and Li (1994) created threshold values for the spectrum, the standard deviation of thrust forces and torque signals in machining operations. Power spectrum of measured displacement for deflections of workpiece was studied by Rahman (1988). Ismail and Kubica (1995) developed a chatter detection indicator, R-value, which was a ratio between two root mean square (RMS) values of dynamic forces pertaining to the chatter and static forces reflecting the cutting geometry. However, one disadvantage of thresholding algorithms is that the empirically selected threshold value may not be valid over a wide range of cutting conditions. As a result, the difficulty in determining the suitable threshold values has led to the studies of artificial neural network-based, multi-sensor-based or fuzzy set theory (Li *et al* 1998 and Du *et al* 1992).

Bailey *et al* (1995) suggested that a more general signal, i.e., an accelerometer signal mounted on machine tool structure close to the cutting region, can be processed to calculate the so-called variance ratio $R = [\sigma_s/\sigma_n]^2$. The parameters σ_s and σ_n are the variances of accelerometer signal in low and high frequency ranges, respectively. A value of $R<<1$ indicates the presence of regenerative chatter. In the work by Li *et al* (1997), two accelerometers were used and the coherence function of two perpendicularly measured acceleration signals was proposed as a normalized index for chatter detection. However, the coherence function index may subject to the non-stationary nature of acceleration signals so that occasionally it appears over-vigilant and causes no evident indication of chatter occurrence as commented in (Xu 2002).

Though it seems that the conventional stationary signal analysis approaches such as fast Fourier transform (FFT) are very appealing in examining chatter, they are not accurate and reliable enough for complicated machining processes due to the presence of non-stationarity and non-linearity. This disadvantage has given rise to the implementation of non-stationary signal analysis approaches. In (Gu *et al* 2001), a singular value decomposition technique based on Choi-Williams time-frequency distribution analysis was presented. The extracted singular values served as signatures for the corresponding time-frequency distribution data and were exploited as features (indexes) to monitor the machining chatter condition.

In recent years, wavelet transforms have been widely studied by many researchers and used as tools to explore the dynamic characteristics of cutting processes (Berger *et al* 1998 and Khraisheh *et al* 1995). As compared to other techniques which are vulnerable to noise contamination and inept to non-stationary signals, wavelet transforms possess the advantages of performing local analysis, handling non-stationary or transitory signals, providing time-frequency domain analysis, and compressing measurement data without appreciable degradation. Choi and Shin (2003) proposed a chatter detection methodology by using wavelet-based maximum likelihood (ML) parameter estimation algorithm. In their study, the physical signals from cutting process were depicted to be $1/f$ process which shows "self-similar" property in terms of invariant behavior on different time scales. A spectral parameter γ was estimated with maximum likelihood approach and used as a chatter detection index. The approach demonstrated a very encouraging way to study randomness involved in cutting processes. In (Wu and Du 1996), the authors proposed a feature extraction and assessment method by using wavelet packet transform to monitor machining processes. Wavelet packets that contain large amounts of information were used as features, called feature packets. The feature packets were further selected and evaluated according to four criteria: cross-correlation and cross-coherence of original signal and reconstructed signal from feature packets, correlation of residual signal, and power spectrum of residual signal. Eventually the index based on peak-to-valley value of the selected feature packets was formulated to detect the onset of chatter.

More advanced signal analysis methods such as the Hidden Markov Model (HMM) have been emerging and applied to condition monitoring area just in past few years. The prevalence of HMM results from its rich mathematical structure and proven accuracy on critical applications once each symptom of the system is well trained. Lee and Hwang (2000) applied the HMM method to chatter signal analysis and prediction. Furthermore, a method using continuous time HMM (CHMM) with autoregressive (AR) coefficients was developed to detect and predict chatter in a lathe machine and to diagnose wear of journal bearing. The accuracy and early trend detection ability were validated with experimental data. In spite of the above mentioned advantages, the application of the HMM approach is still limited by training procedure for different patterns. As a result, the industrial application of HMM for chatter detection has not yet been reported in the accessible literature.

2.1.2 Stability analysis techniques

The second trend to study chatter over the past few decades refers to the stability analysis techniques. These techniques involve the modeling and theoretical analysis of cutting process and structural dynamics. The stability lobe diagram (SLD) approach, one of the earliest analysis techniques, was developed by Merritt (1965) who generated specialized plots from the harmonic solutions of the system's characteristic equation to determine system stability and construct the SLDs. After that, many researchers have used Nyquist techniques to generate SLDs (e.g., Sridhar *et al* 1968, Minis *et al* 1990a and 1990b, Lee and Liu 1991, and Minis and Yanushevsky 1993).

Time domain simulation (TDS) is another technique that can be exploited to generate SLDs (Tlusty and Ismail 1981 and1983, Tlusty 1986, Tsai *et al* 1990, Smith and Tlusty 1993, Elbestawi *et al* 1994, Weck *et al* 1994). In this approach, the closed-loop dynamics model of the machining operation is simulated for a particular set of cutting conditions. Steady states of tool-workpiece displacement and machining force signals are examined to determine system stability. The critical depth of cut is adjusted until marginal stability is encountered. The strength of TDS is that it incorporates all major aspects of the machining operation including nonlinear characteristics. However, the inherent drawback to use TDS is its computational burden.

Several researchers also investigated the effect of the force-uncut chip thickness nonlinear relationship on machining chatter. Zhang and Ni (1995) proposed linearization techniques to investigate the stability of linearized closed-loop machining model via the criteria developed from the phase difference between the current and previous tooth passes. Endres (1996b) used TDS and an extension of a linear energy-based analysis (Endres 1996a) to inspect the stability limits for a one dimensional orthogonal machining system.

Due to the TDS's considerable computational time, the numerical time domain analytical approaches for predicting chatter stability have gained more attention. Minis and Yanushevsky (1993) improved the early analysis work by applying the theory of periodic differential equations, i.e., the Floquet's theorem and Fourier analysis, on the milling dynamic equations. In the studies by Budak and Altintas (1995a and 1995b), an alternative analysis solution was provided. Instead of using the Floquet's periodic system theory, they

solved the stability problem by extending the theory (Tlusty and Polacek 1963) which was based on the physics of orthogonal cutting and the regenerative mechanism.

In addition to the above, some researchers showed their growing interest in the transition of cutting dynamics at the onset of chatter. Grabec (1988) indicated that the cutting process contained chaotic dynamics. This supposition was then utilized in chatter detection through coarse-grained entropy rate which is based on a transition from high dimensional to low dimensional dynamics of cutting at the onset of chatter by Gradisek *et al* (1998). Meanwhile, the authors also suggested that a measure of nonlinearity in a cutting process could be used as a chatter detection index. Nevertheless, the calculation of the measured nonlinearity was so complicated that it is not suitable for on-line implementation.

In summary, it is apparent that the stability analysis techniques are theoretically valuable in chatter study. However, these techniques suffer from the difficulty in identifying and tracking the stability lobes because of the changing machining conditions and structural dynamics (Liang *et al* 2004). As a result, they cannot meet the requirements necessary for industry adoption and are rarely used in an industrial setting.

2.2 Review of chatter suppression and avoidance approaches

Chatter suppression or avoidance by active and passive means has been attempted by a number of researchers for many years. However, these attempts were often premature and only limited success was achieved. It is noticed that the active approaches need the modification of machine tool structure, hence impractical as far as cost and easy implementation are concerned. The passive approaches conduct the control theory together with computer technology to suppress chatter. They are more widely investigated by the academia due to their application potential. The following sub-sections summarize the chatter suppression or avoidance approaches investigated by previous researchers.

2.2.1 Active control approaches to chatter suppression

To avoid chatters, Slavicek (1965) and Vanherck (1967) proposed the use of milling cutters with non-uniform tooth pitch. Stone (1970) used end mills with alternating helix for the same purpose. The problem with these techniques is that the design of non-uniform pitch cutter is process-specific and cannot be applied to other processes such as single point machining.

Wang and Lee (1996) described a complete procedure for suppressing chatter of a machine centre. The weak component of an existing machine centre was first identified by vibration testing and a cutting process. Sensitivity analysis was then performed to optimize the weak component which was recognized to be the spindle. Thus a new spindle was manufactured according to the optimization result. More than 100% improvement in cutting capacity was reported in their work.

Sato and Hori (1981) suggested the use of the perspective view coordinate system, equi-contour amplifier and two-dimensional Fourier analysis for the display and characterization of two-dimensional surface roughness. The self-excited chatter marks were taken as an example. Vibration control has also been achieved with help of an active dynamic absorber to improve the cutting process stability using a boring bar (Tewani *et al* 1995). Alternatively, this was also done by introducing unevenly spaced inserts in milling operation (Choudhury and Mathew 1995).

The modeling of chatter control systems and design of corresponding control strategy were also explored by some researchers. A systematic and unified approach (Pan *et al* 1996) was proposed to the modeling and active control of chatter in a lathe machine with non-collocated actuator. Two advanced control algorithms, least mean square (LMS) adaptive filter and fuzzy cerebellar model arithmetic computer (CMAC) neural network were presented. The objective was to make the system have more damping through the use of specially designed magnetostrictive alloy Terfenol-D actuators and accordingly chatter could be suppressed. In the work of Choudhury *et al* (1997), an on-line vibration control system for turning was developed to control machine tool vibration aiming at increased productivity and improved machining accuracy. With the aid of a bifurcated bunch of optical fibers and by phase-shifting, amplifying and feeding back the signal to a specially designed piezoelectric vibrator, the relative vibration between the workpiece and the cutting tool was sensed. As a result, whenever a vibration occurred which led the system to instability, the closed-loop feedback contour would be able to generate an equal and opposite force to stabilize the vibration using a vibrator-exciter. The mathematical model for the feedback control system was established in their study.

Some other specific techniques for suppressing chatter were also reported. Xiao *et al* (2002) investigated a method to suppress chatter by applying vibration cutting without

relying on the tool geometry. They proposed a cutting model of the vibration cutting process to predict the work displacement amplitudes. Experimental results showed that chatter was effectively suppressed when applying vibration cutting. However, the underlying method required a special experiment setup, in particular, a vibrated tool realized through the use of an ultrasonic electrostriction transducer.

The above review clearly indicates that though many active control methods have been proposed, their inherent problems such as the needs for change of tool geometry, use of specialized devices and structure modification of sub-systems of machine tool have made them technically infeasible or economically unjustified in real application. Therefore this technique has not been widely accepted by industry.

2.2.2 Passive control approaches to chatter suppression

Regenerative chatter may be avoided and productivity may be improved by selecting proper spindle speeds lying in one of the stability lobes. To determine the so-called stability lobe diagrams, the stability analysis of the closed-loop system formed by the machining force process and the tool-part structure must be performed. Weck *et al* (1975) utilized on-line generated stability lobes to select a spindle speed so that the depth-of-cut limit can be maximized. However, it is well known that the utilization of SLDs suffers from modeling inaccuracies and very often it is economically not viable to model the structural characteristics of the machining operations that change frequently. Minis and Yanushevsky (1993) presented a method for the prediction of chatter in milling. The dynamics of a milling process was described by a set of differential-difference equations with periodically time-varying coefficients. The stability was examined using Fourier analysis. However, the proposed method still required the knowledge of cutting process dynamics. In other words, system identification of the complex cutting dynamics for the machine-tool-workpiece structure has to be performed.

To overcome the difficulties in using the stability chart, automatic chatter suppression by adjusting the spindle speed set point was introduced. The rationale is that the change of spindle speed can rectify the phase shift between the inner and outer modulation. An automatic spindle speed selection (SSS) methodology (Delio *et al* 1992) was developed and applied to milling operations. A chatter detection algorithm using sound pressure was developed to determine the chatter frequency. The tooth passing frequency was adjusted to

equal the chatter frequency and this procedure was repeated until chatter was suppressed. Smith and Tlusty (1992), and Tarng *et al* (1996) also used similar strategy to avoid the need for identification of stability lobes. This SSS approach is adaptive since the spindle speed is changed based on feedback measurement of the chatter frequency. It is practical for high spindle speed machining when the stability lobes are well separated as stated in (Al-Regib *et al* 2003).

Spindle speed variation (SSV) is another promising technique for chatter suppression. Lin *et al* (1990) varied the spindle speed in a sinusoidal manner to suppress chatter in a face milling operation. Zhang *et al* (1994) examined the effects of amplitude and frequency when varying the spindle speed in a sinusoidal manner and developed an analytical technique to determine the optimal values of these two parameters. Radulescu *et al* (1997) numerically and experimentally investigated the effects of SSV when face milling parts with complex structures. Soliman and Ismail (1997) proposed an approach by using fuzzy-logic to on-line selection of amplitude and frequency of the forcing speed signal. Furthermore, Yilmaz *et al* (2002) generalized the sinusoidal spindle speed variation (S^3V) technique by introducing multi-level random spindle speed variation, where the spindle speed was varied in a random fashion within the maximum amplitude ratio allowed by the spindle-drive. Recently, Al-Regib *et al* (2003) presented a systematic method to suppress chatter through programming the amplitude and frequency of spindle speed variation signal. Energy-based stability analysis was studied in order to examine the effect of the variable spindle speed signal on the work done by the regenerative force. The simple and heuristic criteria for selecting optimal S^3V amplitude ratio and frequency of the forcing speed signal were developed so that the resulting signal would ensure fast stabilization of the machining process. The SSV methodology may also be implemented passively via the design of milling tools whose teeth are not evenly spaced as suggested in (Altintas *et al* 1999).

Although SSV is a favorable technique, there exists no systematic way to select the proper parameters such as amplitude and frequency of the sinusoidal forcing signal. The selection of the parameters depends on the dynamics of machine system and is also constrained by the spindle-drive system response and its ability to track the forcing speed signal. In some situations SSV could create chatter which may otherwise not occur when using a constant spindle speed.

Besides the spindle speed modulation approaches, some other methods for chatter suppression were also attempted. For example, Subramanian (1976) found that cutting stability could be enhanced by increasing feed rate. Landers and Ulsoy (1996) demonstrated that feed had a monotonic affect on regenerative chatter and sometimes could be used by machine tool operators for chatter suppression. Yang *et al* (1997) proposed a chatter suppression method by adjusting time-varying cutting parameters such as the time-varying rake angle and time-varying feed rate. It should be noted that in theory the depth-of-cut may also be decreased to suppress chatter. However, this approach is typically not employed since the part program must be rewritten and productivity is drastically reduced. In another study (Shiraishi *et al* 1988 and 1991), chatter was suppressed by utilizing state feedback and optimal controls. The displacement signals from a proximeter were fed back to the state estimator and then the instantaneous radial tool position was computed. The updated command was executed by a tool holder servo-actuator. Although different convergence performance was obtained, the approach was only suitable for time-invariant linear systems.

2.3 Summarization of reviewed methods

Pros and cons of above reviewed methods for chatter detection and suppression are summarized as follows.

(1) About the chatter detection methods

- Conventional signal processing techniques such as Fourier transform and short-time Fourier transform suffer from the drawbacks of not providing any information regarding the time localization of spectral components and not knowing how to choose the shape of windowing function and the width of window. Some advanced signal processing techniques like HMM require the training process for pattern recognition, and therefore are not suitable for on-line applications.
- Stability analysis approach has a significant contribution to the theoretical study of chatter and to the understanding of chatter mechanism. However, the stability lobes themselves are not stable and may shift due to the change of tool-workpiece structural dynamics. The "optimal" cutting speed obtained based on stability analysis may not even be feasible in terms of stability, let alone "optimal". This approach is therefore useful to provide some theoretical insights but not suited for real applications.

As discussed in Section 2.1.1, because of wavelet transforms' ability to decompose a signal at different independent scales and to zoom in the localized signal structure, wavelet-based analysis is becoming a new problem-solving tool in science and engineering areas. Moreover, if the signal is represented as a function of time, wavelet transforms can provide efficient analysis in both time and frequency domain no matter the signal is stationary, transitory or not. Therefore it is adopted in this study for chatter detection.

(2) About the chatter suppression methods

- Although the active chatter control strategy provides an alternative means to chatter suppression, it is not viable both technically and economically due to the cost and time required in modifying machines and tools.
- The passive chatter control strategy so far is still the most applicable way to suppress chatter. As reviewed in Section 2.2.2, although it is straightforward to suppress chatter by reducing the depth-of-cut, the consequences is reduced productivity. The stability lobe diagram approach is not practical either due to the inaccuracy in modeling and the "swing" of the stable zones. Recently, the spindle speed variation in either sinusoidal or random manner has attracted more attention than any other approaches. However, the disadvantages include the lack of systematic way to optimize the speed variation and the inflexibility in determining the S^3V parameters due to the change of cutting conditions or system configurations.
- Even though the state feedback control and optimal control have not been successful in a nonlinear and time-varying environment, alternative control methodologies, if versatile and easy to implement, are still appealing because of the existence of control infrastructure on modern machines.

Therefore in this study, two versions of control strategies, i.e., plain fuzzy control and fuzzy control with self-regulating capacity are proposed to suppress chatter. The latter control strategy has the ability to on-line adjust controller parameters. As a result, the fuzzy output can be modified automatically. In addition, both control methods are able to adapt to the dynamic systems with non-linear variation and uncertainties. Thus many of the drawbacks of traditional control techniques mentioned above can be avoided. Furthermore, no model estimation and identification are required and the dependence on domain experts is also limited to a minimal extent.

Chapter 3 Development of Chatter Detection Module

In this chapter, a chatter detection module is presented. The methodology used is based on the study of discrete wavelet transform (DWT) scheme and statistical analysis of wavelet transform modulus maxima (WTMM). Wavelet transform modulus maxima is used to describe any point such that wavelet transform of a signal is locally maximal at corresponding time location. Meanwhile, due to noisy machining environment, a wavelet-based de-noising method including a customized thresholding function and a level-dependent universal threshold rule is proposed. The basic principles and detection procedures are given below and shown in Figure 3.1. For the suggested methodology, since prior knowledge of dynamics of machine-tool-workpiece structure and machining process is not required, in-process sensing data assessment becomes very critical. The key idea is the peak values of wavelet coefficients, i.e., wavelet transform modulus maxima, contain the important information about transient periods such as during machine start-up, machine tool engaging cutting or feed direction changing and maximum changes such as jumps or singularities in the signal. Therefore the statistical distribution of these peaks can show different trends or patterns which are used to distinguish faulty condition from the normal cutting conditions. Moreover, statistical analysis leads to the formulation of a non-dimensional detection index which overcomes the difficulty in empirically determining the threshold values. For this purpose, DWT and wavelet-based de-noising algorithm are first conducted at stage 1 to decompose the vibration signal into individual frequency bands and to reduce the noise contained in the raw sensing signal. The clean wavelet coefficients are then used for next feature extraction. At stage 2 the chatter detection features are extracted in terms of wavelet transform modulus maxima. The suspicious band is targeted according to a relative power discriminating criterion. Consequently the modulus maxima within the targeted band are used to formulate chatter status index via statistical distribution analysis. At stage 3, the final decision on the onset of chatter is made through comparing the calculated detection index with a reference threshold value which is determined by experimental tests.

3.1 Wavelet transforms and wavelet transform modulus maxima

The first step of the proposed methodology is data transformation for feature extraction. Wavelet analysis has been widely used in image and speech processing for several decades. However, recently it has received more and more attention in manufacturing process monitoring field since it proved to be an effective tool and a new method to interpret the information contained in the signal. A brief review of wavelet transforms and wavelet transform modulus maxima is given below.

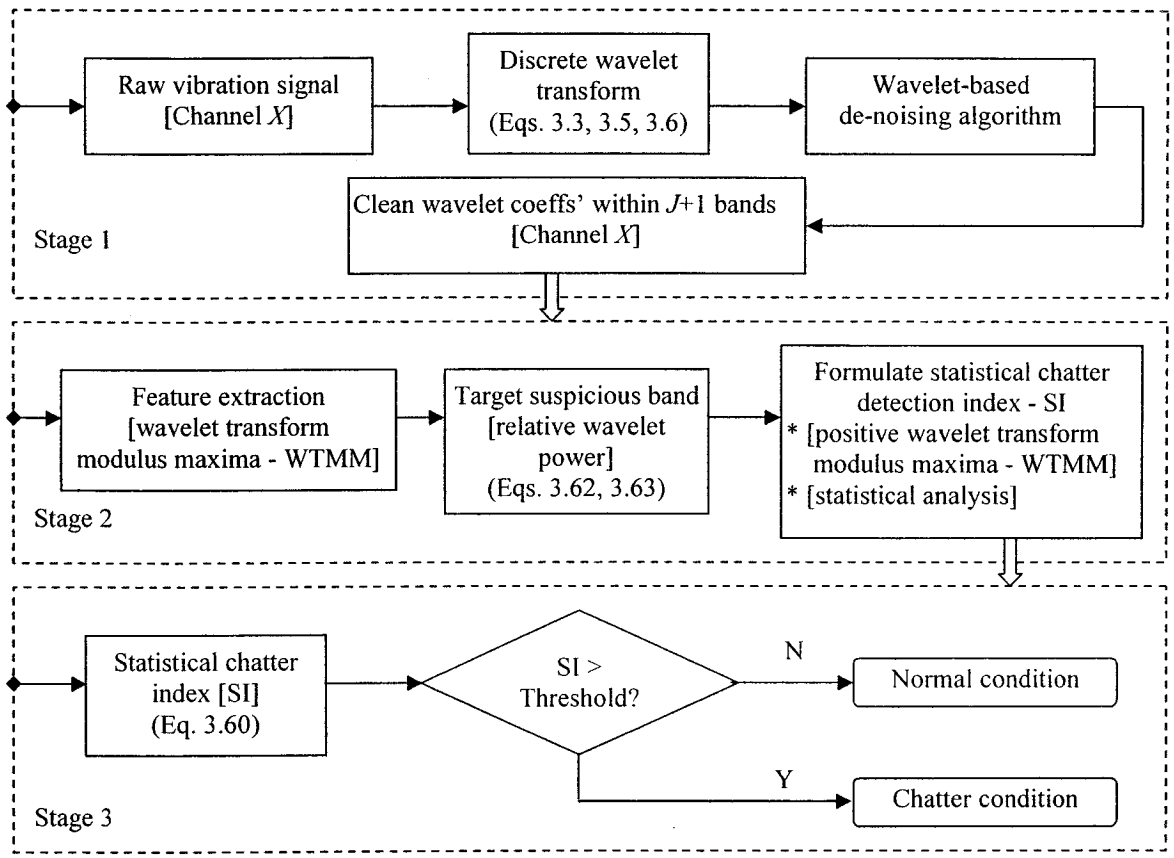


Figure 3.1 Flow chart of chatter detection module

3.1.1 Theoretical background of wavelet transforms

In (Mallat 1998), the continuous wavelet transform (CWT) is defined as follows:

$$W[f(u,s)] = u^{-1/2} \int_{-\infty}^{\infty} f(t)\psi(\frac{t-s}{u})dt \tag{3.1}$$

where $\psi_{u,s}(t)=u^{-1/2}\psi((t-s)/u)$, the functions $\psi_{u,s}(t)$ are called "wavelets" and the function $\psi(t)$ is called "mother wavelet". $f(t)$ is a signal, u and s are named scale and location parameters varying continuously over real domain R, respectively.

When we choose $u=u_0^j$, $s=k\cdot u_0^j\cdot s_0$, where j, k range over integer domain Z, $u_0>1$ and $s_0>0$ are fixed, then $\psi_{u,s}(t)$ is transformed correspondingly into:

$$\psi_{u,s}(t)=\psi_{j,k}(t)=u_0^{-j/2}\psi(\frac{t-ks_0u_0^j}{u_0^j})=u_0^{-j/2}\psi(u_0^{-j}t-ks_0) \tag{3.2}$$

Consequently, the discrete wavelet transform (DWT) is obtained (Daubechies 1992):

$$W[f(j,k)]=u_0^{-j/2}\int_{-\infty}^{\infty}f(t)\psi(u_0^{-j}t-ks_0)dt \tag{3.3}$$

From the perspective of multi-resolution analysis (MRA) theory to view the wavelet transform, a function or signal can be considered as composed of a smooth background and fluctuations or details on top of it. The distinction between the smooth part and the details is determined by the resolution, the scale below which the details of a signal cannot be discerned. At a given resolution, a signal is approximated by ignoring all fluctuations below that scale. When the resolution increases progressively, at each stage finer details are added to the coarser description, providing a successively better approximation to the signal. Eventually as the resolution goes to infinity, the exact signal is recovered. Any signal $f(t)$ can be reconstructed by the equation below in any level of scale (Erlebacher *et al* 1996):

$$f(t)=\sum_{j_0=-\infty}^{\infty}\sum_{k=-\infty}^{\infty}c_{j_0,k}\phi_{j_0,k}(t)+\sum_{j=-\infty}^{j_0}\sum_{k=-\infty}^{\infty}d_{j,k}\psi_{j,k}(t) \tag{3.4}$$

where $c_{j0,k}$ stands for the approximation coefficients at scale j_0 and $d_{j,k}$ represents the detail coefficients at scale j_0 and below. The first part of the above equation is scaling-function-dependent approximation of $f(t)$ at scale j_0. The second part of equation contains wavelet-function-dependent details of $f(t)$ at scales j_0 and below. The function $\phi_{j,k}(t)$ is called the scaling function (or dilation function) in multi-resolution analysis with $\phi_{j,k}(t)=u_0^{-j/2}\phi(u_0^{-j}t-ks_0)$.

As shown in (Erlebacher *et al* 1996), for any given scaling function, the approximation coefficients can be obtained as:

$$c_{j,k}=\int_{-\infty}^{\infty}f(t)\phi_{j,k}(t)dt \tag{3.5}$$

Similarly, the detail coefficients can be obtained as:

$$d_{j,k}=\int_{-\infty}^{\infty}f(t)\psi_{j,k}(t)dt \tag{3.6}$$

Table 3.1 summarizes some relations (Erlebacher *et al* 1996) associated with the MRA theory. Note that in the table, coefficient sets $H=\{h_k\}_{k=0}^{L-1}$ and $G=\{g_k\}_{k=0}^{L-1}$ are related by $g_k=(-1)^k h_{L-k}$ for $k=0,\ldots,L-1$. L is the total number of dilation function coefficients h_k and wavelet function coefficients g_k.

Table 3.1 General relations in the MRA theory

Approximation space V_j	Detail space W_j
$V_j \subset V_{j+1}$	$W_j \subset W_k$ for $j \neq k$
Scaling function ϕ	Mother wavelet ψ
Dilation equation: $\phi(t)=2^{1/2}\sum_k h_k \phi(2t-k)$	Wavelet equation: $\psi(t)=2^{1/2}\sum_k g_k \phi(2t-k)$
Orthonormal basis for V_j: $\{\phi_{j,k}(t)=2^{-j/2}\sum_k \phi(2^{-j}t-k)\}$	Orthonormal basis for W_j: $\{\psi_{j,k}(t)=2^{-j/2}\sum_k \psi(2^{-j}t-k)\}$

In this study, discrete wavelet transform is adopted due to its following advantages (Watson and Addison 2002): (a) providing a fast method of signal decomposition, (b) guaranteeing both energy conservation and exact signal reconstruction, and (c) necessitating the use of orthonormal wavelets, and dilation levels are set in the form of "octaves" (integer powers of two). Moreover, as remarked in (Daubechies 1992) the orthonormal wavelet bases can be viewed as a tool to mathematically describe the "increment in information" needed to go from a coarse approximation to a higher resolution approximation. In view of the above, Daubechies wavelet D_4 is chosen as a data transform since it is orthonormal and commonly used.

The Daubechies D_4 transform has four wavelet and scaling function coefficients (Walker 1999). The scaling function coefficients are:

$$h_0=\frac{1+\sqrt{3}}{4\sqrt{2}}, h_1=\frac{3+\sqrt{3}}{4\sqrt{2}}, h_2=\frac{3-\sqrt{3}}{4\sqrt{2}}, h_3=\frac{1-\sqrt{3}}{4\sqrt{2}} \tag{3.7}$$

The wavelet function coefficients are:

$$g_0=\frac{1-\sqrt{3}}{4\sqrt{2}}, g_1=\frac{\sqrt{3}-3}{4\sqrt{2}}, g_2=\frac{3+\sqrt{3}}{4\sqrt{2}}, g_3=\frac{-1-\sqrt{3}}{4\sqrt{2}} \tag{3.8}$$

The Daubechies D_4 scaling and wavelet functions are calculated by the following equations (Walker 1999):

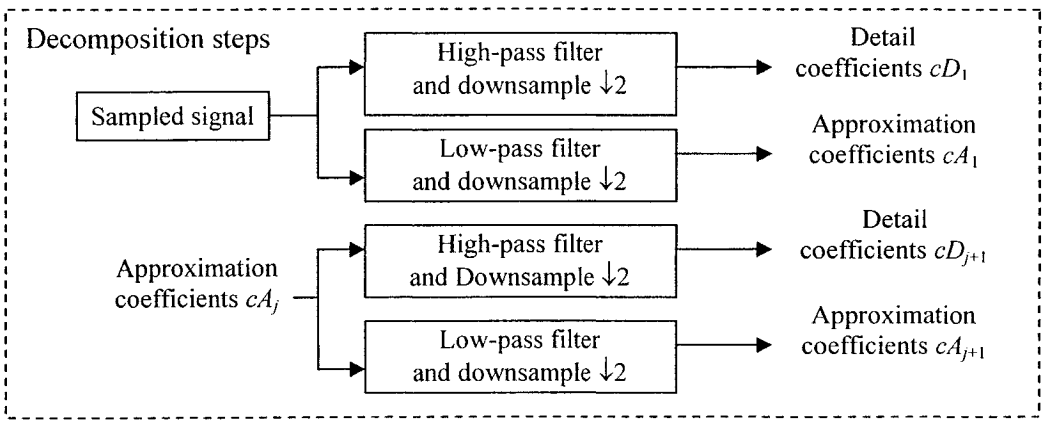
Scaling functions V_i^j at each scale level j for all i:

$$V_i^j = h_0 V_{2i}^{\ j} + h_1 V_{2i+1}^{\ j} + h_2 V_{2i+2}^{\ j} + h_3 V_{2i+3}^{\ j} \quad \forall i = 1,\ldots,N/2 \tag{3.9}$$

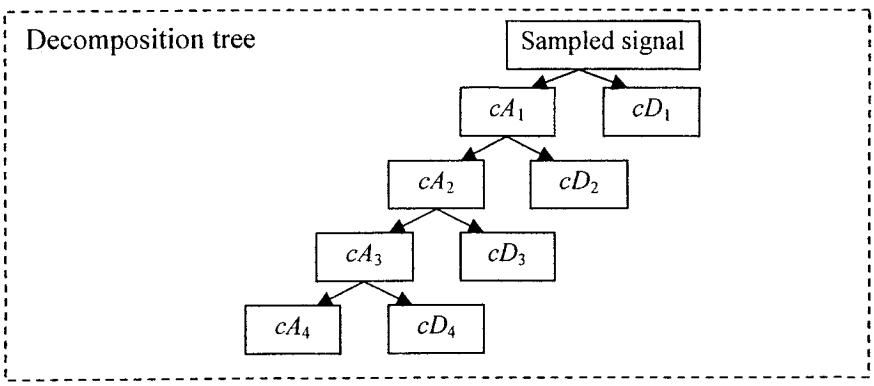
Wavelet functions W_i^j at each scale level j for all i:

$$W_i^j = g_0 V_{2i}^{\ j} + g_1 V_{2i+1}^{\ j} + g_2 V_{2i+2}^{\ j} + g_3 V_{2i+3}^{\ j} \quad \forall i = 1,\ldots,N/2 \tag{3.10}$$

Figure 3.2 (a) shows how discrete wavelet transform decomposes the sampled input data into different resolution levels. Starting with the sampled signal, the first step produces two sets of coefficients: approximation coefficients cA_1, and detail coefficients cD_1. The next step splits the approximation coefficients cA_1 in two parts using the same scheme, replacing sampled signal by cA_1 and producing cA_2 and cD_2, and so on. So the wavelet decomposition of the signal analyzed at level j has the following structure: [cA_j, cD_j, $cD_{j-1},\ldots,cD_1$]. Figure 3.2 (b) contains the terminal nodes of decomposition tree for J=4.



a) Wavelet transform decomposition steps



b) Wavelet transform decomposition tree

Figure 3.2 Wavelet transform decomposition steps and tree

3.1.2 Wavelet transform modulus maxima (WTMM)

As defined in (Mallat 1998), the term "modulus maxima" is used to describe any point (u_0, s_0) such that wavelet transform $|W[f(u, s_0)]|$ has the locally maximal value at $s=s_0$. Here u and s are still the wavelet transform scale and location parameters, respectively, and f is a function or signal. A more precise definition of the local maxima of wavelet transform modulus is given in (Mallat and Hwang 1992) as follows:

"Let $W[f(u,s)]$ be the wavelet transform of a function f:

- We call local extremum any point (u_0, s_0) such that $(\partial W[f(u_0, s)])/(\partial s)$ has a zero-crossing at $s=s_0$, when s varies.
- We call modulus maxima, any point (u_0, s_0) such that $|W[f(u_0, s)]| < |W[f(u_0, s_0)]|$ when s belongs to either a right or the left neighborhood of s_0, and $|W[f(u_0, s)]| \leq |W[f(u_0, s_0)]|$ when s belongs to the other side of neighborhood of s_0."

Figure 3.4 illustrates the wavelet transform of a given signal input as shown in Figure 3.3. The brightness bar on the right-hand side of Figure 3.4 means the visual representation of the magnitude of wavelet coefficients. The concepts of wavelet transform modulus maxima can be understood by referring to Figure 3.5. As shown in the plot (a) of Figure 3.5, for those modulus maxima (e.g., at s=330sec. and s=395sec.), the first partial derivatives with respect to s illustrated in the plot (b) are zeros. That is, the modulus maxima are associated with zero-crossing points. Singularities and irregular structures in the signal often carry the most important information such as the information given by transient phenomena like peaks (Mallat and Hwang 1992). On the other hand, wavelet transform (Mallat 1998) can focus on localized signal structures with a zooming procedure and the decay of wavelet transform modulus maxima across scales is able to characterize the local signal regularity. Thus the singularities of a signal can be detected and the oscillations can be measured from the wavelet transform modulus maxima as concluded in (Mallat and Hwang 1992).

3.2 Wavelet-based de-noising algorithm

The vibration signals during machining process are usually noisy. This is because the accelerometer for signal collection is mounted on the tool holder. The signals obtained from it include vibration information from several sources such as meshing gears in the gear

transmission box, spindle drive, cutting tool and many other running parts in the proximity. Therefore a very essential step for chatter detection is to remove noise from the acquired signal and make it clean. In this study the wavelet coefficients are obtained via the wavelet decomposition and then processed by a de-noising procedure. This procedure is based on a customized thresholding function with a level-dependent universal threshold rule as detailed in the following.

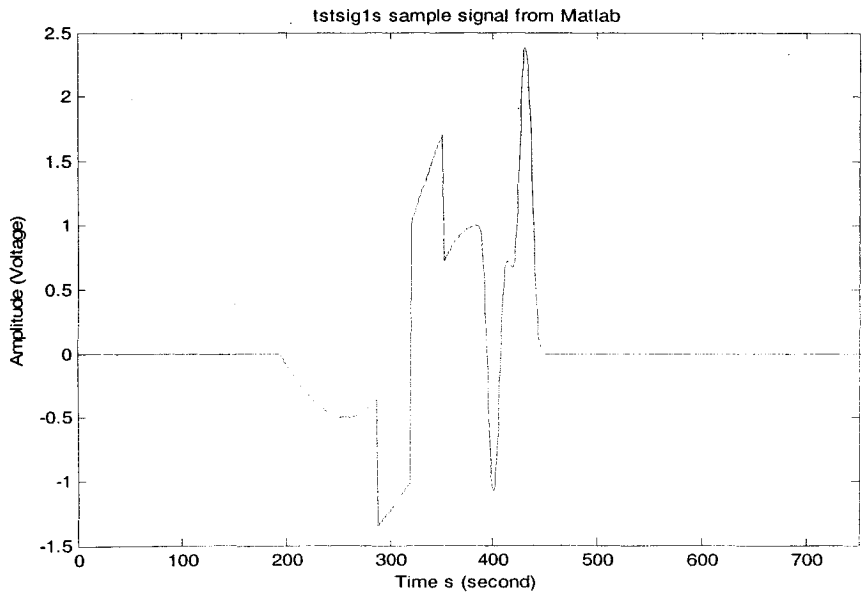


Figure 3.3 Time domain plot of a given input signal f

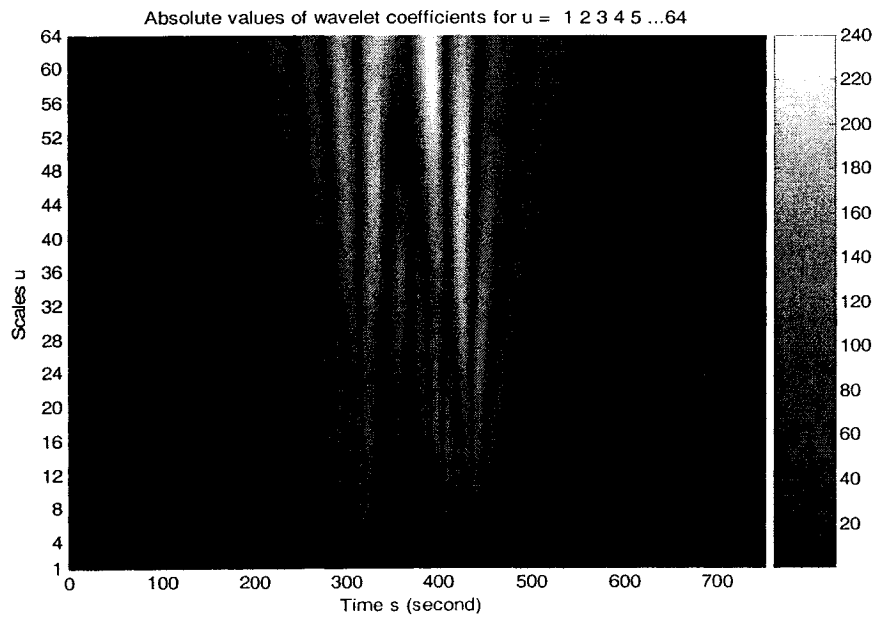


Figure 3.4 Wavelet transform plot (64 scale levels)

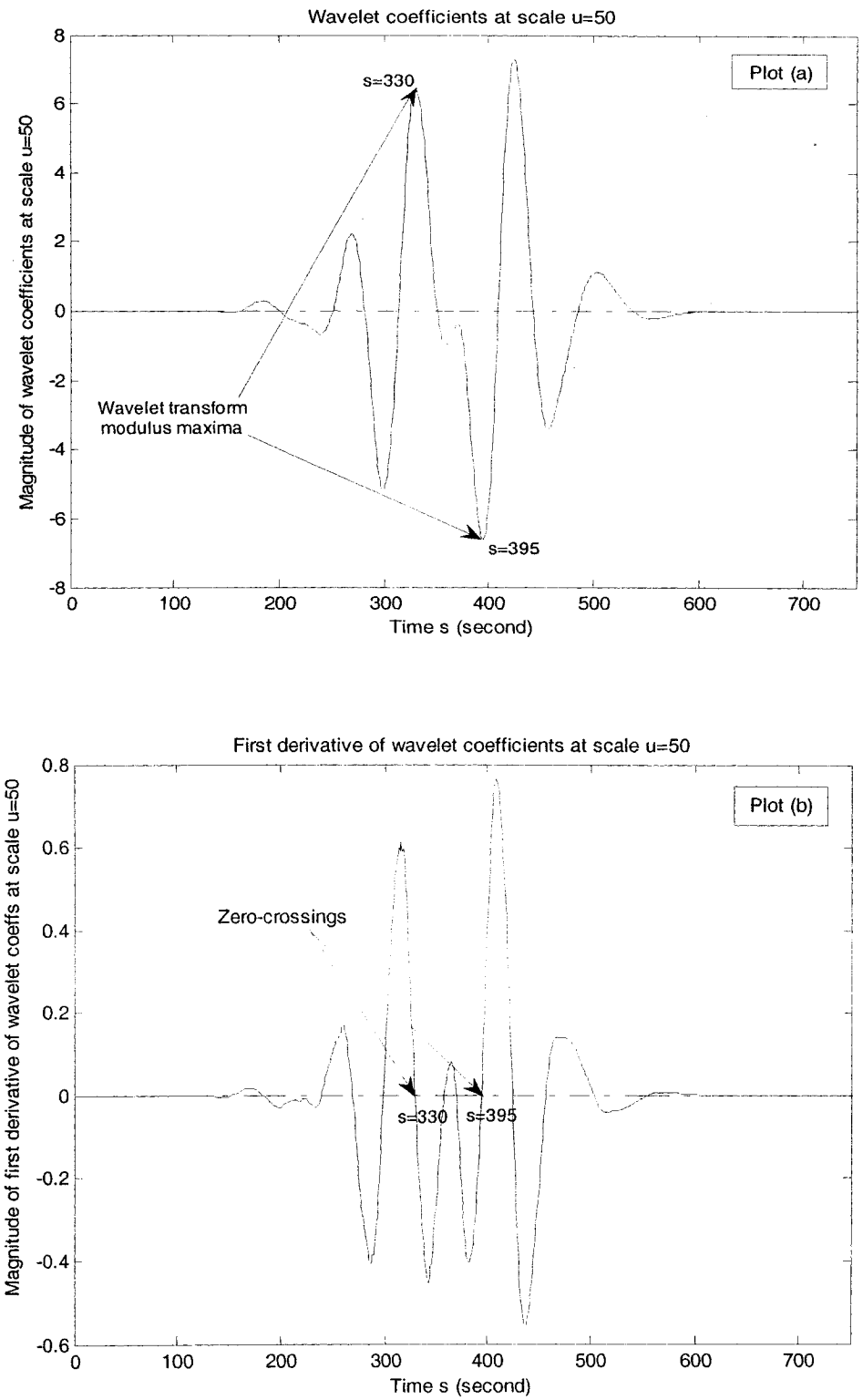


Figure 3.5 Plots of wavelet coefficients (a) and its first derivative (b) at scale u=50

3.2.1 Noise reduction by thresholding wavelet coefficients

Most noise reduction algorithm starts with the following additive model (Jansen 2001):

$$x(n) = f(n) + \eta(n) \tag{3.11}$$

where time n is equally spaced, $f(n)$ is a discrete signal of N data corrupted with noise $\eta(n)$. $\eta(n)$ is assumed to be independent and identically distributed (IID) Gaussian noise with mean zero ($E\eta$=0) and standard deviation σ. The objective of de-noising is to suppress the noise part of the signal $x(n)$ and to recover $f(n)$. The assumption of IID Gaussian noise can be understood in terms of the covariance matrix of noise η as follows.

- Covariance function is defined in the following form (Williams 2003):

$$\mathrm{cov}(x_1, x_2) = E\left[(x_1 - \mu_1)(x_2 - \mu_2)\right] \tag{3.12}$$

where E is the mathematical expectation and μ_i=$E[x_i]$ for i=1,2. The covariance function describes the variability tendency of two data sets x_1 and x_2 as the product of the averages of the deviation of data points from the mean of corresponding data set.

- Consequently, the covariance matrix with respect to noise $\eta(n)$ is defined in the matrix form (Jansen 2001):

$$Q = E\left[(\eta - E\eta)(\eta - E\eta)^T\right] = E\left[\eta\eta^T\right] \tag{3.13}$$

It can be found that on the diagonal of the matrix the variances are σ_i^2=$E[\eta_i^2]$. If this matrix is diagonal, i.e., $E[\eta_i\eta_j]$=0 for $i\neq j$, the noise is called white or uncorrelated. If all data points come from the same probability density (the most general and important one is Gaussian probability density), the points are identically distributed, which implies that, σ_i^2=σ^2, $\forall i$=1,...N. Moreover, if Gaussian noise variables are uncorrelated, they are also independent. Therefore the classical assumption in noise modeling is that the noise is IID.

The motivations by utilizing wavelet transform to de-noise result from the following reasons:

- Wavelet transform can de-correlate a signal with multi-level structures. Small wavelet coefficients are governed by noise, while coefficients with large absolute amplitude values contain more signal information. Hence replacing the smallest, noisy coefficients by zero and a reverse wavelet transform may lead to a reconstruction while retaining the

essential signal characteristics with less noise. Moreover, in (Jansen 2001), the following three observations and assumptions on wavelet transform were stated:

(1) The decorrelating property of a wavelet transform creates a sparse signal: most uncorrelated coefficients are zero or close to zero.

(2) Noise is spread out equally over all coefficients, i.e., it is uniformly distributed.

(3) The noise level is not too high, so that the signal can be discriminated from the signal wavelet coefficients.

- Another important characteristic of wavelet transform is that the wavelet decomposition is not only a sparse representation but also a multi-scale data representation of original signal. This feature provides a solution in terms of scale-dependent thresholds to correlated noise and is more adaptive to signals that contain different characteristics at different scales. Indeed, scale can be seen as the approximate inverse of frequency. It is useful to look across the scales and handle one scale while taking into account the information at adjacent resolutions. In addition, if the noise is correlated instead of white and the standard deviation of noise is dependent on the resolution level, obviously one threshold cannot remove noise decently.

3.2.2 Procedure of wavelet-based de-noising algorithm

The proposed de-noising method in this study consists of the following three steps.

Step 1: Signal decomposition

Choose a wavelet function and decompose the underlying signal x up to scale level J. As mentioned earlier, D_4 is chosen as the wavelet function. As to decomposition levels, theoretically in discrete wavelet transform, the sequence length determines the number of scale levels. In this study, data sequence length of $N=512=2^9$ indicates that there should be nine wavelet levels. However, the major concern of this study is to identify the chatter behavior within certain frequency band, or on certain scale level. It is not imperative to segregate the chatter fault in terms of frequency from other coexistent faults because its unique existence is assumed. Furthermore, due to the limited computing power (Intel Pentium III-based computer) for on-line implementation and the intention of keeping a certain length of data points at each scale, the number of decomposition levels is set to 4. Wavelet coefficients including approximation and detail ones at each level are obtained consequently.

Step 2: Thresholding detail coefficients

Apply the following customized thresholding function with a level-dependent universal threshold rule to the detail coefficients at each scale level.

Conventionally, hard-thresholding function and soft-thresholding function are the most commonly used ones. The hard-thresholding function picks all wavelet coefficients that are greater than the given threshold and sets the others to zeros as defined in (Debnath 2003), here w denotes noisy wavelet coefficients:

$$w_{thr\_h} = \begin{cases} w & |w| \geq thr \\ 0 & otherwise \end{cases} \tag{3.14}$$

The soft-thresholding function has a somewhat different rule from the hard-thresholding function. It shrinks the wavelet coefficients by the given threshold towards zero (Debnath 2003):

$$w_{thr\_s} = \begin{cases} w - thr & w \geq thr \\ 0 & |w| < thr \\ w + thr & w \leq -thr \end{cases} \tag{3.15}$$

In this study, instead of using either of two thresholding functions above, a customized threhsolding function is proposed and defined as follows:

$$w_{thr\_c} = \begin{cases} w - \mathrm{sgn}(w)(1-\zeta) \times thr & ,|w| \geq thr \\ 0 & ,|w| < thr \end{cases} \tag{3.16}$$

This thresholding function can be viewed as the linear combination of hard-thresholding function and soft-thresholding function in the form of $\zeta \cdot w_{thr\_h} + (1-\zeta) \cdot w_{thr\_s}$. Depending on different values of parameter ζ, the above customized function can adapt to either soft-thresholding or hard-thresholding function. Figure 3.6 depicts the customized thresholding function for various ζ's from 0 to 1 with thr=1.

On the other hand, the threshold value appearing in the above customized thresholding function is chosen according to the universal threshold rule (Donoho 1995):

$$thr = \sigma\sqrt{2\log(N)} \tag{3.17}$$

where N is the length of signal, σ is standard deviation of noise at each scale. σ can be estimated on each scale separately by the median of absolute deviation (MAD) value as proposed in (Vidakovic and Guy 2000):

$$\hat{\sigma} = \frac{\text{MAD}}{0.6745} \tag{3.18}$$

In (Jansen 2001) the author remarked that as its name reflects this universal threshold is "valid" for all signals with length N, provided that these signals are "sufficiently" smooth.

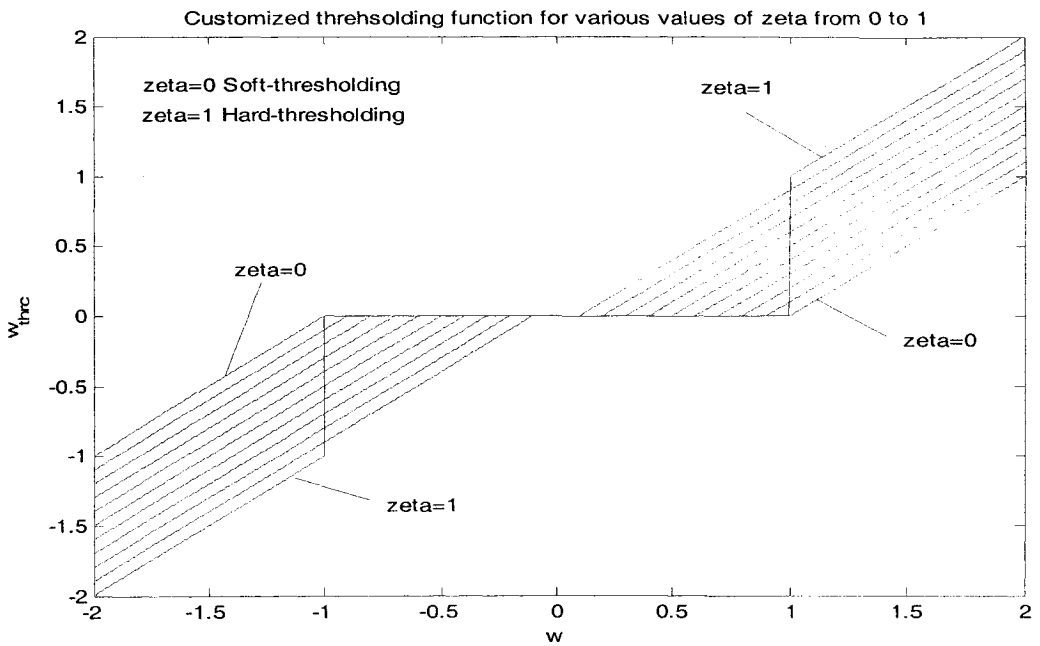


Figure 3.6 Plot of customized threhsolding function for various ζ's from 0 to 1

Next, to determine the most appropriate value of parameter ζ, the generalized cross validation (GCV) function is examined. Theoretically, the minimized mean square error (MSE) function should be used as a criterion to evaluate the best approximation result of noise-free data which corresponds to $f(n)$ in equation (3.11) by removing the noise $\eta(n)$. The MSE in terms of threshold value is defined in wavelet domain as follows (Jansen 2001):

$$MSE(thr) = \frac{1}{N}\|\eta_{thr}\|^2 \tag{3.19}$$

where $\eta_{thr}=w_{thr}-w_0$, w_0 is noise-free wavelet coefficients and w_{thr} is the wavelet coefficients after thresholding. However, it is impossible to know the uncorrupted data $f(n)$ beforehand and hence the MSE function can never be computed exactly. Therefore this MSE has to be estimated. As proposed in (Jansen 2001) GCV is such an estimator of MSE whose definition is given as follows:

$$MSE \approx GCV(thr) = \frac{\frac{1}{N} \cdot \|w - w_{thr}\|^2}{\left(\frac{N_0(thr)}{N}\right)^2} \tag{3.20}$$

where w is the noisy wavelet coefficients, w_{thr} denotes wavelet coefficients after thresholding and $N_0(thr)$ stands for the number of noisy wavelet coefficients below the threshold.

Therefore, instead of using MSE, the GCV function is investigated in this study because it is easy to compute in the wavelet domain and is an asymptotically optimal estimator of MSE function. For each data input cycle, the following procedure will lead to determination of the most suitable value for parameter ζ which minimizes the GCV function.

(1) Initialize ζ=0 since it is always within the range 0 to 1. The initial threshold value is calculated from the universal threshold rule.

(2) Increase ζ by a constant step until it finally reaches 1. For each increment, ζ is a fixed value and treated as a part of new threshold, which means new threshold is set to (1-ζ)×thr (thr is denoted as the original universal threshold). Next, by substituting the new threshold into equation (3.16), w_{thr} and $N_0(thr)$ can be obtained and consequently GCV is calculated based on equation (3.20).

(3) The most appropriate value of ζ minimizing the GCV function is recorded and used for noise reduction.

Step 3: Signal reconstruction

The inverse wavelet transform is performed by using the resultant approximation coefficients and the de-noised detail coefficients from 1 to J levels. Then the approximated original signal is reconstructed and its frequency domain plot is carried out by fast Fourier transform (FFT). Chatter frequency and other typical cutting frequencies can be recognized from the plot.

3.3 Feature extraction and chatter detection index formulation

After capturing the machining process signals, the most important step is to extract feature to assess the health conditions of machine and process. It can usually be fulfilled by conducting signal processing techniques. The extracted features from measured signals should potentially show the highest correlation to the states of monitored phenomenon since

it is critical for decision making and control action at the next phase. In this study, the discrete wavelet transform and proposed noise reduction algorithm have been used to accomplish the preliminary part of this step. The remaining of this section will investigate the formulation of chatter index in detail.

3.3.1 Feature selection and chatter index formulating approach

For each cycle of signal input, discrete wavelet decomposition along with the noise reduction procedure is implemented. As a result, the certain number of sets of clean wavelet coefficients is obtained. The magnitude of wavelet coefficients within each frequency band reflects the amplitude of vibration signal over a time interval. Also as stated in the beginning of this chapter, wavelet transform modulus maxima carry the vital information concerning the characteristics of signals. Maximal magnitude components, i.e., modulus maxima, will be present at the time when maximum changes in the signal have occurred. Considering when chatter tends to occur, most energy of the signal is concentrated on certain frequency band. As a result, the energy accumulated in that band accounts for large portion of total energy and is confined to more local maximal wavelet coefficients than ever, while other coefficients are either reduced or diminished to zeros. Note that wavelet transform does conserve the total energy which a signal holds (Walker 1999). Therefore it is meaningful to investigate wavelet transform modulus maxima and consider it as a key feature to indicate the machining conditions.

For practical applications, the chatter detection index should be varying between a range such as 0 and 1 to avoid the difficulty in determining thresholds. In addition, the derived index should be sensitive to chatters but insensitive to other disturbances.

In this study, the methodology of statistical distribution analysis of wavelet transform modulus maxima is used to analyze random vibration signals and to form chatter detection indicator. This is due to the following considerations. The measured signals are non-deterministic and unpredictable. Their randomness is best characterized with statistical terms. Besides, statistical analysis can produce a non-dimensional index between 0 and 1 in terms of the cumulative distribution function. Moreover, as mentioned earlier in Section 3.3.1, wavelet transform modulus maxima have the ability to detect the transient period and onset of chatters and are employed as the primary feature to formulate the chatter index. The details concerning the proposed methodology are given in Sub-section 3.3.3.

3.3.2 Random process and statistics review pertinent to chatter detection

Before proceeding to the formal subject in next sub-section, a brief review of random process is first provided. Suppose that we take a set of waveforms corresponding to the air temperature in different cities around the world. For each city there is corresponding waveform that is a function of time. The set of all possible waveforms is called an ensemble of time functions or, equivalently, a random process. The waveform for the temperature in any particular city is a single realization or a sample function of the random process. Ensemble averages are measured across the ensemble instead of being measured along a single sample function at different time moments. It is denoted as $E[x]$, which stands for the ensemble averaged value of the quantity x in square bracket. A random process is considered as an ensemble of sample functions. In addition, the random process is said to be stationary if the probability distribution obtained for the ensemble does not depend on time t. A stationary process is called an ergodic process if the average with a sample function is same as the ensemble averages. For the convenience of following study, some important concepts and definitions in random process are introduced as below:

- ***Autocorrelation function***

The autocorrelation function of the random process $X(t)$ is defined as (Williams 2003):

$$R_X(\tau) = E\left[X(t)X(t+\tau)\right] \tag{3.21}$$

where τ is the time difference. For the stationary random process, $R_X(-\tau)=R_X(\tau)$ means an autocorrelation function is an even function of τ (Williams 2003).

- ***Spectral density***

The spectral density, denoted as $S_X(\omega)$, for the random process $X(t)$ is defined as (Newland 1984):

$$S_X(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} R_X(\tau)e^{-i\omega\tau}d\tau \tag{3.22}$$

and autocorrelation function in terms of power spectral density becomes (Newland 1984):

$$R_X(\tau) = \int_{-\infty}^{\infty} S_X(\omega)e^{i\omega\tau}d\omega \tag{3.23}$$

The following is a brief review of statistics to help understand our detection approach. Several basic distributions and their definitions are presented. A few concepts with either one or two random variables are also studied. The review begins with distinguishing the

definitions between a random process and a random variable. A random process is a function of an independent variable, time, and it is treated as a random variable when the variable of time is neglected.

- ***Gaussian distribution***

The probability density function of Gaussian random variable X is given as (Hines *et al* 2003):

$$p_X(x) = \frac{1}{\sqrt{2\pi}\sigma_X} e^{-(x-\mu_X)^2/2\sigma_X^2} \quad -\infty < x < \infty \tag{3.24}$$

where μ_X is mean and σ_X^2 is variance. Gaussian distribution is also known as normal distribution written as $X \sim N(\mu_X, \sigma_X^2)$.

Hines *et al* (2003) also defined the probability density function of Gaussian random variable Z with mean 0 and variance 1 as follows, which is called standard Gaussian distribution and denoted as $Z \sim N(0, 1)$:

$$p_Z(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2} \quad -\infty < z < \infty \tag{3.25}$$

From equations (3.24) and (3.25), it can be seen that by a simple transformation of variables, $z = (x-\mu_X)/\sigma_X$, any non-standard Gaussian distribution of a random variable can be normalized to $N \sim (0, 1)$, i.e., by shifting the data with respect to μ_X and rescaling the data with respect to σ_X.

Gaussian single random variable is readily extended to bi-variate random variables. If X_1 and X_2 are two random variables, the bivariate (joint) probability density function of (X_1, X_2) is defined as (Hines *et al* 2003):

$$p_{X_1,X_2}(x_1,x_2) = \frac{1}{2\pi\sigma_{X_1}\sigma_{X_2}\sqrt{\left(1-\rho_{X_1X_2}^2\right)}} e^{-\frac{1}{2\left(1-\rho_{X_1X_2}^2\right)}\left\{\frac{(x_1-m_{X_1})^2}{\sigma_{X_1}^2}+\frac{(x_2-m_{X_2})^2}{\sigma_{X_2}^2}-\frac{2\rho_{X_1X_2}(x_1-m_{X_1})(x_2-m_{X_2})}{\sigma_{X_1}\sigma_{X_2}}\right\}} \tag{3.26}$$

where m_{X1} and m_{X2} are the means, σ_{X1} and σ_{X2} are the variances, and ρ_{X1X2} is the covariance coefficient.

The covariance coefficient ρ_{X1X2} of bivariate random variables X_1 and X_2 is given by Williams (2003) as:

$$\rho_{X_1X_2} = \frac{E\left[\left(X_1 - \mu_{X_1}\right)\left(X_2 - \mu_{X_2}\right)\right]}{\sigma_{X_1}\sigma_{X_2}} \tag{3.27}$$

One of the most important theorems in probability and statistics is the central limit theorem. It states that "if a random variable Y is the sum of n independent random variables that satisfy certain general conditions, then for sufficiently large n, Y is approximately normally distributed" (Hines *et al* 2003). A special case arises when each of the components has the same distribution, and the following statement is true: if a random variable X is Gaussian so is its derivative X', since a derivative can be expressed as the limiting case of the difference $\{X(t+\Delta t\text{-}X(t)\}/\Delta t$ between two random variables $X(t+\Delta t)$ and $X(t)$ (Newland 1984).

Another important property that will be used in the following section is that the Gaussian random process $X(t)$ and its derivative $X'(t)$ are uncorrelated and so the covariance coefficient $\rho_{XX'}$ is always zero. The proof is as follows.

Equation (3.21) can be written as (Newland 1984):

$$R_X(\tau) = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N} X_i(t)\,X_i(t+\tau) \tag{3.28}$$

Differentiating $X_i(t)X_i(t+\tau)$ with respect to τ yields (Newland 1984)

$$\frac{d}{d\tau}\left\{X_i(t)\,X_i(t+\tau)\right\} = X_i(t)\frac{d}{d\tau}X_i(t+\tau)$$

$$= X_i(t)\frac{d}{d(t+\tau)}X_i(t+\tau)\cdot\frac{d(t+\tau)}{d\tau}$$

$$= X_i(t)\,X_i'(t+\tau) \tag{3.29}$$

This leads to:

$$\frac{d}{d\tau}\left(R_X(\tau)\right) = E\left[X(t)\,X'(t+\tau)\right] \tag{3.30}$$

If random processes $X(t)$ and $X'(t)$ have zero means, from equation (3.27) we have

$$\rho_{XX'} = \frac{E[XX']}{\sigma_X\sigma_{X'}} \tag{3.31}$$

In fact, the Gaussian random variables can always be shifted to have zero means. However, to calculate $\rho_{XX'}$, $E[XX']$ must be known. From equation (3.30), it can be seen that $E[XX']$ corresponds to (Newland 1984)

$$E[XX'] = \frac{d}{d\tau}\left(R_X(\tau)\right)_{\tau=0} \tag{3.32}$$

As defined in equation (3.23), $R_X(\tau)$ is expressed as the Fourier integral of the corresponding spectral density, therefore by considering equations (3.23)) and (3.32) $E[XX']$ can be obtained as (Newland 1984):

$$E[XX'] = i\int_{-\infty}^{\infty} \omega S_X(\omega) d\omega \tag{3.33}$$

It can be derived that $S_X(\omega)$ is a real even function of frequency ω as follows.

(1) $e^{-i\omega\tau} = \cos\omega\tau - i\sin\omega\tau$

(2) Then $S_X(\omega)$ can be written as:

$$S_X(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} R_X(\tau)\cos(\omega\tau)d\tau + \frac{i}{2\pi}\int_{-\infty}^{\infty} R_X(\tau)\sin(\omega\tau)d\tau \tag{3.34}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} R_X(\tau)\cos(\omega\tau)d\tau + i\left(\frac{1}{2\pi}\int_{-\infty}^{0} R_X(\tau)\sin(\omega\tau)d\tau + \frac{1}{2\pi}\int_{0}^{\infty} R_X(\tau)\sin(\omega\tau)d\tau\right)$$

Let $\tau = -t$ with respect only to the term $\frac{1}{2\pi}\int_{-\infty}^{0} R_X(\tau)\ \sin(\omega\tau)d\tau$, the above equation turns to

$$S_X(\omega) =$$

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} R_X(\tau)\cos(\omega\tau)d\tau + i\left(-\frac{1}{2\pi}\int_{-\infty}^{0} R_X(-t)\sin(-\omega t)d(t) + \frac{1}{2\pi}\int_{0}^{\infty} R_X(\tau)\sin(\omega\tau)d\tau\right)$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} R_X(\tau)\cos(\omega\tau)d\tau + i\left(\frac{1}{2\pi}\int_{0}^{\infty} R_X(-t)\sin(-\omega t)d(t) + \frac{1}{2\pi}\int_{0}^{\infty} R_X(\tau)\sin(\omega\tau)d\tau\right)$$

Since $R_X(\tau)$ is an even function and $\sin\omega\tau$ is an odd function, then the above equation can be written as

$$S_X(\omega) =$$

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} R_X(\tau)\cos(\omega\tau)d\tau + i\left(-\frac{1}{2\pi}\int_{0}^{\infty} R_X(t)\sin(\omega t)d(t) + \frac{1}{2\pi}\int_{0}^{\infty} R_X(\tau)\sin(\omega\tau)d\tau\right)$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} R_X(\tau)\ \cos(\omega\tau)d\tau + i\times 0 \tag{3.35}$$

Again since $R_X(\tau)$ is an even function and so is $\cos\omega\tau$, $S_X(\omega)$ is a real even function too.

Therefore the integrand $\omega S_X(\omega)$ in (3.33) is a real odd function of ω and the integral must be zero when ω ranging from $-\infty$ to ∞ and $E[XX']$ is obtained as:

$$E[XX'] = 0 \tag{3.36}$$

This result subsequently leads to $\rho_{XX'}=0$ in equation (3.31), i.e., $X(t)$ and $X'(t)$ are uncorrelated.

Equations (3.22) and (3.23) represent the autocorrelation function and spectral density with respect to Fourier transform, respectively and the above proof is based on spectral density analysis as well. Similar proof of the uncorrelation can be done in terms of wavelet transform as follows.

Wavelet spectrum is defined as (Li and Oh 2002):

$$I(u) := I(u,s) = E\left\{\left|W_u(s)\right|^2\right\} = u\int_{-\infty}^{\infty}\left|\Psi(u\omega)\right|^2 S_X(\omega)\,d\omega \tag{3.37}$$

where u, s are the scale and translation parameters of wavelet transform, $\Psi(\omega)$ is the Fourier transform of $\psi(t)$ which is $\Psi(\omega)=\int\psi(t)e^{-i\omega t}dt$. As mentioned in (Li and Oh 2002), $I(u, s)$ is independent of s for stationary process in many situations and therefore s is omitted in equation (3.37).

We follow the same proof procedure as the above (note that this time X denotes the random variable in the wavelet domain) and know that equation (3.33) can be derived from equations (3.32) and (3.23) as follows:

$$E[XX'] = \frac{d}{d\tau}R_X(\tau)\Big|_{\tau=0} = \frac{d}{d\tau}\int_{-\infty}^{\infty}S_X(\omega)e^{i\omega\tau}d\omega\Big|_{\tau=0} = i\int_{-\infty}^{\infty}\omega S_X(\omega)d\omega \tag{3.38}$$

The relations among $I(u)$, $S_X(\omega)$ and $R_X(\tau)$ can be expressed as $I(u)\leftrightarrow R_X(\tau)\leftrightarrow S_X(\omega)$ (Li and Oh 2002), where $\leftrightarrow$ means one-to-one mapping. Therefore the wavelet spectrum $I(u)$ in equation (3.37) is a real even function since for each $S_X(\omega)$ in equation (3.38), there is only one corresponding $I(u)$ and due to the real even property of $S_X(\omega)$, $I(u)$ is also a real even function. The same conclusion can be drawn that $E[XX']$ is equal to zero and so is $\rho_{XX'}=0$.

Lastly, for Gaussian random variables, there is a remarkable property: if the bivariate Gaussian random variables are uncorrelated, they are also independent (Williams 2003). Therefore the following equation holds by substituting $\rho_{XX'}=0$ into equation (3.26):

$$p_{XX'}(x,x') = \frac{1}{2\pi\sigma_x\sigma_{x'}}e^{-\frac{1}{2}\left\{\frac{x^2}{\sigma_x^2}+\frac{x'^2}{\sigma_{x'}^2}\right\}} = p_X(x)\,p_{X'}(x') \tag{3.39}$$

■ ***Rayleigh distribution***

The probability density function of Rayleigh distribution is expressed as (Williams 2003):

$$p_R(x)=\begin{cases}\dfrac{x}{\sigma_X{}^2}e^{\left(-x^2/2\sigma_X{}^2\right)} & x\geq 0\\ 0 & x<0\end{cases} \tag{3.40}$$

The cumulative distribution function of Rayleigh distribution is (Williams 2003):

$$P_R(x)=\begin{cases}1-e^{\left(-x^2/2\sigma_X{}^2\right)} & x\geq 0\\ 0 & x<0\end{cases} \tag{3.41}$$

where $\sigma_X{}^2$ is the variance for both functions.

■ ***Weibull distribution***

The probability density function of Weibull distribution is given by Hines *et al* (2003) as follows:

$$p_W(x)=\begin{cases}\dfrac{\gamma}{\alpha}\left(\dfrac{x-\mu}{\alpha}\right)^{\gamma-1}e^{-\left(\frac{x-\mu}{\alpha}\right)^{\gamma}} & x\geq\mu\\ 0 & otherwise\end{cases} \tag{3.42}$$

The cumulative distribution function of Weibull random variable is (Hines *et al* 2003):

$$P_W(x)=\begin{cases}1-e^{-\left(\frac{x-\mu}{\alpha}\right)^{\gamma}} & x\geq\mu\\ 0 & otherwise\end{cases} \tag{3.43}$$

where $\mu\in(-\infty,\infty)$ is the location parameter, $\alpha > 0$ is the scale parameter and $\gamma > 0$ is the shape parameter. By appropriate selection of these parameters, the Weibull density function can closely approximate many phenomena.

3.3.3 Statistical distribution analysis of wavelet transform modulus maxima

The probability distribution of wavelet coefficients has been approximated using a generalized Gaussian distribution (Mallat 1989), Gaussian (Basseville *et al* 1992, Luettgen *et al* 1993, and Browne and Cutmore 2002) or Gaussian processes with zero mean (Mihcak *et al* 1999 and Gupta *et al* 2004). In particular, as studied in (Choi and Shin 2003) wavelet coefficients were modeled as mutually independent Gaussian zero-mean random variables. Consequently a chatter detection index was obtained based on a wavelet-based maximum

likelihood (ML) estimation algorithm. It should be pointed out that the Gaussian processes with non-zero mean can be easily converted into zero-mean Gaussian processes as indicated in equations (3.24) and (3.25). Therefore according to the above studies, in this research, the wavelet coefficients are also modeled as random variables following Gaussian distribution. $w_{thr}(t)$ is used to denote the Gaussian random process, random variable w_{thr} represents the wavelet coefficients obtained after thresholding and a_p denotes the wavelet transform modulus maxima with positive magnitude. In the next two sub-sections, we will first demonstrate the wavelet transform modulus maxima with positive amplitude, i.e., the positive peak values of wavelet coefficients that exceed level a follow Rayleigh distribution. Then by considering more general distributions in real situation, Rayleigh distribution is extended to Weibull distribution.

3.3.3.1 Rayleigh distribution of wavelet transform modulus maxima

To show that the positive peak values of the wavelet coefficients follow Rayleigh distribution, the crossing analysis is to be studied (note that equations (3.44) to (3.56) are adopted from (Newland 1984)). It begins with counting the number of "cycles" of wavelet coefficients $w_{thr}(t)$ that have amplitudes greater than level $w_{thr}=a$ during the time period T as shown in Figure 3.7. From the figure, it can be observed that there are totally six crossings with positive slope which exceed the level $w_{thr}=a$ in time T. Equivalently, this means that there exist six peaks during the time period T. Now assume that Figure 3.7 is one sample function of an ensemble of functions which constitute the stationary random process $w_{thr}(t)$. Let $n_a^+(T)$ denote the number of positive slope crossings of $w_{thr}=a$ in period T and $N_a^+(T)$ be the mean value of all the samples, then

$$N_a^+(T) = E\left[n_a^+(T)\right] \tag{3.44}$$

Since the process is assumed stationary, the same result can be obtained if a second interval of duration T is taken immediately after the first. Considering these two intervals together, the following relation holds:

$$N_a^+(2T) = 2N_a^+(T) \tag{3.45}$$

Therefore it can be concluded that for a stationary process the average number of crossings is proportional to the time interval T.

$$N_a^+(T) = v_a^+ T \tag{3.46}$$

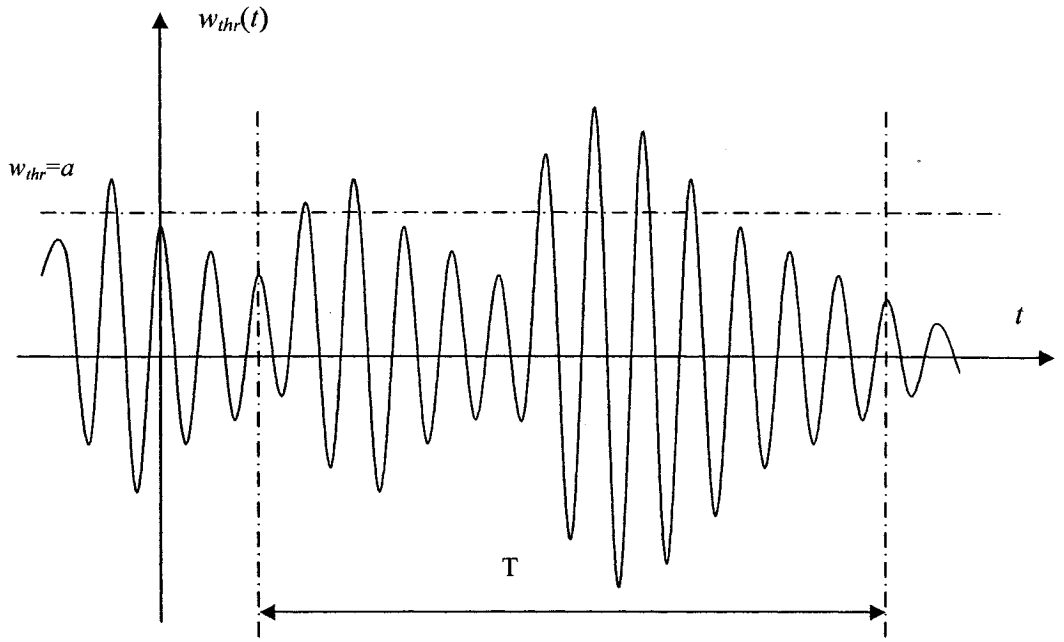


Figure 3.7 Sample of a stationary random process of wavelet coefficients

where v_a^+ is the average frequency of crossings the level $w_{thr}=a$ with positive slope. This frequency v_a^+ can be derived from the probability distribution of w_{thr} as presented in (Newland 1984) and the result is in terms of the joint probability density function $p(w_{thr}, w'_{thr})$ of random variables w_{thr} and its derivative w'_{thr}:

$$v_a^+ = \int_0^\infty p(a, w'_{thr}) w'_{thr} dw'_{thr} \tag{3.47}$$

Equation (3.47) is a general form which can be applied to any probability distribution of $w_{thr}(t)$. As indicated earlier, the bivariate random variable (w_{thr}, w'_{thr}) approximately follows joint Gaussian distribution, thus from equations (3.39) and (3.47) we can obtain:

$$v_a^+ = \frac{1}{2\pi\sigma_{w_{thr}}\sigma_{w'_{thr}}} e^{-a^2/2\sigma_{w_{thr}}^2} \int_0^\infty e^{-w'^2_{thr}/2\sigma_{w'_{thr}}^2} w'_{thr} dw'_{thr} \tag{3.48}$$

where σ_{wthr}, $\sigma_{w'thr}$ are the standard deviations of random variables w_{thr} and w'_{thr}, respectively.

Because

$$\int_0^\infty \left[e^{-\dot{w}_{thr}^2/2\sigma_{w_{thr}}^2} \right] \dot{w}_{thr} d\dot{w}_{thr} = \frac{1}{2\ 1/2\sigma_{\dot{w}_{thr}}} \int_0^\infty e^{-\dot{w}_{thr}^2/2\sigma_{w_{thr}}^2} d\left(\frac{\dot{w}_{thr}^2}{2\sigma_{\dot{w}_{thr}}^2} \right)$$

$$= \sigma_{\dot{w}_{thr}}^2 \left[e^{-\dot{w}_{thr}^2/2\sigma_{\dot{w}_{thr}}^2} \right]_0^\infty = \sigma_{\dot{w}_{thr}}^2 \tag{3.49}$$

equation (3.48) becomes

$$v_a^+ = \frac{\sigma_{w'_{thr}}}{2\pi\sigma_{w_{thr}}} e^{-a^2/2\sigma_{w_{thr}}^2} \tag{3.50}$$

A special case occurs if the level a is equal to zero. It gives a statistical average frequency for crossing level w_{thr}=0 (zero-crossing) and is denoted as v_0^+:

$$v_0^+ = \frac{\sigma_{w'_{thr}}}{2\pi\sigma_{w_{thr}}} \tag{3.51}$$

Having attained the frequency of crossings of $w_{thr}=a$, the probability distribution of positive peak wavelet coefficients can be determined by extending the above calculation. Here positive peak wavelet coefficients are considered equivalent to wavelet transform modulus maxima with positive magnitude a_p. Let $p(a_p)da_p$ be the probability of a peak with positive magnitude which is randomly chosen and lies in the range a to $a+da_p$, as shown in Figure 3.8. The probability that any positive peak is greater than a is therefore,

$$\text{Prob}(a_p \geq a) = \int_a^\infty p(a_p) da_p \tag{3.52}$$

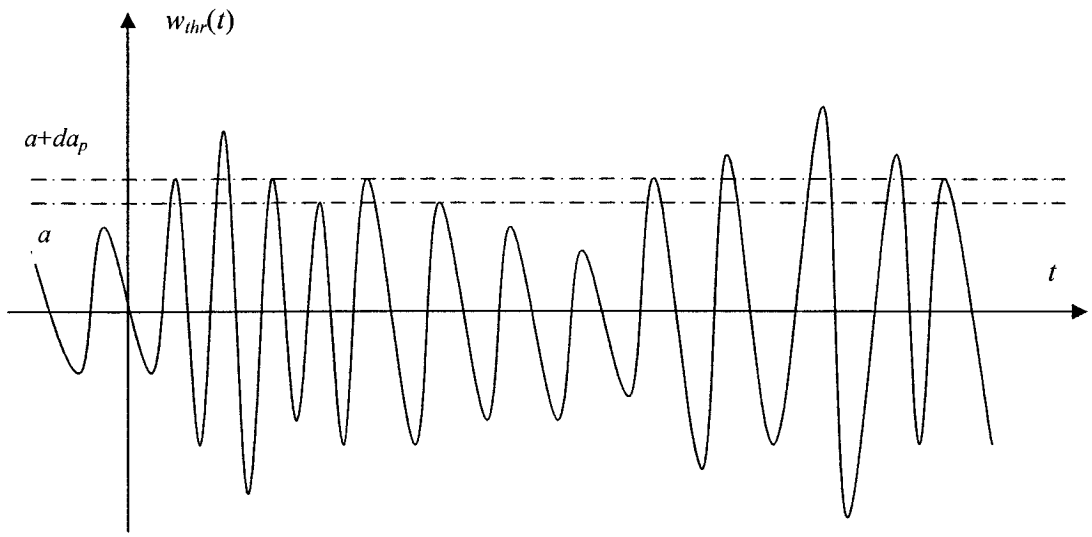


Figure 3.8 Peaks identification in the range $w_{thr}=a$ to $w_{thr}=a+da_p$

Now during time period T, there will be only an average of v_a^+T out of v_0^+T cycles (since one positive crossing of w_{thr}= 0 occurs for each full cycle of the process) which will have peak values exceeding the level a. Accordingly, the proportion of cycles whose peak value exceeds a is v_a^+/v_0^+ and this is equivalent to the probability that a positive peak value is above a. Hence the following equation is obtained:

$$\text{Prob}(a_p \geq a) = \int_a^{\infty} p(a_p) da_p = \frac{v_a^+}{v_0^+} \tag{3.53}$$

Differentiating equation (3.53) with regard to a_p gives

$$-p(a_p) = \frac{1}{v_0^+} \frac{d}{da_p}(v_a^+) \tag{3.54}$$

Equation (3-54) is a general result and applicable to any probability density distribution for the occurrence of positive peaks. However, if the random process is Gaussian, there is a simple and important result for $p(a_p)$. Substituting equations (3.50) with $a = a_p$ and (3.51) into equation (3.54) gives

$$-p(a_p) = \frac{d}{da_p}\left(e^{-a_p^2/2\sigma_{w_{thr}}^2}\right) = -\frac{a_p}{\sigma_{w_{thr}}^2} e^{-a_p^2/2\sigma_{w_{thr}}^2}$$

$$\Rightarrow p(a_p) = \frac{a_p}{\sigma_{w_{thr}}^2} e^{-a_p^2/2\sigma_{w_{thr}}^2} \quad 0 \leq a_p \leq \infty \tag{3.55}$$

Comparing with equation (3.40), equation (3.55) shows the probability density function of Rayleigh distribution and is depicted in Figure 3.9. The function $p(a_p)$ has its maximum value at σ_{wthr}, the standard deviation of the process $w_{thr}(t)$ as proved follows

$$\frac{dp(a_p)}{da_p} = \frac{1}{\sigma_{w_{thr}}^2} e^{-\frac{a_p^2}{2\sigma_{w_{thr}}^2}} + \frac{a_p}{\sigma_{w_{thr}}^2}\left(\frac{2a_p}{2\sigma_{w_{thr}}^2}\right) e^{-\frac{a_p^2}{2\sigma_{w_{thr}}^2}} = \frac{1}{\sigma_{w_{thr}}^2} e^{-\frac{a_p^2}{\sigma_{w_{thr}}^2}}\left(1 - \frac{a_p^2}{\sigma_{w_{thr}}^2}\right) = 0$$

This leads to $a_{p\max} = \sigma_{wthr}$.

It is clear that the majority of peaks have the magnitude around σ_{wthr}. The probability of finding very small or very large peaks is low. Since the cumulative distribution function of Rayleigh distribution is $P(a) = 1 - e^{-a^2/2\sigma_{w_{thr}}^2}$, therefore the probability of any positive peaks a_p exceeding a is written as:

$$\text{Prob}(a_p \geq a) = 1 - P(a) = e^{-a^2/2\sigma_{w_{thr}}^2} \tag{3.56}$$

So far the derivations and proofs demonstrated above are associated with those positive peak values of wavelet coefficients. For those peak values of wavelet coefficients with negative magnitudes, if we flip them 180 degree over the horizontal axis, i.e., taking their absolute values, and do the same crossing analysis as shown in Section 3.3.3.1, similar results can be obtained in terms of Rayleigh distribution since the initial wavelet coefficients

are shifted to have zero mean. And mathematically both positive and negative portions of peaks of wavelet coefficients should have same probability distribution. Therefore, only the wavelet transform modulus maxima with positive magnitudes, i.e., those peaks of wavelet coefficients with positive magnitudes, are considered and actually in use.

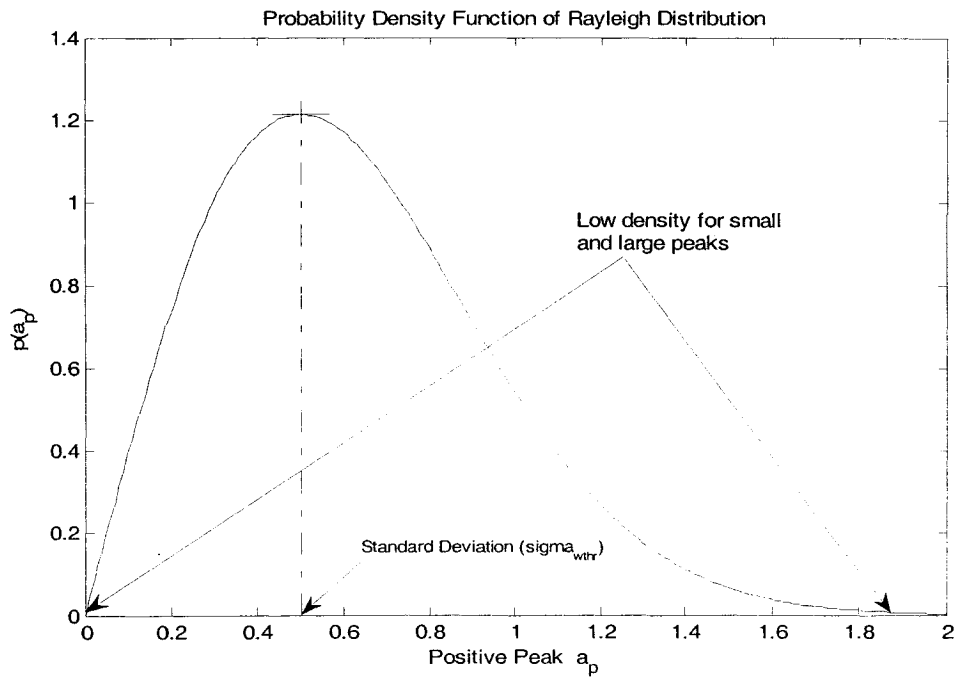


Figure 3.9 Rayleigh distribution of wavelet coefficients with positive peaks for a Gaussian process $w_{thr}(t)$

3.3.3.2 Weibull distribution of wavelet transform modulus maxima

At the end of the previous sub-section, it is concluded that the probability of positive peaks exceeding level a_p follows Rayleigh distribution. However in a number of applications, it has been found that Gaussian assumption may not be valid and the distribution of positive peaks may depart from a Rayleigh distribution (Newland 1984). Therefore to accommodate more general distributions, Weibull distribution is used for the broader applications (Note that Rayleigh distribution is a special case of Weibull distribution). The details of equations from (3.57) to (3.61) in the following are available in (Newland 1984).

Let a_0 be the mean peak height and from equation (3.56) we have

$$\text{Prob}\left(a_p \geq a_0\right) = e^{-a_0^2/2\sigma_{w_{thr}}^2} = \frac{1}{2} \tag{3.57}$$

From the above equation, a_0 can be expressed as:

$$a_0 = \sigma_{w_{thr}} \sqrt{2\ln 2} \text{ or } \sigma_{w_{thr}} = \frac{a_0}{\sqrt{2\ln 2}} \tag{3.58}$$

Substituting equation (3.58) into (3.56) gives

$$\text{Prob}\left(\frac{a_p}{a_0} \geq \frac{a}{a_0}\right) = 1 - P\left(\frac{a}{a_0}\right) = e^{-(\ln 2)(a/a_0)^2} \tag{3.59}$$

Replacing the exponent 2 in equation (3.59) by the general shape parameter γ gives us the following general probability distribution function (Figure 3.10 illustrates the plots corresponding to several shape parameters), or the statistical chatter index, *SI*,

$$SI = P\left(\frac{a}{a_p}\right) = 1 - e^{-(\ln 2)(a/a_0)^\gamma} \tag{3.60}$$

The above equation (3.60) gives a special form of cumulative distribution function of Weibull distribution. Actually it is a type of Weibull distribution with zero mean, scale parameter $\alpha = a_0/(\ln 2)^{1/\gamma}$ and integral interval of $a/a_0 \sim \infty$. The corresponding probability density function is obtained by differentiating (3.60) with respect to a/a_0 and Figure 3.1 illustrates the plots with respect to several different shape parameter values:

$$p\left(\frac{a}{a_0}\right) = \gamma(\ln 2)\left(\frac{a}{a_0}\right)^{\gamma-1} e^{-(\ln 2)(a/a_0)^\gamma} \tag{3.61}$$

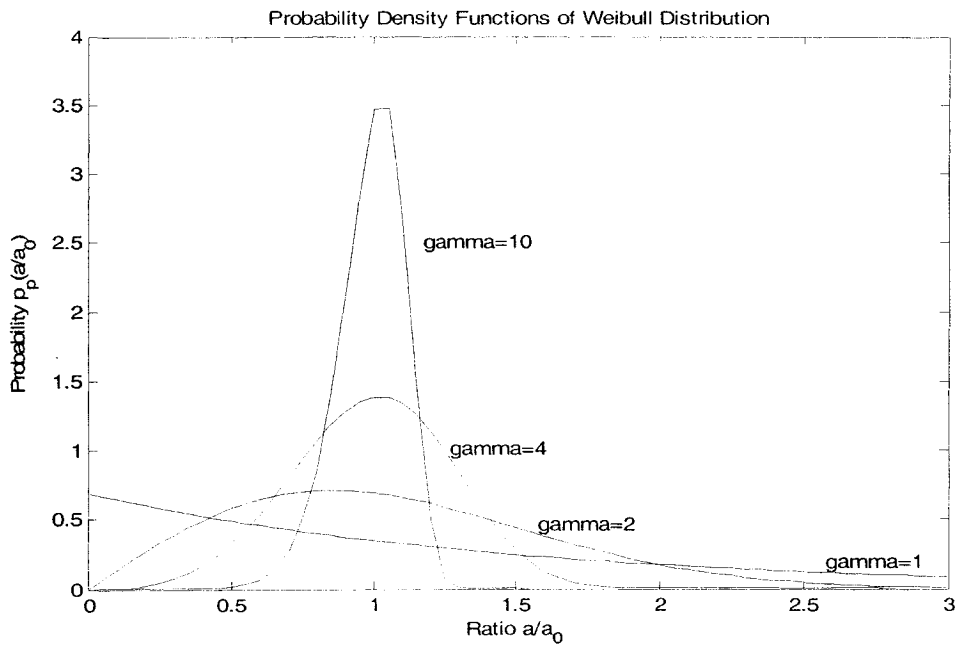


Figure 3.10 Plot of Weibull probability density function

Once the cumulative distribution function of Weibull distribution shown in equation (3.60) is finally acquired, our intention to develop a chatter index having the dimensionless property [0~1] can be realized since the value calculated by equation (3.60) always varies in that range. The concrete procedure to formulate the detection index is presented in the following sub-section.

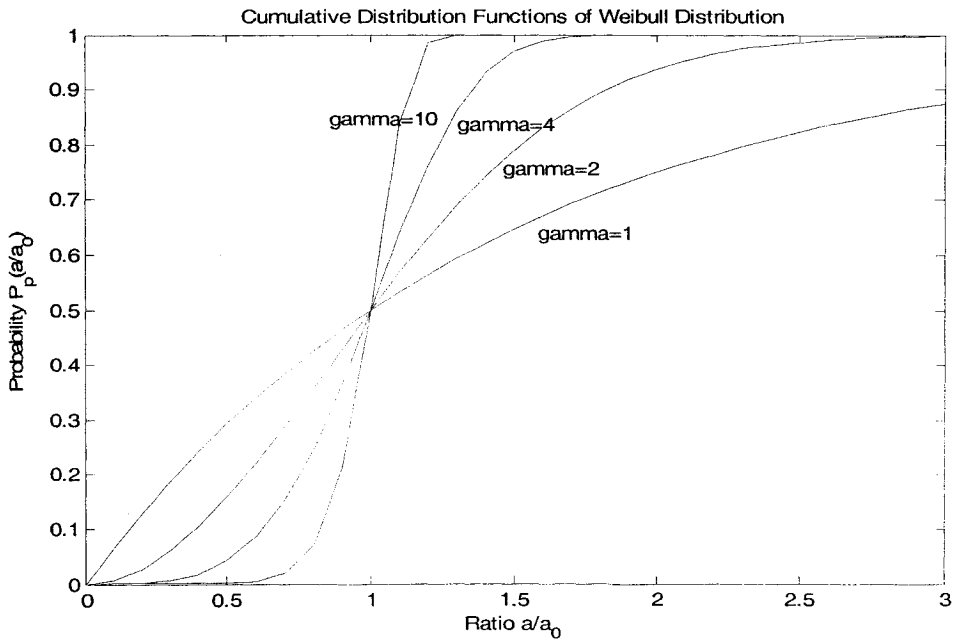


Figure 3.11 Plots of Weibull cumulative distribution function

3.3.3.3 Procedure of statistical chatter detection index formulation

The sub-bands in which chatter could potentially occur must be monitored during the machining period. However, in this study, to identify chatters more efficiently and reduce the computation burden, chatter-prone fault is targeted at certain frequency band by using a percentage-power discriminating criterion. It is calculated according to the normalized wavelet power. After the calculation of ratios of power in each band to total power of all bands, the sub-band with maximal percentage is considered as the suspicious one, which means chatter could occur within the band. However, the occurrence of chatter has to be confirmed by whether or not the chatter index value in the targeted band exceeding the reference threshold.

The normalized wavelet power (NWP) in band j is calculated as follows (Torrence and Compo 1998):

$$NWP(j) = \frac{1}{M}\sum_{k=0}^{M-1}\left(w_{thr}{}^{j,k}\right)^2 \tag{3.62}$$

where w_{thr} stands for the wavelet coefficients after thresholding in terms of scale j and location k, M is the number of wavelet coefficients in each frequency band, which is calculated by $M = N/2^j$, N is the total number of data points in each data acquisition cycle.

The band with largest relative wavelet power (RP) is identified as the most chatter-prone frequency range:

$$RP(j) = NWP(j) \Big/ \sum_{j=1}^{J} NWP(j) \tag{3.63}$$

where J is the total number of scale levels.

After identifying the suspicious band, special attention is then paid to it. The following procedure consisting of four steps is executed to identify chatters. Figure 3.2 shows the flow chart of this procedure.

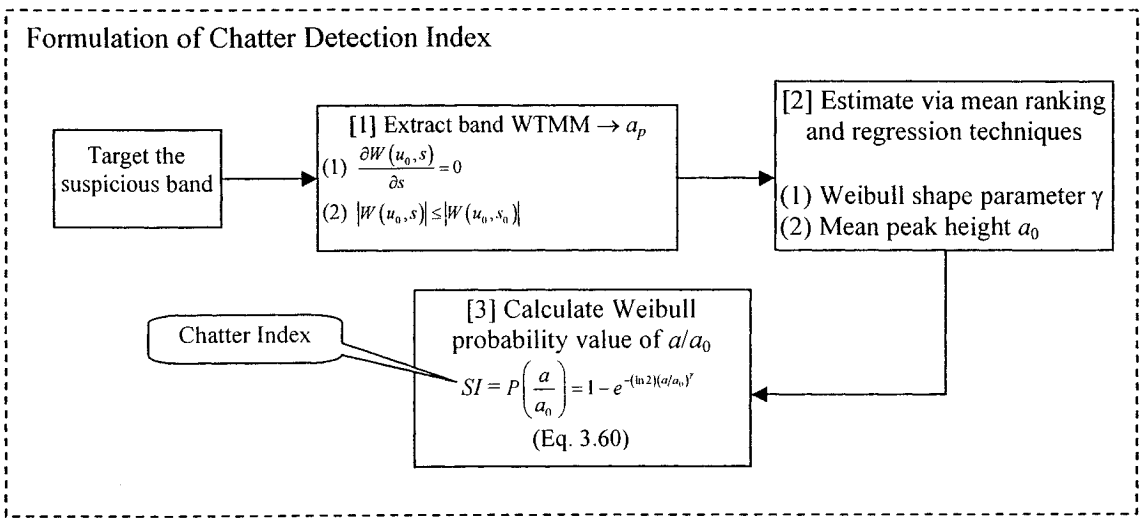


Figure 3.12 Flow chart of chatter detection index formulation

Step 1: Extract wavelet transform modulus maxima from clean wavelet coefficients

The extraction of wavelet transform modulus maxima with positive magnitude from the targeted band is performed according to its definition as shown in Section 3.1.2.

Step 2: Estimate Weibull shape parameter γ and median peak height a_0

So far it can be seen that to calculate statistical index for chatter detection as specified in equation (3.60), the shape parameter γ and median peak height a_0 must be determined. In

this step, according to (Williams 2003) γ and a_0 can be estimated by using regression technique as shown below. Starting with equation (3.60) with only consideration of a (from now on a is considered as wavelet transform modulus maxima), one obtains:

$$P\left(\frac{a}{a_0}\right) = 1 - e^{-(\ln 2)(a/a_0)^{\gamma}} \tag{3.64}$$

Applying the natural logarithm to the above equation, we have

$$-\ln\left[1 - P\left(\frac{a}{a_0}\right)\right] = (\ln 2)\left(\frac{a}{a_0}\right)^{\gamma} \tag{3.65}$$

Using the natural logarithm again

$$\ln\left(-\ln\left[1 - P\left(\frac{a}{a_0}\right)\right]\right) = \ln(\ln 2) + \gamma \ln a - \gamma \ln a_0 \tag{3.66}$$

Equation (3.66) can be identified as an equation of a straight line:

$$y = mx + b \tag{3.67}$$

where

$$y = \ln\left(-\ln\left[1 - P\left(\frac{a}{a_0}\right)\right]\right) \tag{3.68}$$

$$x = \ln a \tag{3.69}$$

$$b = \ln(\ln 2) - \gamma \ln a_0 \tag{3.70}$$

To calculate y in (3.68), we need an estimate of the cumulative distribution function of $P(a/a_0)$. It can be done with the use of mean ranking technique (Williams 2003):

$$P\left(\frac{a_k}{a_0}\right) \approx \frac{k}{n+1} \tag{3.71}$$

where n is the total number of positive WTMM in the chatter-prone band, k=1,...n, a_k is the sorted WTMM value in ascending order. Equation (3.71) implies that the larger the k value, the larger the probability value of $P(a_k/a_0)$ as illustrated in Figure 3.1.

With x and y data available, the regression technique will give the shape parameter γ in terms of m directly and mean peak height a_0 in terms of b as follows (Williams 2003):

$$m = \frac{c_{xy}}{\sigma_x^2} \tag{3.72}$$

$$b = \bar{y} - m\bar{x} = \bar{y} - \gamma\bar{x} \tag{3.73}$$

where c_{xy} is the sample covariance, σ_x^2 is the sample variance and $a_0 = e^{(\ln(\ln 2)-b)/\gamma}$.

Again according to (Williams 2003), c_{xy} and σ_x^2 can be individually estimated as:

$$c_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{3.74}$$

$$\sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{3.75}$$

where $\bar{x}$ and $\bar{y}$ are the sample mean values of data x and y.

Step 3: Compute Weibull probability of a/a_0 to obtain chatter detection index

Finally the band-based chatter detection index is formulated according to equation (3.60). As the number of samples generally has an influence on the accuracy of parameter estimation, a moving window with a specified size is proposed. Accordingly, with the increased sample size, the index shows clear trend and tends to be smoother.

3.3.3.4 Reference threshold setting and decision making

Detection of chatter fault is achieved by the implementation of a monitoring strategy, which is based on pre-determined reference threshold. In this study, the reference threshold value is chosen to be 0.65 based on a series of testing results. Tests are performed according to the conditions that will be illustrated in chapter 5. Upon the onset of chatter, a crossing of reference threshold will occur and the chatter detection index will be larger than 0.65 occasionally or continuously depending on the severity of chatter. As a result chatter is identified and an alarm is triggered.

3.4 Conclusions

In this chapter, a new methodology for detecting chatter is developed with the use of discrete wavelet transform without the prior fault information and knowledge of the process. A wavelet-based noise reduction method based on customized thresholding function with a universal threshold rule is also performed. Then the focus is on the formulation of chatter detection index by investigating the statistical distribution of positive WTMM. From equation 3.60, we can see that the proposed chatter recognition index does not involve parameters related to the material of workpiece, structure characteristics, and machining

conditions. It only reveals the statistical property of wavelet transform modulus maxima which in turn can detect the occurrence of chatter. Moreover, it always varies between 0 and 1.The need to re-calibrate the threshold value is avoided when each time cutting condition or process changes.

Chapter 4 Development of Chatter Suppression Module

In this chapter, a chatter suppression module is developed. It is based on a proposed novel control strategy. The strategy takes advantage of fuzzy logic control while incorporating a self-regulating algorithm which can on-line adjust the fuzzy control outputs and tune the controller parameters. The principal idea is to import the artificial intelligence and adaptability concepts into the conventional controller to deal with the conditions such as load disturbances, nonlinear dynamics of the process and the variation of plant parameters. The flow chart of the module is shown in Figure 4.1. From the chart, it can be seen that the basic fuzzy logic controller carries out controls and the self-regulating mechanism is responsible for adapting to the system variation. The self-regulating mechanism contains three components: performance measurement, controller parameter optimization and control output modification. The purpose of control system performance measurement is to establish a successful correcting criterion for the self-regulating controller. In this study, two inputs, i.e., the energy error and change of the energy error are monitored to measure the controller performance. The suitable values of controller parameters are found according to the optimization of a cost function. Consequently the correction value is calculated to modify the fuzzy control output. In this study, the objective of chatter suppression is to maintain a stable and productive cutting process as it is known that feed rate is relate to machining productivity. Therefore as soon as chatter is detected, the control system will act by adjusting the spindle speed and/or feed rate. In addition, a two-way adjustment, i.e., both increase and decrease of spindle speed or feed rate to preserve the productivity is proposed. The suppression criterion used in this study is chatter detection index. Chatter is considered to be suppressed when the index falls below its reference threshold.

4.1 Fuzzy logic controller for end milling processes

As shown in Figure 4.1, a basic fuzzy logic controller consists of five major parts: (1) a fuzzifier, which relates real numbers to fuzzy sets, (2) an inference engine to perform fuzzy reasoning using fuzzy rules, (3) a membership function, which defines how a fuzzy input in

the input space is mapped to a degree of membership between 0 and 1, (4) a rule base, which provides the inference engine with fuzzy control rules, and (5) a defuzzifier to transform the outcome of fuzzy inferences from fuzzy sets into crisp numbers. The design procedure of the underlying fuzzy logic controller for end milling process is described as follows.

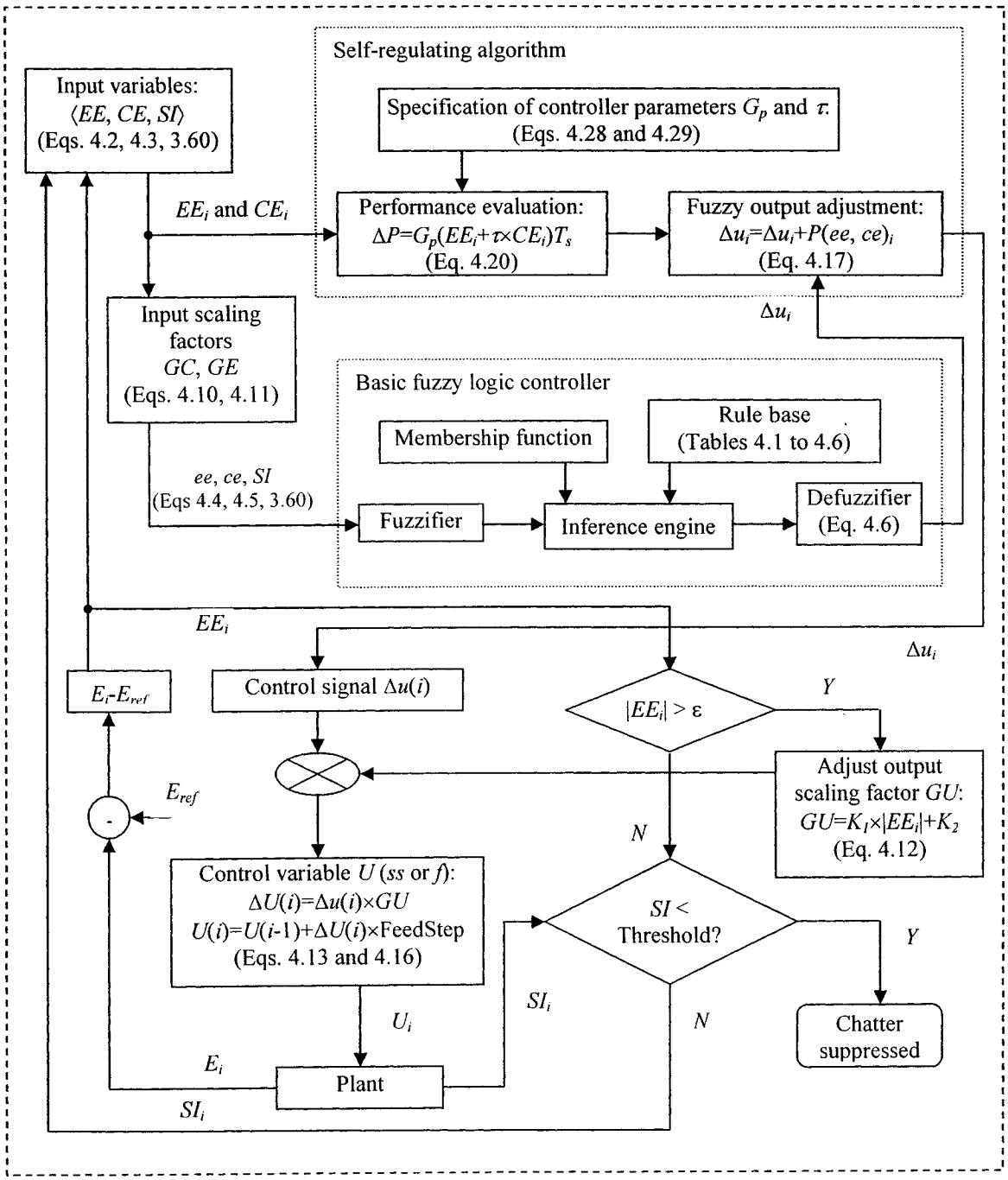


Figure 4.1 Flow chart of chatter suppression module

4.1.1 Design procedure of fuzzy logic controller for end milling

1) *Define input variables*

In this study, the mean vibration energy of a signal is defined as (Liang *et al* 2004):

$$Ei = \frac{1}{N}\sum_{i=1}^{N} x_i^{\,2} \tag{4.1}$$

where x_i is the input vibration signal at time i and N is the number of samples in a time window. Accordingly, three variables, energy error EE_i, change of energy error CE_i and chatter detection index SI_i, at a sampling time i, are used as input variables. In (Liang, *et al* 2003), EE_i and CE_i are respectively given by:

$$EE_i = E_{ref} - E_i \tag{4.2}$$

$$CE_i = EE_i - EE_{i-1} \tag{4.3}$$

where E_{ref} is the reference energy level, E_i is the current energy value and E_{i-1} is the energy value at previous time instant. EE_i and CE_i are then fuzzified to linguistic variables *ee* and *ce* by multiplying the corresponding input scaling factors. Consequently, *ee* and *ce* can be calculated as (Hsu and Fann 1996):

$$ee_i = \left(E_{ref} - E_i\right) \times GE = EE_i \times GE \tag{4.4}$$

$$ce_i = \left(EE_i - EE_{i-1}\right) \times GC = CE_i \times GC \tag{4.5}$$

where GE and GC are the input scaling factors for the energy error and change of the energy error, respectively. Moreover, the fuzzified linguistic variables *ee* and *ce* are divided into seven equal span fuzzy sets within the range [-1, 1]. The *SI* value is directly mapped into four fuzzy sets within interval [0, 1] since it always varies between 0 and 1. Note that in this study, the singleton fuzzifier is used.

2) *Construct fuzzy sets and membership functions*

The seven fuzzy sets for *ee* and *ce* are {*NL*, *NM*, *NS*, *ZE*, *PS*, *PM*, *PL*}, where *P*, *N*, *Z*, *S*, *M* and *L* correspond to positive, negative, zero, small, medium and large, respectively. Four fuzzy sets are defined for the input variable *SI*, they are {*ZE*, *PS* or *NS*, *PM* or *NM*, *PL* or *NL*}. For computation efficiency, the commonly used triangular membership functions are adopted and shown in Figure 4. in the domains of *ee* or *ce*. Figure 4. and Figure 4. depict the triangular membership functions in the domains of *SI* and fuzzy output universe *DU*.

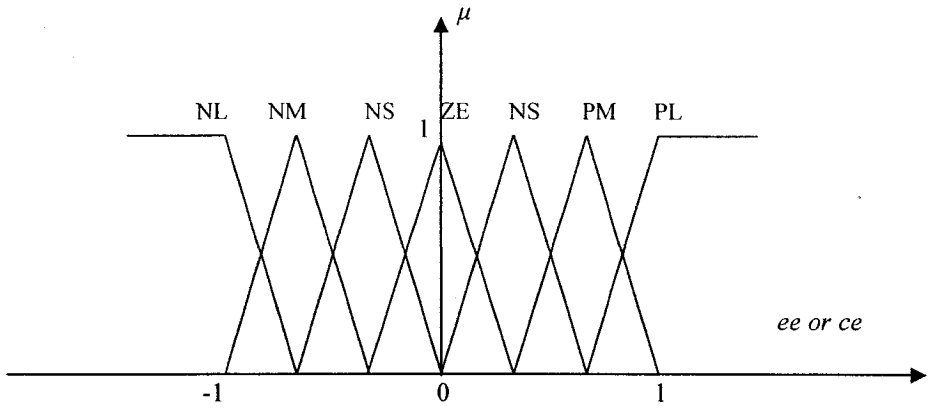


Figure 4.2 Membership functions of linguistic variables, *ee* and *ce*

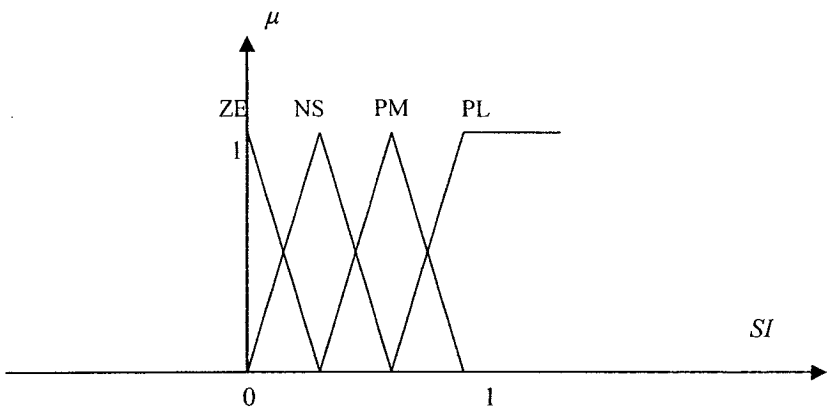


Figure 4.3 Membership functions of linguistic variable *SI*

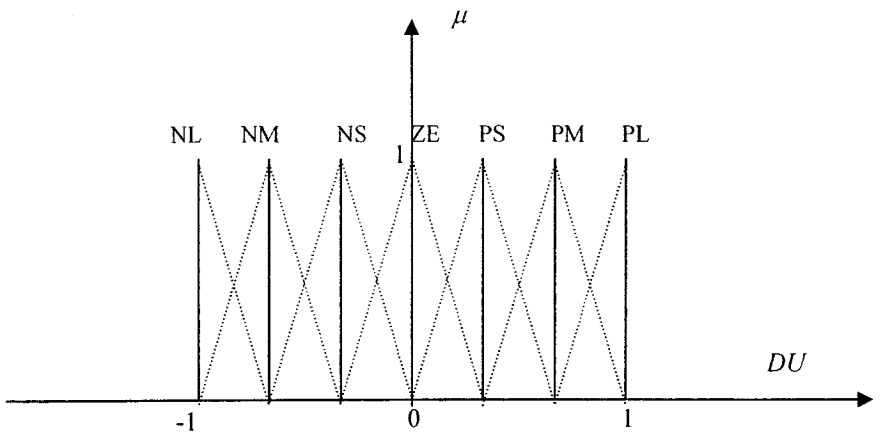


Figure 4.4 Membership functions of linguistic variable *DU*

3) *Construct fuzzy rule base*

In this study either spindle speed (*ss*) or both spindle speed and feed rate (*f*) are used as control variables for process control depending on whether single (spindle speed) or multi-parameter (spindle speed and feed rate) adjustment approach is applied. Traditionally, two-input fuzzy controller is used to perform control task. In our case, they would be *ee* and *ce*. In this research, the third input variable *SI* is added into the control strategy to reflect the chatter status and suppression effect. Its control rule base is constructed by taking productivity into account. As we know, chatter suppression can be achieved by conservatively selecting cutting parameters such as depth of cut and feed rate. However, it will lead to low productivity. In this study, a two-way strategy is considered, i.e., both increase and decrease of spindle speed and/or feed rate are allowed for chatter suppression. Increasing feed rate and accordingly spindle speed will improve productivity and in the meantime may either lead to a stable process or worsen the chatter condition, depending on the on-line location of stability lobes. On the other hand, reducing feed rate and accordingly spindle speed may not necessarily suppress chatter but will certainly reduce productivity. The stability lobes shift dynamically with the cutting conditions. Our purpose is to place the cutting process in a stable (chatter-free) and productive zone. Figure 4. and Figure 4. illustrate the above idea. In Figure 4., for the same depth of cut, i.e., the same productivity if feed rate is fixed, two-way adjustment can bring the spindle speed to a closer stability lobe. The dotted line is the stability lobe diagram at a previous time instant. In Figure 4., if the current spindle speed equals ss_1, we should bring it to ss_2 not ss_3 because ss_1 is closer to ss_2 but far away from ss_3. As a result, a total of six fuzzy rule bases associated with *ee*, *ce* and *SI* are initially constructed as shown in Table 4.1 to Table 4.6.

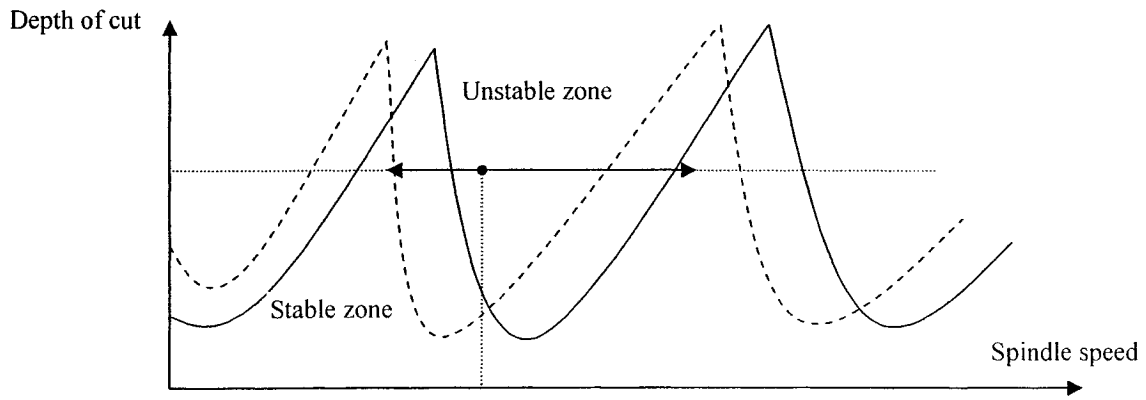


Figure 4.5 Shifted stability lobe and two-way adjustment of spindle speed

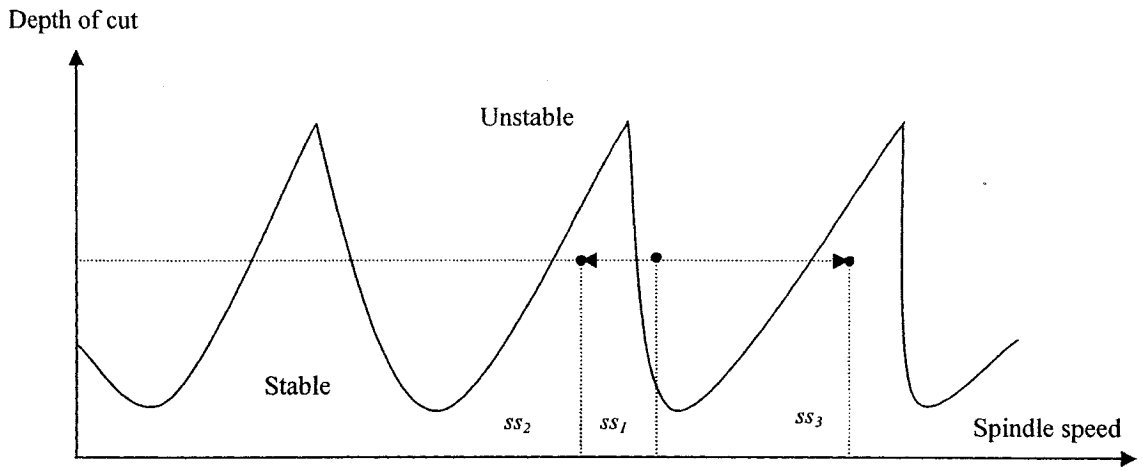


Figure 4.6 Stability lobes and two-way adjustment strategy with respect to spindle speed

Table 4.1 Feed rate fuzzy rule base for (*SI=ZE* and *PS*) based on *ee* and *ce*

ce \ *ee* (SI=ZE 1, SI=PS 2)	**NL 1**	**NM 2**	**NS 3**	**ZE 4**	**PS 5**	**PM 6**	**PL 7**
NL 1	NL	NL	NM	NM	NS	ZE	ZE
NM 2	NL	NM	NM	NS	ZE	ZE	ZE
NS 3	NM	NM	NS	ZE	ZE	ZE	PS
ZE 4	NM	NS	ZE	ZE	ZE	PS	PM
PS 5	NS	ZE	ZE	ZE	PS	PM	PM
PM 6	ZE	ZE	ZE	PS	PM	PM	PL
PL 7	ZE	ZE	PS	PM	PM	PL	PL

Table 4.2 Feed rate fuzzy rule base for (*SI=PM*) based on *ee* and *ce*

ce \ *ee* (SI=PM 3)	**NL 1**	**NM 2**	**NS 3**	**ZE 4**	**PS 5**	**PM 6**	**PL 7**
NL 1	NM	NM	NS	NS	ZE	PS	PS
NM 2	NM	NS	NS	ZE	PS	PS	PS
NS 3	NS	NS	ZE	PS	PS	PS	PM
ZE 4	NS	ZE	PS	PS	PS	PM	PL
PS 5	ZE	PS	PS	PS	PM	PL	PL
PM 6	PS	PS	PS	PM	PL	PL	PL
PL 7	PS	PS	PM	PL	PL	PL	PL

Table 4.3 Feed rate fuzzy rule base for (*SI=PL*) based on *ee* and *ce*

ce \ *ee* SI=PL 4	NL 1	NM 2	NS 3	ZE 4	PS 5	PM 6	PL 7
NL 1	NS	NS	ZE	ZE	PS	PM	PM
NM 2	NS	ZE	ZE	PS	PM	PM	PM
NS 3	ZE	ZE	PS	PM	PM	PM	PL
ZE 4	ZE	PS	PM	PM	PM	PL	PL
PS 5	PS	PM	PM	PM	PL	PL	PL
PM 6	PM	PM	PM	PL	PL	PL	PL
PL 7	PM	PM	PL	PL	PL	PL	PL

Table 4.4 Spindle speed fuzzy rule base for (*SI=ZE* and *PS*) based on *ee* and *ce*

ce \ *ee* SI=ZE 1 SI=PS 2	NL 1	NM 2	NS 3	ZE 4	PS 5	PM 6	PL 7
NL 1	NM	NM	NS	NS	ZE	ZE	ZE
NM 2	NM	NS	NS	ZE	ZE	ZE	ZE
NS 3	NS	NS	ZE	ZE	ZE	ZE	ZE
ZE 4	NS	ZE	ZE	ZE	ZE	ZE	PS
PS 5	ZE	ZE	ZE	ZE	ZE	PS	PS
PM 6	ZE	ZE	ZE	ZE	PS	PS	PM
PL 7	ZE	ZE	ZE	PS	PS	PM	PM

Table 4.5 Spindle speed fuzzy rule base for (*SI=PM*) based on *ee* and *ce*

ce \ *ee* SI=PM 3	NL 1	NM 2	NS 3	ZE 4	PS 5	PM 6	PL 7
NL 1	NS	NS	ZE	ZE	PS	PS	PS
NM 2	NS	ZE	ZE	PS	PS	PS	PS
NS 3	ZE	ZE	PS	PS	PS	PS	PS
ZE 4	ZE	PS	PS	PS	PS	PS	PM
PS 5	PS	PS	PS	PS	PS	PM	PM
PM 6	PS	PS	PS	PS	PM	PM	PL
PL 7	PS	PS	PS	PM	PM	PL	PL

Table 4.6 Spindle speed fuzzy rule base for (*SI=PL*) based on *ee* and *ce*

ce \ *ee* SI=PL 4	NL 1	NM 2	NS 3	ZE 4	PS 5	PM 6	PL 7
NL 1	ZE	ZE	PS	PS	PM	PM	PM
NM 2	ZE	PS	PS	PM	PM	PM	PM
NS 3	PS	PS	PM	PM	PM	PM	PM
ZE 4	PS	PM	PM	PM	PM	PM	PL
PS 5	PM	PM	PM	PM	PM	PL	PL
PM 6	PM	PM	PM	PM	PL	PL	PL
PL 7	PM	PM	PM	PL	PL	PL	PL

The fuzzy inference engine is used to determine the fuzzy output based on the fuzzy inputs and fuzzy rules. In this study, to reduce the computational time, the Sugeno-style fuzzy inference engine is adopted which uses singleton output membership functions as illustrated in Figure 4.. The T-norm (product) inferencing method as proposed in (Rodolfo *et al* 1998) is chosen and fuzzy implication is conducted by implementing Min-Max operation (Xu 2002). Defuzzification involves the process of converting the set of fuzzy output values into a crisp control command. Since singleton membership functions are used, the defuzzification result is simply the weighted average of the location of a few involved spikes as stated in (Liang *et al* 2003). As a result, the output change Δu which is used to adjust spindle speed or feed rate of CNC milling machine is obtained as:

$$\Delta u = \frac{\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{l=1}^{L}\mu_{jkl}C_{jkl}}{\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{l=1}^{L}\mu_{jkl}} \tag{4.6}$$

where each of subscripts j, k, l denotes the rules corresponding to their respective input variable, i.e., j is for the energy error, k is for change of the energy error, and l is for the *SI*. J, K, L are the number of fuzzy sets associated with j, k, l respectively. In this study, J=7, K=7, L=4. C_{jkl} is the location of the spike associated with output rule corresponding to j, k and l. μ_{jkl} is the height of the spike associated with j, k and l, and its value will be determined using the min-max operation as described later. For non-active rules, or non-active (j, k, l) combinations μ_{jkl}=0. Therefore only the activated rules in each control loop have net effect on the control action. For *ee* and *ce*, the fuzzy sets NL, NM, NS, ZE, PS, PM, and PL are

coded as 1, 2, 3, 4, 5, 6, and 7 respectively. For *SI*, the fuzzy sets ZE, PS, PM and PL are coded as 1, 2, 3 and 4.

The following example demonstrates how the fuzzy logic controller works. Suppose that *ee*, *ce* and *SI* have the fuzzy inputs of 0.35, -0.75 and 0.6, respectively. From Figure 4., it can seen that these fuzzy inputs are associated with six fuzzy sets (PS and PM sets in the *ee* domain, NM and NL sets in the *ce* domain, and PS and PM sets in the *SI* domain) and trigger the following eight fuzzy rules from Table 4.4 and Table 4.5 (assuming that the crisp control variable is spindle speed):

$Rule(5,1,2)$: (IF $ee \in PS \wedge ce \in NL \wedge SI \in PS$, THEN $\Delta ss \in ZE$)

$Rule(5,1,3)$: (IF $ee \in PS \wedge ce \in NL \wedge SI \in PM$, THEN $\Delta ss \in PS$)

$Rule(6,1,2)$: (IF $ee \in PM \wedge ce \in NL \wedge SI \in PS$, THEN $\Delta ss \in ZE$)

$Rule(6,1,3)$: (IF $ee \in PM \wedge ce \in NL \wedge SI \in PM$, THEN $\Delta ss \in PS$)

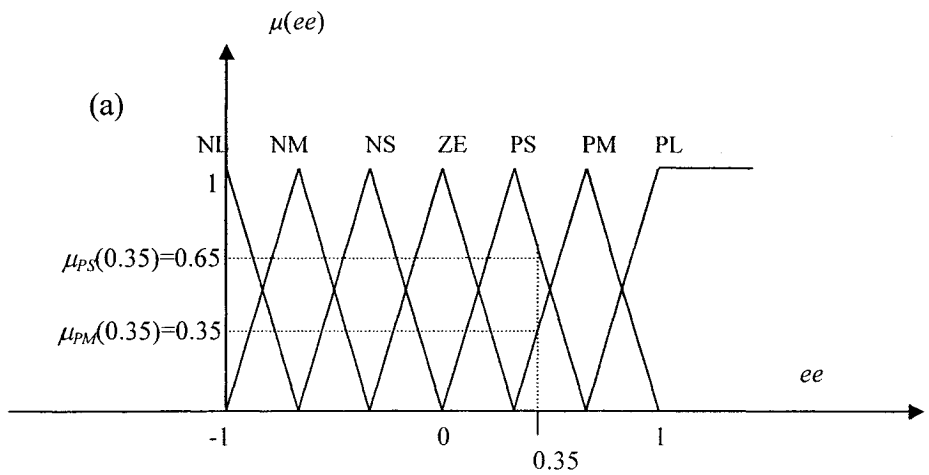
$Rule(5,2,2)$: (IF $ee \in PS \wedge ce \in NM \wedge SI \in PS$, THEN $\Delta ss \in ZE$)

$Rule(5,2,3)$: (IF $ee \in PS \wedge ce \in NM \wedge SI \in PM$, THEN $\Delta ss \in PS$)

$Rule(6,2,2)$: (IF $ee \in PM \wedge ce \in NM \wedge SI \in PS$, THEN $\Delta ss \in ZE$)

$Rule(6,2,3)$: (IF $ee \in PM \wedge ce \in NM \wedge SI \in PM$, THEN $\Delta ss \in PS$)

Figure 4. (a), (b), and (c) illustrate the membership degrees associated with the fuzzy sets based on the individual fuzzy input.



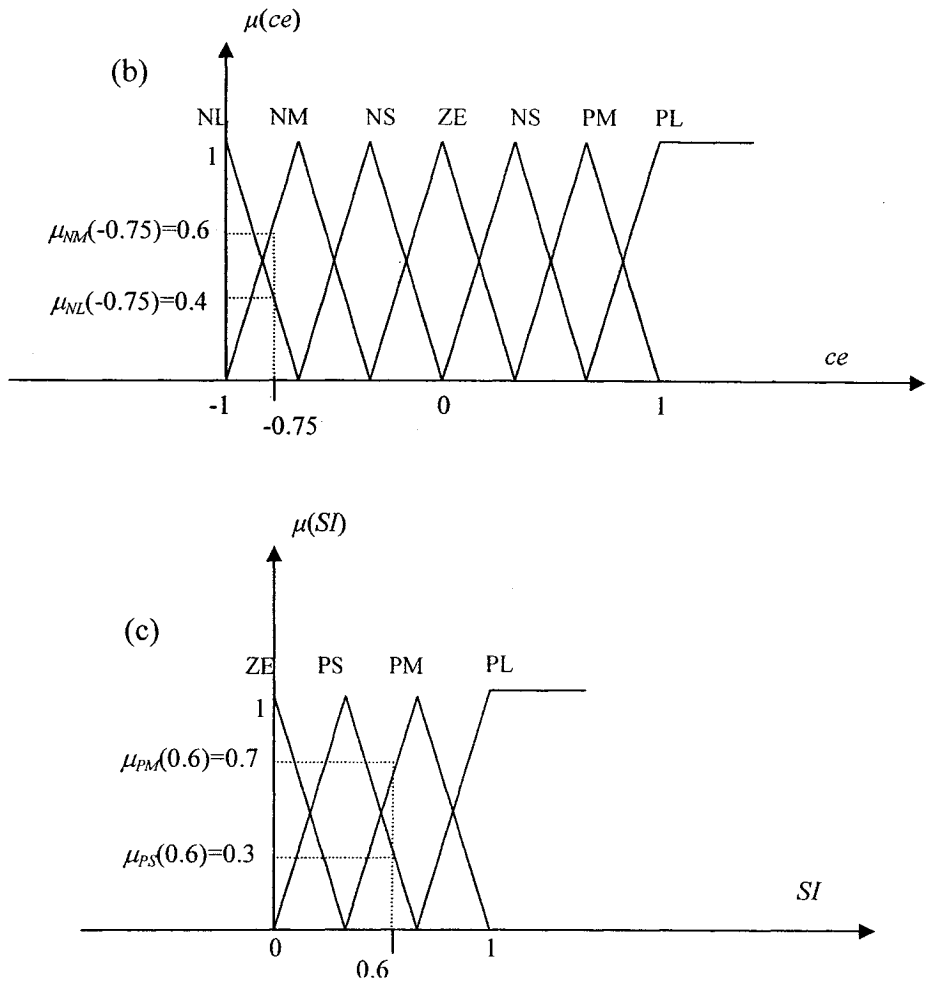


Figure 4.7 Illustrations of input membership degrees associated with three fuzzy inputs *ee*, *ce*, *SI*

The membership function output value for each rule is determined according to the min-max operation rule as follows (Xu 2002):

$$\mu_{jkl} = \min\left[\mu(ee)_j, \mu(ce)_k, \mu(SI)_l\right] \tag{4.7}$$

where μ_{jkl} is the output membership function value of rule (j, k, l) specified by inputs *ee*, *ce* and *SI*. $\mu(ee)_j$, $\mu(ce)_k$ and $\mu(SI)_l$ are the membership function values in *ee*, *ce* and *SI* domains. To illustrate, suppose that μ_{jkl}=min[$\mu(ee)_j$, $\mu(ce)_k$, $\mu(SI)_l$]=$\mu(ce)_k$. C_{jkl} is the location (i.e., the fuzzy value in the associated universe of discourse) of peak value associated with rule (j, k, l). For instance, in the above example, μ_{523}=min[$\mu(ee)_5$, $\mu(ce)_2$, $\mu(SI)_3$]=min[0.65, 0.60, 0.70]=$\mu(ce)_2$=0.60, and C_{523} = 0.333 since the output fuzzy set is PS for rule (5,2,3) and the location of the peak of PS set in the output domain is 0.333.

Continuing the above example, μ_{jkl} is obtained after the calculations as shown in Table 4.7. Meanwhile, eight activated rules illustrated above give us eight output fuzzy sets. The linguistic values of these eight fuzzy sets are PS, ZE, PS, ZE, PS, ZE, PS, ZE, respectively and their corresponding fuzzy values C_{jkl} are:

$$\begin{aligned} &C_{523}=0.333,\ C_{522}=0,\ C_{513}=0.333,\ C_{512}=0; \\ &C_{623}=0.333,\ C_{622}=0,\ C_{613}=0.333,\ C_{612}=0; \end{aligned} \tag{4.8}$$

These fuzzy values have to be converted into physical value before they can be used as control commands. This task is accomplished through defuzzification using the weighted average method as shown in Figure 4.. As both μ_{jkl} (shown in Table 4.7) and C_{jkl} (listed in equation (4.8)) are available, the final crisp output for the above example is calculated based on equation (4.6) as below:

$$\Delta u=\frac{0.6\times0.333+0.3\times0+0.4\times0.333+0.3\times0+0.35\times0.333+0.3\times0+0.35\times0.333+0.3\times0}{0.6+0.35+0.3+0.3+0.4+0.35+0.3+0.3}=0.195 \tag{4.9}$$

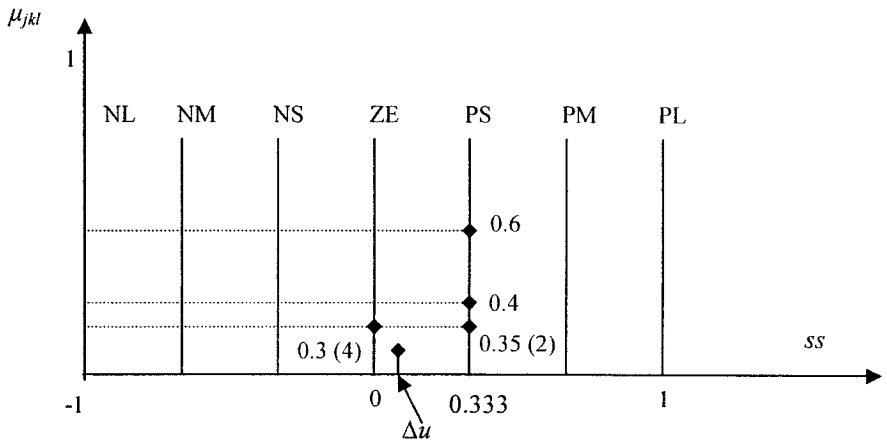


Figure 4.8 Defuzzification using singleton membership functions

Table 4.7 Calculations of the coefficients of output fuzzy rules

μ_{jkl}	j=5⇒ee:μ_{PS}(0.35)=0.65	j=6⇒ee:μ_{PM}(0.35)=0.35
	l=3⇒SI:μ_{PM}(0.6)=0.7 l=2⇒SI:μ_{PS}(0.6)=0.3	l=3⇒SI:μ_{PM}(0.6)=0.7 l=2⇒SI:μ_{PS}(0.6)=0.3
k=2⇒ce:μ_{NM}(-0.75)=0.6	μ_{523}=min[0.65,0.6,0.7]=0.6 μ_{522}=min[0.65,0.6,0.3]=0.3	μ_{623}=min[0.35,0.6,0.7]=0.35 μ_{622}=min[0.35,0.6,0.3]=0.3
k=1⇒ce:μ_{NL}(0.75)=0.4	μ_{513}=min[0.65,0.4,0.7]=0.4 μ_{512}=min[0.65,0.4,0.3]=0.3	μ_{613}=min[0.35,0.4,0.7]=0.35 μ_{612}=min[0.35,0.4,0.3]=0.3

4.1.2 Input and output scaling factors

The input scaling factors *GE*, *GC* and output scaling factor *GU* are defined in this sub-section. Input scaling factor *GE* for the energy error is given as follows (Liang *et al* 2003):

$$\begin{cases} GE_{(up)} = \dfrac{1}{E_{up} - E_{ref}} & \text{if } E_i \geq E_{ref} \\ GE_{(low)} = \dfrac{1}{E_{ref} - E_{low}} & \text{if } E_i < E_{ref} \end{cases} \tag{4.10}$$

It can be seen that the energy error scaling factor is separately considered in the zones above and below the reference energy due to the asymmetry of the two zones. As the change of energy error *CE* may vary over the entire range between the maximal and minimal allowable limits, E_{up} and E_{low} respectively, it is not necessary to separately consider the scaling factor for the change of energy error. Therefore as in (Liang *et al* 2003) it is given by:

$$GC = \frac{1}{CE_{\max} - CE_{\min}} \tag{4.11}$$

The output from fuzzy controller after the defuzzification has to be multiplied by the output scaling factor *GU* to become the physical control command. In this study, the output scaling factor *GU* is defined as per equation (4.12) (Rodolfo *et al* 1998). The logic behind this equation is that the output scaling factor functions as an overall controller gain factor (Hsu and Fann 1996). In addition, the control system's behavior in terms of the accuracy of controller can be evaluated by mean square error (MSE) of energy and the relation among K_1, K_2 and MSE can be depicted as a spatial surface if plotted in a 3-D coordinate system (Rodolfo *et al* 1998). Minimizing the MSE which can be achieved through the tangent of the surface ensures a constant cutting despite disturbances. Therefore the output scaling factor *GU* in equation (4.12) is expressed as a form of linear function and the absolute value of the energy error is used as an adjustment criterion:

$$GU = K_1 \times |EE_i| + K_2 \tag{4.12}$$

where K_1 and K_2 are constants and will be determined during the experiments by minimizing the mean square energy error.

The adjustment per control cycle, $\Delta U(i)$ is accordingly calculated as follows:

$$\Delta U(i) = \Delta u(i) \times GU \tag{4.13}$$

4.1.3 Constraints to the proposed control system

The output control command $\Delta u(i)$ from fuzzy controller could lead to an excessively large amount of adjustment of spindle speed or feed rate. It can not be followed by the underlying milling machine. Therefore, severely unstable cutting process and even tool or machine failure could occur. For this reason, a relation between control cycle and maximum allowable machine adjustment is established. The feed step and speed step, namely the feed rate change and spindle speed change during a single control cycle, are defined as follows (Liang *et al* 2003):

$$FeedStep = \frac{FeedOutputRange}{CyclesPerSecond} \tag{4.14}$$

$$SpeedStep = \frac{SpeedOutputRange}{CyclesPerSecond} \tag{4.15}$$

where FeedOutputRange and SpeedOutputRange are the voltage values associated with the adjustment ranges of feed rate and spindle speed, respectively. CyclesPerSecond is the number of control cycles per second. With these restrictions, the machine is constrained to make a full range adjustment within one second rather than within a single control cycle. Therefore the control output equation for either the spindle speed or feed rate command $U(i)$ is given by:

$$U(i) = U(i-1) + \Delta U(i) \times FeedStep \text{ or } SpeedStep \tag{4.16}$$

4.1.4 Some other constraints

In addition to the above adjustment constraints, other technical restrictions should also be considered to secure a reliable machining process as listed in Table 4.8.

Table 4.8 Some additional constraints to control system

Constraints	Conditions
$(f, ss)=(f, ss)_{min}$	$E > E_{max}$
$(f, ss)=(f, ss)_{max}$	$E < E_{min}$
$(f, ss)=0$	$E > E_{upperlimit}$
$(f, ss)=(f, ss)_{lowerlimit}$	$(f, ss) < (f, ss)_{lowerlimit}$
$(f, ss)=(f, ss)_{upperlimit}$	$(f, ss) > (f, ss)_{upperlimit}$

4.2 Development of self-regulating algorithm

The machining process control involves systems whose behaviors are difficult to describe mathematically, especially for the milling process due to its nonlinear dynamics and time-varying cutting parameters. An adaptive fuzzy logic control has the potential to achieve a better control performance than model-based controllers as commented in (Hsu and Fann 1996). Therefore in this study a fuzzy logic controller with self-regulating capability is developed to tackle the nonlinear system involved in end milling process.

From Figure 4.1, it can be seen that the self-regulating algorithm consists of three major components. Their functions are stated at the beginning of this chapter. The idea is to let this adjustment mechanism monitor the energy error EE_i, change of the energy error CE_i and accordingly modify the fuzzy control output Δu_i based on the performance of controller. If the performance is poor, the output receives a change so that next time it can make better adjustment and the performance of controller will improve. The magnitude of each change to the fuzzy output is decided by the performance measure $P(ee, ce)$ which can either be the preset numbers as shown in Table 4.9, expressing the desired transient response, or a practical adjustment equation as it will be studied in the following sub-section. Also due to the time delay, i.e., the time for a control signal to propagate to the output of plant, the output to be modified cannot be the current output and thus the mechanism has to step back in time in order to find the responsible one. Furthermore, by taking a cost function into account, the fuzzy controller parameters can be determined automatically. As a result, the suggested self-regulating algorithm can be incorporated with fuzzy control strategy to perform effectively.

4.2.1 Control output adjustment mechanism

The current states of system are evaluated and a performance measure $P(ee, ce)_i$ is returned by the self-regulating adjustment mechanism. Here a zero performance measure, $P(ee, ce)_i = 0$, implies that the state $(ee, ce)_i$ is satisfactory because there is no control action needed to take. Any non-zero $P(ee, ce)_i$ which corresponds to a nonzero quantity will indicate an unsatisfactory state. In the latter case, the mechanism assumes that the control output must be reinforced by the amount of $P(ee, ce)_i$ to reflect the deviation from an ideal condition. However, since some time is required before a control action can take effect in the

plant output, and the current control output does not account for the action, it needs to go back a number of samples d in time to correct an earlier one. The self-regulating algorithm must therefore know the time lag of the plant. The integer d is chosen as comparable to the plant time lag and here d is called the reinforcement delay. Now the precise adjustment for the control output can be expressed as (Note that the following equations from (4.17) to (4.23) are adopted from (Jan 1998)):

$$\Delta u_{i-d} = \Delta u_{i-d} + P(ee, ce)_i \tag{4.17}$$

where the time subscript i denotes the current sample. In other words, the self-regulating algorithm regards the performance measure as an extra contribution to the control output in order to push the plant output to a state with zero deviation.

Table 4.9 Example of performance measures (Yamazaki 1982)

		ce												
ee		**-6**	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	-6	-6	-6	-6	-6	-6	-6	-6	-5	-4	-3	-2	-1	0
	-5	-6	-6	-6	-6	-5	-4	-4	-4	-3	-2	-1	0	0
	-4	-6	-6	-6	-5	-4	-3	-3	-3	-2	-1	0	0	1
	-3	-6	-6	-5	-4	-3	-2	-2	-2	-1	0	0	1	2
	-2	-6	-5	-4	-3	-2	-1	-1	-1	0	0	1	2	3
	-1	-5	-4	-3	-2	-1	-1	0	0	0	1	2	3	4
	0	-5	-4	-3	-2	-1	0	0	0	1	2	3	4	5
	1	-3	-2	-1	0	0	0	0	1	1	2	3	4	5
	2	-2	-1	0	0	0	1	1	1	2	3	4	5	6
	3	-1	0	0	0	1	2	2	2	3	4	5	6	6
	4	0	0	0	1	2	3	3	3	4	5	6	6	6
	5	0	0	1	2	3	4	4	4	5	6	6	6	6
	6	0	1	2	3	4	5	6	6	6	6	6	6	6

For example, assume d=2 and variable data are given in Table 4.10. From t =1 to t =4 the plant follows the desired trajectory because the performance measures $P(ee, ce)_i$ are zeros. At t =5, error changes the sign indicating an overshoot, and the performance measure reacts by indicating $P(ee, ce)_5$ = -1. Since d is 2, the output to be adjusted will be at the previous position corresponding to $t = i\text{-}d$ = 5-2 = 3. At that sampling instant, the state was $(ee, ce)_3$=(1, -2) and the control was Δu_3=-1. Hence the adjusted command is $\Delta u_3 = \Delta u_3 + P(ee, ce)_5$=-1-1= -2.

Table 4.10 Data for the above example

	Time t				
Variables	1	2	3	4	5
ee	6	3	1	0	-1
ce		-3	-2	-1	-1
u		0	-1	-1	-2
P(ee, ce)		0	0	0	-1

Table 4.9 illustrates an example of desired performance measure table. Noticing that the zero diagonal on the table expresses the following linear relation:

$$GE \times EE + GC \times \frac{dEE}{dt} = ee + ce = 0 \tag{4.18}$$

Based on the above equation, to obtain performance measures in a more general way, the expression in equation (4-18) can be modified as:

$$GE \times EE + GC \times \frac{dEE}{dt} = ee + ce = P(ee, ce) \tag{4.19}$$

However, in this study as GE and GC are already separately specified in equations (4.10) and (4.11), instead of using GE and GC, a performance measure $P(ee, ce)$ is established by introducing the target time constant τ and adaptation gain G_p which affects the convergence rate and sample period T_s (Jan 1998). Therefore equation (4.19) is rewritten as:

$$GE \times EE + GC \times \frac{dEE}{dt} = ee + ce = P(ee, ce)$$

$$\Rightarrow GE \times EE + GC \times CE = P(ee, ce)$$

$$\Rightarrow GE \times (EE + (GC/GE) \times CE) = P(ee, ce)$$

Let $GE=G_p$, $GC/GE=\tau$ and consider the instantaneous value of EE, CE within the sampling period T_s, then the increment of $P(ee,ce)$ during T_s, defined as ΔP, becomes:

$$\Rightarrow \Delta P = P(ee, ce) \times T_S = GE \times (EE + (GC/GE) \times CE) \times T_S$$

$$\Rightarrow \Delta P = G_p \times (EE_i \times T_s + \tau \times T_s \times CE_i) = G_p \times (EE_i + \tau \times CE_i) \times T_s \tag{4.20}$$

The proposed adjustment mechanism in terms of equation (4.20) is simple and eliminates the difficulty in establishing an appropriate performance decision table such as Table 4.9. It also can adapt to the change of system dynamics faster by tuning the parameters

τ and G_p, therefore increasing the computing speed (Jan 1998). The principles in tuning these parameters of self-regulating controller are described below:

- ***Target time constant τ***

Smaller τ leads to faster response. If it is too small, however, the closed loop system cannot possibly follow the desired trajectory. As a result a large overshoot may appear. Therefore the range of τ is given as follows:

$$T_p \le \tau \le T_p + \tau_p \tag{4.21}$$

where time constant τ_p and dead time T_p are plant parameters. In this study, time constant can be determined by observing the time lapsed between two consecutive overshoots with self-regulating algorithm disabled. Dead time can be measured roughly by the time between consecutive changes of energy values.

- ***Reinforcement delay d***

Basically d is related to both plant and control system, therefore it should be chosen with respect to target time constant and sample period. The following equation shows that d can be obtained by dividing target time constant by sample period and rounding the result to the nearest integer:

$$d = \left[\tau / T_s\right] \tag{4.22}$$

- ***Adaptation gain G_p***

Larger G_p can result in faster convergence of control system to steady state. However if it is too large, the regulating process may become unstable. The following inequality gives a reasonable upper bound of G_p:

$$G_p \le \frac{0.2 \times \left|F\left(ee, ce\right)\right|_{\max}}{\left|\left(EE_i + \tau \times CE_i\right)\right|_{\max} \times T_s} \tag{4.23}$$

In equation (4.23), $F(ee, ce)_{max}$ denotes the largest control value calculated by fuzzy controller without the involvement of self-regulating algorithm. $|EE_i + \tau \times CE_i|_{max}$ can be obtained according to the largest recorded EE_i and CE_i values during the occurrence of chatter while keeping the self-regulating mechanism deactivated and τ at its upper bound.

4.2.2 Auto-tuning of controller parameters

From the above discussion, we notice that if τ and G_p are determined, that is, they can be tuned to the appropriate values so that the controller is able to achieve quick convergence

to move the cutting process to stable status, the objective of chatter suppression is achieved.

In this study, τ and G_p are obtained by considering a cost function, i.e., through the optimization. The term of optimization is used here because the selections of τ and G_p are based on a criterion of minimizing the instantaneous control energy as proposed in (Hazem and Kevin 2004):

$$J(A,\rho) = J_{te}(A) + J_u(\rho, \varsigma_{te}) \tag{4.24}$$

where $J_{te}(A)=te^2(i)$, $J_u(\rho, \varsigma_{te})=u^2(i)$, $te(i)$ is the instantaneous tracking error, $u(i)$ is the instantaneous control, ρ is the adaptation gain, ς_{te} is an incremental vector and A is defined as the approximation of the ideal approximator parameter vector A^* (The fuzzy system can be used as the approximator to approximate either the plant or the controller). Since A is related to the transfer function of machining process which is unknown and the purpose of this study is to develop a fuzzy control strategy without the prior knowledge of the plant, the determination of A and the first term of equation (4.24) is not of our particular interest. Only the second term of equation (4.24) is considered for this research. The pair of (ρ, ς_{te}) has the same interpretation as (G_p, τ), i.e., ρ corresponds to G_p denoting the adaptation gain on which the magnitude of each new control value depends, and ς_{te} corresponds to τ expressing how fast the control system can follow the desired response, and they are related to control energy $u^2(i)$. Therefore, the second term of equation (4.24) is rewritten, in terms of (G_p, τ), as:

$$J_u(\rho, \varsigma_{te}) = J_u(G_p, \tau) = u^2(i) \tag{4.25}$$

Temporarily disregarding the reinforcement delay d in equation (4.17) or if d equals to 0, the control output Δu can be obtained as follows:

$$\Delta u_i = \Delta u_i + P(ee, ce)_i = \Delta u_i + G_p(EE_i + \tau \times CE_i)T_s \tag{4.26}$$

where the first term is the fuzzy controller output and the second term is the performance measure.

To obtain the "optimal" values of G_p and τ within the ranges given by equations (4.21) and (4.23), the cost function as shown in equation (4.25) has to be minimized. Since the current control $u(i)$ is the sum of previous control $u(i\text{-}1)$ and $\Delta u(i)$, and the previous control $u(i\text{-}1)$ is a constant, to minimize $u(i)$ implies the minimization of Δu and equation (4.25) is expressed as:

$$\min J_u(G_p, \tau) \rightarrow \min \Delta u(i) \tag{4.27}$$

From equations (4.26) and (4.27), it can be observed that to minimize the new Δu, $P(ee, ce)_i$ must be minimized since the first term of equation (4.26) is an old Δu which is a constant calculated by previous adjustment. Additionally, due to the fact that G_p is always greater than zero, the "optimal" target time constant τ^* can be determined by fixing G_p and searching via the following criteria (assuming the ranges of τ and G_p are $[\tau^l, \tau^u]$ and $[G_p^l, G_p^u]$, respectively):

$$\begin{cases} \text{I: For } \forall \tau \in [\tau^l, \tau^u], \text{ if } (EE_i + \tau \times CE_i) < 0, \\ \text{select } \tau^* \in [\tau^l, \tau^u] \text{ such that for } \forall \tau \in [\tau^l, \tau^u] \cap \tau \neq \tau^*, \left|EE_i + \tau^* \times CE_i\right| \geq \left|EE_i + \tau \times CE_i\right| \\ \text{II: For } \forall \tau \in [\tau^l, \tau^u], \text{ if } (EE_i + \tau \times CE_i) > 0, \\ \text{select } \tau^* \in [\tau^l, \tau^u] \text{ such that for } \forall \tau \in [\tau^l, \tau^u] \cap \tau \neq \tau^*, \left|EE_i + \tau^* \times CE_i\right| \leq \left|EE_i + \tau \times CE_i\right| \end{cases} \quad (4.28)$$

Once τ^* is found, the "optimal" value of adaptation gain G_p^* can be searched by fixing τ at its "optimal" value and considering the following rules:

$$\begin{cases} \text{I: if} (EE_i + \tau^* \times CE_i) < 0, \; G_p^* = G_p^u \\ \text{II: if} (EE_i + \tau^* \times CE_i) > 0, \; G_p^* = G_p^l \end{cases} \quad (4.29)$$

For example, target time constant τ varies within the range of $5 \leq \tau \leq 15$ seconds and the upper bound of adaptation gain G_p is 20, i.e., $G_p \leq 20$. Current error EE_i is -5 and change of the error CE_i is -2. Last control signal Δu^{old} is 2.5 and sample period is 0.001 seconds. According to equation (4.26), we have:

$$\begin{aligned} \Delta u^{new}(i) &= \Delta u^{old}(i) + P(ee, ce)_i = \Delta u^{old}(i) + G_p (EE_i + \tau \times CE_i) T_s \\ &= 2.5 + G_p (-5 + \tau \times (-2)) \times 0.001 \end{aligned} \quad (4.30)$$

By the criteria of equation (4.28), it is obvious that τ = 15 is the "optimal" value by searching its range. Again the "optimal" G_p is determined by the rules shown in equation (4.29) and apparently G_p=20 is the best value which meets the requirements. The new control signal Δu^{new} is then calculated via equation (4.30) and equals to 3.2.

4.3 Implementation procedure of proposed control strategy

The control objective in this study is to maintain a stable cutting process, in particular, to keep the *SI* value below the threshold because it is the only index indicating the existence of chatter. The implementation procedure of proposed control strategy is described below.

Step1: Determine some control-related parameter values

Some control-related parameters have to be known in advance. Otherwise the control strategy cannot be applied. The set point of mean vibration energy needs to be determined through experiments, as well as the maximal and minimal values of EE, CE, and mean vibration energy E. In addition, to set the upper bound of G_p, the largest control output value calculated by fuzzy controller, $F(ee, ce)_{max}$, is necessary to be determined. CyclesPerSecond value is used for determining the FeedStep and SpeedStep and should also be specified.

Step 2: Determine plant-related parameter values

In this study a self-regulating algorithm is applied to the underlying fuzzy controller to make it adaptive to system variation and process disturbances. From the discussion in Section 4.2.1, it is known that the determination of G_p and τ is crucial since they are related to the controller's adaptability. This can be accomplished by first pinpointing their ranges. According to equations (4.21) and (4.23), time constant τ_p, dead time T_p and upper bound of G_p are the key elements in determining these ranges. Section 4.2.1 specifies the way of how to decide τ_p, T_p and G_p, and Section 4.2.2 gives the detailed description of how to obtain G_p and τ by considering a cost function.

Step 3: Calculate vibration energy and obtain input variables

The control strategy in chatter suppression module is triggered once the detection index SI identifies the occurrence of chatter. The mean vibration energy is then calculated according to equation (4.1). Three input variables, i.e., the energy error EE and change of the energy error CE calculated by equations (4.2) and (4.3) together with the index SI are fed into the proposed fuzzy controller. Consequently, the change of control output is generated from the fuzzy control system.

Step 4: Perform self-regulating algorithm to obtain G_p^*, τ^* and $P(ee,ce)_i$

As demonstrated in Section 4.2.2, the "optimal" values of controller parameters G_p and τ can be obtained. The performance measure $P(ee, ce)$ is calculated based on equation (4.20). Thus a new control command value which is either spindle speed or feed rate is calculated as per equations (4.13), (4.16) and (4.17), and transmitted to the plant.

Step 5: Verify control performance for chatter suppression

By observing the energy error EE and the index value SI, the chatter status can be evaluated. If the energy error decreases and eventually reaches the reference energy level or

below, the cutting process tends to become stable. However, the confirmation of chatter suppression must be convinced by the *SI* value which falls below the reference threshold.

4.4 Conclusions

This chapter presents two chatter suppression strategies which are fuzzy control and self-regulating fuzzy control. The latter features a cost function which leads to a tuning mechanism to obtain the controller parameters comparing with other conventional fuzzy controllers. Equipped with a regulating function for output scaling factor, the proposed controller is expected to achieve better control performance. The two-way adjustment approach is proposed to preserve productivity. Moreover, since the fuzzy control output modification can be fulfilled by on-line adjustment mechanism, it reduces significantly the trial-and-error efforts that other control strategies may need to conduct.

Chapter 5 Experiments

This chapter presents the experimental work and results obtained from end milling machining. The proposed chatter detection and suppression methodologies have been tested on 1018 cold rolled steel workpiece with different profiles and depth-of-cuts. The purpose is to assess the performance of the presented approaches and to verify their effectiveness and feasibility when applied directly to industrial CNC machine. It should be noted that the *SI* value is used as the only analytical criterion for chatter detection in the following tests. The chatter, once detected, i.e., the *SI* is above its reference level, will be suppressed based on both *SI* and the energy related information, i.e., *ee*, and *ce*.

5.1 Experimental apparatus

The systematic configuration is illustrated in Figure 5.1. The chatter detection and suppression system consists of the following major hardware components:

- CNC SERVO 2000 milling machine
- Wilcoxon 993B-6 hermetic tri-axial accelerometer
- Wilcoxon P703BT power unit
- National Instruments SCB-100 interface box
- National Instrument AT E Series Board: AT-MIO-16DE-10 data acquisition card
- Intel Pentium III desktop PC with LabWindows/CVI 5.0 installed

5.1.1 CNC SERVO 2000 milling machine

The experiments were conducted on a three-axis vertical CNC milling machine, CNC SERVO 2000, made by Servo Products Company as shown in Figure 5.2. Its 3 HP spindle motor is driven by a three-phase G3 adjustable frequency AC drives and the manufacturer is Saftronics Inc. The gearbox provides two speed range selections. Low gear gives the speed variation from 50 to 500 rpm and high gear covers the range of 500 to 5000 rpm. The spindle speed can be adjusted either by the commands from the servo operator console (SOC) which is a built-in computer, or manually by regulating the meter on the spindle panel. For chatter suppression, the spindle speed is controlled by an external computer. The digital

spindle speed commands are calculated by the external computer and then converted into the analog signals via A/D card on a DAQ board. Feed rate adjustment commands can be programmed using the built-in computer. Another way to change feed rate is through the use of pendant. The SOC paired with a pendant can regulate the feed rate override ranging 0 ~ 150%. In this study, feed rate control commands are acquired from the external computer via the pendant.

The CNC SERVO 2000 has a three-axis sliding table which can drive the workpiece moving along a specified direction. The movements in X-axis (table) and Y-axis (cross) are manipulated by SERVO II axis drive motors, and Z-axis is manually operated.

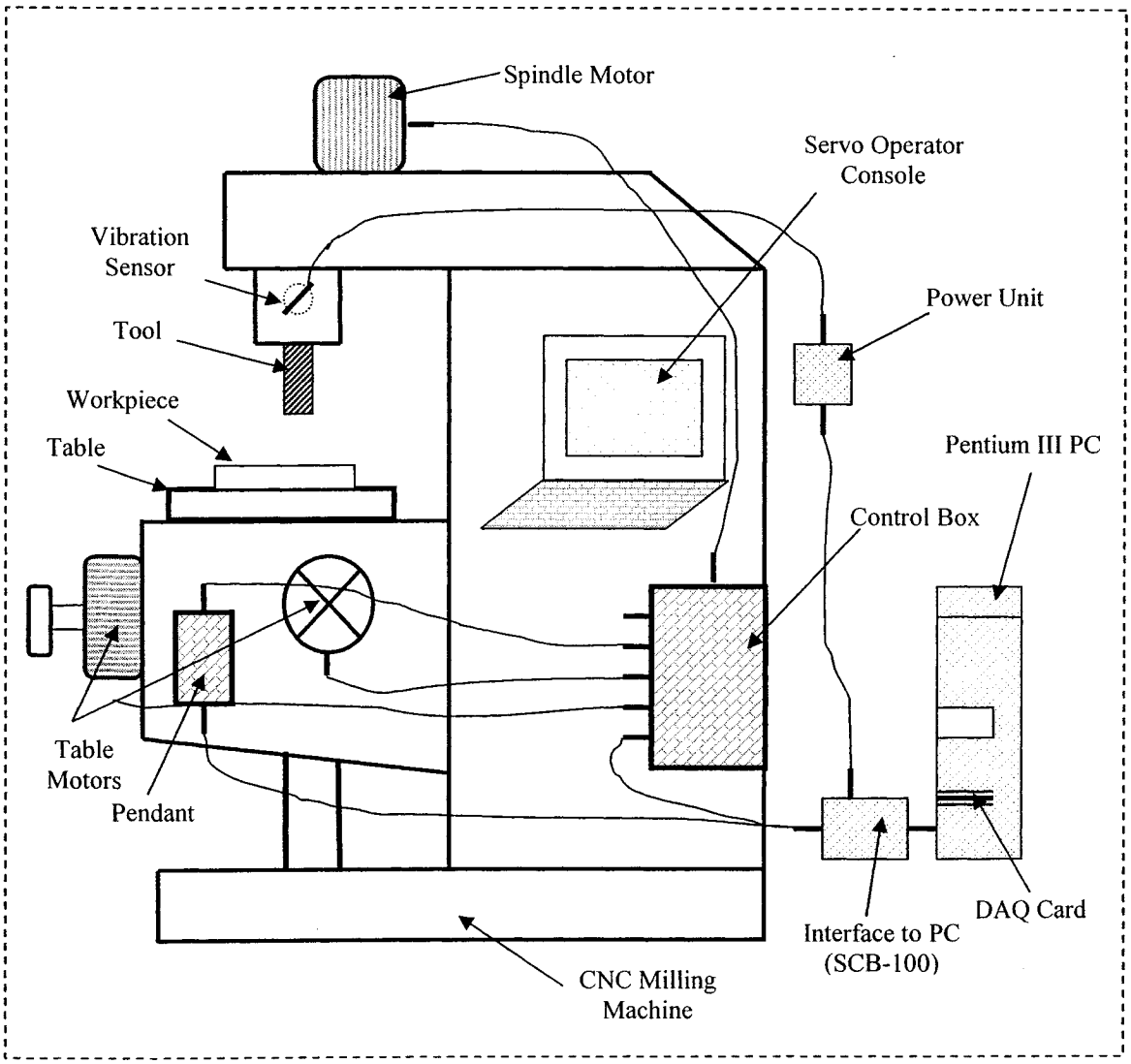


Figure 5.1 System hardware configuration

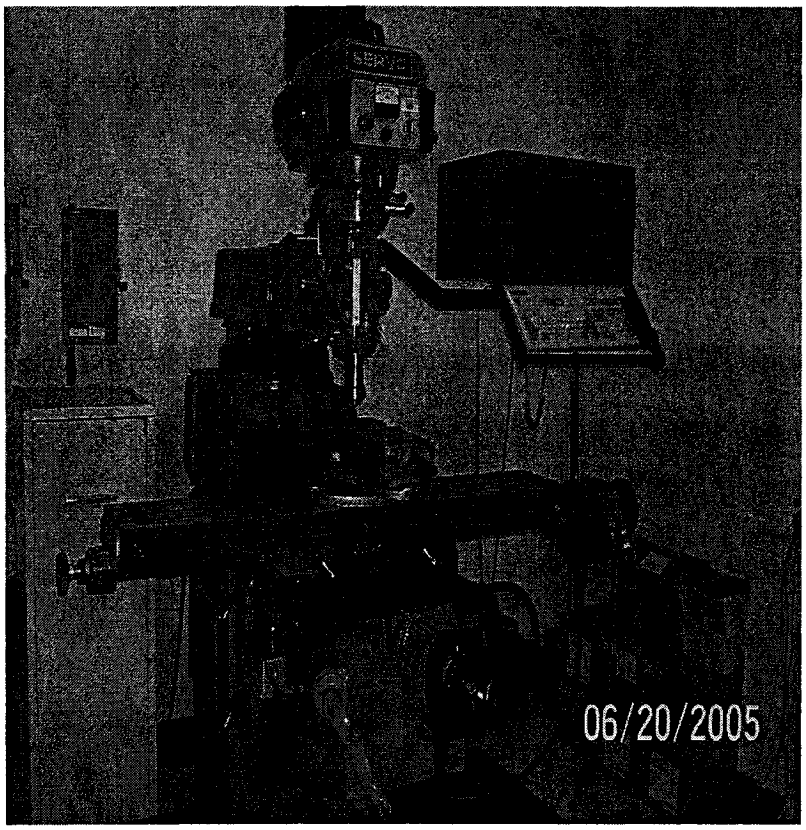


Figure 5.2 CNC SERVO 2000 milling machine

5.1.2 Vibration sensor

Vibration sensors provide a low-cost and easy-to-install solution to a variety of machining processes. The one used in this study is a hermetic tri-axial accelerometer (993B-6) supplied by Wilcoxon Research Inc. It is mounted on CNC machine near the tool holder where the vibration signal that dynamically reflects the cutting conditions is able to be collected. The Wilcoxon 993B-6 sensor can acquire the signals with frequency response ranges of 2~10,000 Hz for Z-axis and 2~7000 Hz for X and Y axes. A power unit, P730BT supplying this sensor with power, has three identical outputs in X, Y and Z directions. However, in our tests only the X direction signal is used because the main strength of the signal is in X direction which is the cutting direction.

5.1.3 Data acquisition (DAQ) board

The DAQ board used in this study is the National Instrument AT E Series Board (AT-MIO-16DE-10) which is a multifunction input-output device with up to 100kS/sec reading

capability. It also characterizes a 12-bit analog-to-digital converter for 16 single ended or 8 differential analog inputs, two channels of 12-bit digital-to-analog converters for outputs, two up/down 24-bit counter/timer, and 32 digital I/O lines. The analog signal from the vibration sensor can be digitized at up to 40,960 samples/second for each direction. To ensure data integrity, a double buffering method is utilized. The digital control commands in terms of spindle speed and feed rate from the external computer are converted to analog voltage output signals which range from 0 to 5V for feed rate (corresponds to 0 to 150% feed rate override) and 0 to 10V for spindle speed (corresponds to 50 to 500 rpm in low gear and 500 to 5000 rpm in high gear forward spindle rotation), respectively.

5.1.4 Software and development kit

LabWindows™/CVI 5.0 is adopted as the development platform. The entire chatter detection and suppression system is programmed in C language for on-line implementation. In particular, some library functions in the wavelet transform toolbox of National Instruments (NI) Signal Processing Toolset are used to perform discrete wavelet transform algorithm. A modified "fuzzyeng.c" file which is originally developed for fuzzy logic toolbox in MATLAB is applied for implementing the fuzzy engine. A "realft.c" file is used to carry out fast Fourier transform (FFT) for the data with real number. The LabWindows™ panel is designed to on-screen display the collected sensor data, to conduct chatter detection analysis, and to view suppression effect as shown in Figure 5.3.

5.2 Determination of the experimental parameters

Through the experiments, the following parameters required for system have been determined as:

- Value of reference energy level E_{ref}=0.8
- Values of maximal and minimal energy levels, E_{max}=4.5 and E_{min}=0
- Time constant τ_p=0.5 and dead time T_p=0.08
- Coefficients of output scaling factor function, K_1 and K_2
- Maximal energy error and change of energy error EE_{max}=3.7, CE_{max}=3.7
- Absolute value of maximal output from fuzzy system $|F(ee, ce)_{max}|$=1.0
- The number of control cycles per second, CyclesPerSecond=14

Chapter 4 presented the explanations of above parameters in chatter suppression context. Experimental cuts have been conducted to determine their values. Each cut is carried out using the feed rate and spindle speed recommended by the experienced operators and a handbook (Meng 1996). Except for finding the parameters K_1 and K_2, the self-regulating algorithm is disabled to determine the above parameter values. K_1 and K_2 can be known according to the criterion whether or not they can minimize the MSE of cutting energy. An initial value of K_1 or K_2 is selected based on reference and the MSE of vibration energy will be calculated. Then the K_1 and K_2 values will be changed leading to different MSE values. Repeat this procedure until a combination of K_1 and K_2 is obtained which gives the lowest MSE. In addition, the reference threshold value 0.65 as specified in Chapter 3 for chatter detection index needs to be verified. Maximal adaptation gain G_p can be settled in terms of equation (4.23).

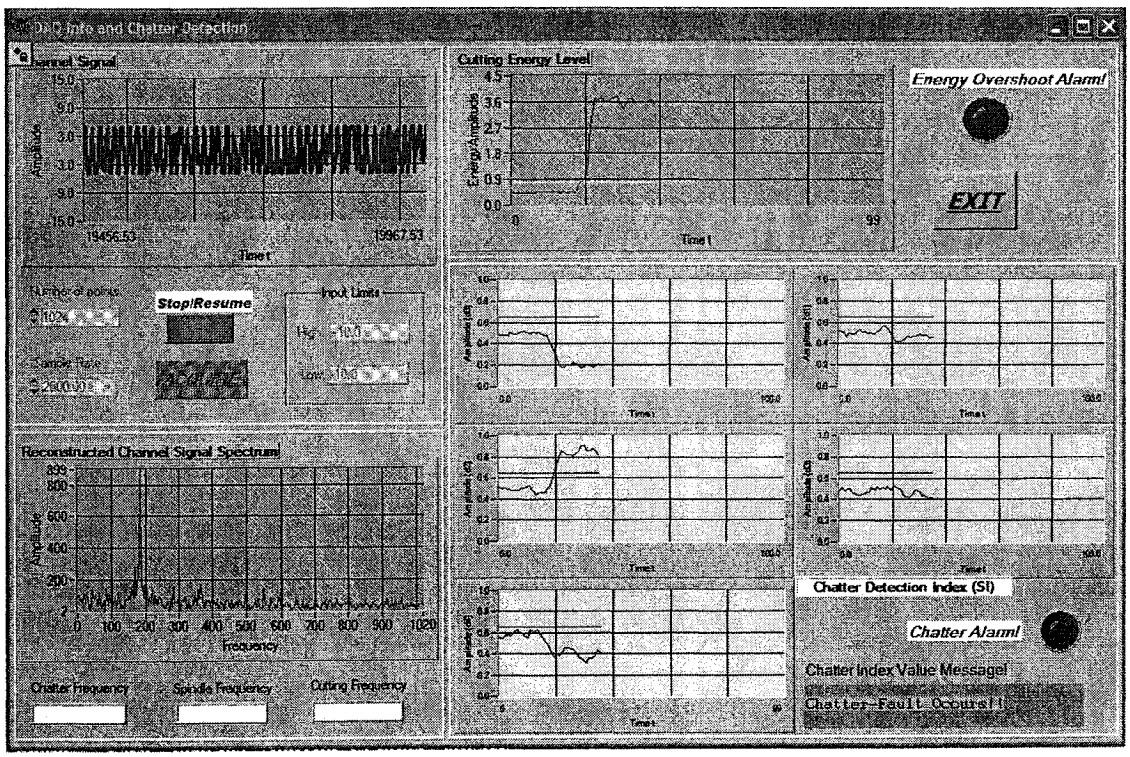


Figure 5.3 Designed LabWindowsTM panel

5.3 Experimental set-up and implementation

Figure 5.4 shows the experimental block diagram (Hermansyah, 2000) and the set-up conditions are listed as follows (The setting of accelerometer, end-milling cutter, workpiece and slotting is illustrated in Figure 5.5):

- CNC SERVO 2000 milling machine at the high gear mode
- Wilcoxon 993B-6 tri-axial accelerometer with X axis activated
- High Speed Steel (HSS) 3/8" (9.525mm) end-milling cutter with four helical flutes, 30 degree of helix angles and 38 mm overhang
- 1018 cold rolled steel workpiece (rigidly mounted on the table)
- Spindle speed range is between 500 and 1500 rpm
- Feed rate range is from 50 to 150 mm/min
- Feed per tooth is set to 0.025 mm/rev
- Slot cutting with coolant, the profile and geometric dimension of two steel workpieces are shown in Figure 5.6 and Figure 5.7, respectively.
- Vibration signals are acquired at the sampling rate of 2.0 kHz with a window size of N=1024 samples and chatter index SI is computed for every 512 samples.

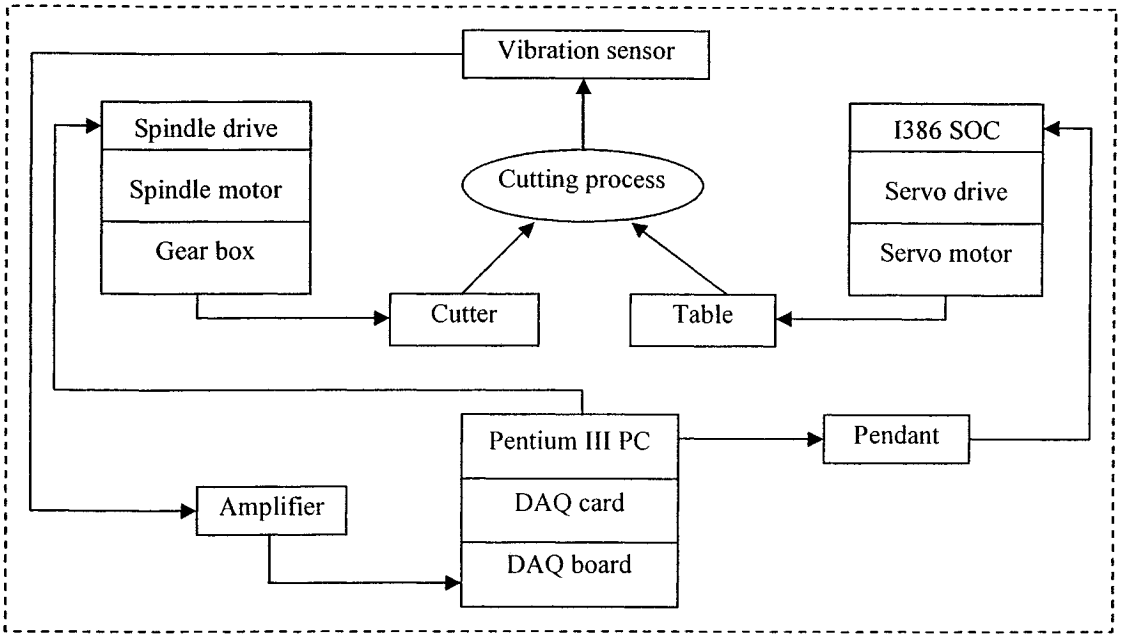


Figure 5.4 Systematic block diagram for experiments

For chatter detection, tests are performed at the constant spindle speed with corresponding feed rate as shown in Table 5.1 based on feed/tooth=0.025 mm/rev while increasing the depth of cut. For suppression, two kinds of adjustment approaches are used, i.e., adjusting spindle speed only and adjusting both spindle speed and feed rate. Since the feed adjustment alone cannot effectively suppress chatters (Liang *et al* 2004), this approach is not examined in our study. For each approach, two control strategies are implemented,

plain fuzzy control and fuzzy control with self-regulating. Table 5.1 summarizes all testing conditions along with the related observations such as maximal detection index *SI*. The chatter detection tests are mainly conducted using workpiece 2 (WP2) with several combinations of spindle speed, feed rate and depth of cut as shown in Table 5.1. To further validate the proposed detection method, an additional test is also carried out on workpiece 1 (WP1) with a larger depth of cut (4.572 mm), a spindle speed of 1000 rpm and a feed rate of 90 mm/min. The above cutting conditions are selected with reference to the recommended parameters for similar cutters (Meng 1996 and Zhao 1992): feed per tooth=0.025 mm/rev, depth of cut $\leq$ 5 mm, and surface speed (SF) $\leq$ 30 m/min.

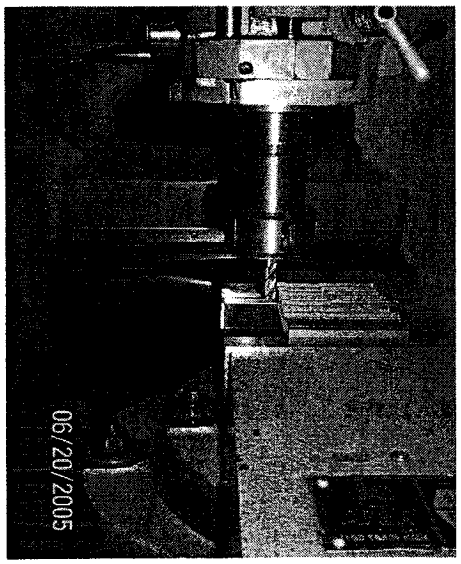


Figure 5.5 Setting of sensor, cutter, workpiece and slotting

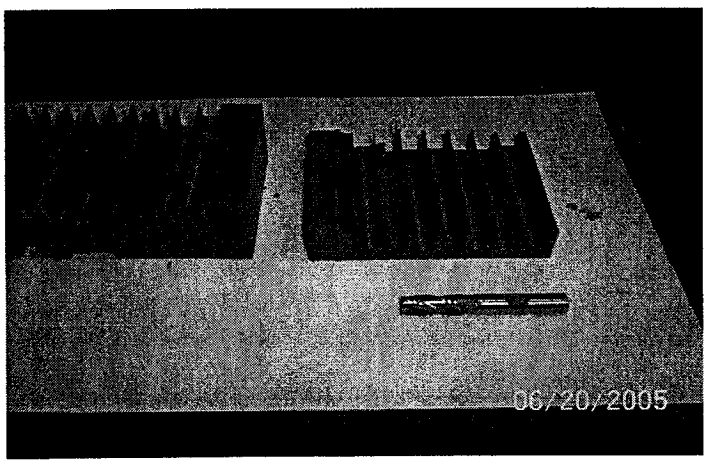


Figure 5.6 Profile of two workpieces

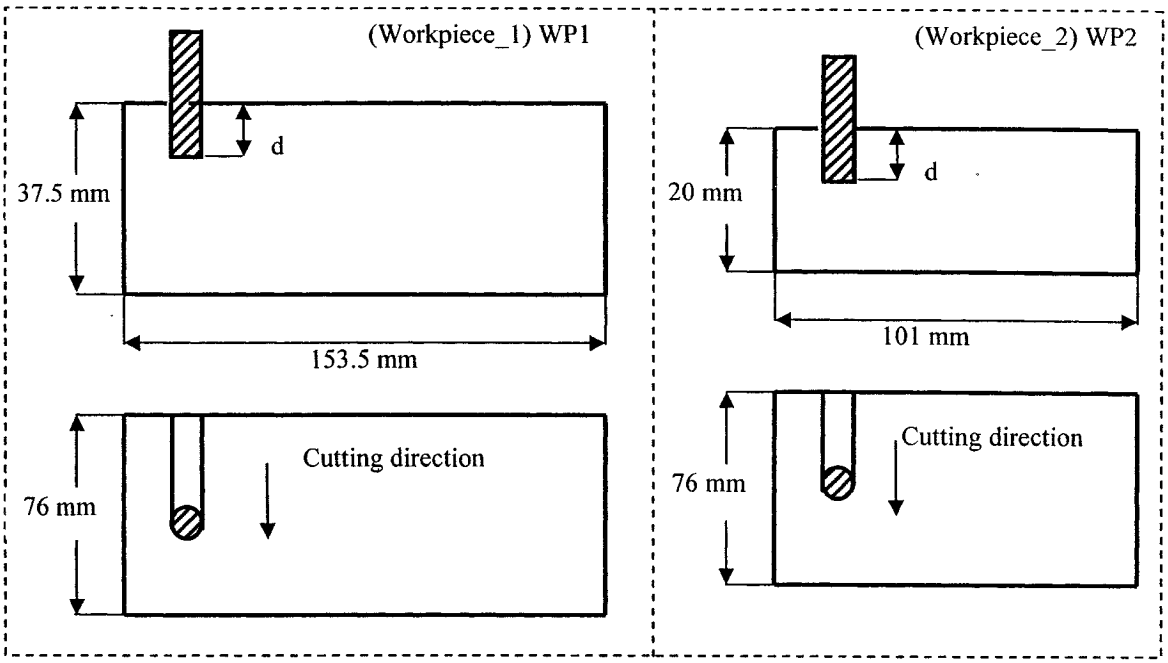


Figure 5.7 Dimensional measurement of two workpieces used for tests

5.4 Experimental results and discussions

The experiments were conducted to examine: (1) the performance of the proposed wavelet based chatter detection module, (2) the performance of the proposed fuzzy control based chatter suppression module, (3) the effects of adjusting single parameter (spindle speed) and multiple parameters (spindle speed and feed rate) on chatter suppression, (4) the behavior of plain fuzzy control and self-regulating fuzzy in chatter suppression, and (5) the overall system behavior under different cutting conditions in chatter detection and suppression. For the above purposes, a total of 21 tests were carried out on two different types of steel workpiece geometry. It should be also noted that, unlike most of the previous chatter suppression systems as reviewed in chapter 2, the proposed chatter suppression module attempts to adjust cutting parameters in both directions (increase or decrease spindle speed and/or feed rate) for chatter suppression with lowest possible productivity loss.

Based on cutting results, the reference threshold levels for chatter detection index and energy are selected as 0.65 (out of the range 0 ~ 1.0) and 0.8 (out of the range 0 ~ 4.5), respectively. In chatter suppression, the units for control commands of spindle speed and feed rate are both voltages. For example, a reading of 7.0V from the spindle motor is equivalent to 1000 rpm spindle speed, 3.71 V and 3.41 V from the feed motor correspond to

Table 5.1 Experimental test conditions of end milling process

Test No.	*ss* [3] (rpm)	*f* [3] (mm/min)	Work-piece	*doc* (mm)	Control method	Critical band[5]	Max *SI*[6]	PSD[6] (V^2)	Vibrat.[6] energy	Condition / control strategy
1	700[1]	70	WP2	2.540	N/A[4]	N/A	$< SI_{ref}$	-	$< E_{ref}$	Stable
2	700[1]	70	WP2	3.048	N/A	3	0.75-0.90	1300	2.5-3.0	Severe
3	900[1]	90	WP2	2.032	N/A	N/A	$< SI_{ref}$	-	$< E_{ref}$	Stable
4	900[1]	90	WP2	2.540	N/A	3	0.60-0.75	240	0.5-1.2	Moderate
5	900[1]	90	WP2	3.048	N/A	3	0.80-0.90	1350	2.7-3.6	Severe
6	1000 [1]	100	WP2	3.048	N/A	5	0.60-0.90	450	0.9-1.9	Moderate ~ Severe
7	1000 [1]	100	WP2	3.048	PF	5	-	-	-	*ss* & *f*
8	1000 [1]	100	WP2	3.048	FSR	5	-	-	-	*ss* & *f*
9	1000 [2]	100	WP2	3.048	N/A	2	0.60-0.70	155	0.5-0.7	Moderate
10	1000 [2]	100	WP2	3.048	PF	2	-	-	-	*ss* only
11	1000 [2]	100	WP2	3.048	FSR	2	-	-	-	*ss* only
12	1000[2]	90	WP1	4.572	N/A	2	0.60-0.75	120	0.5-0.6	Moderate
13	1000[2]	90	WP1	4.572	PF	2	-	-	-	*ss* & *f*
14	1000[2]	90	WP1	4.572	FSR	2	-	-	-	*ss* & *f*
15	1000[2]	90	WP1	4.572	PF	2	-	-	-	*ss* only
16	1000[2]	90	WP1	4.572	FSR	2	-	-	-	*ss* only
17	1000[1]	100	WP2	3.556	N/A	5	0.75-1.0	1680	2.5-4.0	Severe
18	1000[1]	100	WP2	3.556	PF	4,5	-	-	-	*ss* & *f*
19	1000[1]	100	WP2	3.556	FSR	3,4,5	-	-	-	*ss* & *f*
20	1000[1]	100	WP2	3.556	FSR	3,4,5	-	-	-	*ss* only
21	1000[1]	100	WP2	3.556	PF	4,5	-	-	-	*ss* only

Note:
[1] New cutter.
[2] Used cutter.
[3] For suppression tests, these are the initial values.
[4] N/A indicates detection only, PF means suppression using plain fuzzy control, FSR means suppression using fuzzy control with self-regulating.
[5] Critical band is the wavelet scale level with highest *SI* when chatter is observed.
[6] For chatter detection tests only.

feed rates of 100 mm/min and 90 mm/min, respectively. In this study, surface profiles, sound, and FFT plot of vibration sensor data are used to confirm the occurrence of chatters. To be consistent, the testing results plotted in Figure 5.8 to Figure 5.28 are in the same order as the test number listed in Table 5.1. It is noted that the critical band or scale (the band where chatter is observed) is different for some tests due to different cutting conditions or tools. For example, the critical band of tests #2, #4 and #5 is the same, i.e., band 3, because of similar cutting conditions. The critical band of tests #6 to #8 is band 5 due to similar cutting conditions and the same new cutter. Though the cutting conditions of tests #9 to #11 are identical to those of #6 to #8, the critical band now is band 2 caused by the use of an old cutter. The critical band of the next five tests, i.e., #12 to #16 is identical, namely band 2 because of the same cutting conditions and the same tool. For severe chatter cases conducted through tests #17 to #21, all the critical bands are listed due to the shift of chatter frequency. Chatter detection test #17 shows that the chatter band appears in band 5. For tests #18 and #21, the critical bands are bands 4 and 5 because of the implementation of plain fuzzy control. Bands 3, 4 and 5 become the critical bands in tests #19 and #20 since the fuzzy control with self-regulating is performed.

5.4.1 System behaviors in chatter detection

The results for chatter detection tests #1 to #6, #9, #12 and #17 are plotted in Figure 5.8 to Figure 5.13, Figure 5.16, Figure 5.19 and Figure 5.24. It can be seen that under the specified cutting conditions, the energy level and index *SI* level during the stable machining processes (test #1 and #3 as shown in Figure 5.8 and Figure 5.10) are wide spread in all frequency ranges or bands and there are no clear dominant frequencies. This is an indication of stable cutting process (Dong *et al* 1992). All calculated chatter detection indexes (SI) in five sub-bands are below the threshold value 0.65. Meanwhile, the energy level is around the reference level 0.8 which demonstrates the cut is maintained in a steady state.

On the other hand, moderate or severe chatter vibrations were observed in tests #2, #4, #5, #6, #9, #12 and #17 as shown in Figure 5.9, Figure 5.11, Figure 5.12, Figure 5.13, Figure 5.16, Figure 5.19 and Figure 5.24. Figure 5.9 for test #2 clearly shows that the chatter detection index, *SI*, of band 3 is in the range of 0.75 to 0.90 which is substantially higher than that in other bands and significantly above the threshold 0.65 of band 3 as well. Accordingly, the chatter state was captured by the high *SI* level in band 3. From energy stand

point, this indicates that band 3 contains the largest portion of signal energy, i.e., most of vibration energy is absorbed into band 3 and it causes other bands to have the fewer energy distribution which leads to the lower *SI* value in those bands. It can be noticed that the mean vibration energy plot in Figure 5.9h) illustrates the substantially higher energy level in band 3 than the reference level, also signifying severe chatter. The FFT plot further validates the above observations. As shown in Figure 5.9e), the chatter occurs at 200 Hz which belongs to band 3 or scale 3 in the wavelet domain. Meanwhile Figure 5.9b) shows a very high power spectral density (about 1300) reflecting a severe chatter condition.

Similar observations can be made for tests #4, #5, #6, #9, #12 and #17. However, the chatter severity may be different and can be evaluated by *SI* values. For example, the *SI* value for test #4 fluctuates between 0.60 and 0.75 (Figure 5.8), narrowly around the reference level. This signals a moderate chatter condition. The moderate power spectral density level (240), energy level (0.5-1.25), and the amplitude of the time domain signals as shown in Figure 5.11 can all confirm this conclusion. Following the same approach, the process conditions for tests #5, #6, #9, #12 and #17 can also be classified based on Figure 5.12, Figure 5.13, Figure 5.16, Figure 5.19 and Figure 5.24. The results are listed in Table 5.1. It is noticed that in some plots the energy level is above the reference level and the suppression action is not taken until the *SI* is above SI_{ref}. This is because the detection is solely based on the *SI* value. The suppression is triggered only when the *SI* value is greater than its reference level. However, once the suppression starts, the energy related information *ee* and *ce* will be used along with *SI* for fuzzy control decisions.

In summary, the proposed chatter detection method can effectively detect chatter and evaluate chatter severity. It should be pointed out that, though FFT is exploited in the above to verify the results, the FFT based approaches cannot be easily used in practice since the threshold values (the power spectral density values) are often different for different machine-tool-workpiece combinations and different machining parameters. This requires re-calibration of the threshold value when process changes. In addition, the FFT plot provides only an aggregated result of the whole process and does not give information as for when the chatter will occur. The proposed wavelet based method, on the other hand, can be easily used for any machining process without re-calibrating the threshold level due to the dimensionless nature of the *SI* indicator and can simultaneously provide both time and

frequency (scale) domain information. Therefore chatter onset and severity can be monitored and evaluated for any cutting processes.

5.4.2 Comparison between single and multiple parameter adjustments

Twelve cutting cases (tests #7 and #10, #13 and #15, #8 and #11, #14 and #16, #18 and #21, #19 and #20) are conducted to make comparisons and the results are summarized in Table 5.2. As noticed in the table, adjusting both spindle speed and feed rate is significantly more productive than adjusting spindle speed alone except for the severe tests #18 and #21, #19 and #20. For plain fuzzy control with a depth of cut of 3.048 mm, the cutting time of test #7 (adjusting both speed and feed) is 43 seconds as compared to 51 seconds for test #10 (adjusting speed only), representing 16% of reduction. Similar observation can be made when the depth of cut increases to 4.572 mm using another workpiece. This is demonstrated by test #13 (multi-parameter adjustment) with a cutting time of 50 seconds and test #15 (single parameter adjustment) lasted 56 seconds. Consistent high productivity of multi-parameter adjustment is also recorded even if the control method is switched to self-regulating fuzzy control. Table 5.2 shows that, for the same cutting conditions as used in plain fuzzy control tests, the machining time reduces from 51 seconds (test #11) to 43 seconds (test #8) with a depth of cut of 3.048 mm, and from 56 seconds (test #16) to 50 seconds (test #14) with an increased depth of cut (4.572 mm). The high productivity of multi-parameter adjustment is achieved by on-line increasing feed rate. This can be clearly seen in Figure 5.14, Figure 5.20, Figure 5.15, and Figure 5.21 (for tests #7, #13, #8, and #14 respectively) where feed rates increase as part of the chatter suppression action, leading to reduced machining time.

However, for the same cutting condition as test #17 where the severe chatter occurs, tests #18 to #21 show the different scenarios. For plain fuzzy with depth of cut of 3.556 mm, test #18 (multi-parameter adjustment) takes 90 seconds in cutting time which is substantially longer than 50 seconds for test #21 (single-parameter adjustment) since feed rate is dropped to seek the stable cutting. Similarly, when control strategy is changed to fuzzy control with self-regulating, the cutting time for test #19 becomes 70 seconds compared with 50 seconds for test #20. Figure 5.25 to Figure 5.28 illustrate the above observations. This may be caused by the fact that the chatter is so severe that the only way for control system to suppress it and stabilize the process is to slow down speed and/or feed.

Table 5.2 Comparison of control and adjustment methods

Machining parameters			Performance indicators	Control method			
				Plain fuzzy control		Self-regulating fuzzy control	
Spindle speed (rpm)[a]	Feed rate (mm/min)[b]	Depth of cut (mm)		Feed & speed	Speed only	Feed & speed	Speed only
Workpiece 2			Test #	#7	#10	#8	#11
1000	100	3.048	Cutting time (sec) [e]	43	51	43	51
			No. of adj pts where $SI > SI_{ref}$	2/2[c]	5[d]	2/2	5
			Smoothness [f] (feed & speed)	0.2869 & 0.4076	0.2620	0.2862 & 0.3826	0.1662
Workpiece 1			Test #	#13	#15	#14	#16
1000	90	4.572	Cutting time (sec)	50	56	50	56
			No. of adj pts where $SI > SI_{ref}$	1/1	4	1/1	3
			Smoothness (feed & speed)	0.3470 & 0.4570	0.6195	0.2850 & 0.3740	0.2657
Workpiece 2			Test #	#18	#21	#19	#20
1000	100	3.556	Cutting time (sec) [e]	90	50	70	50
			No. of adj. pts Where $SI > SI_{ref}$	6/6	7	38/38	27
			Smoothness (feed & speed)	0.5333 & 0.7502	0.6452	0.1095 & 0.1686	0.2027

Note:
[a] Initial spindle speed.
[b] Constant feed rate for spindle speed adjustment and initial feed rate for adjusting both speed and feed.
[c] "2/2" means that there are two adjustment points (for both speed and feed) where SI is above SI_{ref}.
[d] This number indicates the number of spindle speed adjustment points where the SI is above SI_{ref}.
[e] The time information is recorded during the cutting and can be verified from the energy plot.
[f] In terms of the root-mean-square of feed change (mm/min) or speed change (rpm) per adjustment as per Equation 5.1.

In addition to the improved productivity, the multi-parameter adjustment approach seems also to be more effective for chatter suppression in our non-severe tests. This can be revealed by the time duration when the SI value is above its reference level. Such time duration can be measured by the number of adjustments made to bring the SI to SI_{ref} or below because the time duration between two adjacent adjustments is fixed. As summarized

in Table 5.2, during the entire machining process, there are only two adjustment points where the *SI* is above its reference level for tests #7 (multi-parameter adjustment and plain fuzzy control) and #8 (multi-parameter adjustment and self-regulating fuzzy control) as compared to five points for single parameter adjustment tests #10 and #11 under identical cutting conditions. When the depth of cut is raised to 4.572 mm, the above figures drop to one point for multi-parameter adjustment tests (#13 and #14) in contrast with four and three for tests #15 and #16, respectively when the speed alone is adjusted.

Nevertheless, for severe chatter tests what is observed does not completely comply with the above statement. For instance, test #18 (multi-parameter approach and plain fuzzy control) shows six adjustments of speed and feed in comparison with seven speed only adjustments for test #21 (single-parameter approach and plain fuzzy control). This is in agreement with the conclusion that multi-parameter adjustment is more effective that single-parameter. On the contrary, comparing with the only 27 speed adjustments in test #20 for single-parameter approach and fuzzy control with self-regulating, the number of adjustments in test #19 becomes 38 for multi-parameter approach and the same control strategy. Such a behavior could be caused by the additional disturbance to the process dynamics and hence further oscillated stability lobes when both speed and feed are regulated against the severe chatter. This obviously requires more frequent machining parameter adjustments to "chase" the volatile stability lobes.

In summary, the above comparison suggests that multi-parameter adjustment is more productive in machining and more effective in chatter suppression for non-severe tests. As mentioned before, the reduced cutting time is contributed mainly by the increased feed rate. The enhanced effectiveness may be interpreted as additional and/or joint curbing effect to chatter. For the cutting processes with severe chatter, the productivity and suppression effectiveness are less certain due to the volatile nature of the processes.

5.4.3 Comparison between plain fuzzy and fuzzy with self-regulating controls

The comparison between plain fuzzy control and fuzzy control with self-regulating can be made using the same twelve testing cases (i.e., tests #7 and #8, #10 and #11, #13 and #14, #15 and #16, #18 and #19, #20 and #21) discussed in the previous sub-section. To examine productivity, test pairs of #7 and #8, #13 and #14, and #18 and #19 are compared since they

are all related to feed rate. For non-severe chatter tests, it can be observed that the cutting time for tests #7 (plain fuzzy) and #8 (fuzzy control with self-regulating) is both about 43 seconds. Similarly, plain fuzzy and fuzzy control with self-regulating for respective tests #13 and #14 show almost the same cutting time, approximate 51 seconds. However, the latter control strategy performs better than the former one for severe chatter tests. Test #19 (fuzzy control with self-regulating) takes only 70 seconds in contrast with 90 seconds in test #18 (plain fuzzy). Therefore, in terms of productivity, the fuzzy control with self-regulating turns out more advantageous than the plain fuzzy control.

Another comparison, i.e., time needed to reach steady state, can also be made between the above six pairs. Except for the severe chatter tests (#18 and #19, #20 and #21), each of other pairs conducts almost the same number of adjustments, i.e., two adjustments of speed and feed for tests #7 and #8, five speed only adjustments for tests #10 and #11, one adjustment (speed and feed) for tests #13 and #14, and four and three adjustments (speed alone) for respective tests #15 and #16. However, for the severe chatter test pairs, plain fuzzy control takes the lead since there are merely six adjustments of speed and feed for test #18 (plain fuzzy) but 38 adjustments for test #19 (fuzzy control with self-regulating). Likewise, seven speed adjustments are conducted for test #21 (plain fuzzy) in contrast with 27 adjustments for test #20 (fuzzy control with self-regulating). Therefore on the whole, plain fuzzy control performs well in terms of adjustment efficiency. However, this advantage of plain fuzzy has its deficiencies in two aspects. First, when multi-parameter approach is implemented, the drastic decrement of spindle speed also causes the decrease of feed rate which results in much lower productivity. For example, in this study, test #18 quickly drops the speed and feed to 600 rpm and 50 mm/min, respectively after a few number of adjustments. This results in a longer cutting time than test #19 as illustrated in Table 5.2. Second, when single-parameter adjustment is used, the abrupt increase or decrease of spindle speed may cause mis-match between the changed spindle speed and the fixed feed rate, which could accelerate tool wear and impact surface quality. Moreover, the large speed swing can cause heavy burden to the spindle system and it may lead to the premature machine failure (Liang *et al* 2004).

The last comparison for the two control approaches is the smoothness of control process. For this purpose, all suppression tests are investigated. In this study, we quantify the smoothness of control process according to one of the following criteria:

$$RMS_f = \sqrt{\frac{\sum_{i=1}^{n} \Delta f_i^2}{n}} \quad \text{and} \quad RMS_{ss} = \sqrt{\frac{\sum_{i=1}^{n} \Delta ss_i^2}{n}} \tag{5-1}$$

The results are shown in Table 5.2. It is clear to see that fuzzy control with self-regulating tends to yield smoother control processes since the magnitude of adjustment with respect to speed and/or feed is smaller than that in plain fuzzy control. Therefore it can be concluded that fuzzy with self-regulating control strategy helps avoid the large vibration caused by over adjustment of speed or feed.

5.4.4 System behavior in cutting workpieces under different conditions

In this study, tests #1 and #2, #3, #4 and #5, and #6 and #17 as shown in Figure 5.8 and Figure 5.9, Figure 5.10, Figure 5.11 and Figure 5.12, and Figure 5.13 and Figure 5.24 indicate that chatter is directly related to the depth of cut and workpiece geometry. For example, in Figure 5.12 (test #5), the chatter detection index in band 3 shows much higher values than that in Figure 5.11 (test #4) when both comparing to the index values of stable cut (test #3) plotted in Figure 5.10. Therefore tests #4 and #5 suggest the occurrence of moderate and severe chatters, respectively. In addition, the difference in the profile of workpiece demonstrates that it is easier to trigger chatter for WP2. This may be due to its small dimensions in length and thickness, which could cause WP2 less rigidly clamped by fixtures. The comparisons among tests #6, #9, #12 and #17 further confirm this observation. For test #12 (WP1), the cut has to be conducted much deeper (4.572mm) to produce moderate chatter. However, the depth of cut 3.048mm for tests #6 and #9 (both WP2), and 3.556mm for test #17 (WP2) with the same spindle speed and even larger feed rate can produce moderate or severe chatters. Furthermore, the above tests indicate that, even for identical depth of cut and workpiece, the cutting processes could be either stable or unstable depending on spindle speed. Test #1 tells us that for WP2 there is no chatter when speed is set to 700rpm and depth of cut is 2.540mm. However, for the same workpiece and depth of cut, test #4 immediately shows the occurrence of chatter at the spindle speed 900rpm. On the

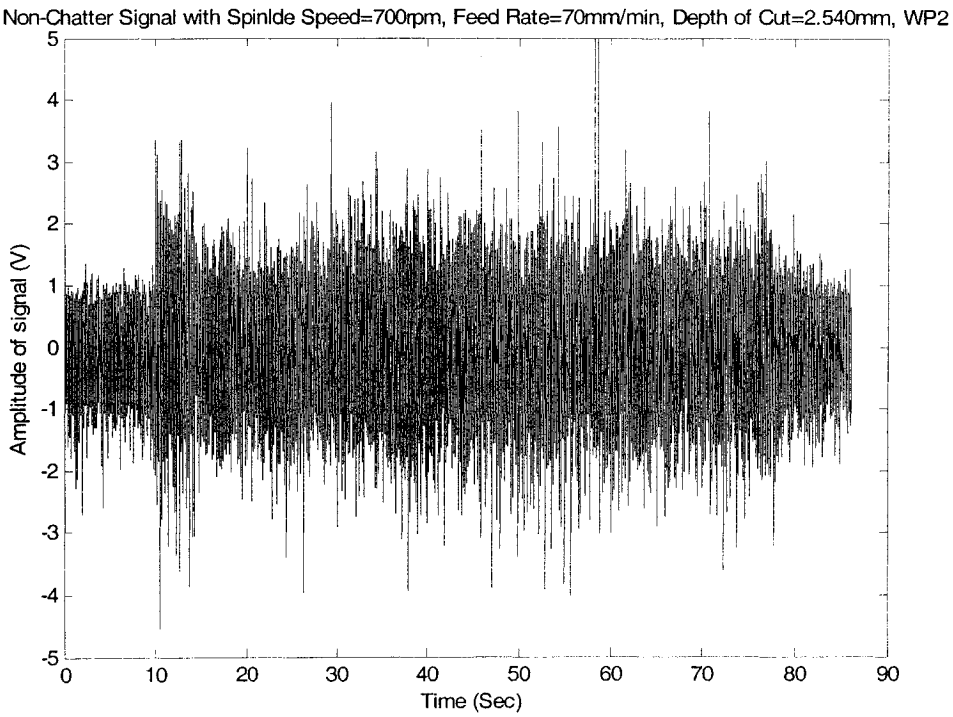
other hand, both 900rpm and 1000rpm speed generate chatters in tests #5 and #6 for the depth of cut 3.048mm and WP2.

Another observation for chatter detection is that chatters occur within different frequency bands for different cuttings. For instance, for tests #2, #4 and #5 the chatter frequency is located in band 3 (around 200Hz). Tests #6 and #17 show that chatter occurs in band 5 and the chatter frequencies are approximately 25Hz and 40Hz. Moreover, chatters are detected in band 2 for tests #9 and #12 with the chatter frequency about 400Hz. The explanation has been given earlier in this chapter.

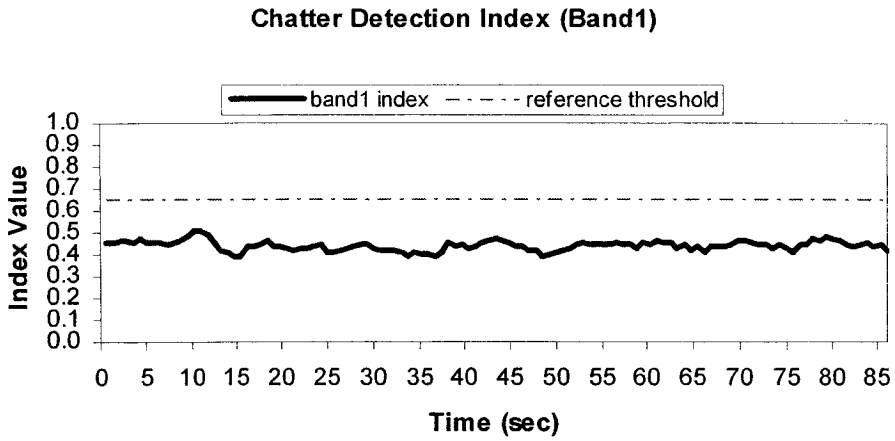
For chatter suppression, the control system demonstrates a quite consistent performance in most non-severe chatter tests. The cutting energy level is suppressed around reference level when speed and/or feed is adjusted and plain fuzzy control or fuzzy control with self-regulating is implemented. The detection index, *SI* value, is then maintained under the threshold throughout the process. However, for severe chatter tests, the performance of the controller is quite different. It can be noticed that the *SI* value fluctuates. In particular, for tests #19 and #20, when the fuzzy control with self-regulating is conducted, the *SI* value fluctuates significantly and it occasionally shoots up to much higher magnitude than its threshold level within certain bands. This phenomenon can be explained as follows. When speed or feed is regulated continuously, chatter develops at a certain frequency that depends on the underlying spindle speed. On the other hand, it takes some time for chatter to evolve. Therefore the fluctuation comes from the intermittent adjustment of control variables and the development of chatter. It is also noticed that the *SI* fluctuates in several bands and shows higher magnitude which is above its reference threshold within those bands. An explanation for this behavior could be that chatter frequency is related to both spindle speed and machine-tool-workpiece structure characteristic frequencies. Though as we know chatter frequency occurs near a dominant structural frequency when it is fully developed, while at a different stage when chatter has not yet completely evolved, the variation of spindle speed and excitement of other structural system's frequencies may alter the chatter frequency band. It should be pointed out that the vibration observed in tests #18, #19, #20 and #21 may no longer be the regenerative chatter, i.e., the self-excited vibration due to the dynamics between the cutter and workpiece. Instead, it could be largely contributed by the lack of the rigidity of the workpiece (WP2), which in this case is very thin after the first few tests and

sufficient clamping torque cannot be applied since otherwise the workpiece will be severely distorted. For our tests, in addition to the regulation of spindle speed caused by control system and attenuation of chatter, the additional instability because of the lack of rigidity in set-up could also result in chatter frequency swing between different bands. Back to the suppression effects, it can be noted that for all chatter suppression tests conducted in this study, our proposed control strategies are able to not only decrease the severity of chatter considerably (This can be observed by comparing the *SI* patterns before and after the control is invoked), but also eventually suppress chatter (This can be confirmed by both the *SI* values and vibration energy levels after the control brings the process to the steady state).

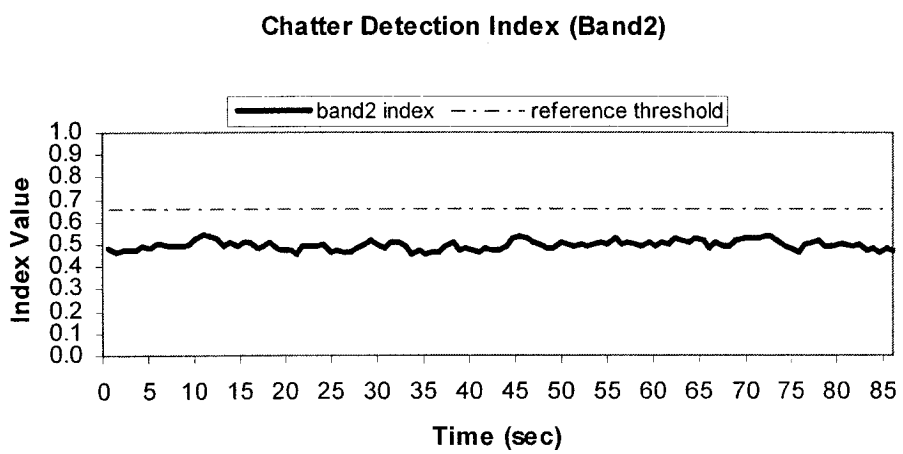
In summary, the implemented fuzzy control system can effectively mitigate the difficulty in selecting certain parameters for spindle speed modulation which is one of approaches to suppress chatter as proposed in some literatures. In addition, it is known that the fuzzy control approach can also tolerate imprecise data and incomplete knowledge of systems (Liang *et al* 2004). Therefore the new chatter suppression module proposed in this study works reasonably well for most normal chatters. However, its performance was adversely affected when non-regenerative vibration, e.g., additional instability is introduced due to the lack of clamping rigidity.



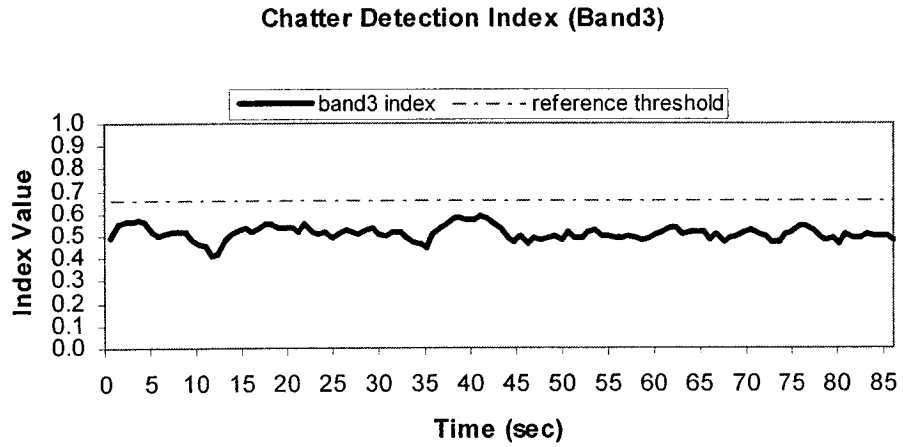
a) Time domain plot



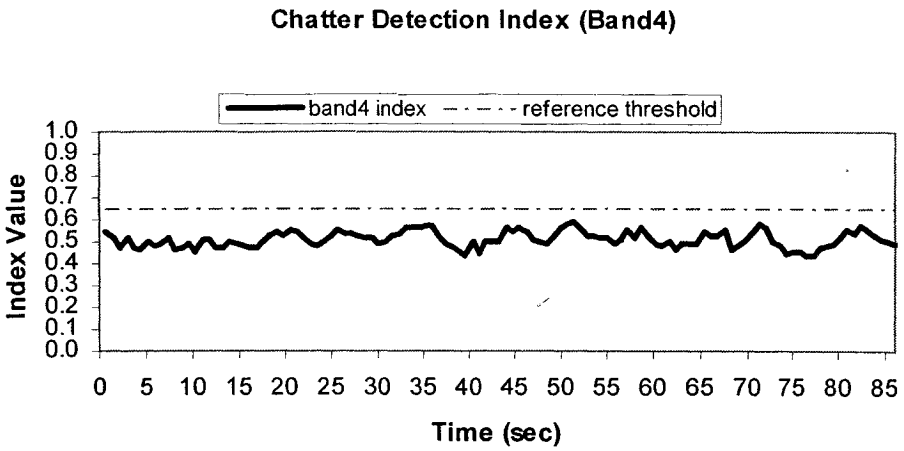
b) Wavelet domain plot: scale 1



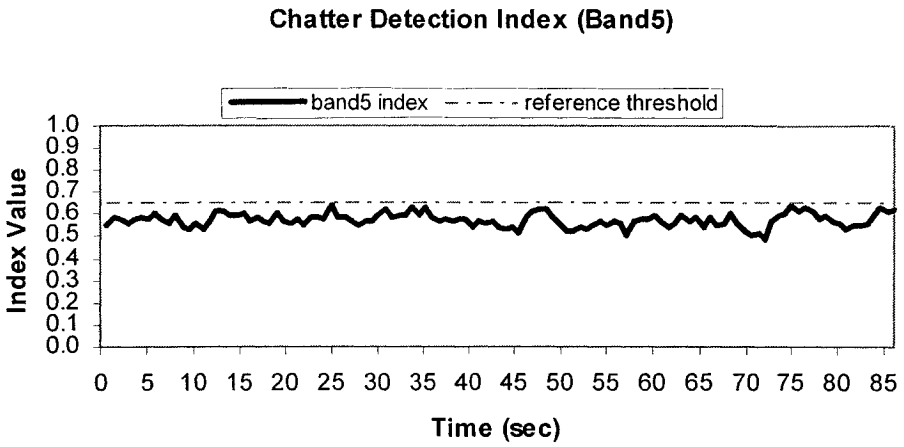
c) Wavelet domain plot: scale 2



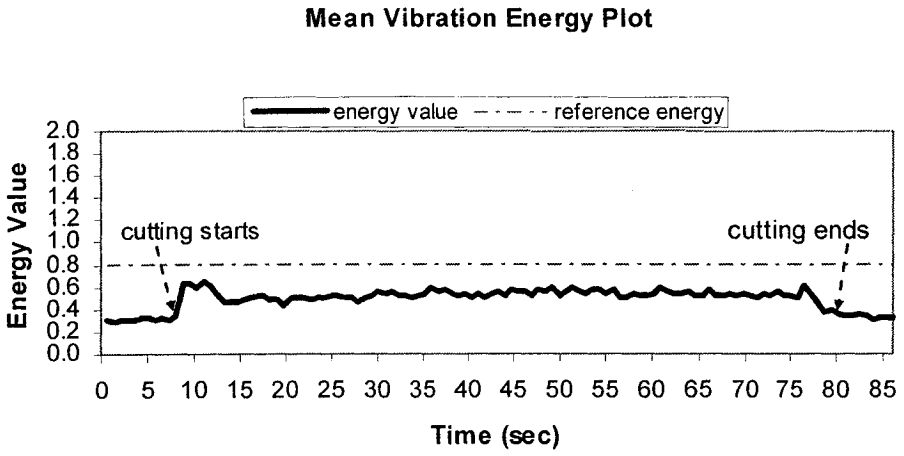
d) Wavelet domain plot: scale 3



e) Wavelet domain plot: scale 4

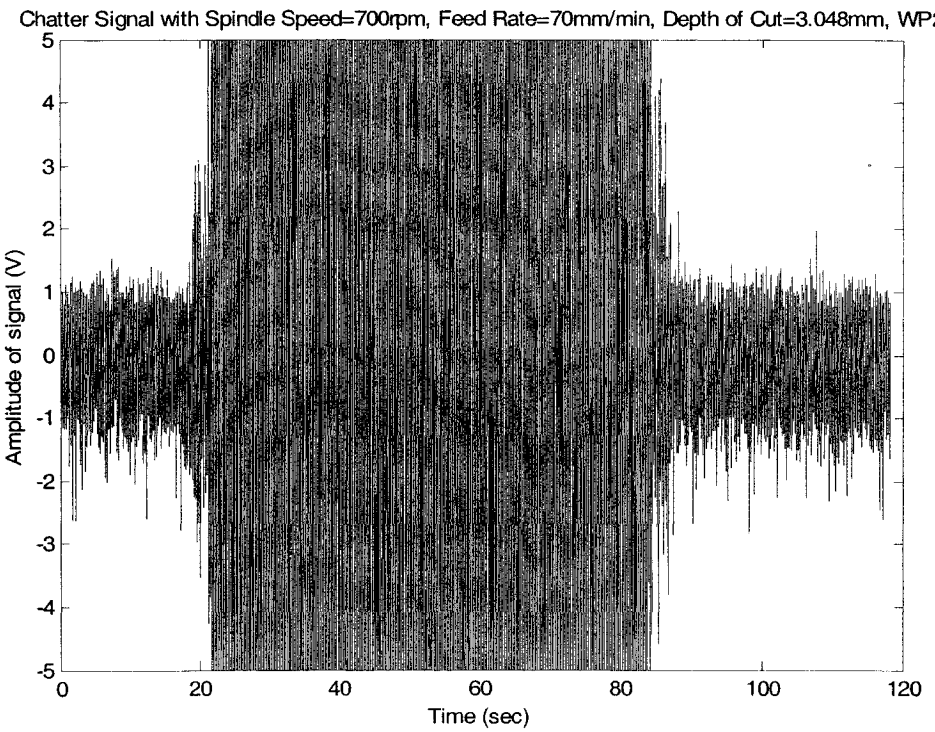


f) Wavelet domain plot: scale 5

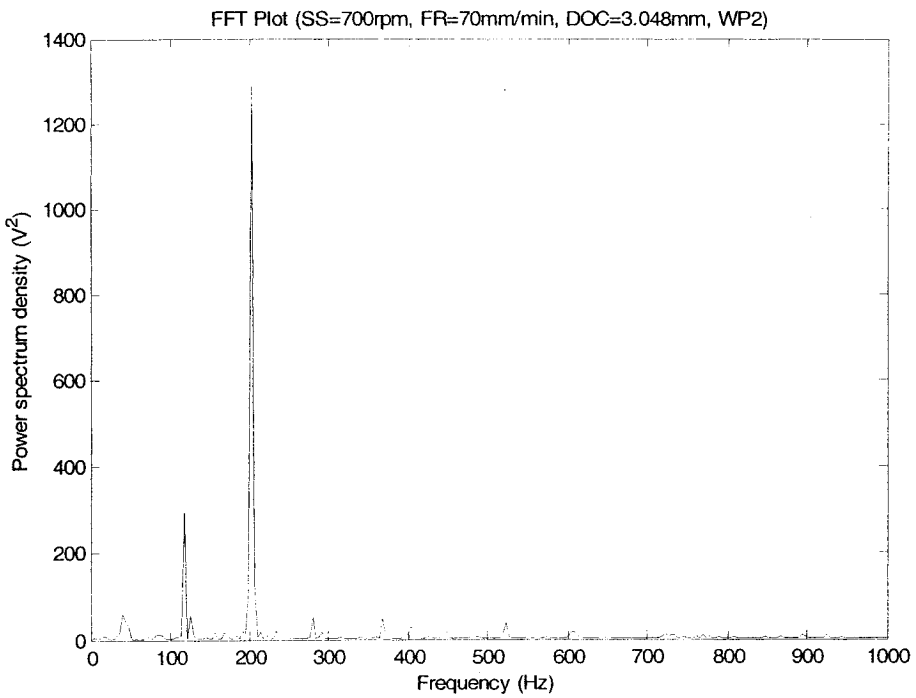


g) Mean vibration energy plot

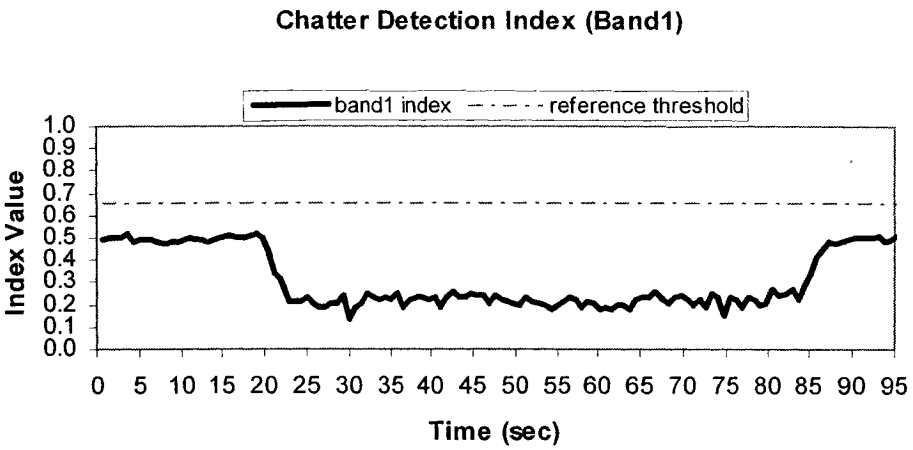
Figure 5.8 Chatter detection result of test #1 (Spindle speed 700rpm, feed rate 70mm/min, depth of cut 2.54mm, WP2)



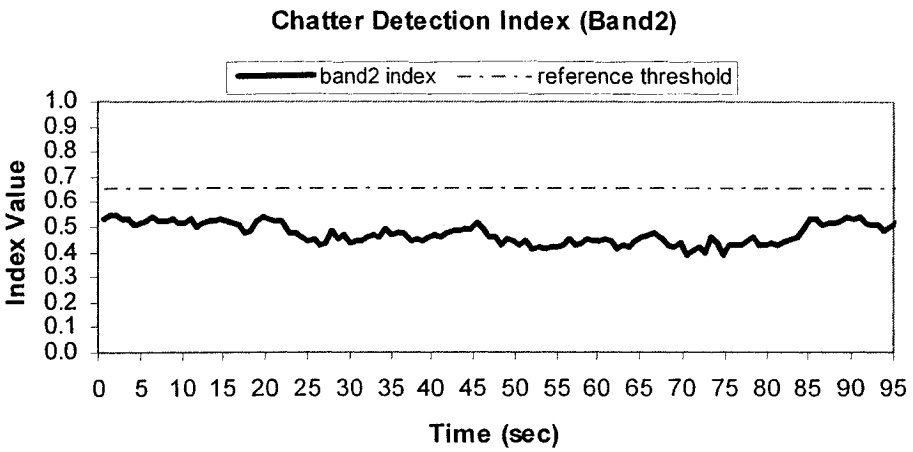
a) Time domain plot



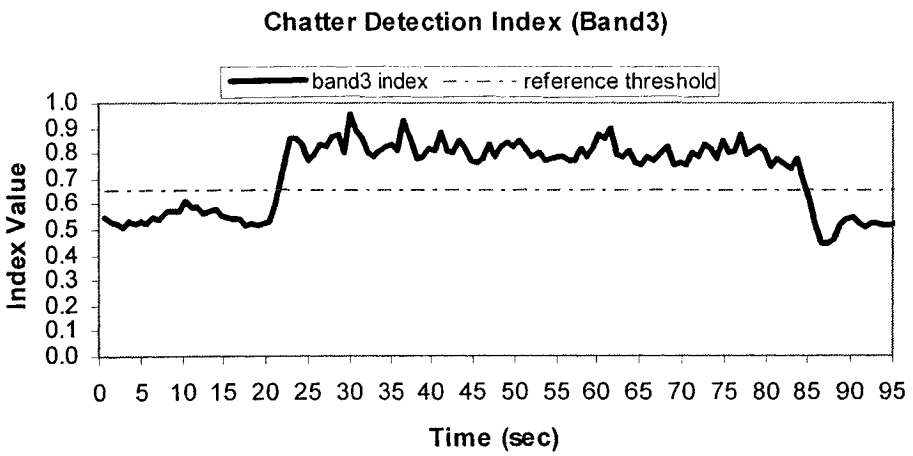
b) FFT plot



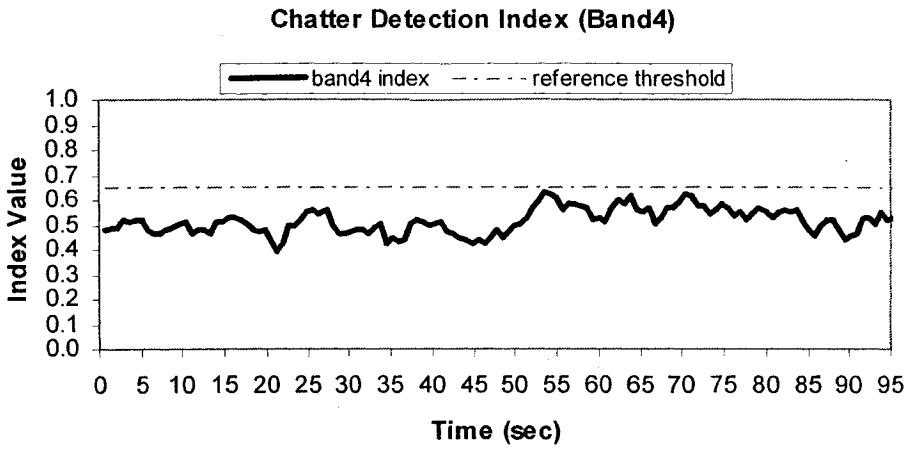
c) Wavelet domain plot: scale 1



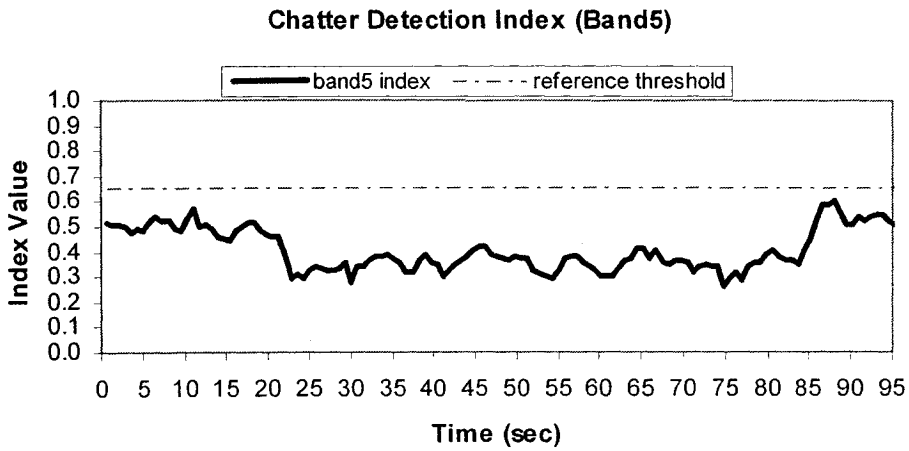
d) Wavelet domain plot: scale 2



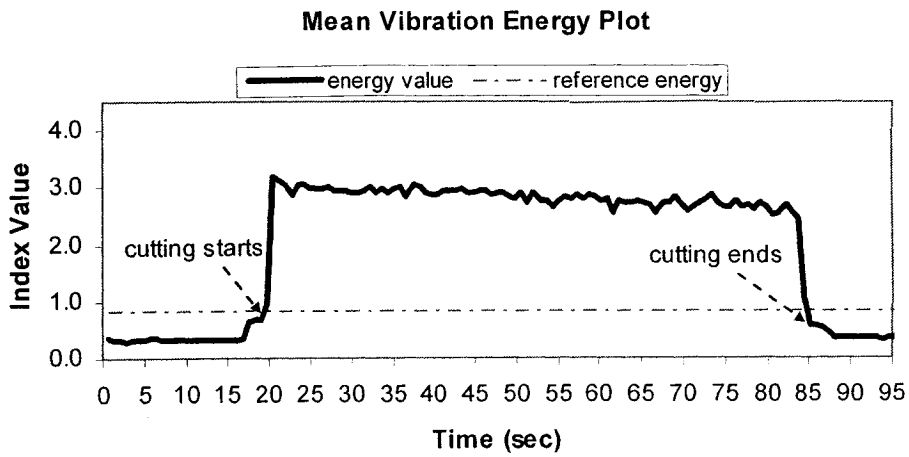
e) Wavelet domain plot: scale 3



f) Wavelet domain plot: scale 4

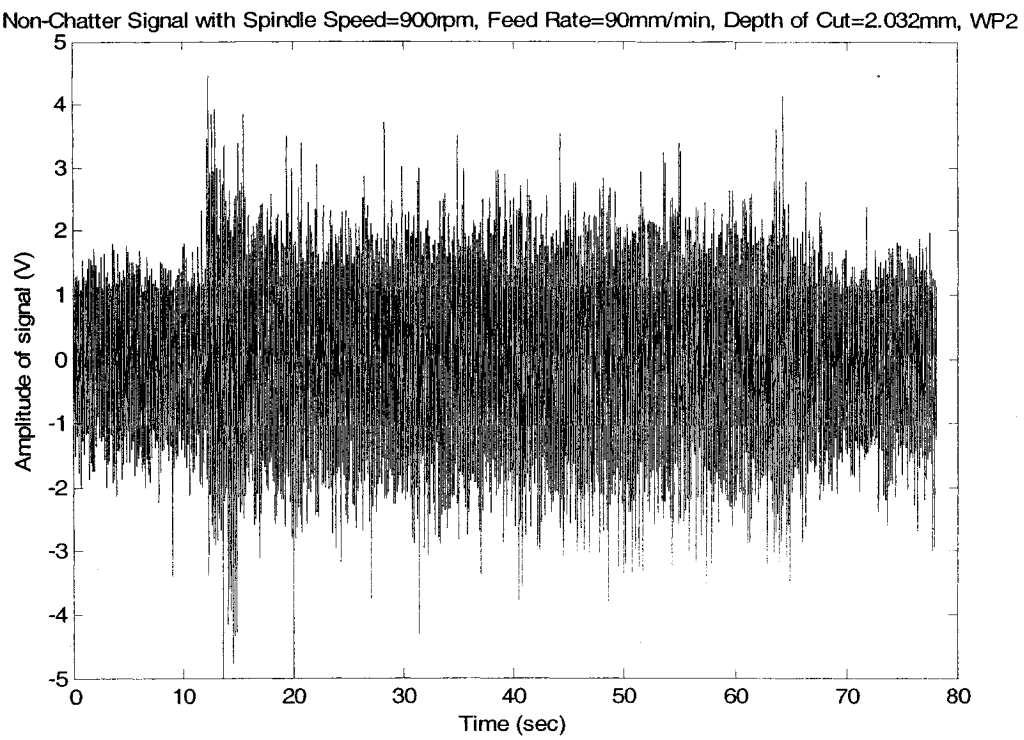


g) Wavelet domain plot: scale 5

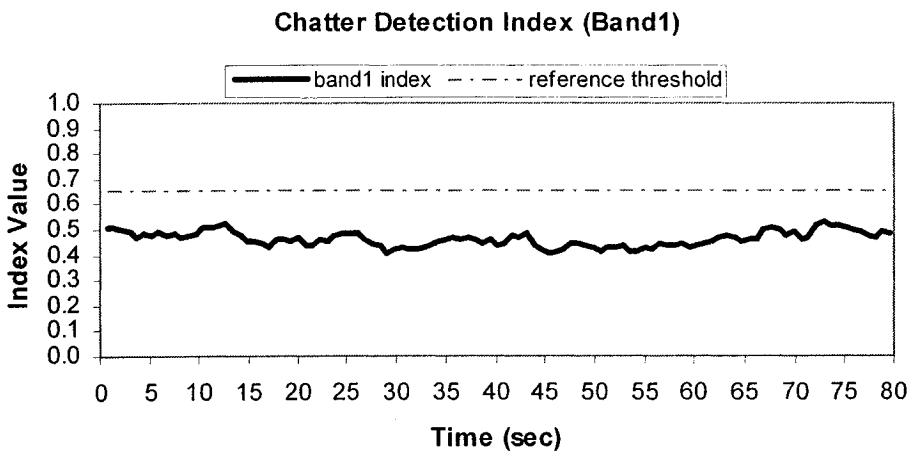


h) Mean vibration energy plot

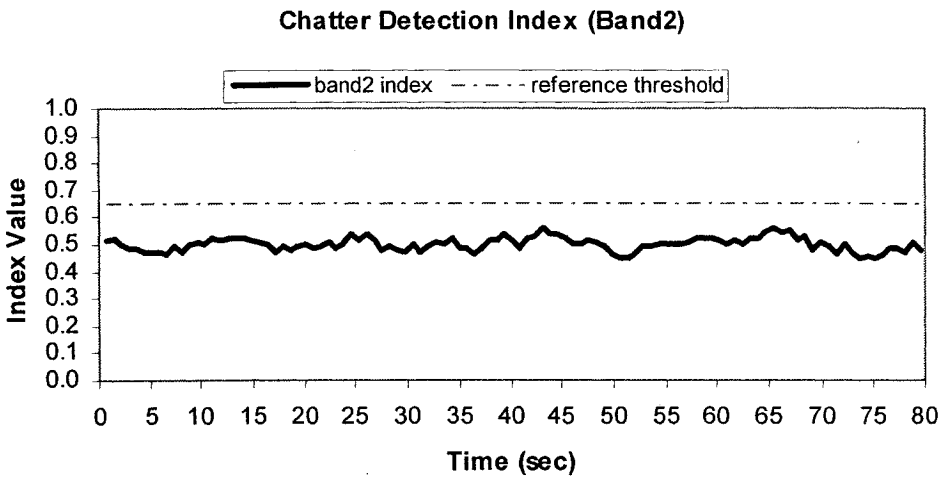
Figure 5.9 Chatter detection result of test #2 (Spindle speed 700rpm, feed rate 70mm/min, depth of cut 3.048mm, WP2)



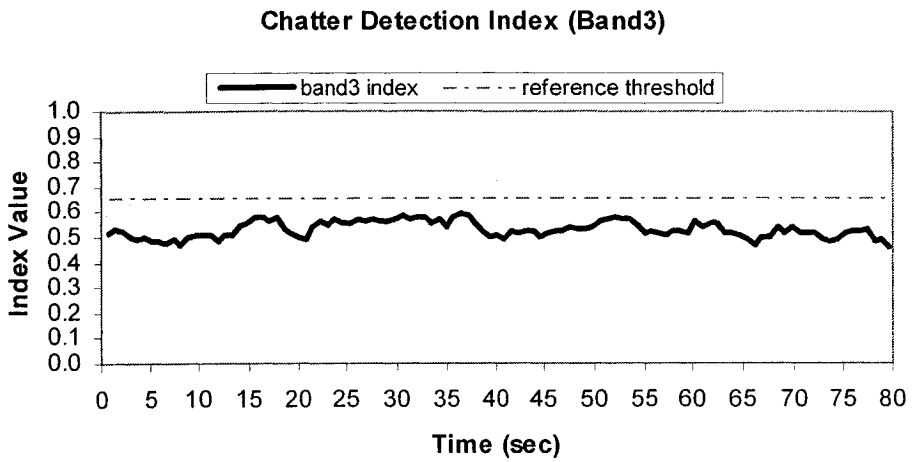
a) Time domain plot



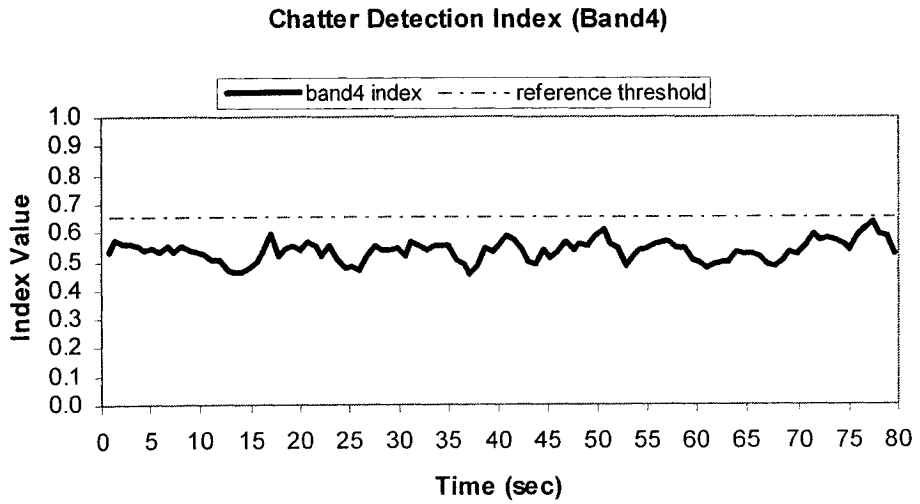
b) Wavelet domain plot: scale 1



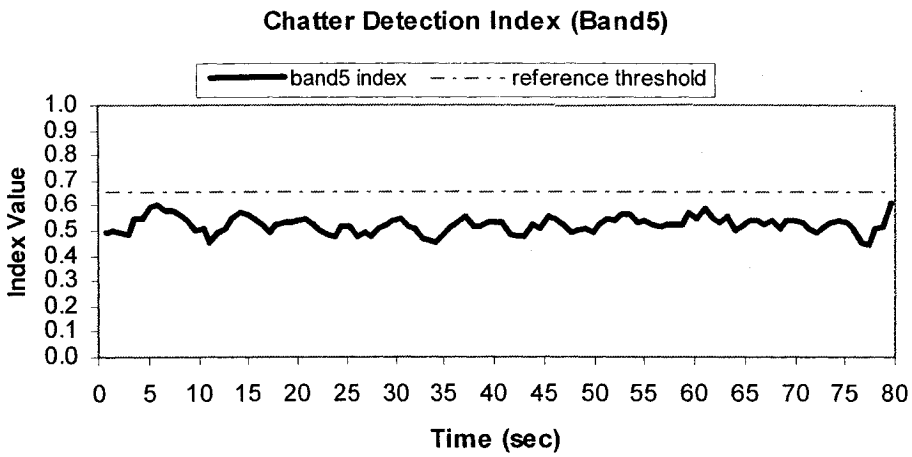
c) Wavelet domain plot: scale 2



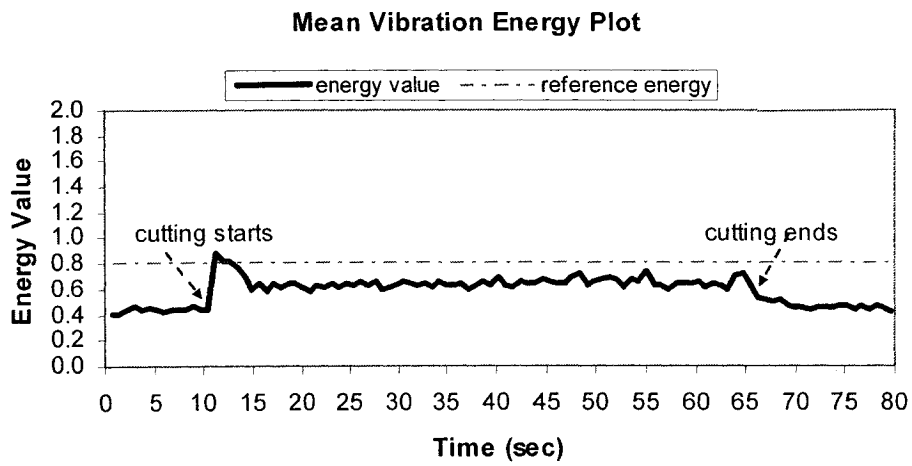
d) Wavelet domain plot: scale 3



e) Wavelet domain plot: scale 4

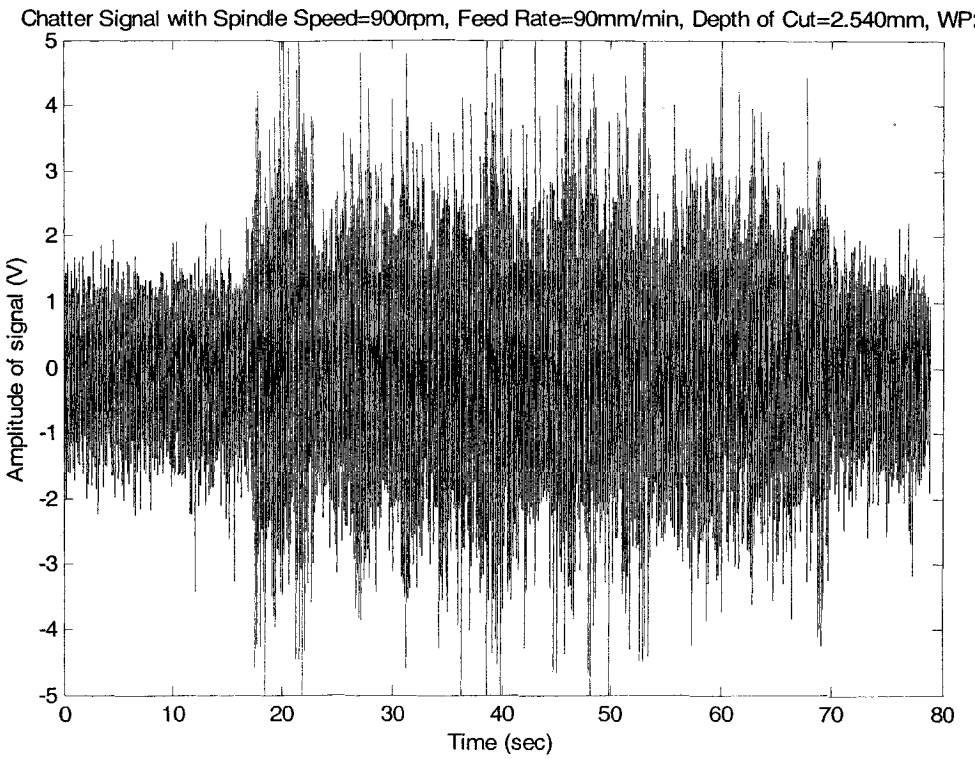


f) Wavelet domain plot: scale 5

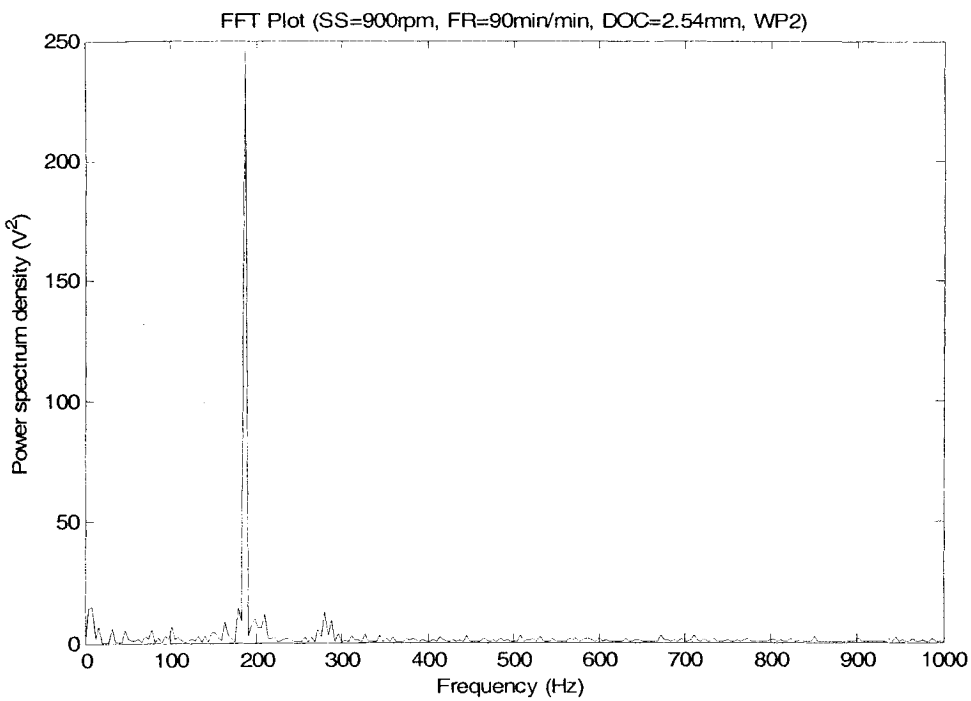


g) Mean vibration energy plot

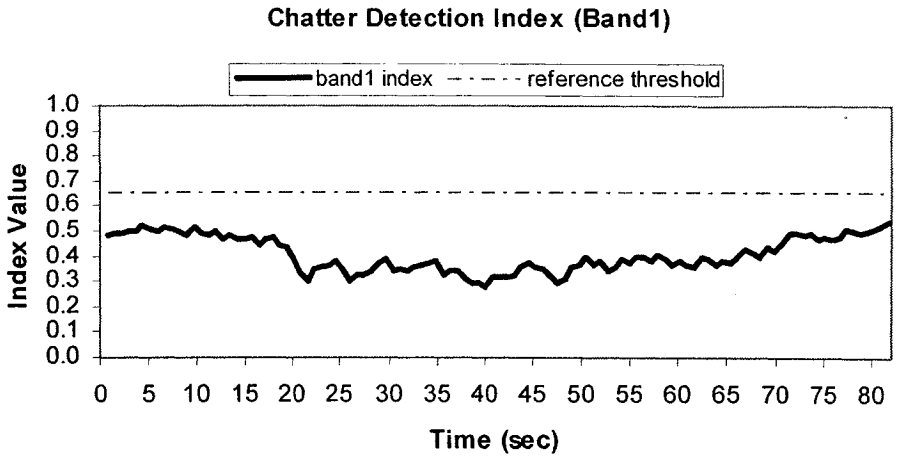
Figure 5.10 Chatter detection result of test #3 (Spindle speed 900rpm, feed rate 90mm/min, depth of cut 2.032mm, WP2)



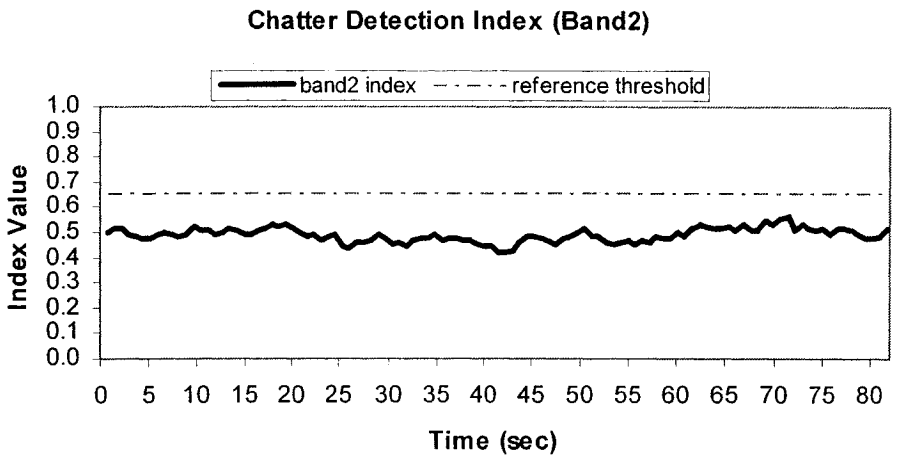
a) Time domain plot



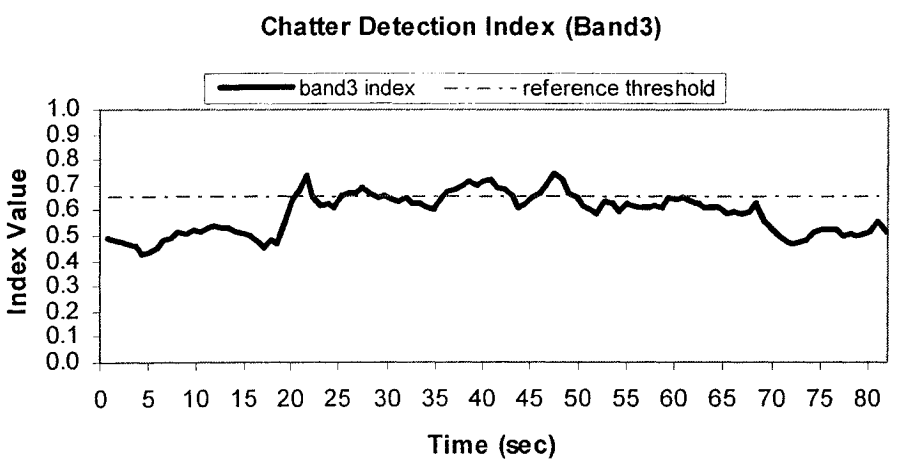
b) FFT plot



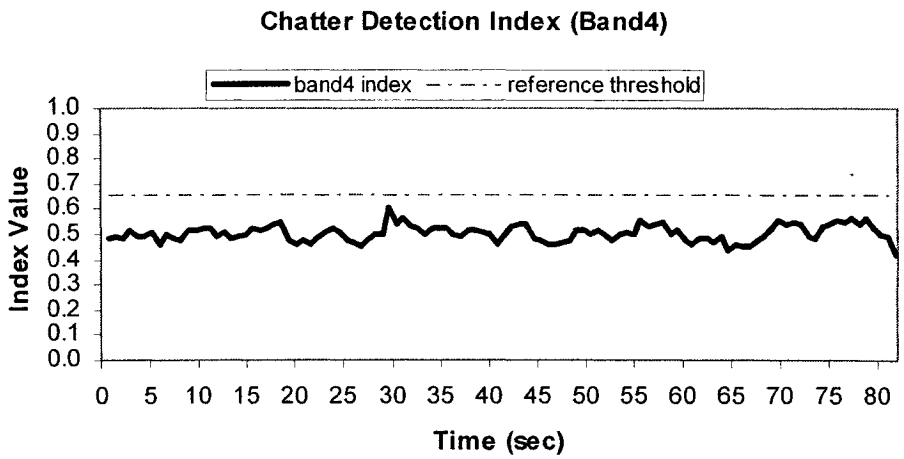
c) Wavelet domain plot: scale 1



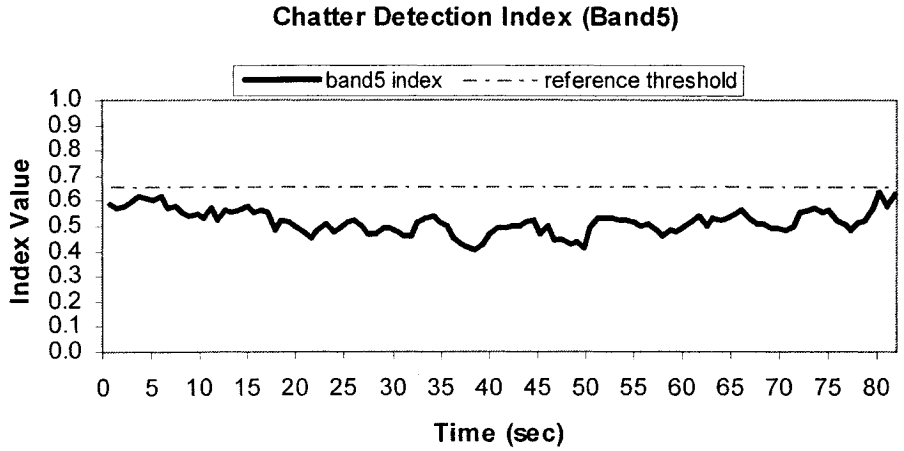
d) Wavelet domain plot: scale 2



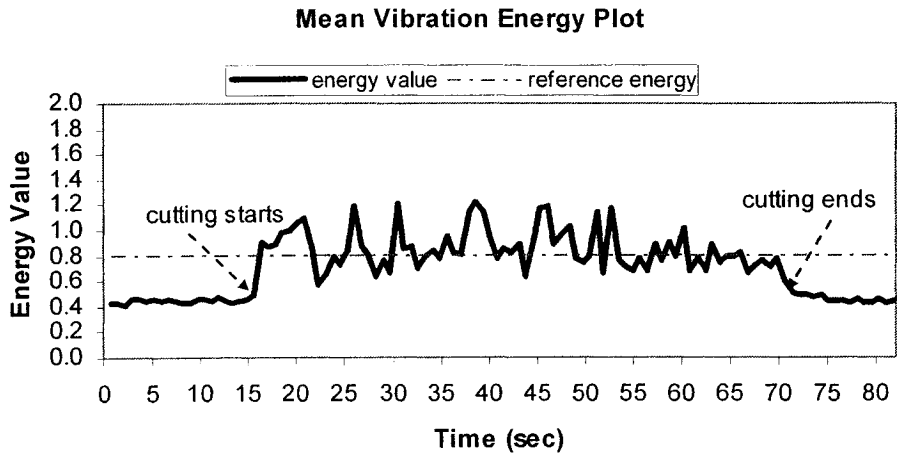
e) Wavelet domain plot: scale 3



f) Wavelet domain plot: scale 4

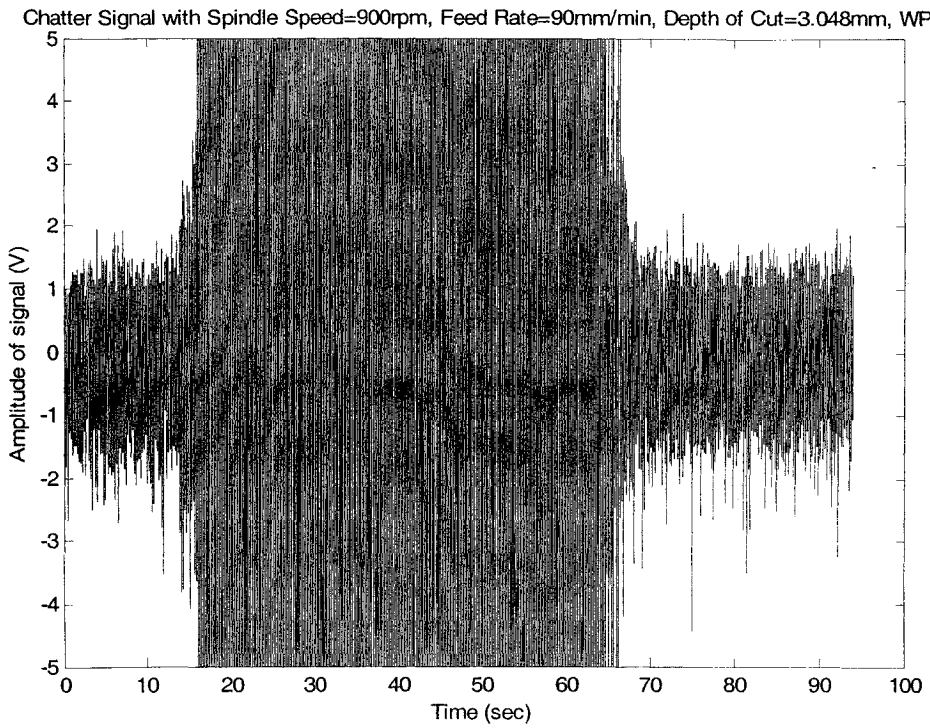


g) Wavelet domain plot: scale 5

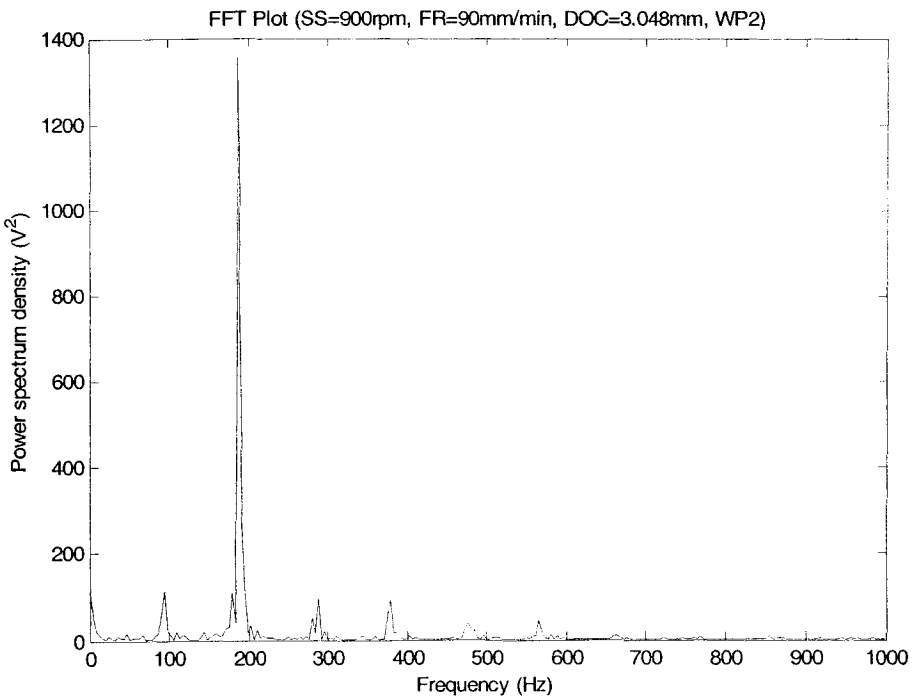


h) Mean vibration energy plot

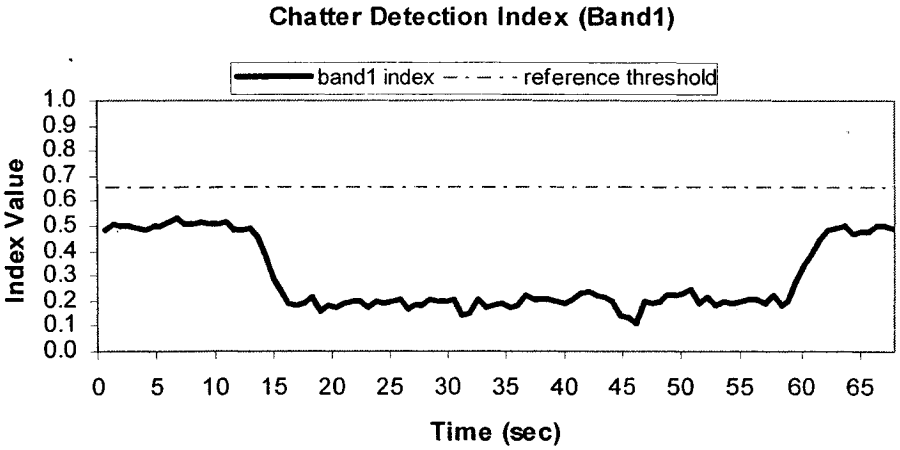
Figure 5.11 Chatter detection result of test #4 (Spindle speed 900rpm, feed rate 90mm/min, depth of cut 2.54mm, WP2)



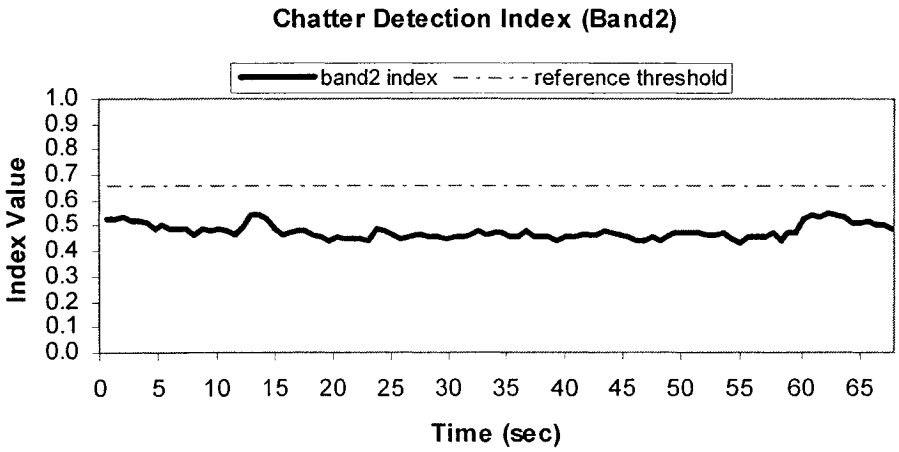
a) Time domain plot



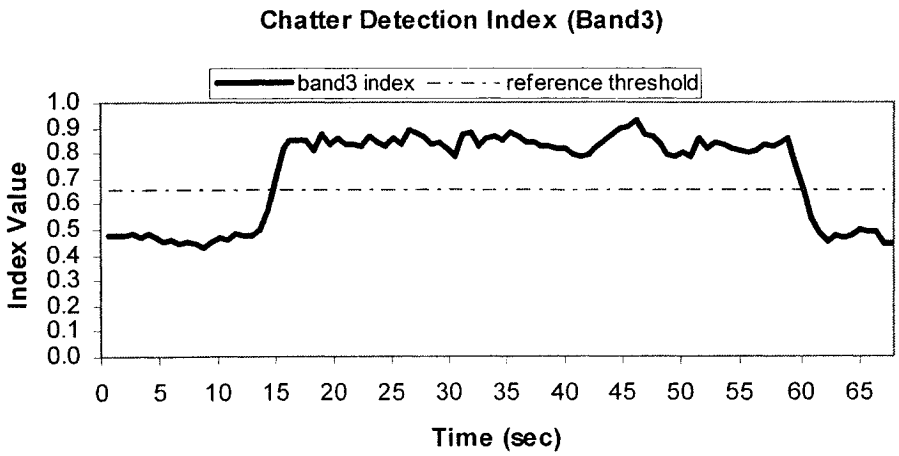
b) FFT plot



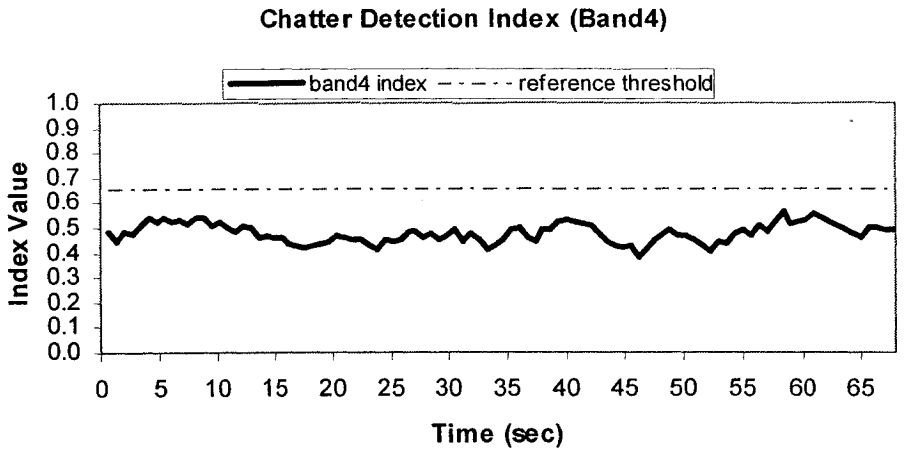
c) Wavelet domain plot: scale 1



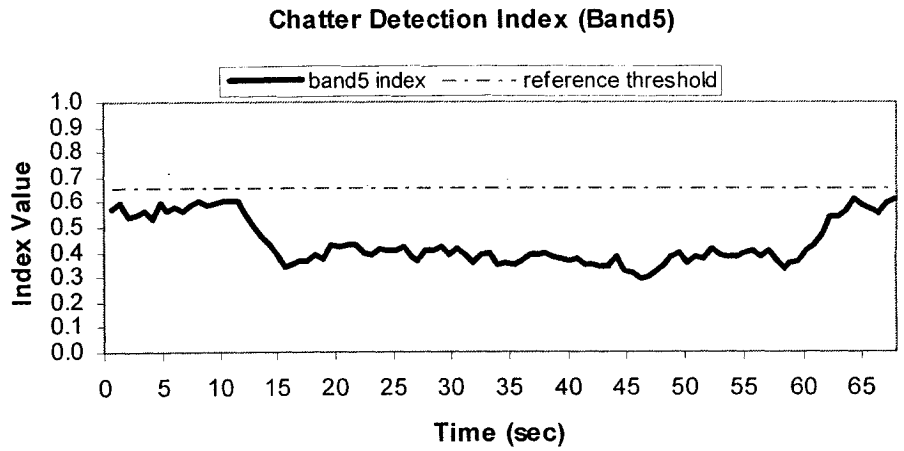
d) Wavelet domain plot: scale 2



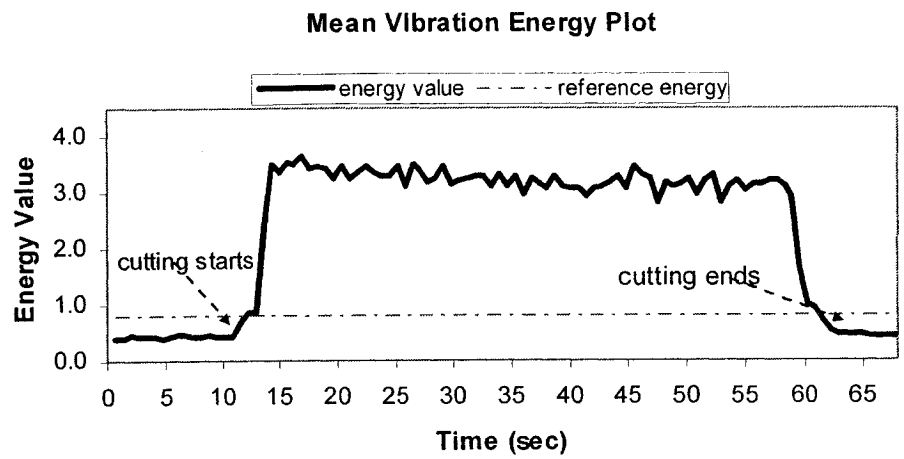
e) Wavelet domain plot: scale 3



f) Wavelet domain plot: scale 4

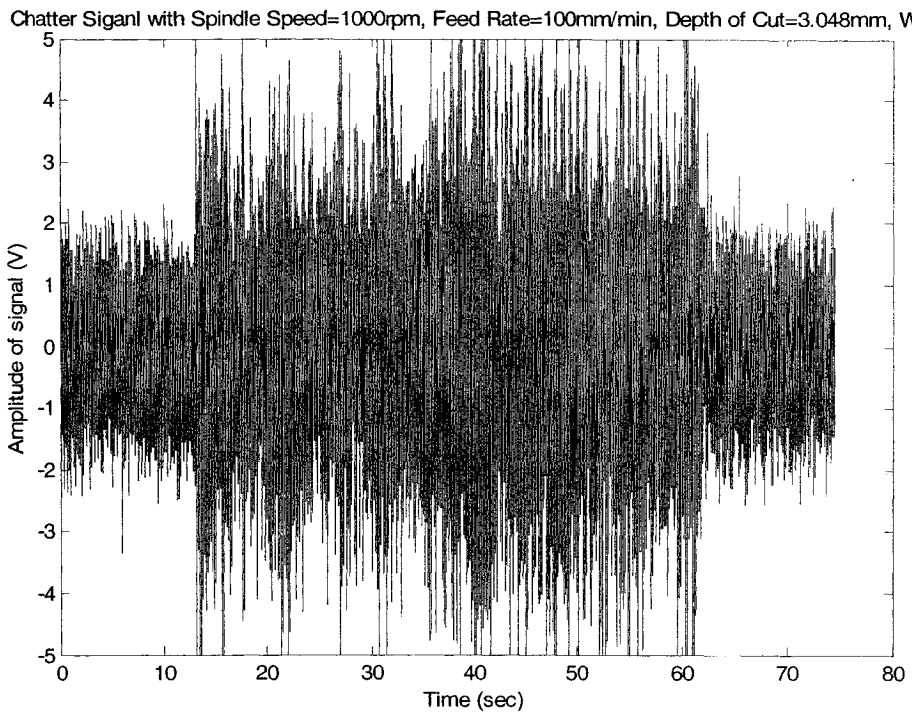


g) Wavelet domain plot: scale 5

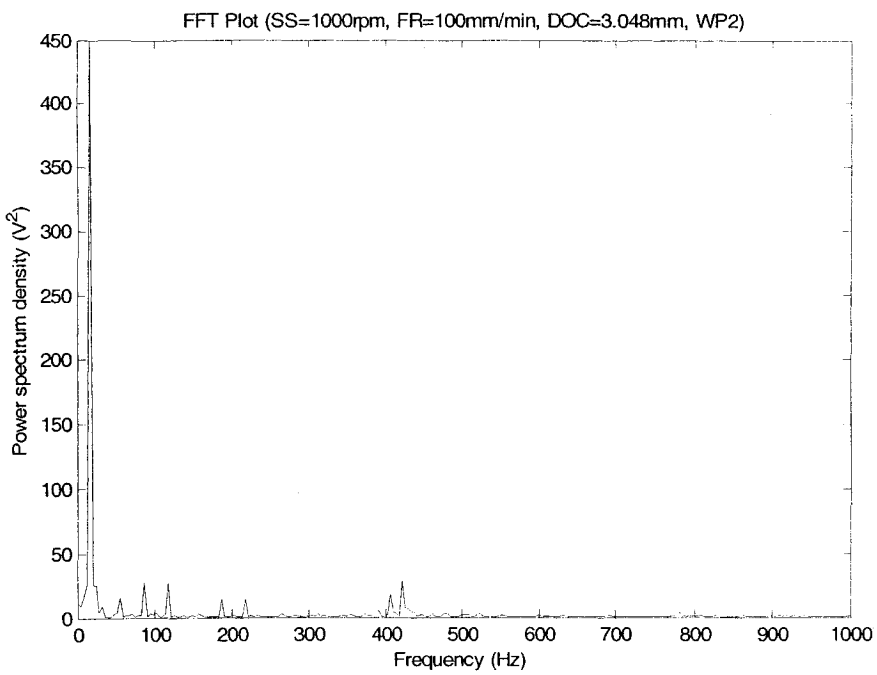


h) Mean vibration energy plot

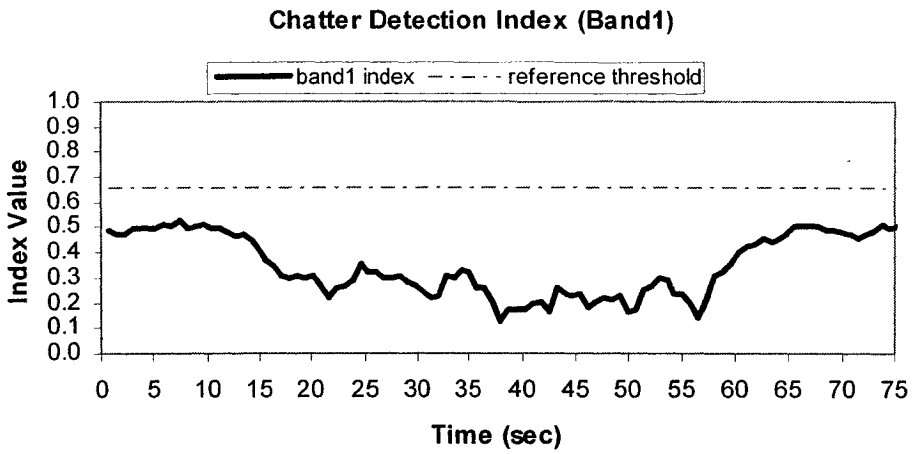
Figure 5.12 Chatter detection result of test #5 (Spindle speed 900rpm, feed rate 90mm/min, depth of cut 3.048mm, WP2)



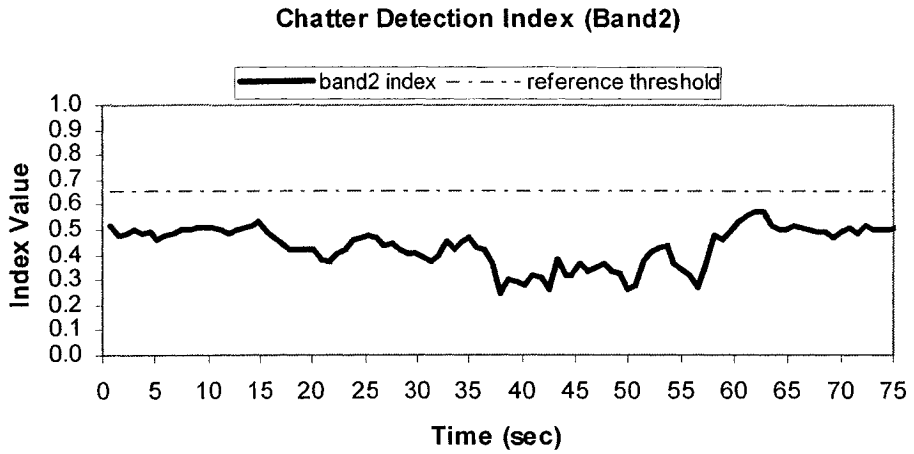
a) Time domain plot



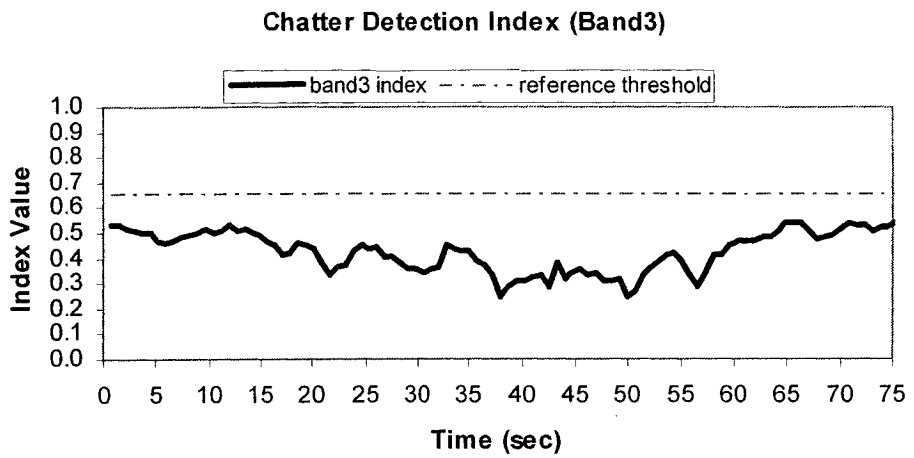
b) FFT plot



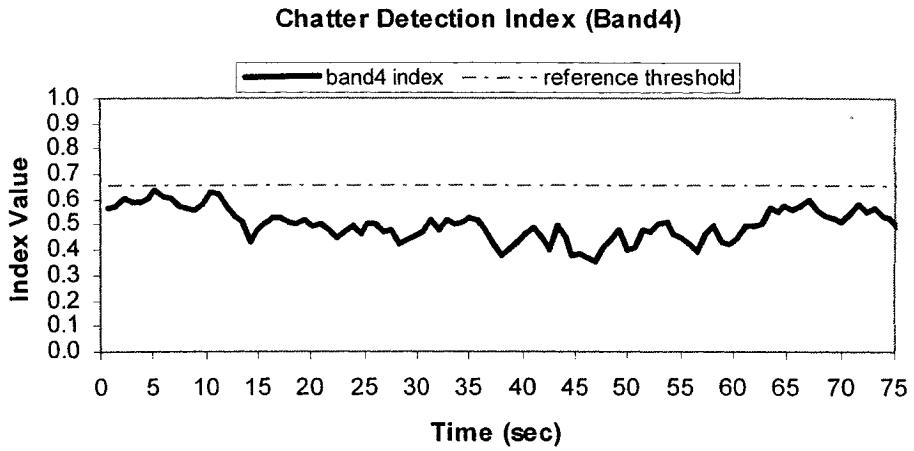
c) Wavelet domain plot: scale 1



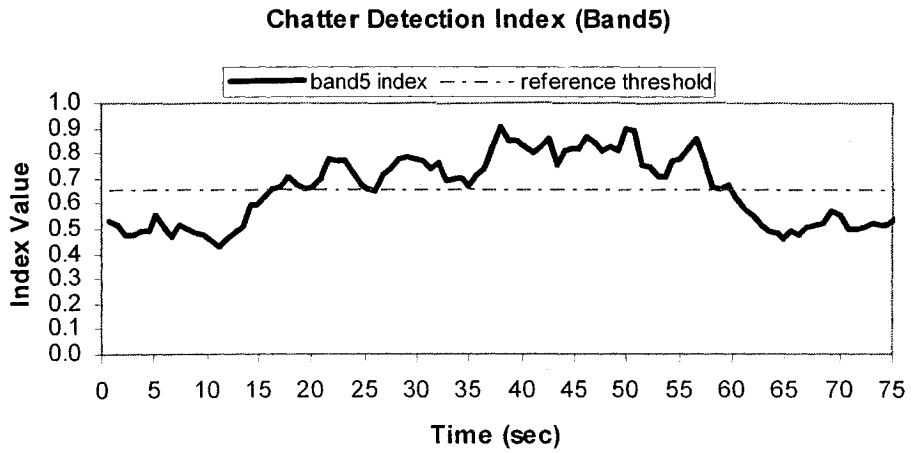
d) Wavelet domain plot: scale 2



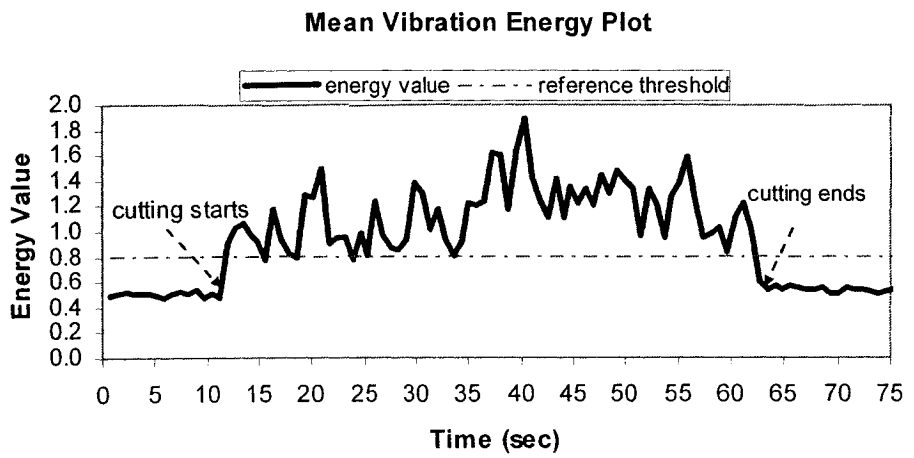
e) Wavelet domain plot: scale 3



f) Wavelet domain plot: scale 4

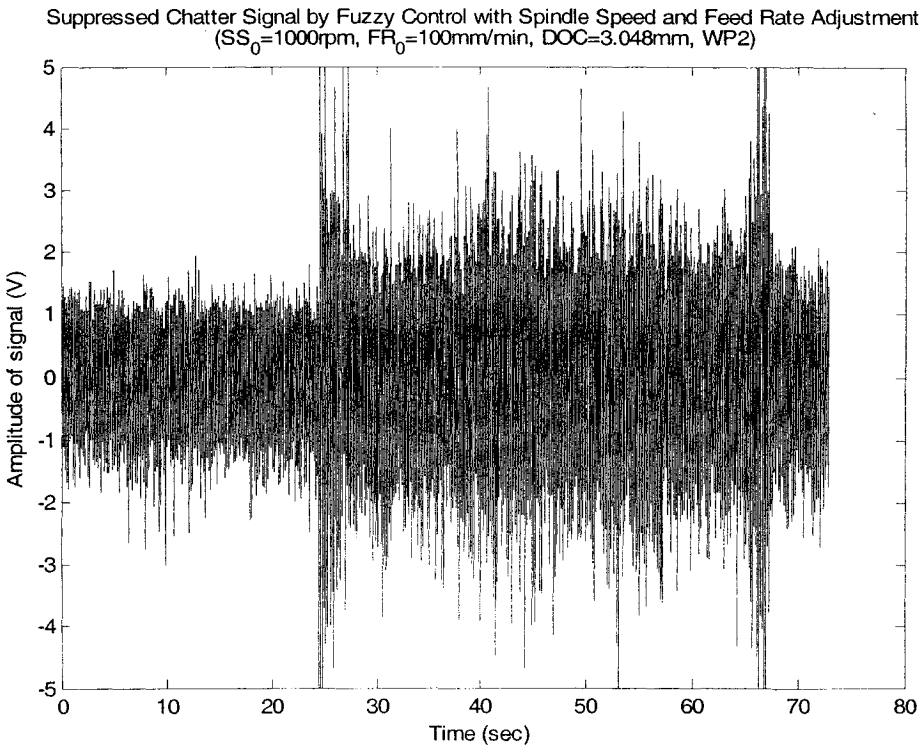


g) Wavelet domain plot: scale 5

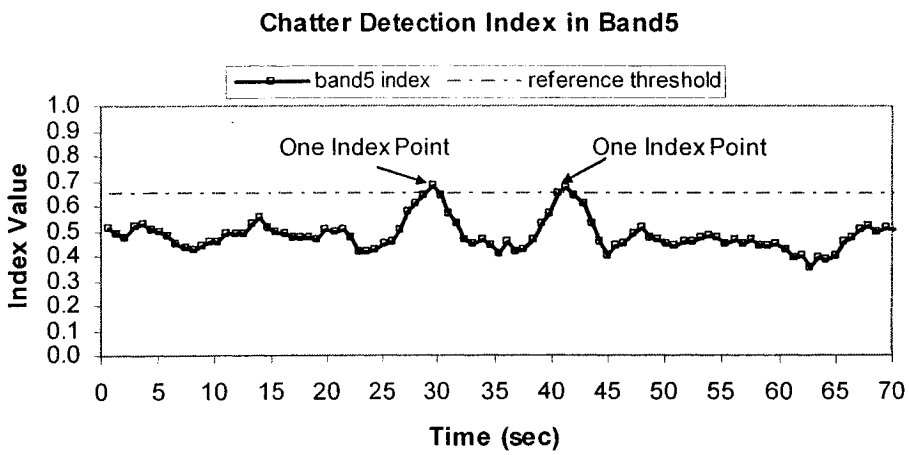


h) Mean vibration energy plot

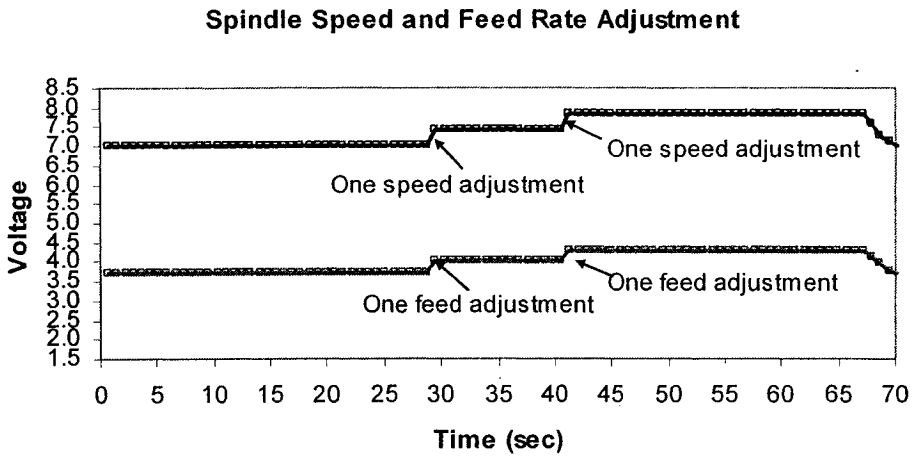
Figure 5.13 Chatter detection result of test #6 (Spindle speed 1000rpm, feed rate 100mm/min, depth of cut 3.048mm, WP2)



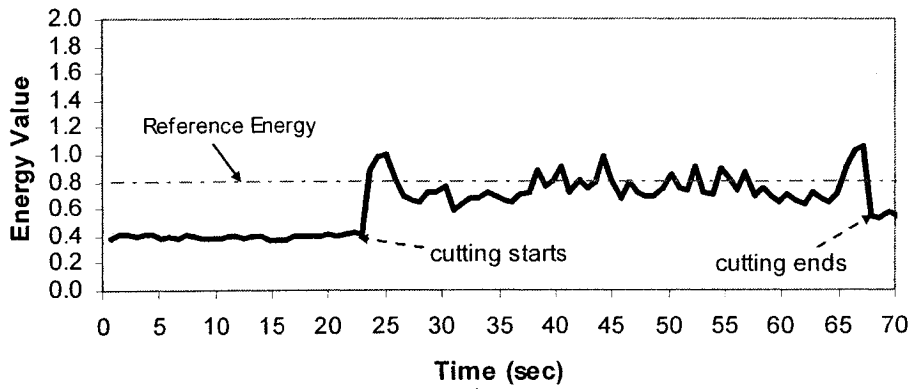
a) Time domain plot



b) Band 5 chatter index plot

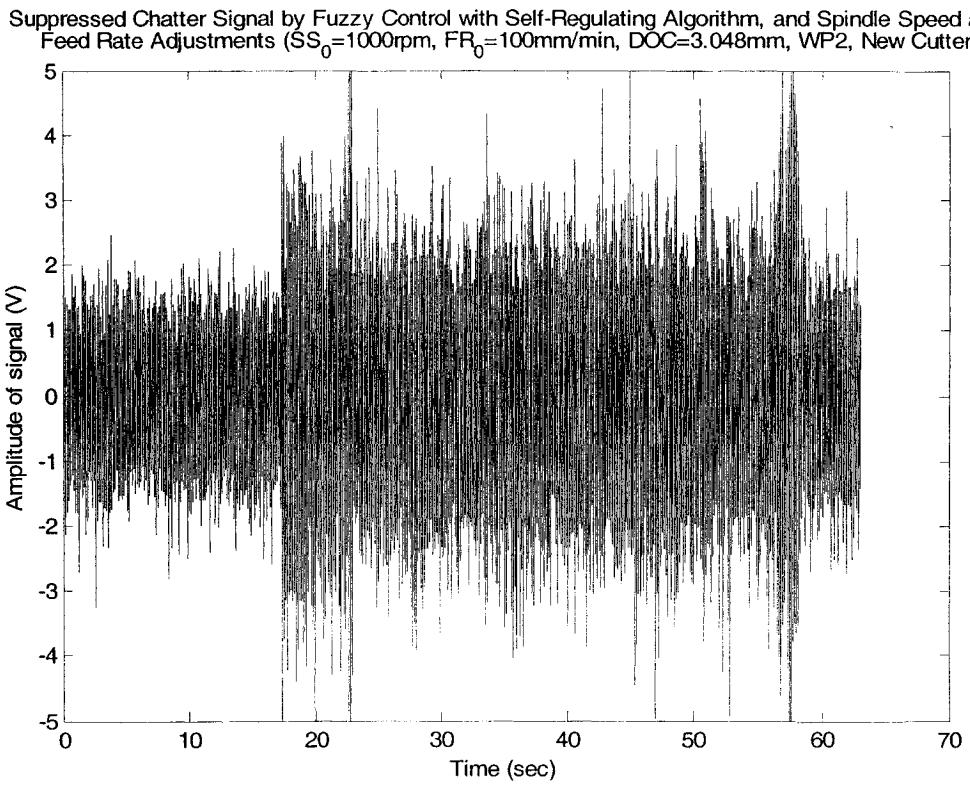


c) Spindle speed and feed rate adjustment plot

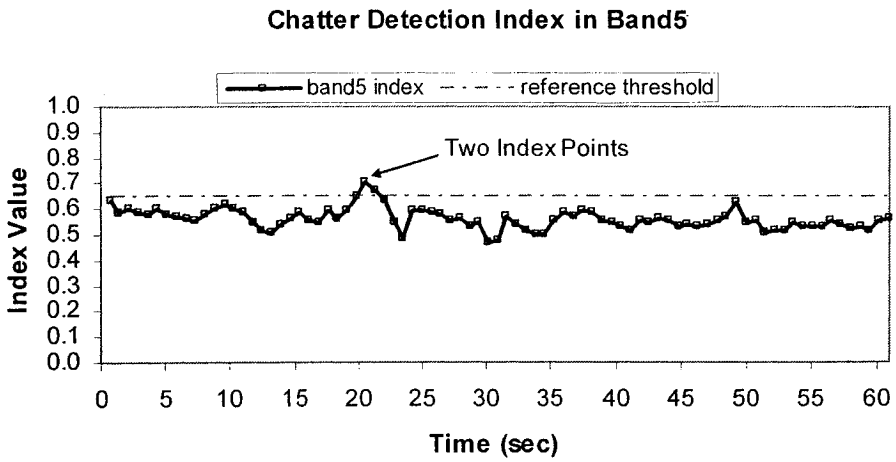


d) Mean vibration energy plot

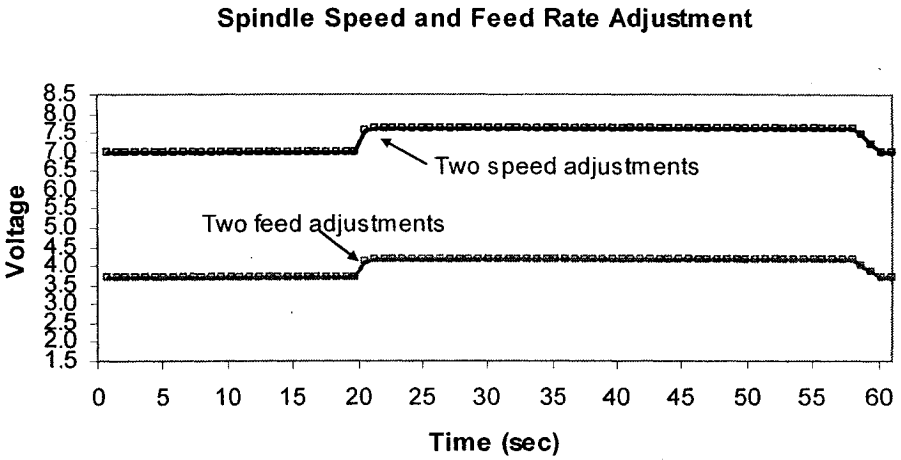
Figure 5.14 Chatter suppression result of test #7 by adjusting speed and feed (Fuzzy control only, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.048mm, WP2)



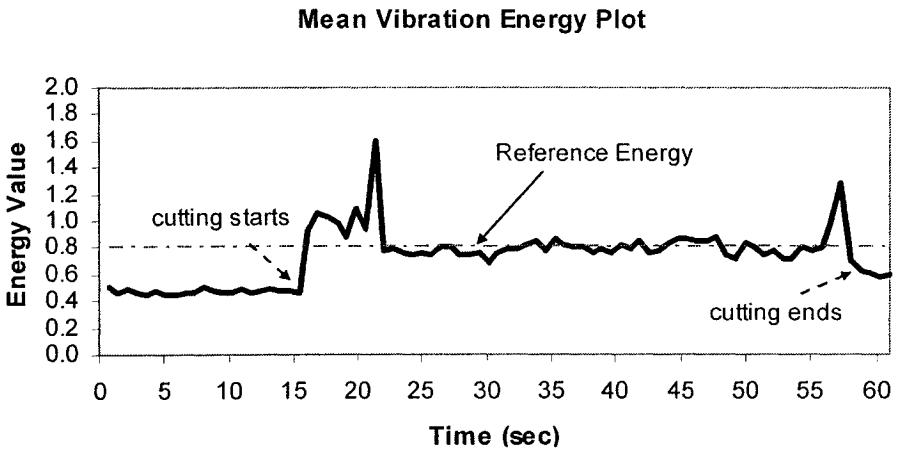
a) Time domain plot



b) Band 5 chatter index plot

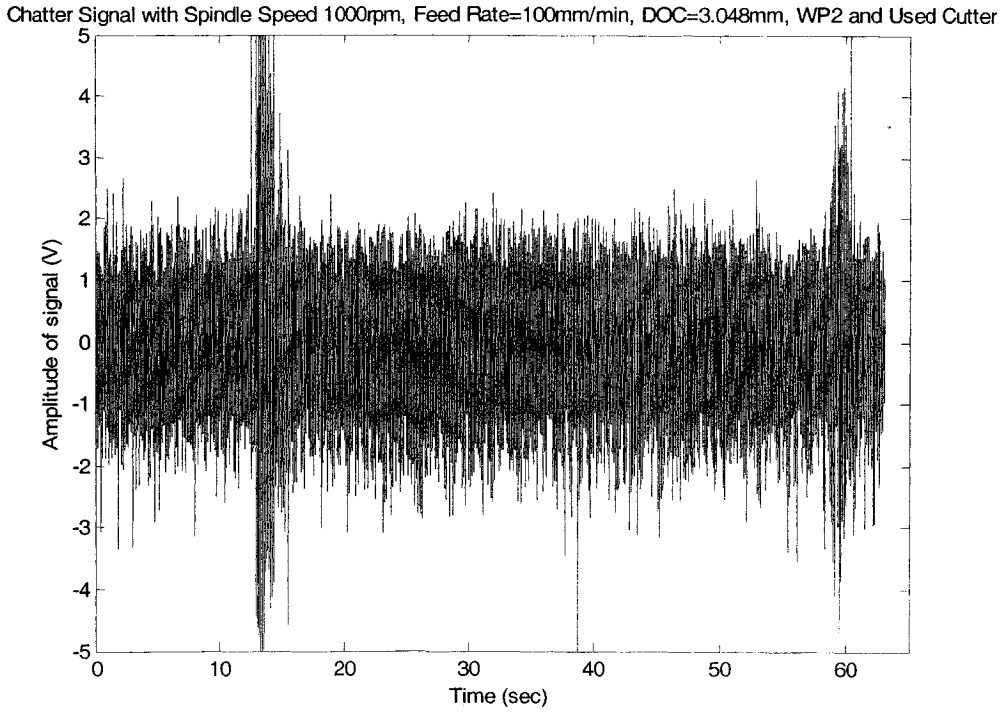


c) Spindle speed and feed rate adjustment plot

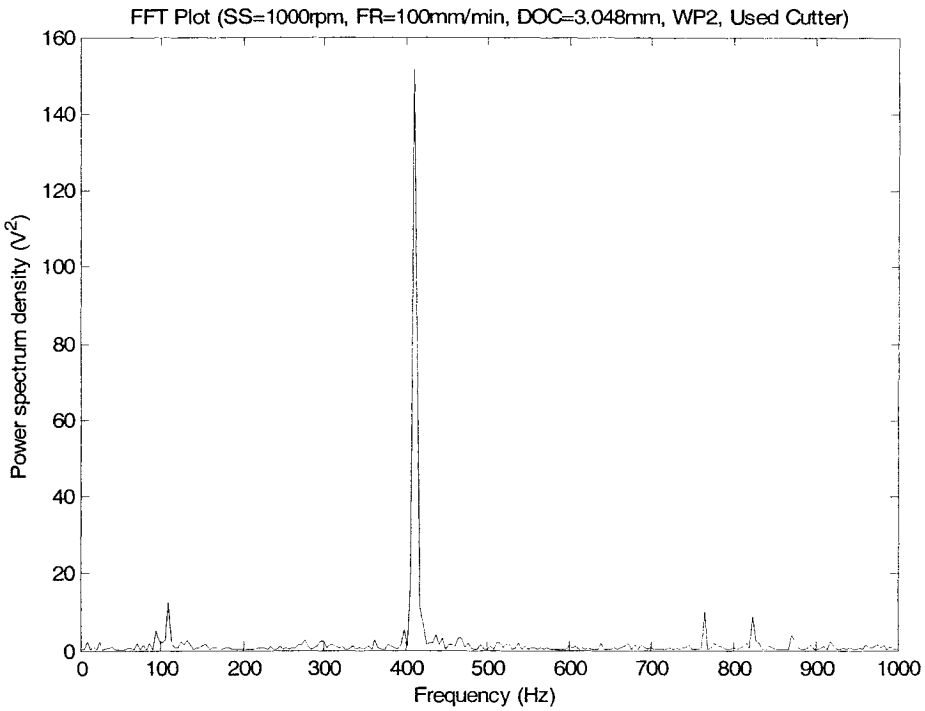


d) Mean vibration energy plot

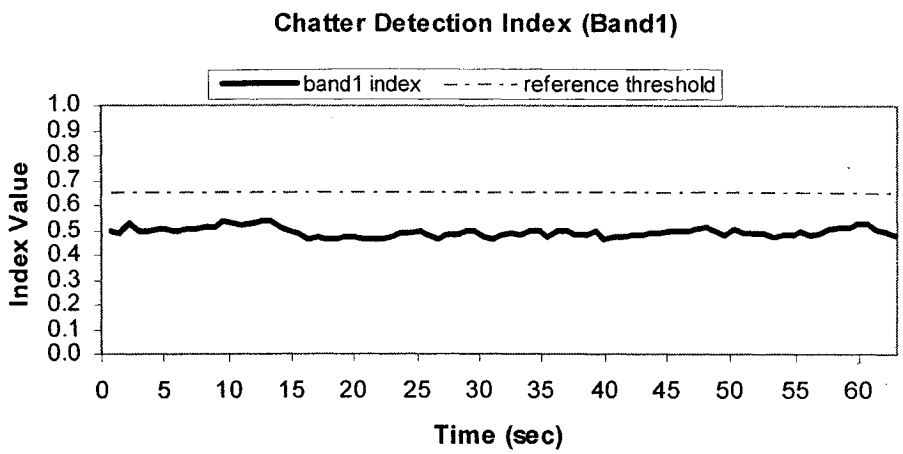
Figure 5.15 Chatter suppression result of test #8 by adjusting speed and feed (Fuzzy control with self-regulating, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.048mm, WP2)



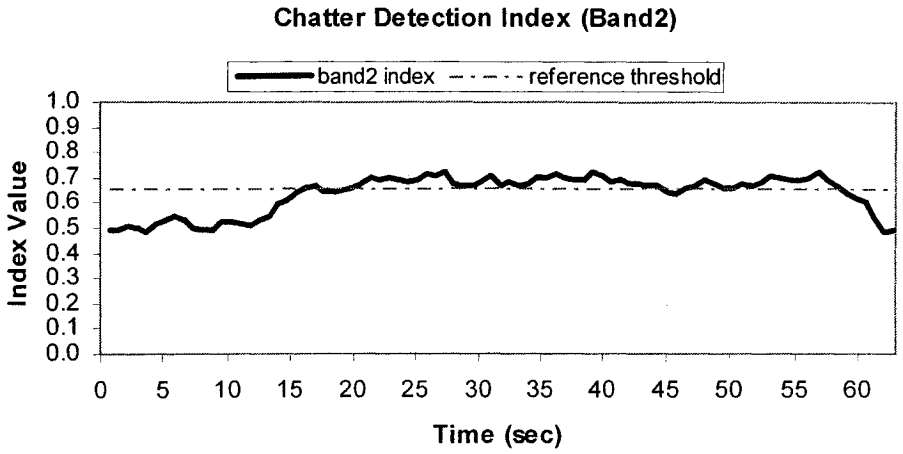
a) Time domain plot



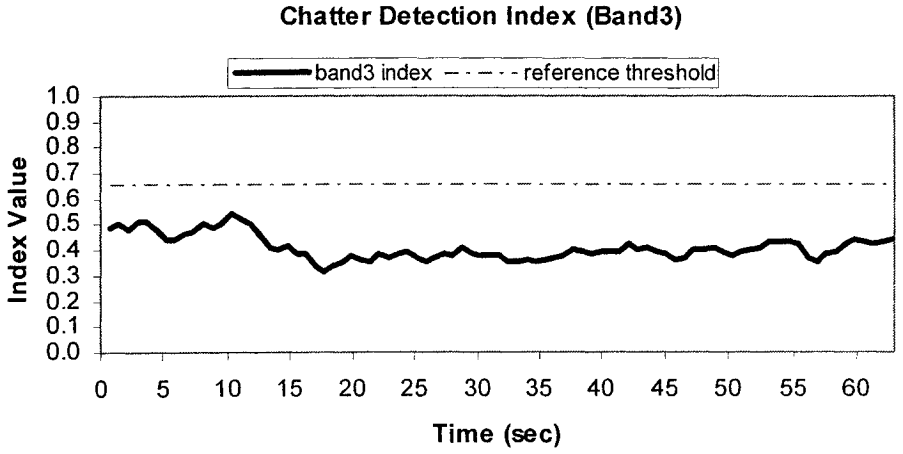
b) FFT plot



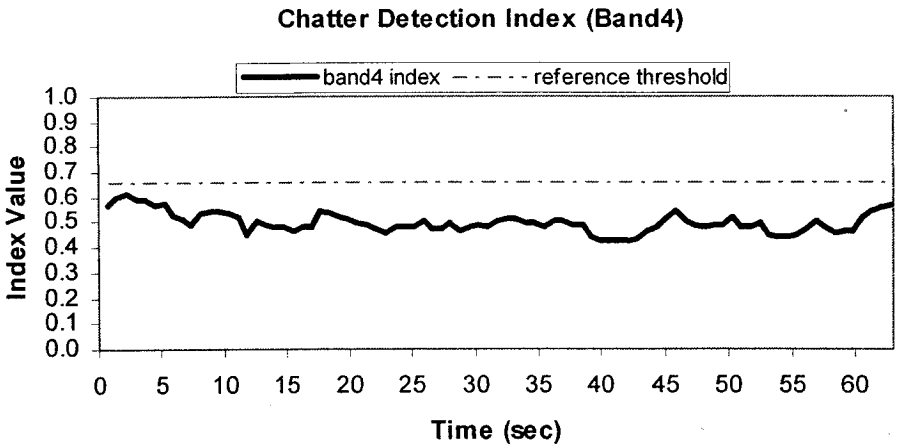
c) Wavelet domain plot: scale 1



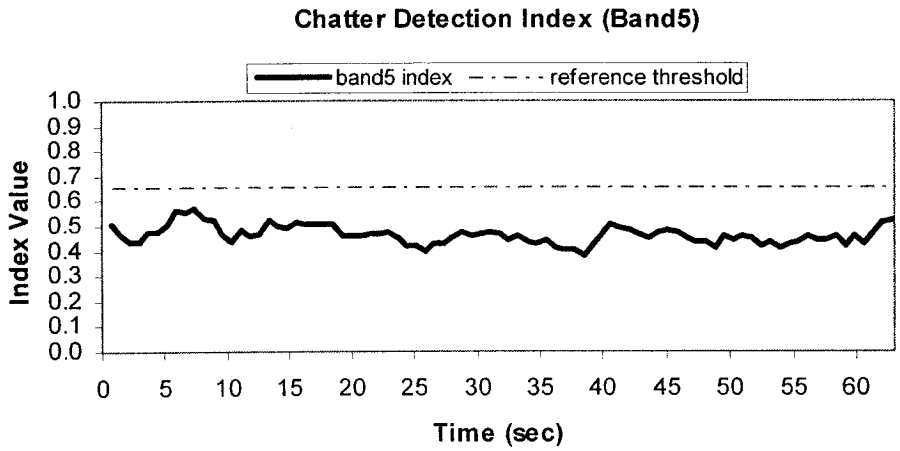
d) Wavelet domain plot: scale 2



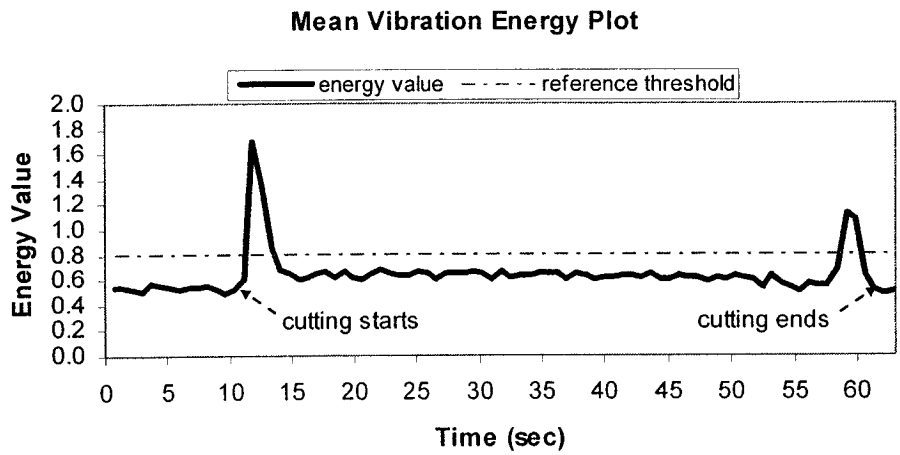
e) Wavelet domain plot: scale 3



f) Wavelet domain plot: scale 4

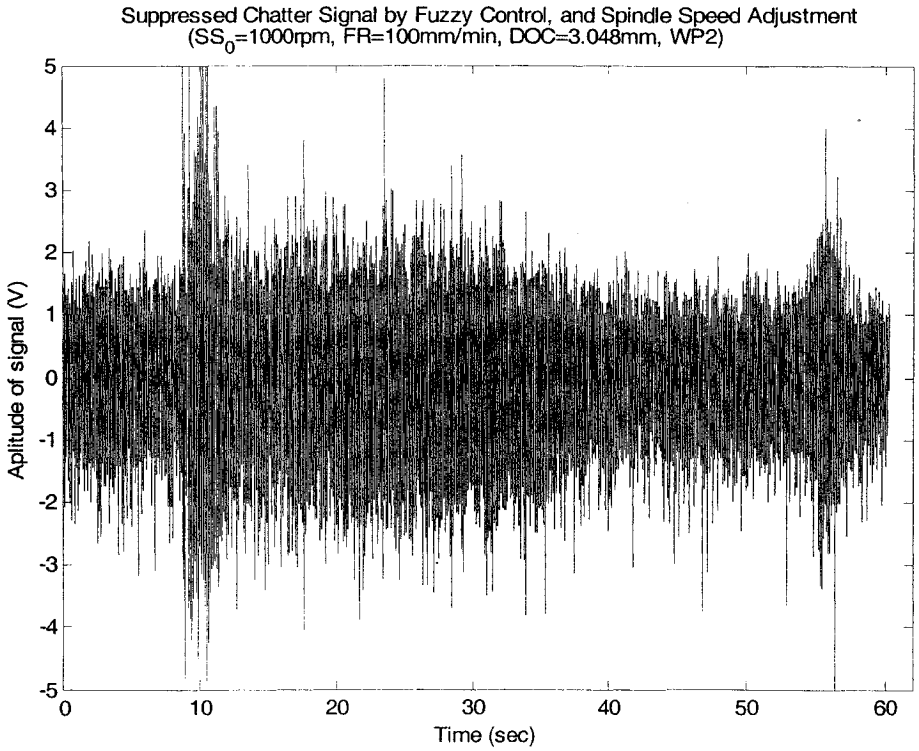


g) Wavelet domain plot: scale 5

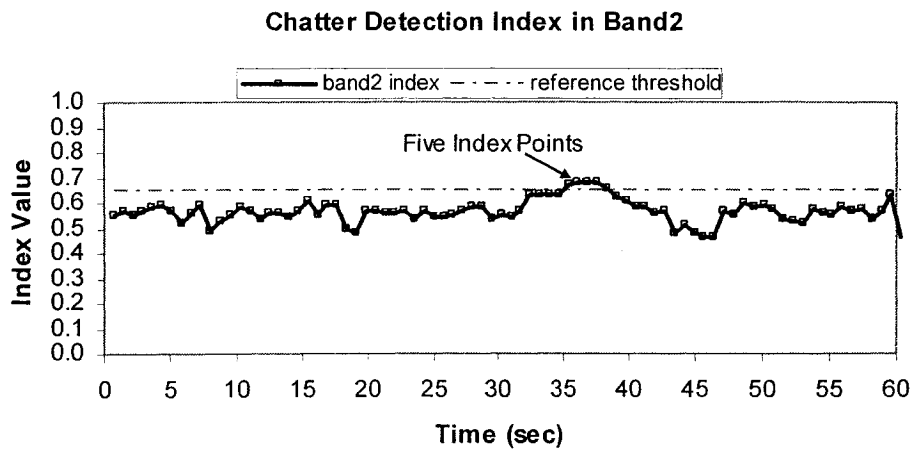


h) Mean vibration energy plot

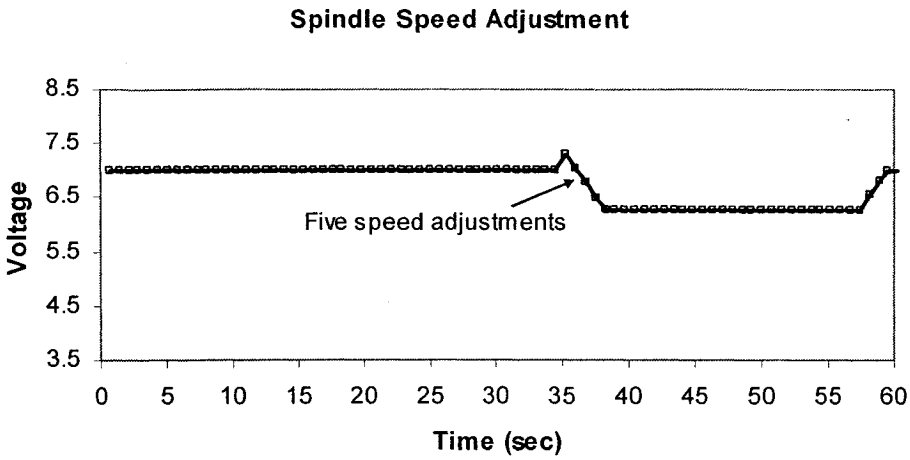
Figure 5.16 Chatter detection result of test #9 (Spindle speed 1000rpm, feed rate 100mm/min, depth of cut 3.048mm, WP2)



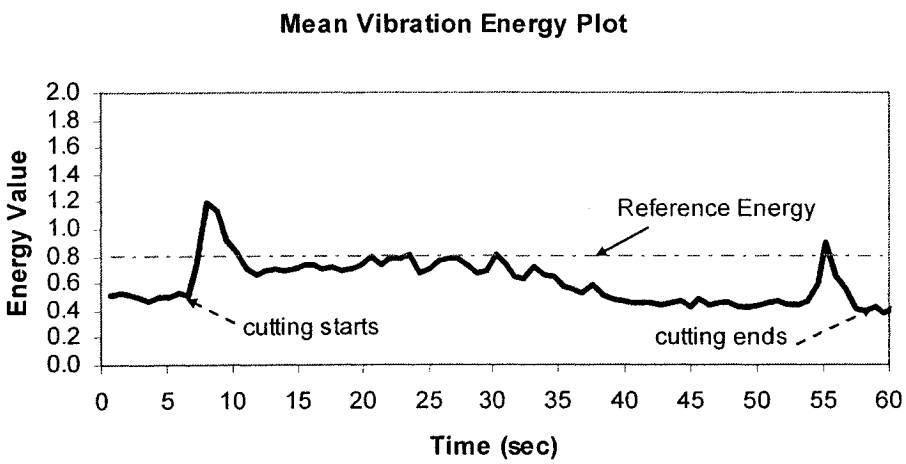
a) Time domain plot



b) Band 2 chatter index plot

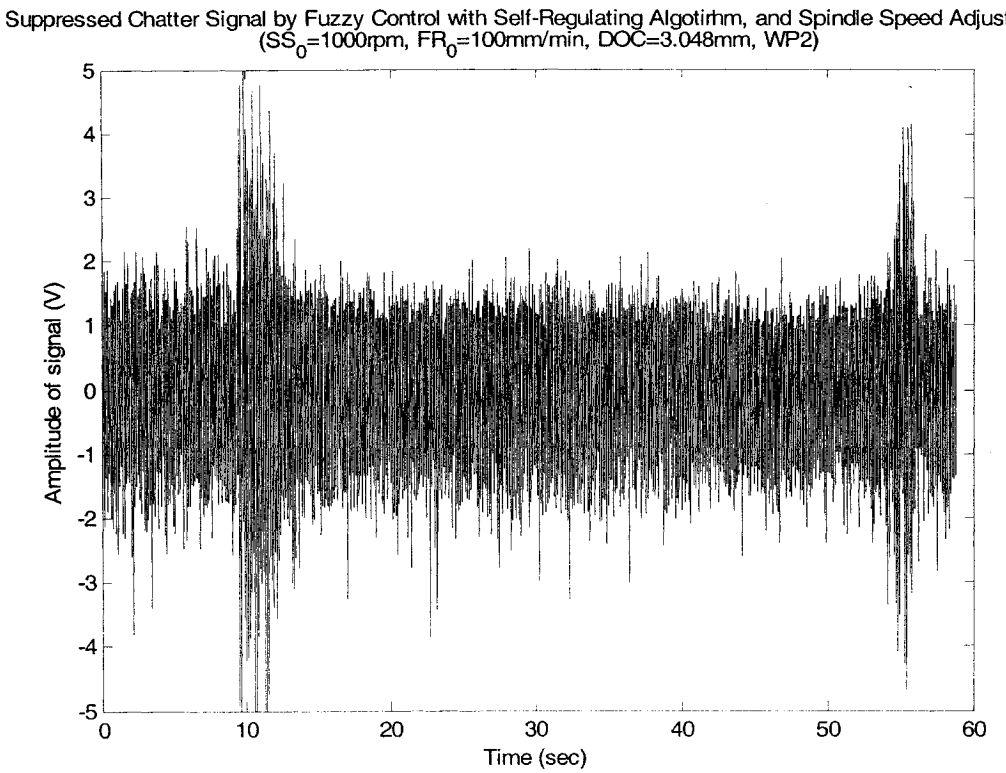


c) Spindle speed adjustment

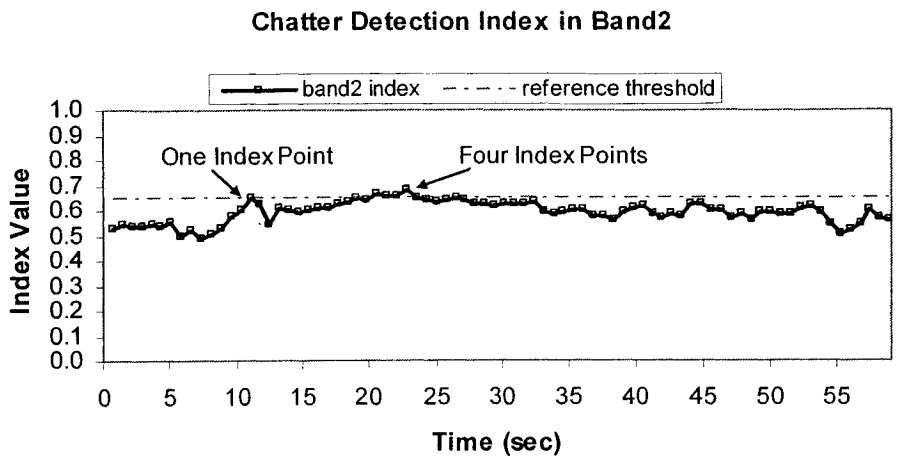


d) Mean vibration energy plot

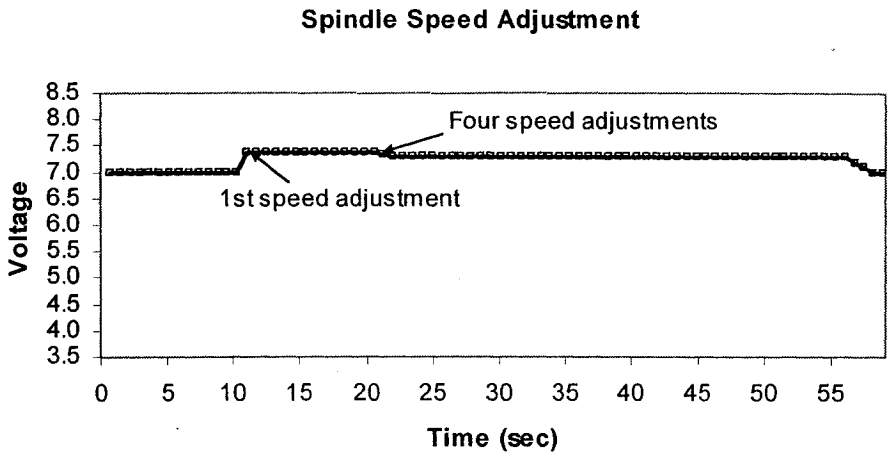
Figure 5.17 Chatter suppression result of test #10 by adjusting spindle speed only (Fuzzy control only, initial spindle speed 100rpm, feed rate 100mm/min, depth of cut 3.048mm, WP2)



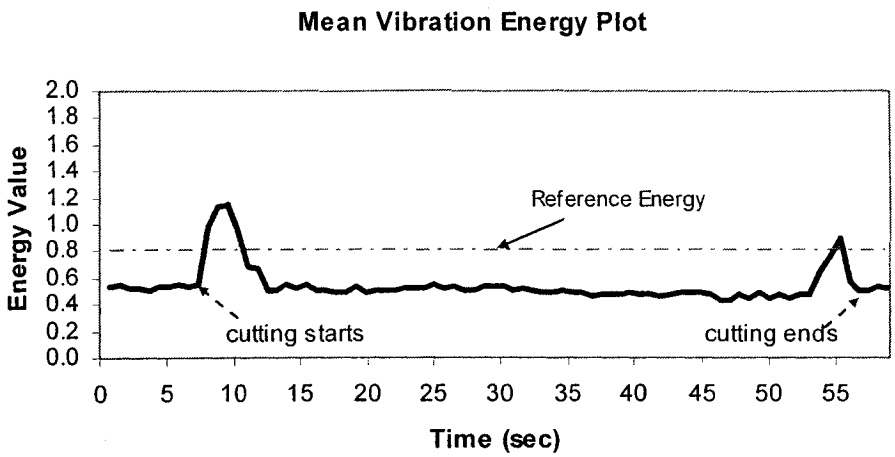
a) Time domain plot



b) Band 2 chatter index plot

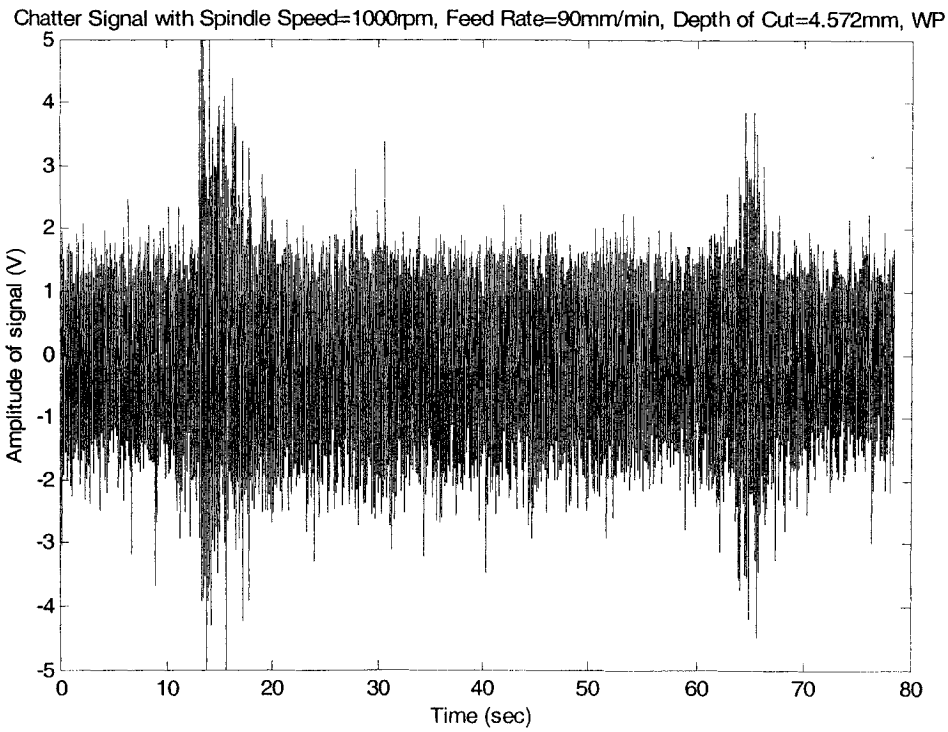


c) Spindle speed adjustment plot

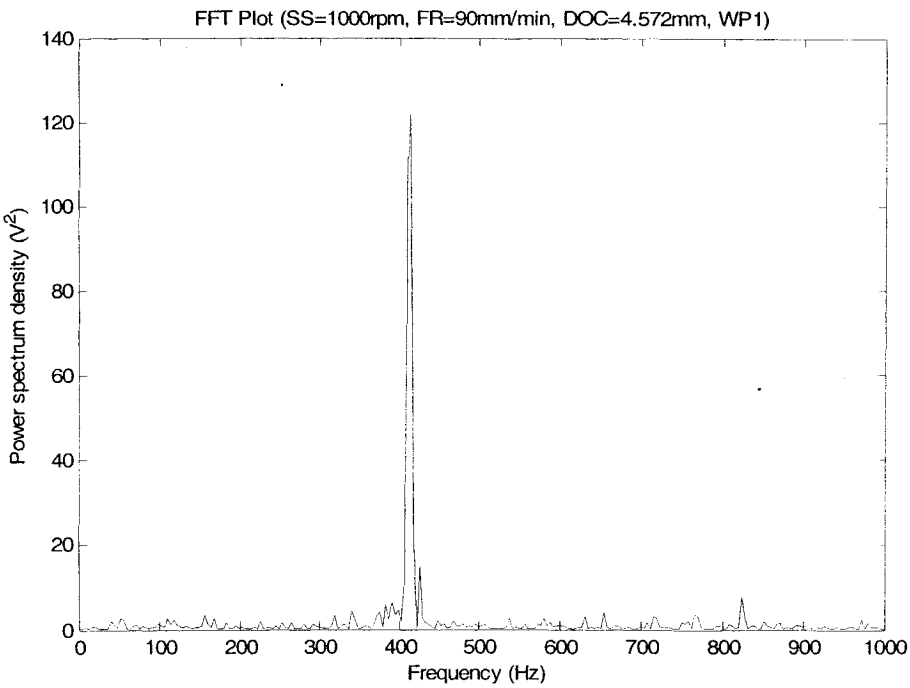


d) Mean vibration energy plot

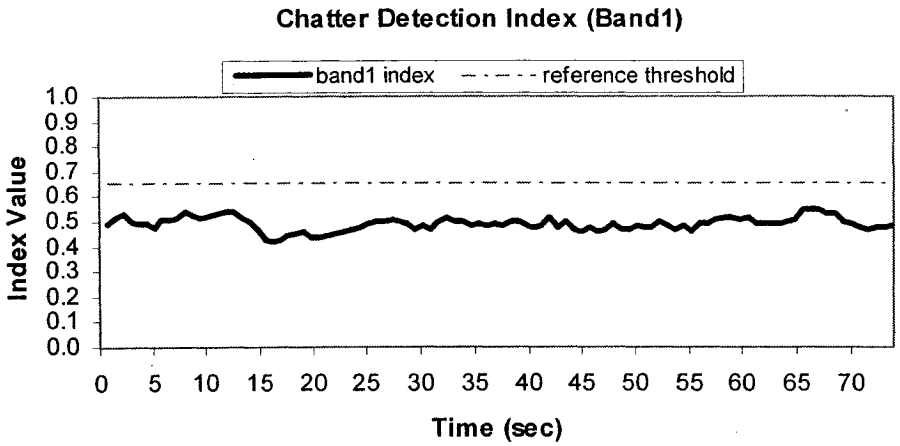
Figure 5.18 Chatter suppression result of test #11 by adjusting speed only (Fuzzy control with self-regulating, initial spindle speed 1000rpm, feed rate 100mm/min, depth of cut 3.048mm, WP2)



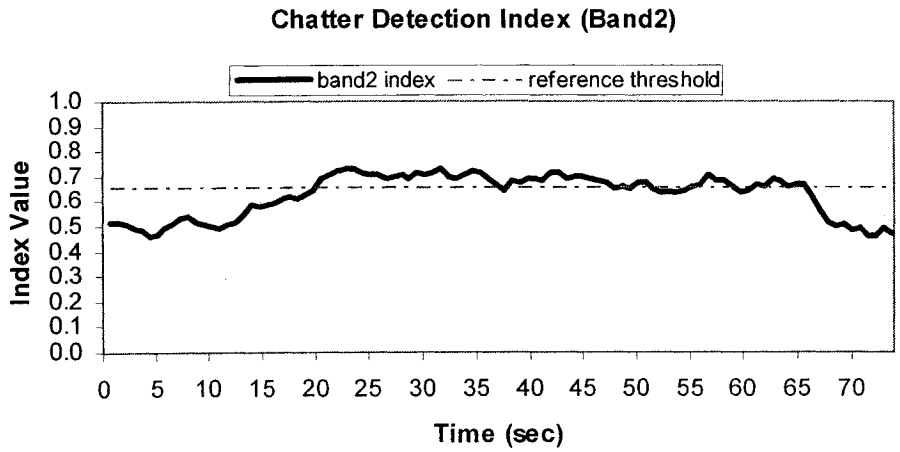
a) Time domain plot



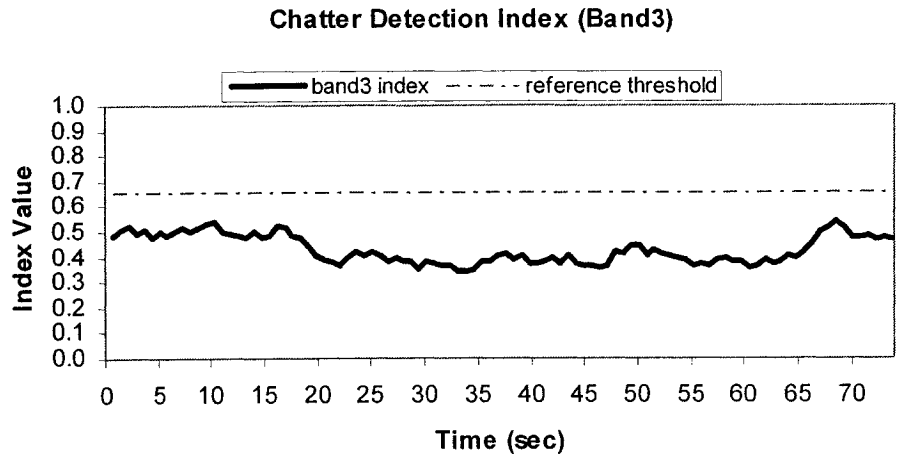
b) FFT plot



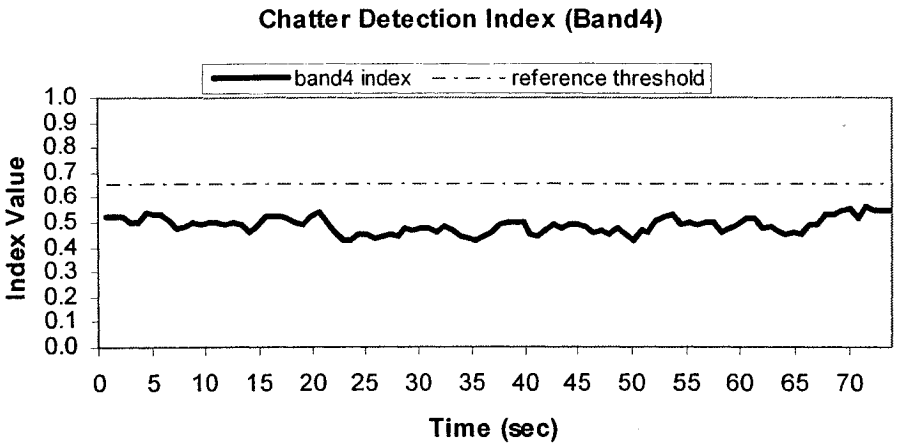
c) Wavelet domain plot: scale 1



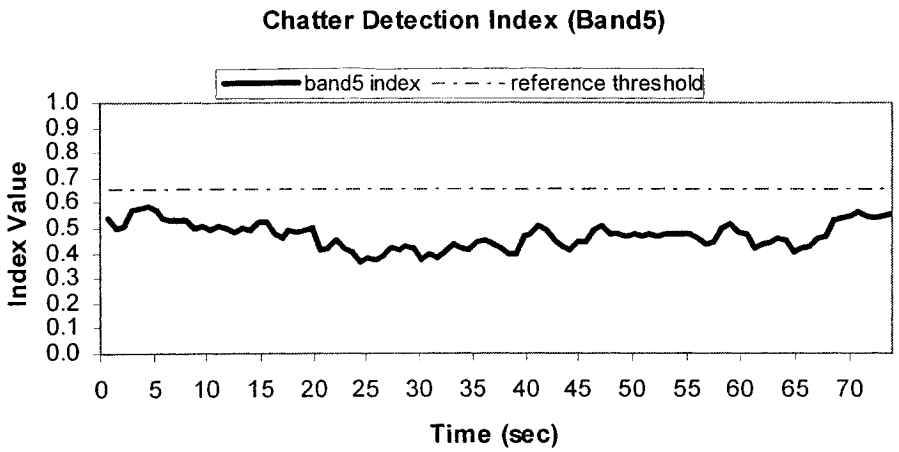
d) Wavelet domain plot: scale 2



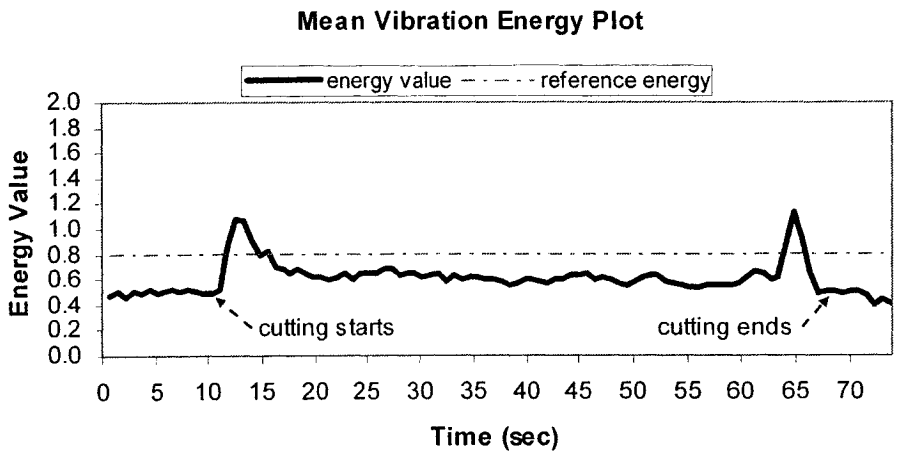
e) Wavelet domain plot: scale 3



f) Wavelet domain plot: scale 4

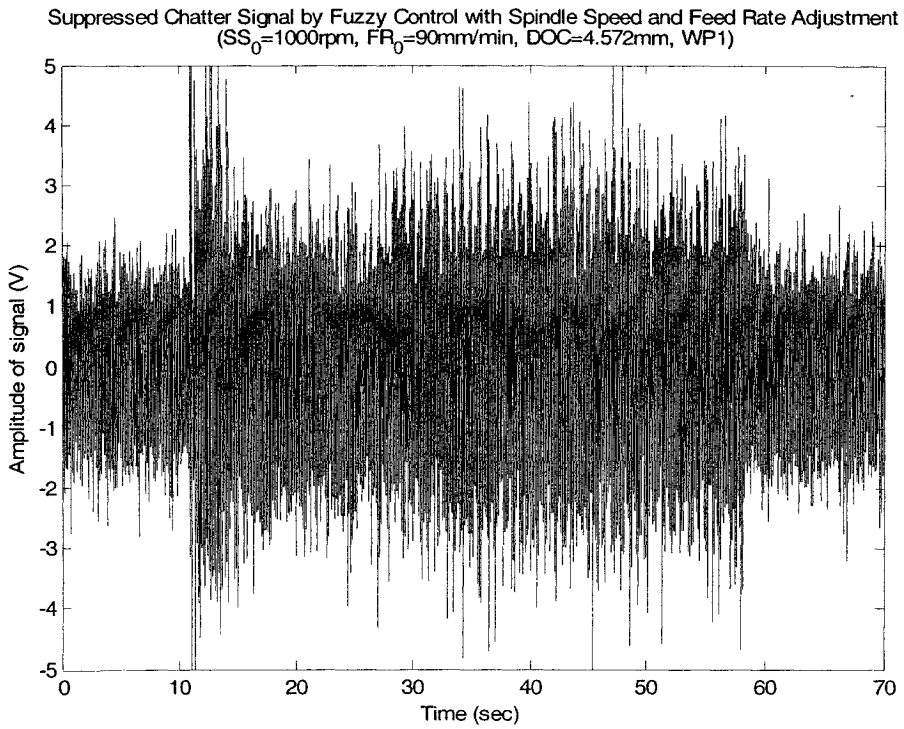


g) Wavelet domain plot: scale 5

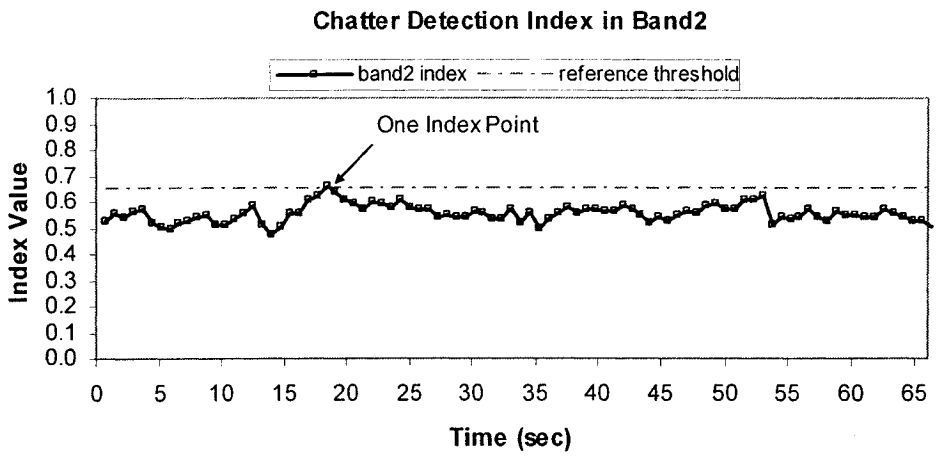


h) Mean vibration energy plot

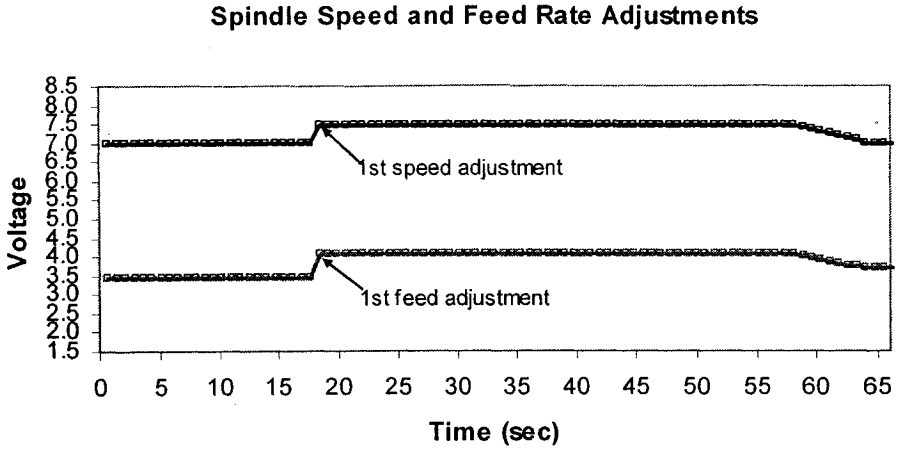
Figure 5.19 Chatter detection result of test #12 (Spindle speed 1000rpm, feed rate 90mm/min, depth of cut 4.572mm, WP1)



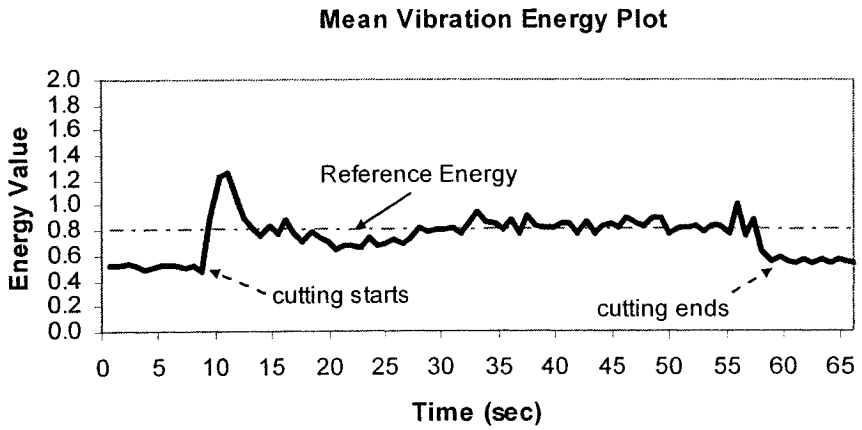
a) Time domain plot



b) Band 2 chatter index

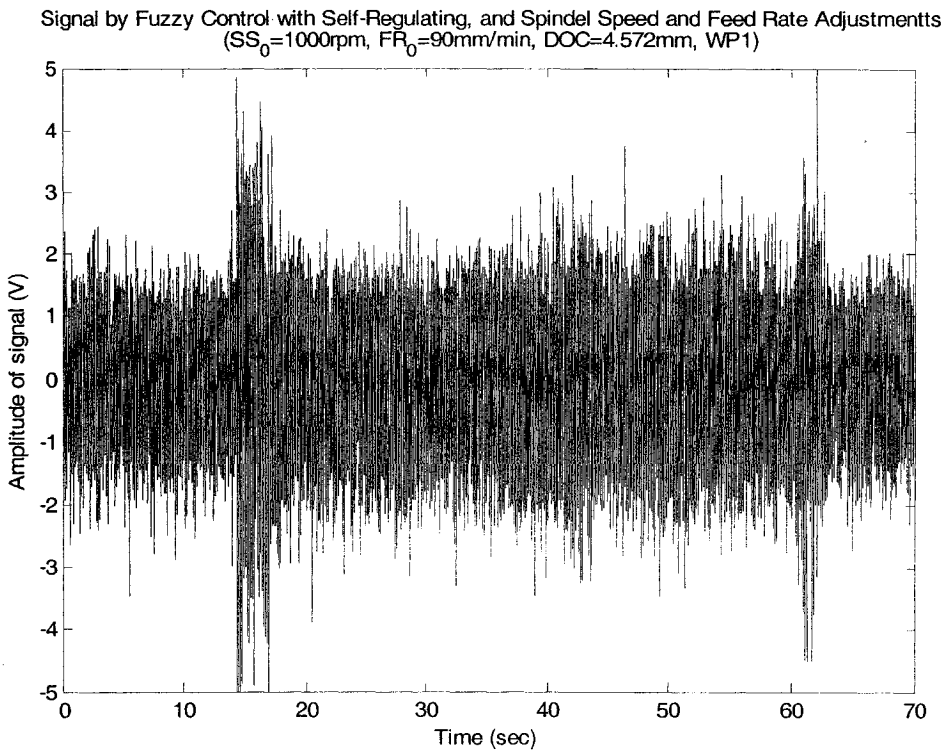


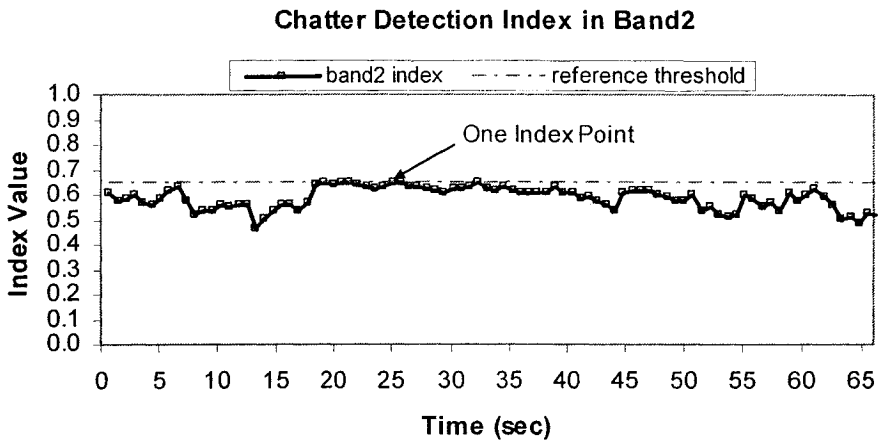
c) Spindle speed and feed rate adjustment plot



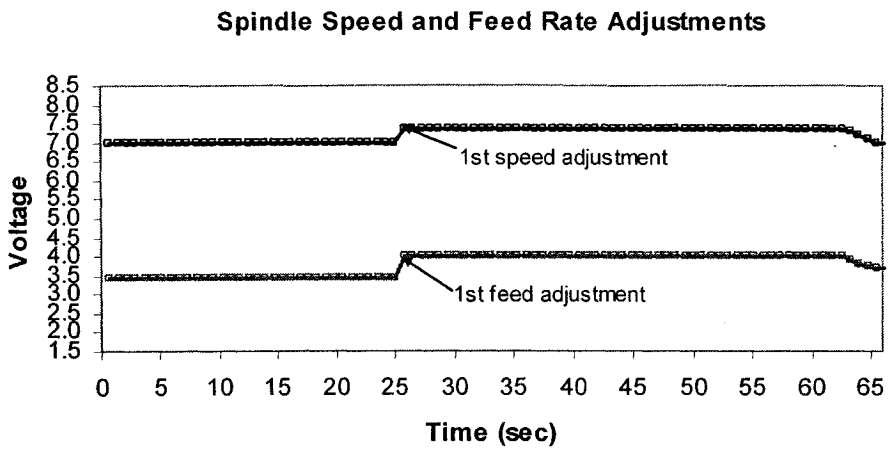
d) Mean vibration energy plot

Figure 5.20 Chatter suppression result of test #13 by adjusting speed and feed (Fuzzy control only, initial spindle speed 1000rpm, initial feed rate 90mm/min, depth of cut 4.572mm, WP1)

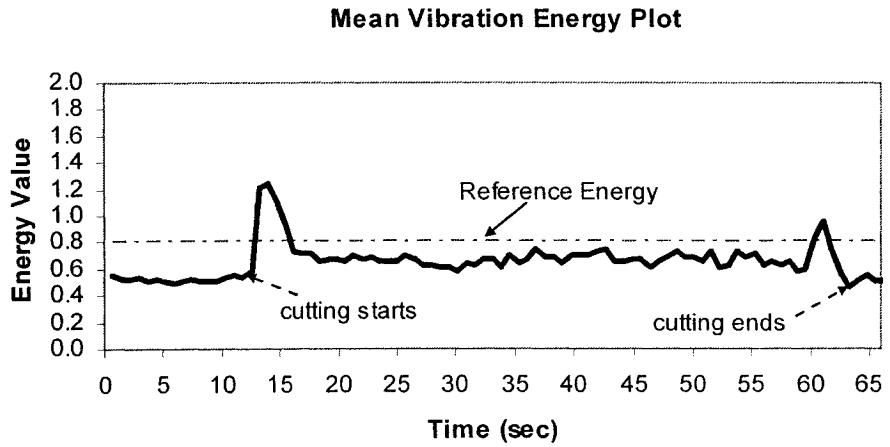




b) Band 2 chatter index plot

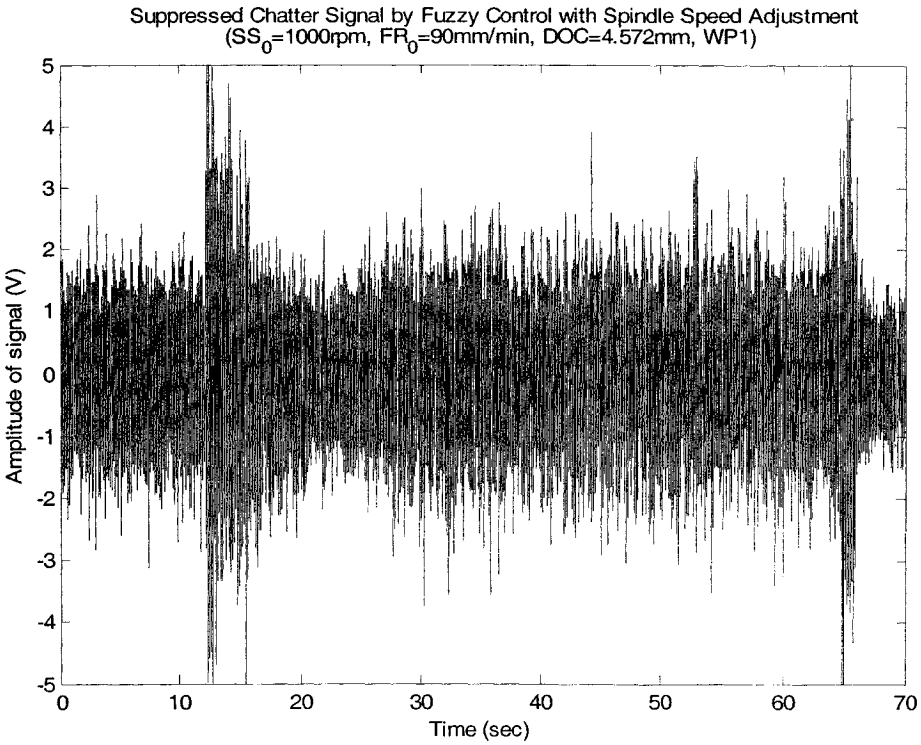


c) Spindle speed and feed rate adjustment plot

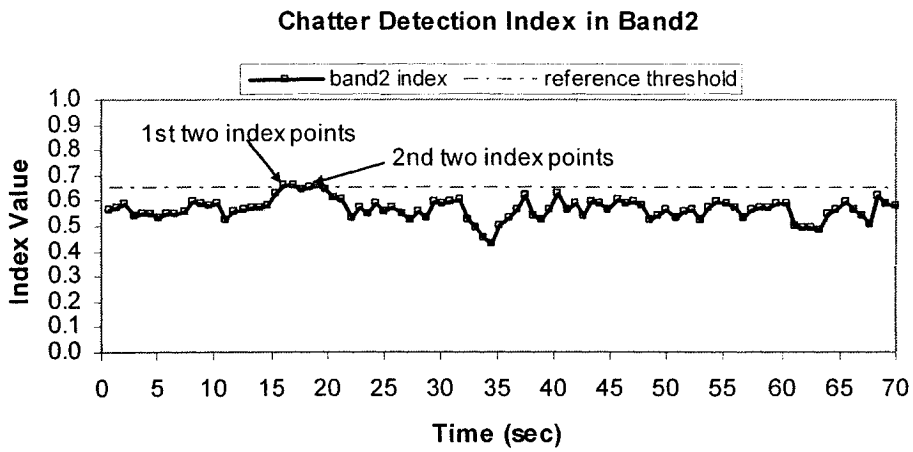


d) Mean vibration energy plot

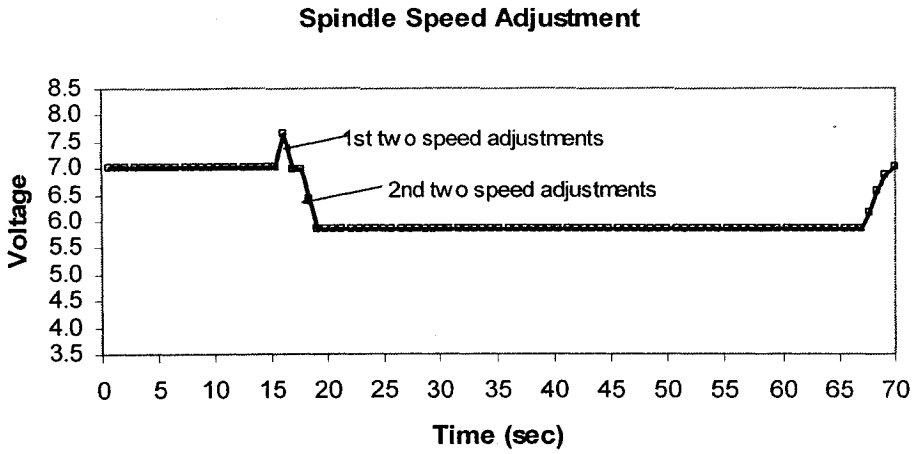
Figure 5.21 Chatter suppression result of test #14 by adjusting speed and feed (Fuzzy control with self-regulating, initial spindle speed 1000rpm, initial feed rate 90mm/min, depth of cut 4.572mm, WP1)



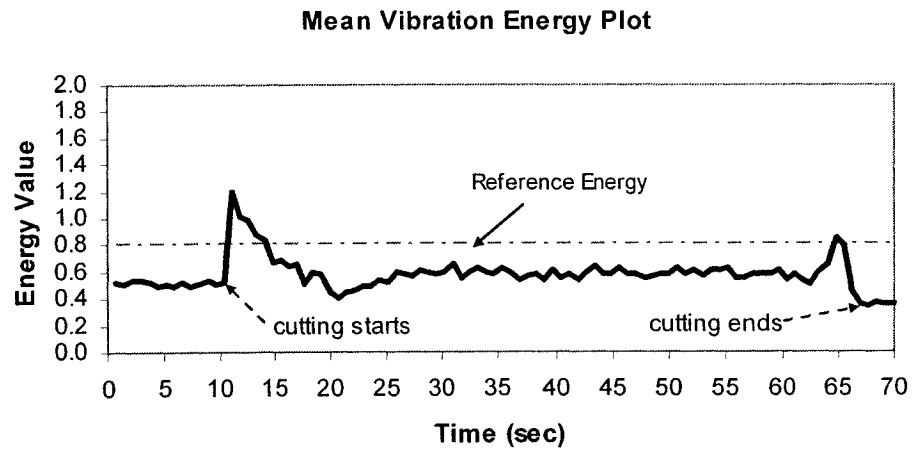
a) Time domain plot



b) Band 2 chatter index plot

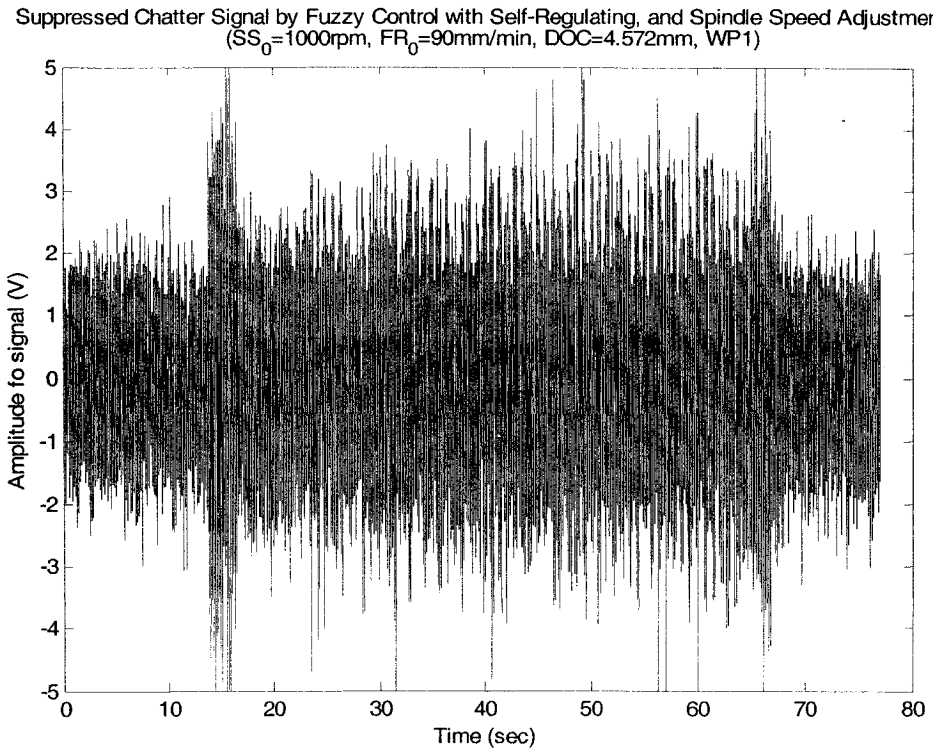


c) Spindle speed and feed rate adjustment plot

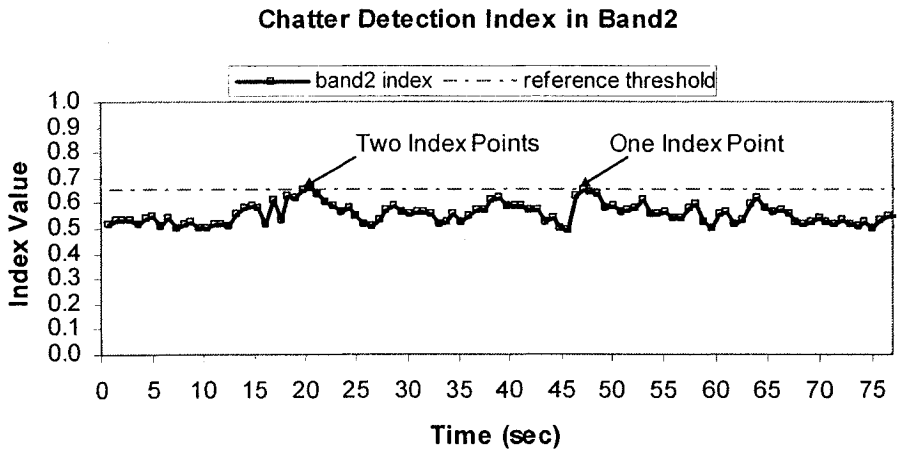


d) Mean vibration energy plot

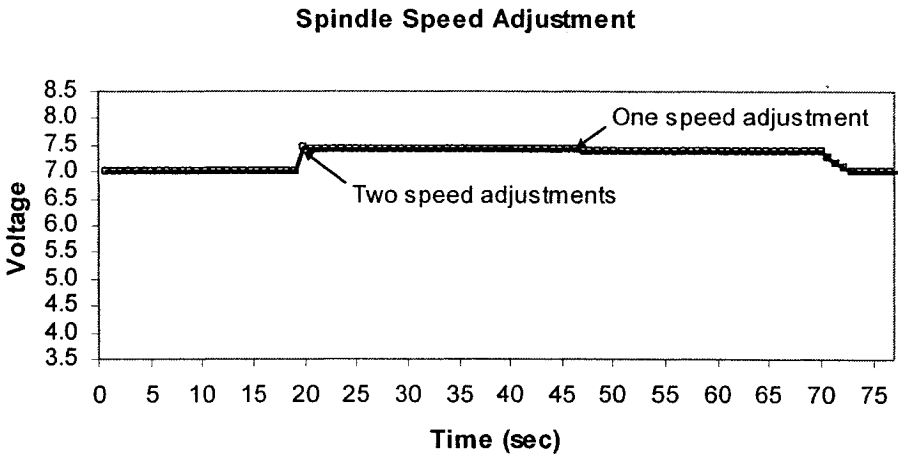
Figure 5.22 Chatter suppression result of test #15 by adjusting speed only (Fuzzy control only, initial spindle speed 1000rpm, feed rate 90mm/min, depth of cut 4.572mm, WP1)



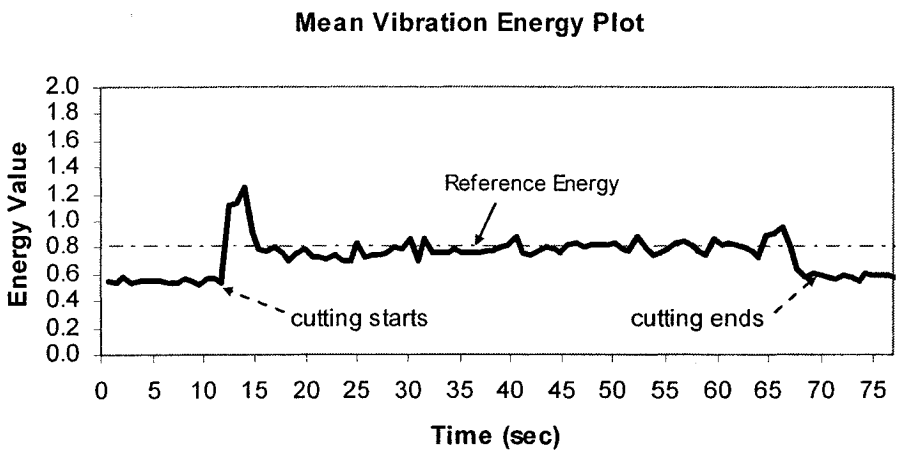
a) Time domain plot



b) Band 2 chatter index plot

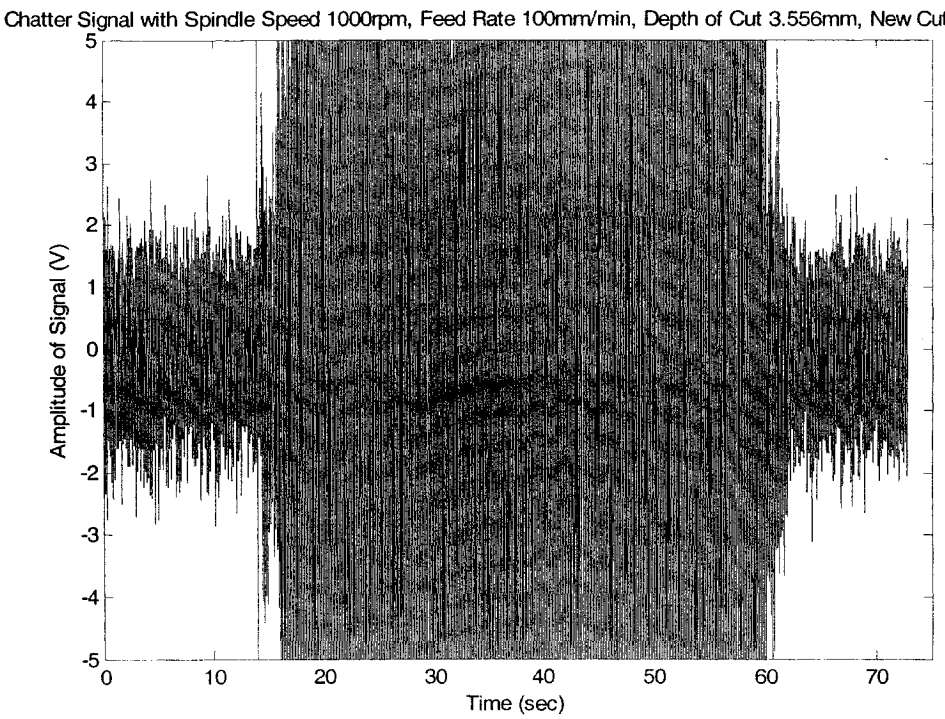


c) Spindle speed adjustment plot

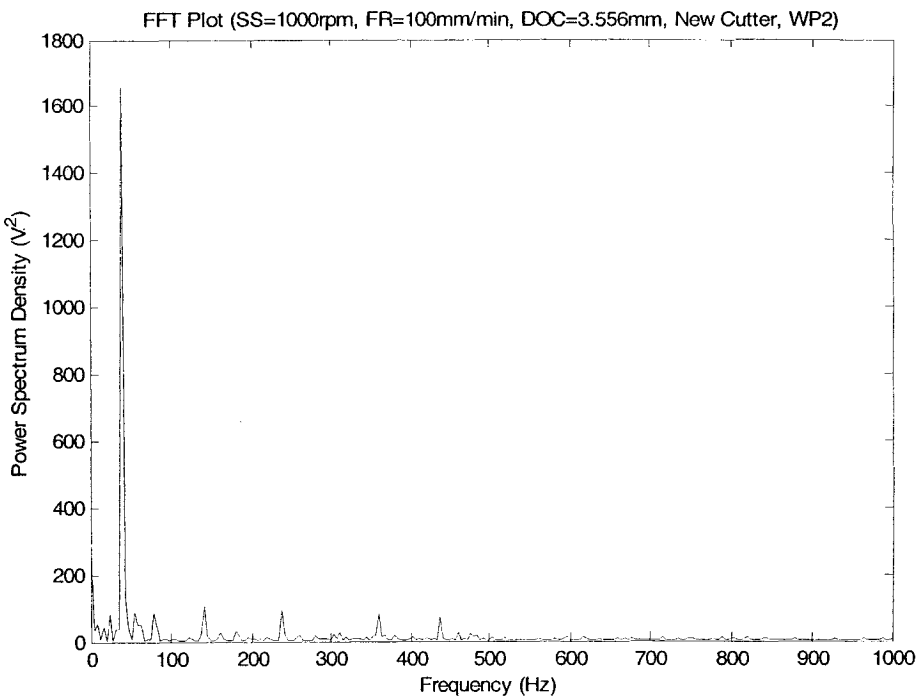


d) Mean vibration energy plot

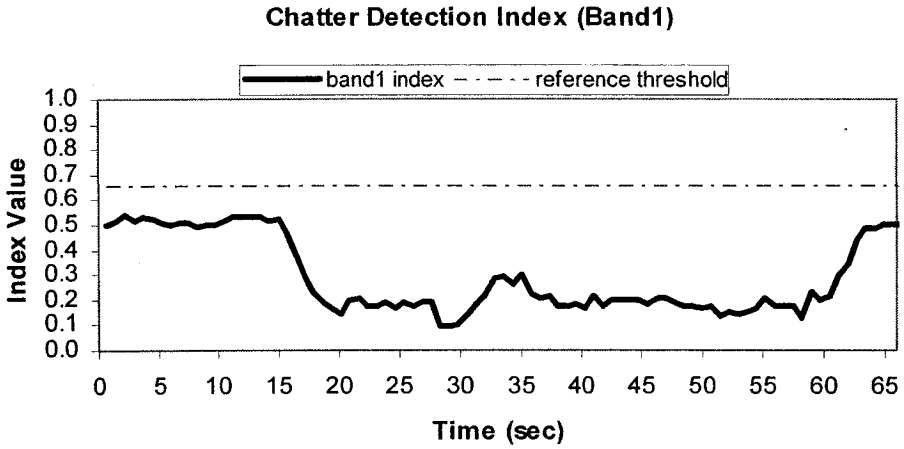
Figure 5.23 Chatter suppression result of test #16 by adjusting speed only (Fuzzy control with self-regulating, initial spindle speed 1000rpm, feed rate 90mm/min, depth of cut 4.572mm, WP1)



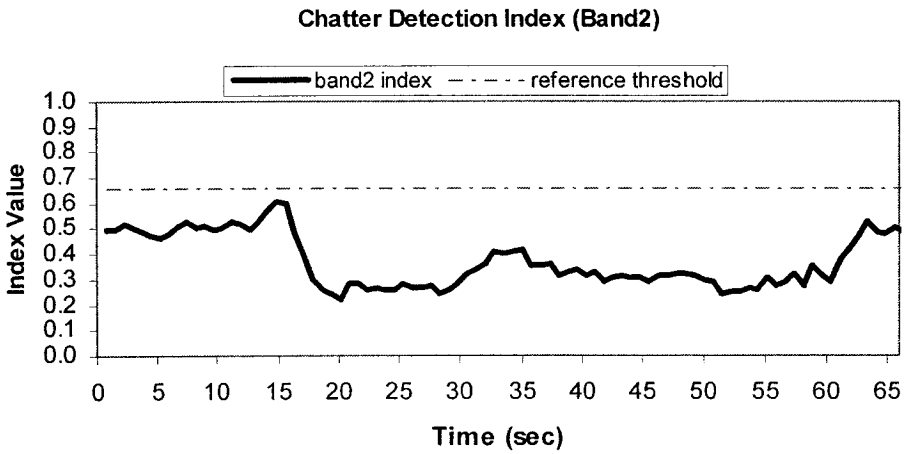
a) Time domain plot



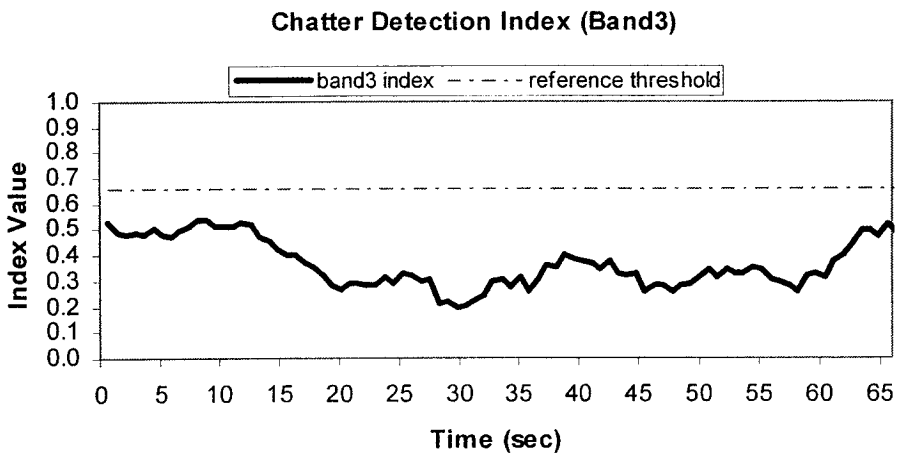
b) FFT plot



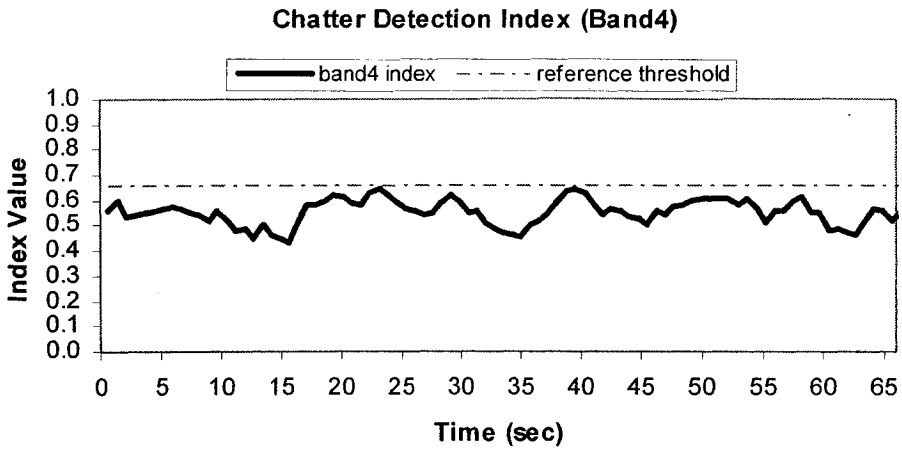
c) Wavelet domain plot: scale 1



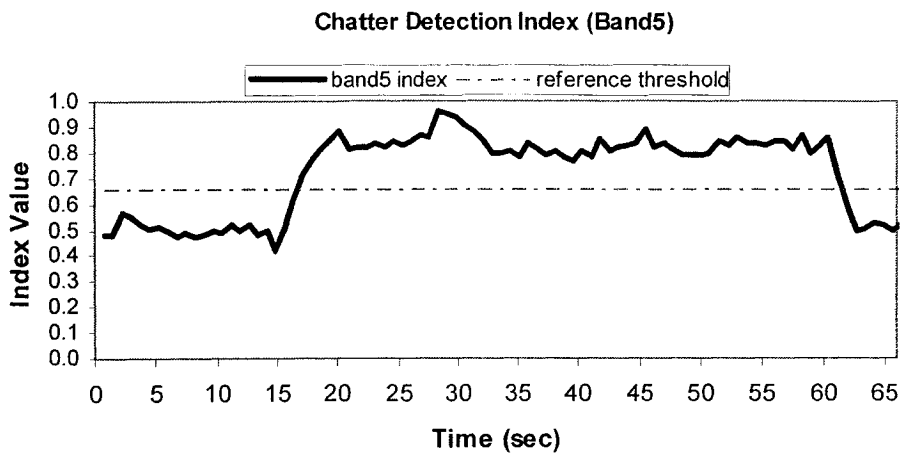
d) Wavelet domain plot: scale 2



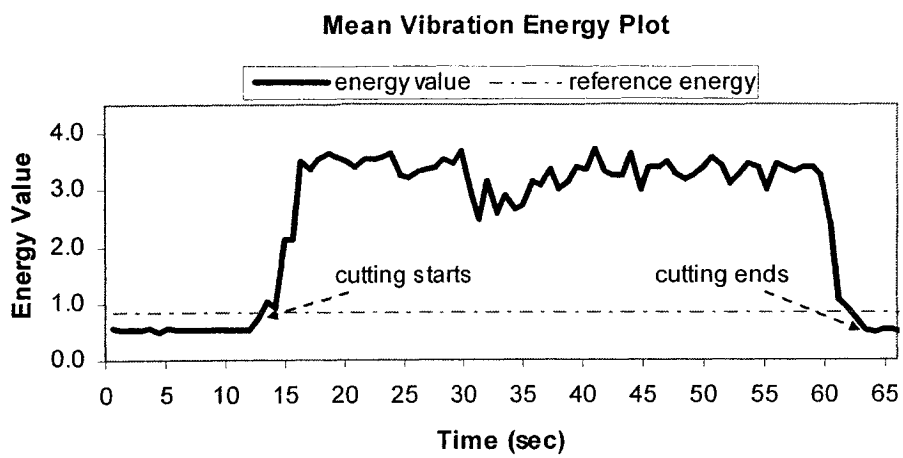
e) Wavelet domain plot: scale 3



f) Wavelet domain plot: scale 4

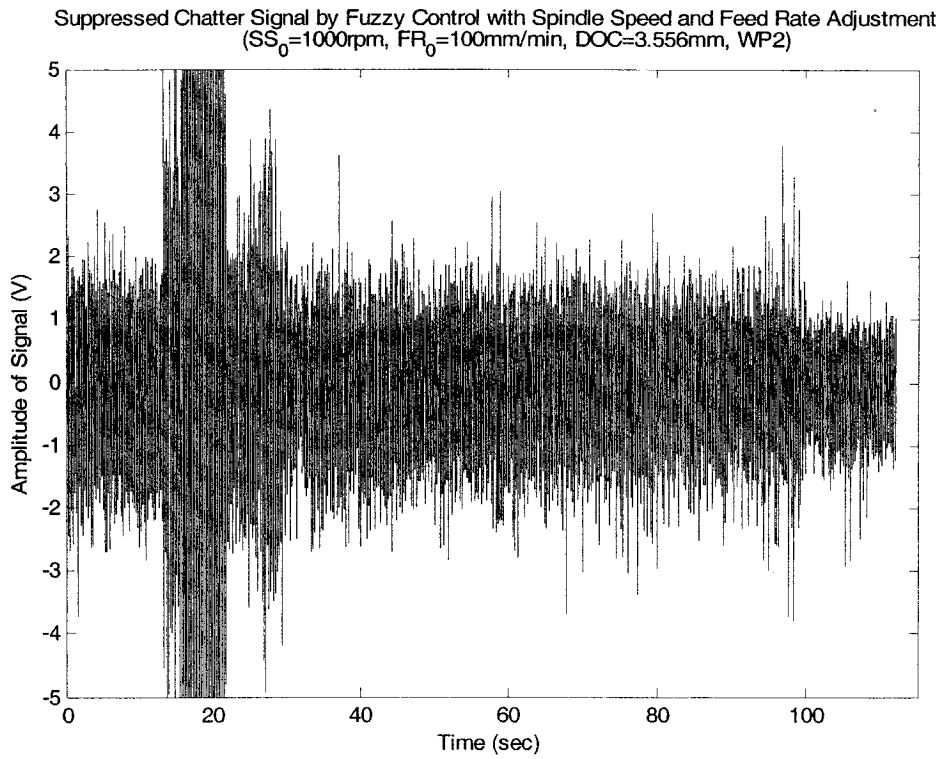


g) Wavelet domain plot: scale 5

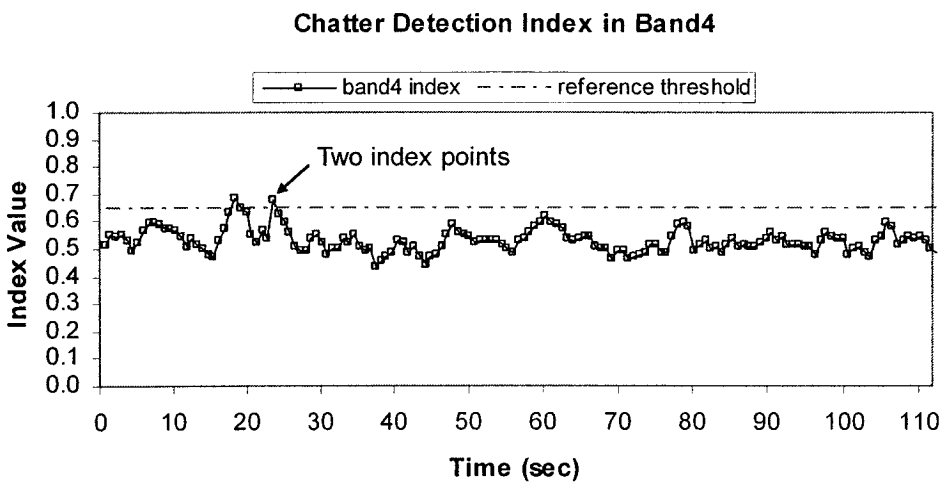


h) Mean vibration energy plot

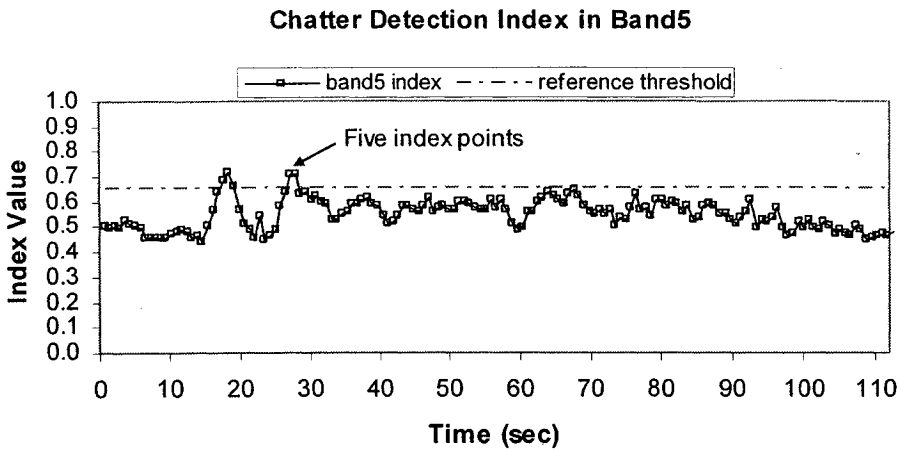
Figure 5.24 Chatter detection result of test #17 (Spindle speed 1000rpm, feed rate 100mm/min, depth of cut 3.556mm, WP2)



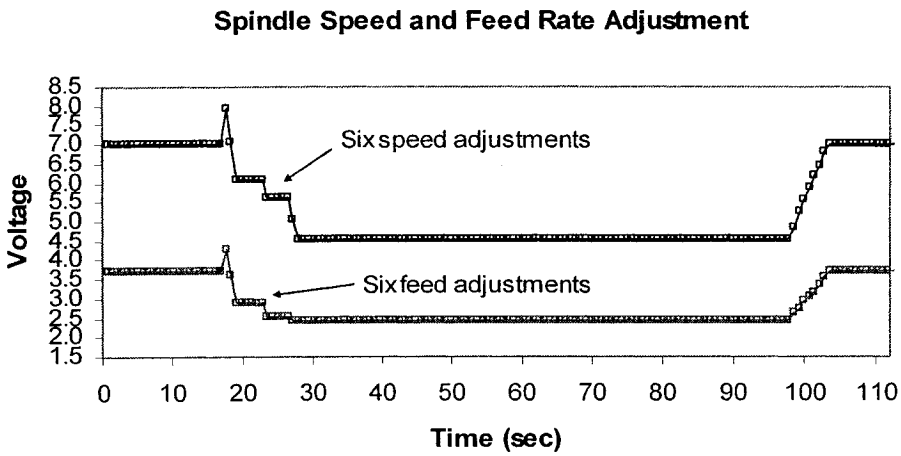
a) Time domain plot



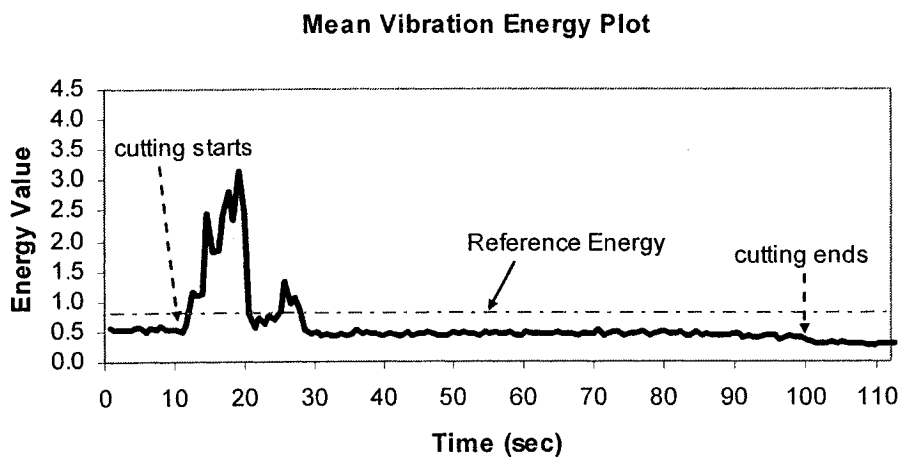
b) Band 4 chatter index plot



c) Band 5 chatter index plot

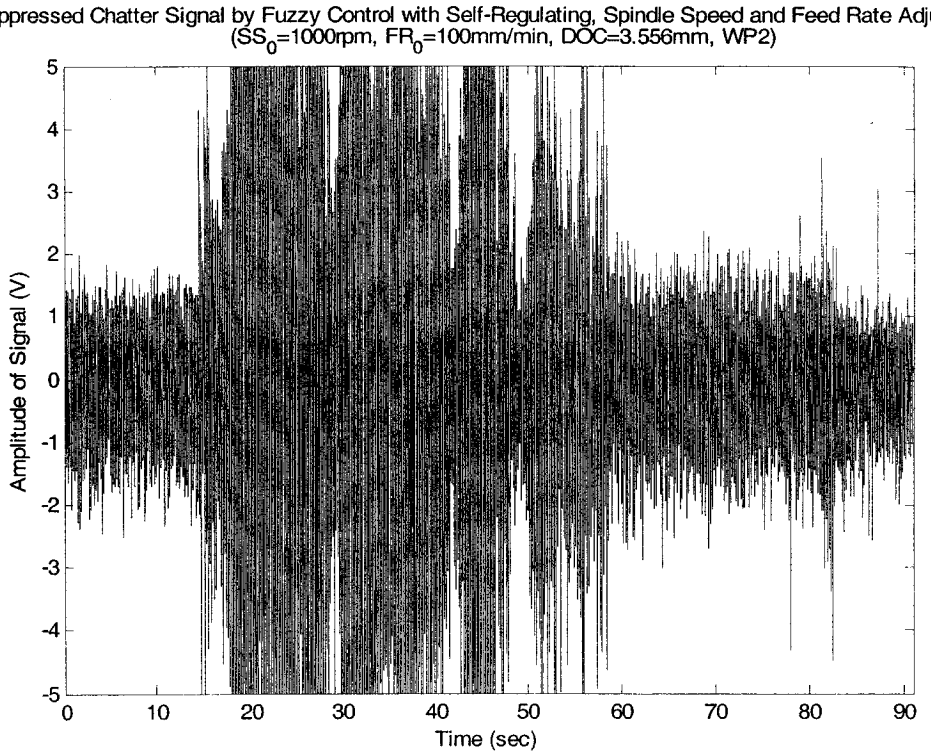


d) Spindle speed and feed rate adjustment plot

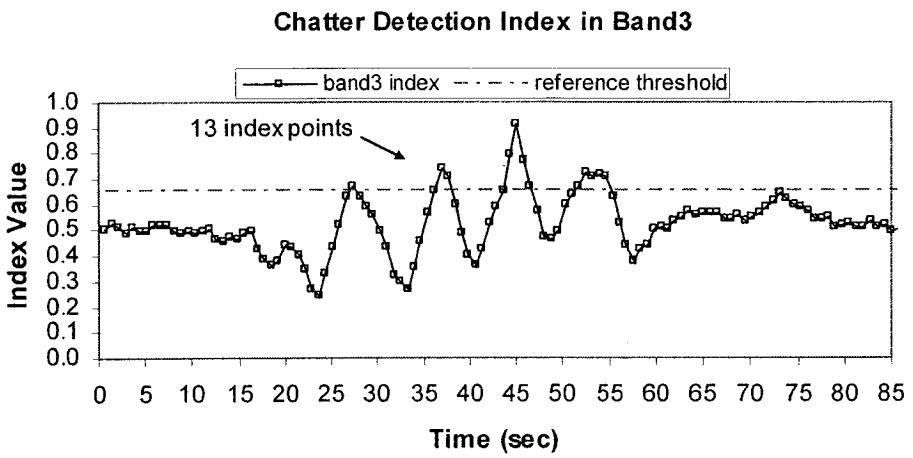


e) Mean vibration energy plot

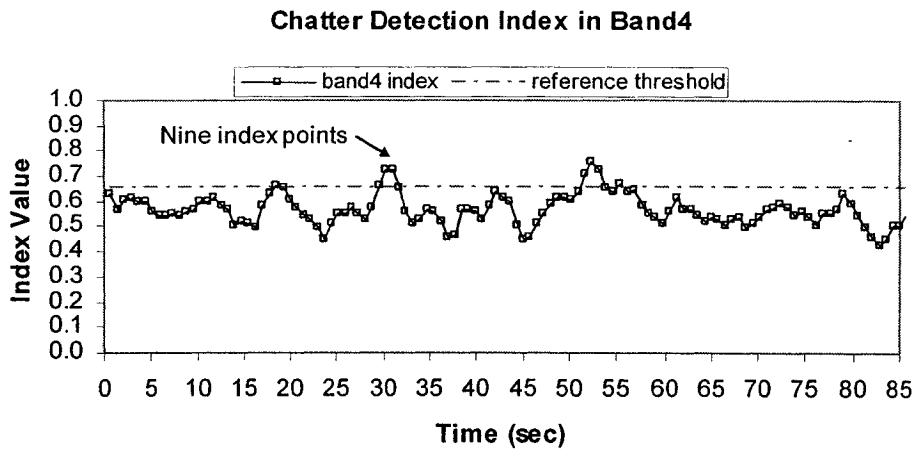
Figure 5.25 Chatter suppression result of test #18 by adjusting speed and feed (Fuzzy control only, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.556mm, WP2)



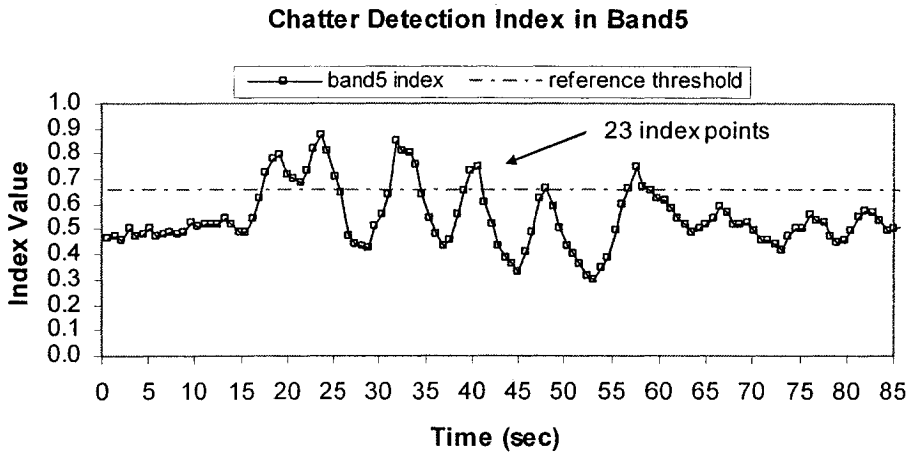
a) Time domain plot



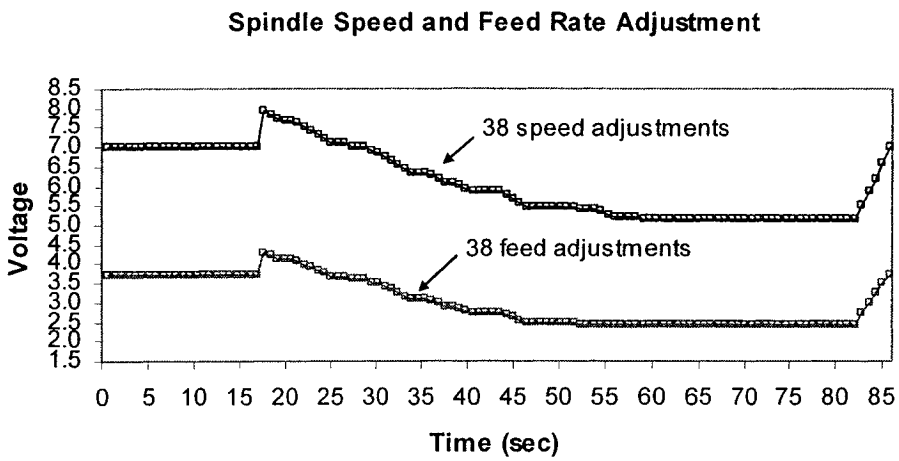
b) Band 3 chatter index plot



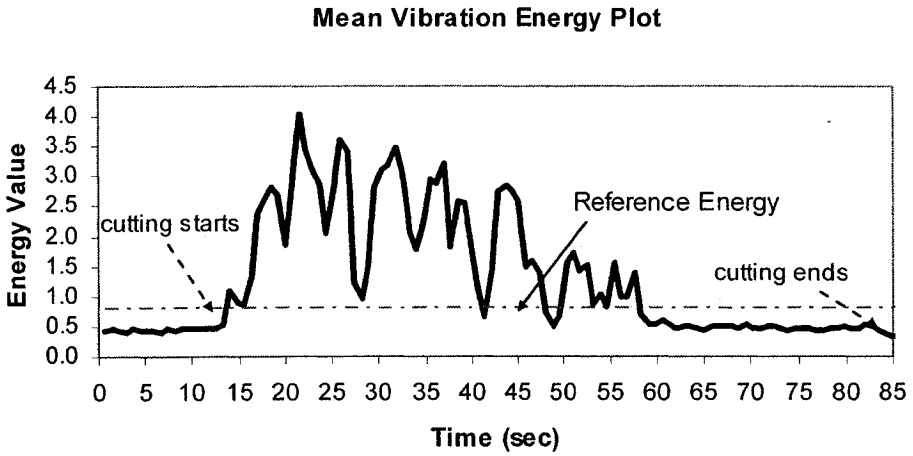
c) Band 4 chatter index



d) Band 5 chatter index plot

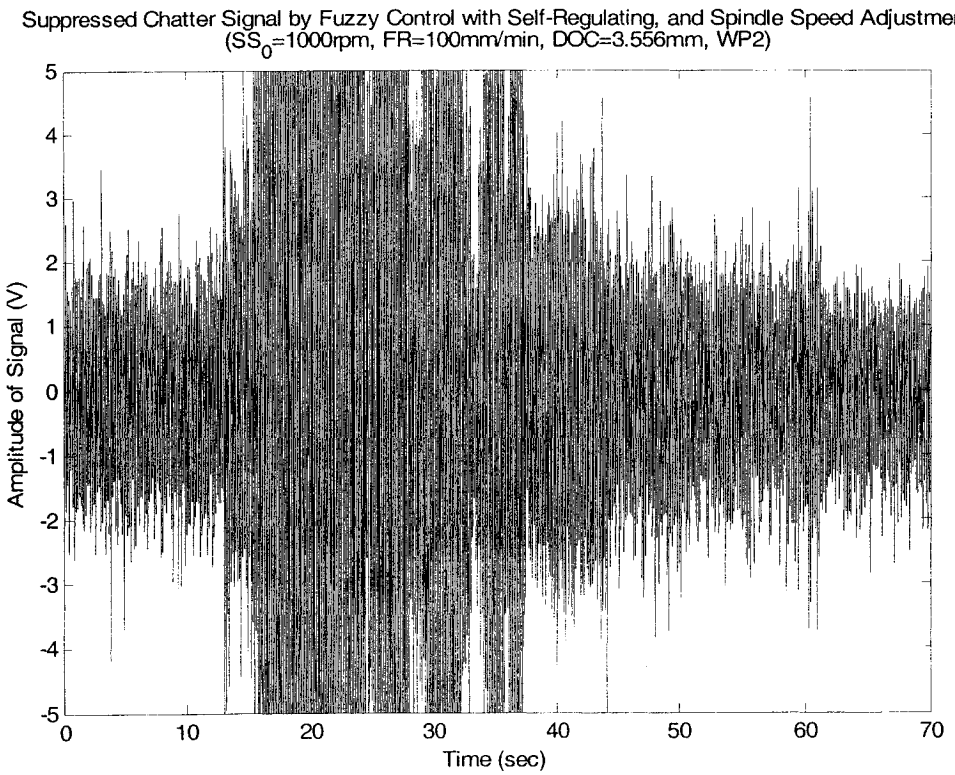


e) Spindle speed and feed rate adjustment plot

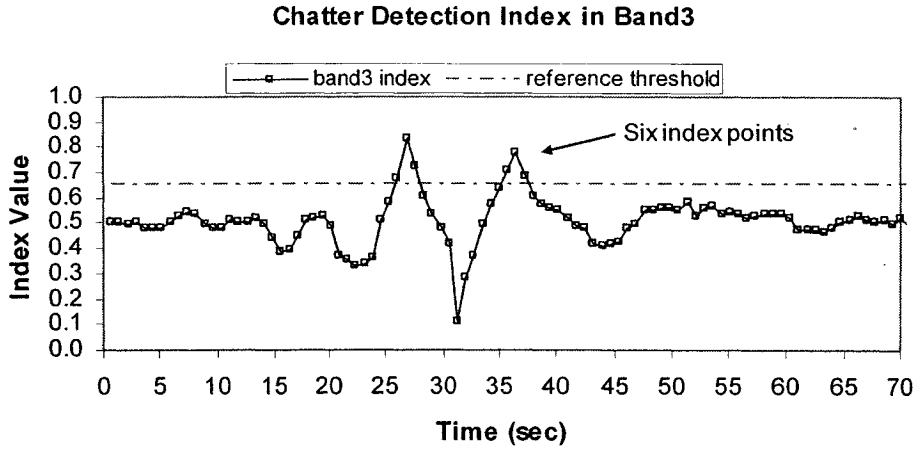


f) Mean vibration energy plot

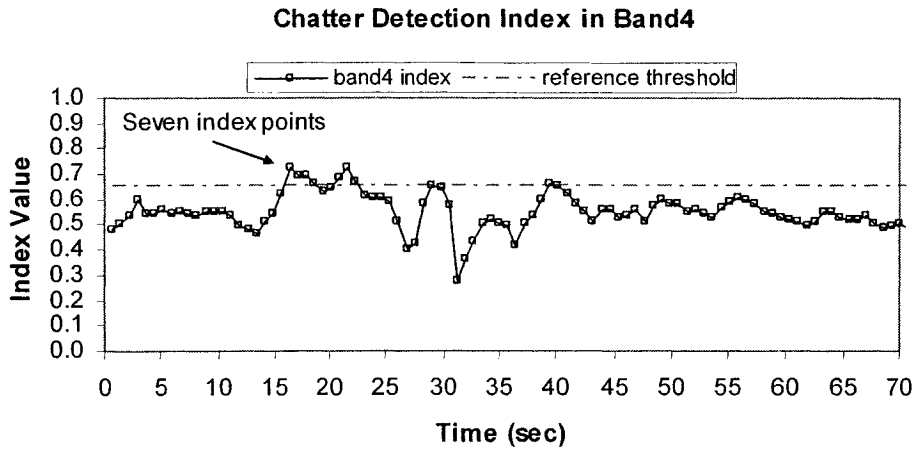
Figure 5.26 Chatter suppression result of test #19 by adjusting speed and feed (Fuzzy control with self-regulating, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.556mm, WP2)



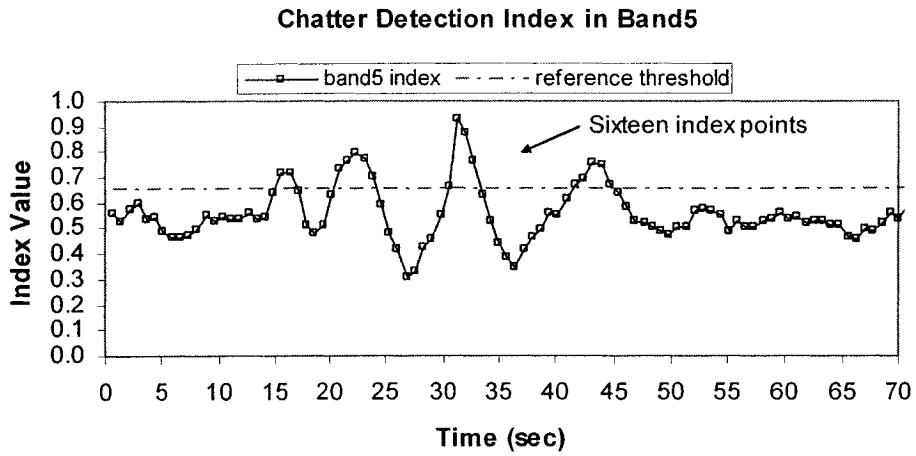
a) Time domain plot



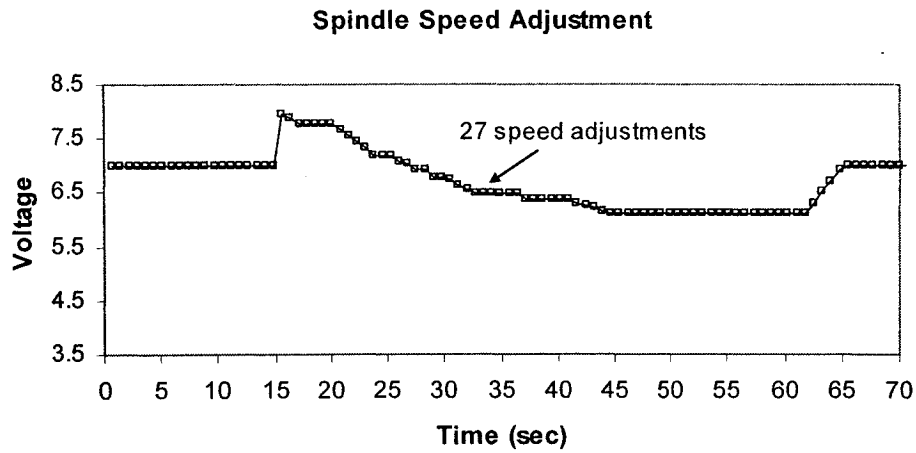
b) Band 3 chatter index plot



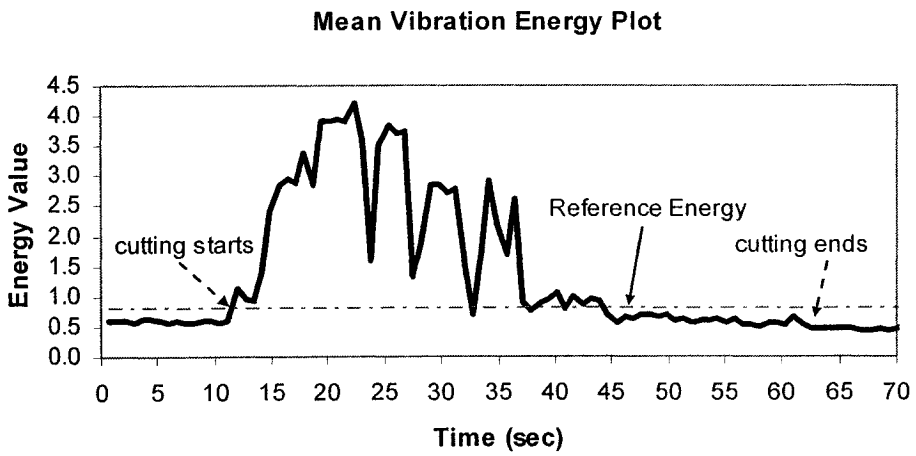
c) Band 4 chatter index plot



d) Band 5 chatter index plot

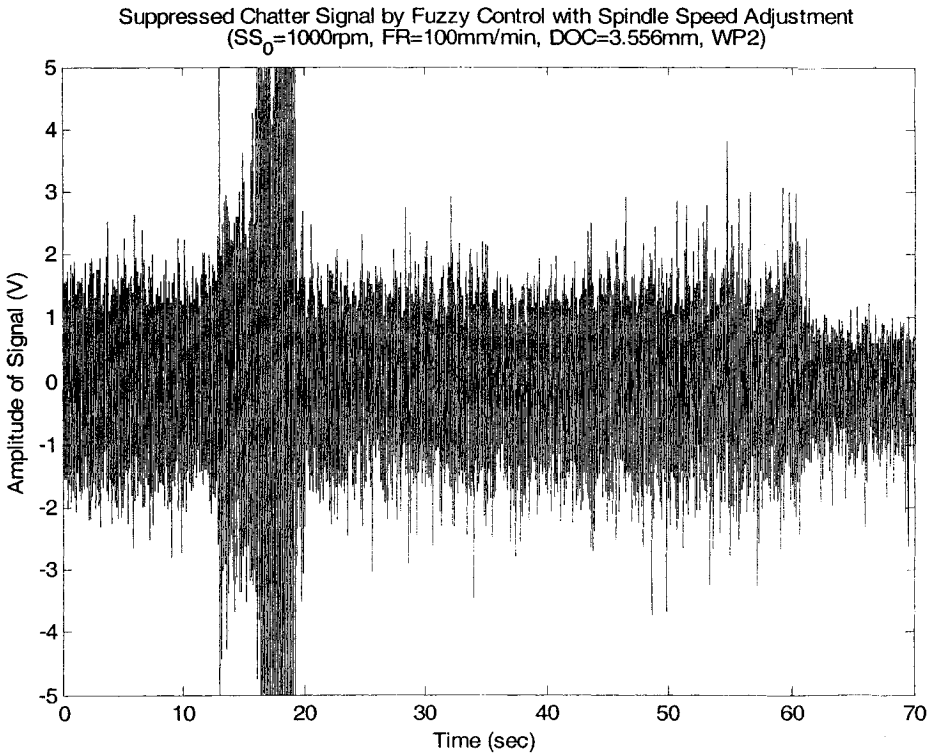


e) Spindle speed adjustment plot

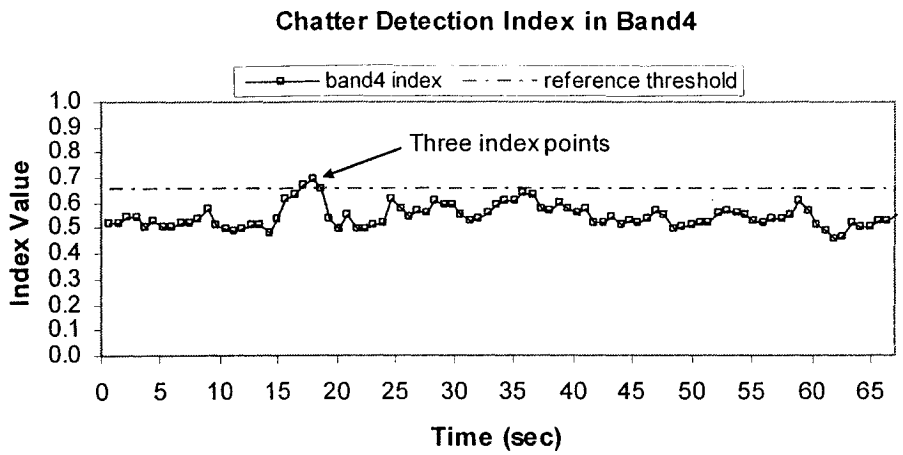


f) Mean vibration energy plot

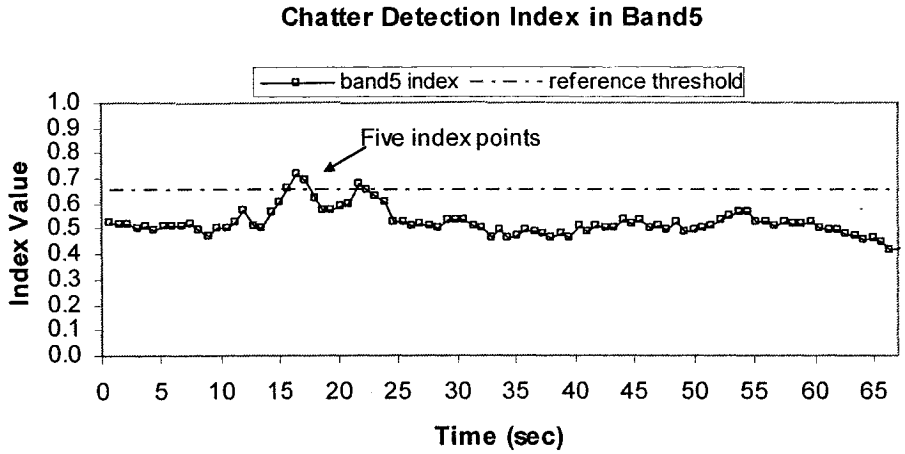
Figure 5.27 Chatter suppression result of test #20 by adjusting speed only (Fuzzy control with self-regulating, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.556mm, WP2)



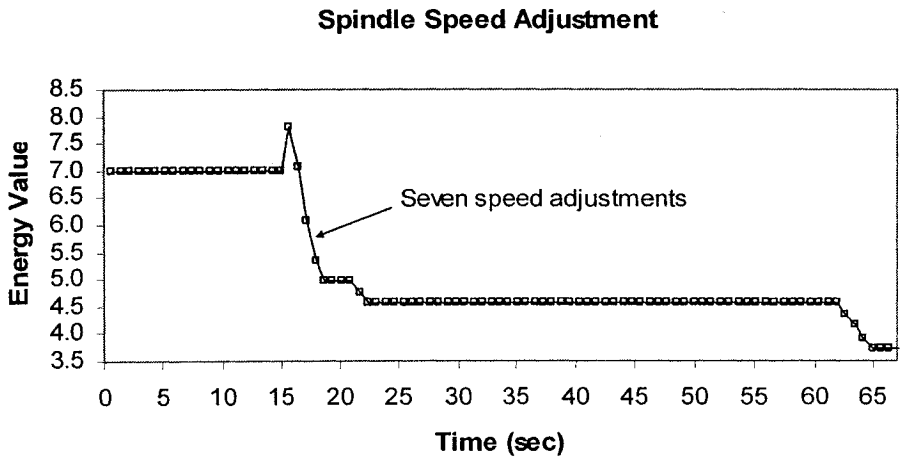
a) Time domain plot



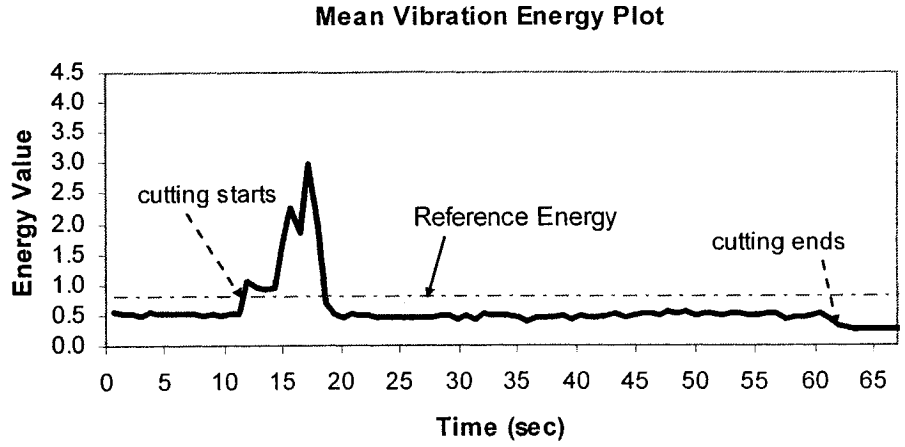
b) Band 4 chatter index plot



c) Band 5 chatter index plot



d) Spindle speed adjustment plot



e) Mean vibration energy plot

Figure 5.28 Chatter suppression result of test #21 by adjusting speed only (Fuzzy control only, initial spindle speed 1000rpm, initial feed rate 100mm/min, depth of cut 3.556mm, WP2)

Chapter 6 Conclusions and Future Research

6.1 Conclusions

In this study, a new chatter detection method using a wavelet transform and statistical analysis of wavelet transform modulus maxima is developed. The experimental results demonstrate that this new detection method can effectively detect chatter under different cutting conditions. Moreover, the severity of chatter can also be distinguished, which eliminates frequent re-calibration of chatter threshold values when cutting condition changes. Although the method is for milling process, it may also be applied to other machining processes. For chatter suppression, a fuzzy control based approach featuring two-way and multi-parameter adjustment is proposed. It does not require pre-specification of a nominal spindle speed and speed modulation parameters as needed in some spindle speed variation approaches. The effectiveness of the developed method is verified experimentally. The cutting tests conducted in this research show that the chatters detected can be suppressed and the control system is always activated whenever the index *SI* indicates the occurrence or recurrence of chatter. The use of self-regulation improves the smoothness of control process. For normal chatters, multi-parameter adjustment is more productive and effective than single-parameter adjustment.

6.2 Future research

Future studies may be carried out in the following directions:

- One enhancement of the proposed fuzzy control with self-regulating would be to seek an improved equation for adaptation gain G_p for convergence to steady state.
- The chatter suppression module may be improved to deal with non-regenerative vibrations caused by the lack of workpiece or setup rigidity.
- Re-examining the two-way adjustment to make the suppression system more productive and effective would be an important and challenging topic in the future, especially when multi-parameter adjustment is implemented for severe chatters.
- A self-learning approach by on-line modification of the fuzzy rule base could also be considered to make the fuzzy suppression module more adaptive.

- On-line chatter prediction can be investigated by conducting some modeling or learning techniques.

References

1. Al-Regib, E., Ni, J. and Lee, S. H., 2003, Programming spindle speed variation for machine tool chatter suppression. *International Journal of Machine Tools and Manufacturing*, **43** (12), pp. 1229 - 1240.

2. Altintas, Y. and Chan, P. K., 1992, In-process detection and suppression of chatter in milling. *International Journal of Machine Tools and Manufacturing*, **32** (3), pp. 329 - 347.

3. Altintas, Y., Engin, S. and Budak, E., 1999, Analytical stability prediction and design of variable pitch cutters. *ASME Journal of Manufacturing Science and Engineering*, **121** (2), pp. 173 - 178.

4. Arnold, R. N., 1946, The mechanism of tool vibration in the cutting of steel. *Proceedings of the Institution of Mechanical Engineers*, **154** (4), pp. 261 - 284.

5. Bailey, T., Ruget, Y., Spence, A. and Elbestawi, M. A., 1995, Open-architecture controller for die and mold machining. *American Control Conference*, Seattle, Washington, June 21 - 23, pp. 194 - 199.

6. Basseville, M., Benveniste, A., Chou, K. C., Golden, S. A., Nikoukhah, R. and Willsky, A. S., 1992, Modeling and estimation of multiresolution stochastic processes. *IEEE Transactions on Information Theory*, **38** (2), pp. 766 – 784.

7. Berger, B. S., Minis, I., Harley, J., Rokni, M. and Papadopoulos, M., 1998, Wavelet-based cutting state identification. *Journal of Sound and Vibration*, **213** (5), pp. 813 - 827.

8. Browne, M. and Cutmore, T.R.H., 2002, Low-probability event-detection and separation via statistical wavelet thresholding: an application to psychophysiological denoising. *Clinical Neurophysiology*, **113** (9), pp. 1403 - 1411.

9. Budak, E. and Altintas, Y., 1995a, Analytical prediction of chatter stability in milling: Part I General formulation. *Proceedings of the ASME Dynamic Systems and Control Division*, **57** (1), pp. 545 - 556.

10. Budak, E. and Altintas, Y., 1995b, Analytical prediction of chatter stability in milling: Part II Application of the general formulation to common milling systems. *Proceedings of the ASME Dynamic Systems and Control Division*, **57** (1), pp. 557 - 565.

11. Choi, T. and Shin, Y. C., 2003, On-Line chatter detection using wavelet-based parameter estimation. *ASME Journal of Manufacturing Science and Engineering*, **125** (1), pp. 21 - 28.

12. Choudhury, S. K. and Mathew, J., 1995, Investigations of the effect of non-uniform insert pitch on vibration during face milling. *International Journal of Machine Tools & Manufacturing*, **35** (10), pp. 1435 -1444.

13. Choudhury, S. K., Goudimenko, N. N. and Kudinov, V. A., 1997, On-line control of machine tool vibration in turning. *International Journal of Machine Tools & Manufacturing*, **37** (6), pp. 801-811.

14. Daubechies, I., 1992, *Ten Lectures on Wavelets*, (Philadelphia: Society for Industrial and Applied Mathematics).

15. Debnath, L., 2003, *Wavelets and Signal processing*, (Boston: Birkhauser).

16. Delio, T., Tlusty, J. and Smith, S., 1992, Use of audio signals for chatter detection and control. *ASME Journal of Engineering for Industry*, **114** (2), pp. 146 - 157.

17. Dong, W., Au, Y. and Mardapittas, A., 1992, Machine tool chatter monitoring by coherence analysis, *International Journal of Production Research*, **30** (8), pp. 1901 - 1924.

18. Donoho, D. L., 1995, De-noising by soft-thresholding. *IEEE Trans. Inf. Theory*, **41** (3), pp. 613 - 627.

19. Du, R., Elbestawi, M. A. and Li, S., 1992, Tool condition monitoring in turning using fuzzy set theory. *International Journal of Machine Tools and Manufacturing*, **32** (6), pp. 781 - 796.

20. Elbestawi, M. A., Ismail, F., Du, R. and Ullagaddi, B. C., 1994, Modeling machining dynamics including damping in the tool-workpiece interface. *ASME Journal of Engineering for Industry*, **116** (4), pp. 435 - 439.

21. Endres, W. J., 1996a, A quantitative energy-based method for predicting stability limit as a direct function of spindle speed for high-speed machining. *Transaction of the NAMRI/SME*, **24**, pp. 27 - 32.

22. Endres, W. J., 1996b, The effect of uncut chip thickness nonlinearity and linear process gain calculation on machining stability analysis. *Proceedings of the Manufacturing Engineering Division, ASME International Mechanical Engineering Conference and Exposition*, Atlanta, Georgia, Nov., pp. 17 -22.

23. Erlebacher, G., Hussaini, M. Y, and Jameson, M. L., 1996, *Wavelet: Theory and Applications*, (Oxford: Oxford University Press).

24. Grabec, I., 1988, Chaotic dynamics of the cutting process. *International Journal of Machine Tools and Manufacturing*, **28** (1), pp. 19 - 32.

25. Gradisek, J., Govekar, E. and Grabec, I., 1998, Using coarse-grained entropy rate to detect chatter in cutting. *Journal of Sound and Vibration*, **214** (5), pp. 941 - 952.

26. Gu, S., Ni, J. and Yuan, J., 2001, Non-stationary signal analysis and transient machining process condition monitoring. *International Journal of Machine Tools & Manufacturing*, **42** (1), pp. 41-51.

27. Gupta, S., Chauhan, R. C. and Sexana, S. C., 2004, Wavelet-based statistical approach for speckle reduction in medical ultrasound images. *Medical & Biological Engineering & Computing*, **42** (2), pp. 189 - 192.

28. Hazem, N. N. and Kevin, M. P., 2004, Stable auto-tuning of adaptive fuzzy/neural controller for nonlinear discrete-time systems. *IEEE Transaction on Fuzzy Systems*, **12** (1), pp. 70 - 83.

29. Hermansyah, A., 2000, *Intelligent Suppression of Chatter in End Milling Processes,* Master Thesis, Department of Mechanical Engineering, University of Ottawa, Canada.

30. Hsu, P. L. and Fann, W. R., 1996, Fuzzy adaptive control of machining processes with a self-learning algorithm. *ASME Journal of Manufacturing Science and Engineering*, **118** (1), pp. 522 - 530.

31. Ismail, F. and Kubica, E. G., 1995, Active suppression of chatter in peripheral milling, Part 1: A statistical indicator to evaluate the spindle speed modulation method. *International Journal of Advanced Manufacturing Technology*, **10** (5), pp. 299 - 310.

32. Jan, J., 1998, *The self-organizing fuzzy controller*, Tech. report no 98-H 869, Department of Automation, Technical University of Denmark, Denmark.

33. Jansen, M., 2001, *Noise Reduction by Wavelet Thresholding*, (New York: Springer-Verlag).

34. Khraisheh, M. K., Pezeshki, C. and Bayoumi, A. E., 1995, Time series-based nalysis for primary chatter in metal cutting. *Journal of Sound and Vibration*, **180** (1), pp. 67 - 87.

35. Koenigsberger, F. and Tlusty, J., 1967, *Machine Tools Structures - Vol. I: Stability against Chatter*, (London: Pergamon Press).

36. Landers, R. G. and Ulsoy, A. G., 1996, Chatter analysis of machining systems with nonlinear force processes. *ASME International Mechanical Engineering Congress and Exposition*, Atlanta, Georgia, November 17 - 22, DSC - Vol. 58, pp. 183 - 190.

37. Lee, A. C. and Liu, C. S., 1991, Analysis of chatter vibration in the end milling process. *International Journal of Machine Tools and Manufacturing*, **31** (4), pp. 471 - 479.

38. Lee, J. M. and Hwang, Y., 2000, Diagnosis of machine signal using hidden markov model. *KSME*, Summer Conference, pp. 230 - 236.

39. Li, T. H. and Oh, H. S., 2002, Wavelet spectrum and its characterization property for random processes. *IEEE Transactions on Information Theory*, **48** (11), pp. 2922 - 2936.

40. Li, X. Q., Wong, Y. S. and Nee, A. Y. C, 1997, Tool wear and chatter detection using the coherence function of two crossed accelerations. *International Journal of Machine Tools and Manufacturing*, **37** (4), pp. 425 - 435.

41. Li, X. Q., Wong, Y. S. and Nee, A. Y. C, 1998, A comprehensive identification of tool failure and chatter using a parallel multi-ART2 neural network. *ASME Journal of Manufacturing Science and Engineering*, **120** (2), pp. 433 - 442.

42. Liang, M., Yeap, T., Hermansyah, A. and Rahmati, S., 2003, Fuzzy control of spindle torque for industrial CNC machining. *International Journal of Machine & Manufacture*, **43** (14), pp. 1497 - 1508.

43. Liang, M., Yeap, T. and Hermansyah, A., 2004, A fuzzy system for chatter suppression in end milling. *Proceedings of the Institution of Mechanical Engineers - Part B: Journal of Engineering Manufacture*, **128**, pp. 403 - 417.

44. Lin, S. C., Devor, R. E. and Kapoor, S. G., 1990, The effects of variable speed cutting on vibration control in face milling. *ASME Journal of Engineering for Industry*, **112** (1), pp. 1 - 11.

45. Luettgen, M. R., Karl, W. C., Willsky, A. S. and Tenney, R. R., 1993, Multiscale representations of markov random fields. *IEEE Transactions on Signal Processing*, **41** (12), pp. 3377 - 3396.

46. Mallat, S. G., 1998, *A Wavelet Tour of Signal Processing*, (London: Academic Press Limited).

47. Mallat, S. G., 1989, A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **2** (7), pp. 674 - 693.

48. Mallat, S. and Hwang, W. L., 1992, Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, **38** (2), pp. 617 - 643.

49. Meng, S., 1996, *Handbook of Metal Cutting Processes*, (Beijing: Machinery Industry Press).

50. Merritt, H. E., 1965, Theory of self-excited machine tool chatter: Contribution to machine-tool chatter research - 1. *ASME Journal of Engineering for Industry*, **87** (4), pp. 447 - 454.

51. Mihcak, M., Kivanc, K. I. and Ramchandran, K., 1999, Low-complexity image denoising based on statistical modeling of wavelet coefficients. *IEEE Signal Processing Letters*, **6** (12), pp. 300 - 303.

52. Minis, I., Magrab, E. and Pandelidis, I., 1990a, Improved methods for the prediction of chatter in turning, Part III: A generalized linear theory. *ASME Journal of Engineering for Industry*, **112** (1), pp. 28 - 35.

53. Minis, I., Yanushevsky, R. and Tembo, A., 1990b, Analysis of linear and nonlinear chatter in milling. *Annals of CIRP*, **39** (1), pp. 459 - 462.

54. Minis, I. and Yanushevsky, R., 1993, A new theoretical approach for the prediction of machine tool chatter in milling. *ASME Journal of Engineering for Industry*, **115** (1), pp. 1 - 8.

55. Newland, D. E., 1984, *An Introduction to Random Vibration and Spectral Analysis*, 2nd ed., (England: Longman Group Limited).

56. Pan, G., Xu, H., Kwan, C. M., Liang, C., Haynes, L. and Geng, Z., 1996, Modeling and intelligent chatter control strategies for a lathe machine. *Control Engineering Practice*, **4** (12), pp. 1647 - 1658.

57. Radulescu, R., Kapoor, S. G. and Devor, R. E., 1997, An investigation of variable spindle speed face milling for tool-work structures with complex dynamics, Part 1: Simulation results and Part 2: Physical explanation. *ASME Journal of Manufacturing Science and Engineering*, **119** (3), pp. 266 - 280.

58. Rahman, M., 1988, In-process detection of chatter threshold. *ASME Journal of Engineering for Industry*, **110** (1), pp. 44 - 50.

59. Rodolfo, E. H., Clodeinir, R. P., Angel, A., Salvador, R., Carlos, G. and Jose, R. A., 1998, Toward intelligent machining: Hierarchical fuzzy control for the end milling process. *IEEE Transaction on Control Systems Technology*, **6** (2), pp. 188 – 199.

60. Sato, H. and Hori, M. O., 1981, Characteristics of two-dimensional surface roughness - Taking self excited chatter marks as objective. *Annals of CIRP*, **30** (1), pp. 481 - 486.

61. Shiraishi, M., Kume, E. and Hoshi, T., 1988, Suppression of machine-tool chatter by state feed-back control. *Annals of the CIRP*, **37** (1), pp. 369 - 372.

62. Shiraishi, M., Yamanaka, K. and Fujita, H., 1991, Optimal control of chatter in turning. *International Journal of Machine Tools and manufacturing*, **31** (1), pp. 31 - 43.

63. Slavicek, J., 1965, The effect of irregular tooth pitch on stability of milling. *Proceeding of the 6th MTDR Conference*, (London: Pergamon Press).

64. Smith, S. and Tlusty, J., 1992, Stabilizing chatter by automatic spindle speed regulation. *Annals of the CIRP*, **41** (1), pp. 433 - 436.

65. Smith, S. and Tlusty, J., 1993, Efficient simulation programs for chatter in milling. *Annals of the CIRP*, **42** (1), pp. 463 - 466.

66. Sridhar, R., Hohn, R. E. and Lang, G. W., 1968, A stability algorithm for the general milling process: Contribution to machine tool chatter research -7. *ASME Journal of Engineering for Industry*, **90** (2), pp. 330 - 334.

67. Soliman, S. and Ismail, F., 1997, Chatter suppression by adaptive speed modulation. *International Journal of Machine Tools and Manufacturing*, **37** (3), pp. 355 - 369.

68. Stone, B. J., 1970, The effect on the chatter behavior of machine tools of cutters with different helix angles on adjacent teeth. *Advance in Machine Tool Design and Research, Proceedings of 11th International MTDR Conference*, University of Birmingham, England, Vol. A, pp. 169 - 180.

69. Subramanian, T. L., 1976, An investigation of computer control of machine chatter. *ASME Journal of Engineering for Industry*, **98** (4), pp. 1209 - 1214.

70. Tarng, Y., Hseih, Y. and Li, T., 1996, Automatic selection of spindle speed for suppression of regenerative chatter in turning. *International Journal of Advanced Manufacturing Technology*, **11** (1), pp. 12 - 17.

71. Tarng, Y. S. and Li, T. C., 1994, Detection and suppression of drilling chatter. *ASME Journal of Dynamic Systems, Measurement, and Control*, **116** (4), pp. 729 - 734.

72. Tewani, S. G., Rouch, K. E. and Walcott, B. L., 1995, A study of cutting process stability of a boring bar with active dynamic absorber. *International Journal of Machine Tools & Manufacturing*, **35** (1), pp. 91 - 108.

73. Tlusty, J., 1986, Dynamics of high-speed milling. *ASME Journal of Engineering for Industry*, **108** (2), pp. 59 - 67.

74. Tlusty, J. and Ismail, F., 1981, Basic nonlinearity in machining chatter. *Annals of the CIRP*, **30** (1), pp. 299 - 304.

75. Tlusty, J. and Ismail, F., 1983, Special aspects of chatter in milling. *ASME Journal of Vibration, Acoustics, Stress, and Reliability in Design*, **105** (1), pp. 24 - 32.

76. Tlusty, J. and Polacek, M., 1963, The stability of machine tools against self excited vibrations in machining. *ASME Proceedings of the International Research in Production Engineering Conference*, Pittsburgh, PA, pp. 465 - 474.

77. Tobias, S. A. and Fishwick, W., 1958, *A Theory of Regenerative Chatter*, (London: The Engineer).

78. Torrence, C. and Compo, G. P., 1998, A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, **79** (1), pp. 61 - 79.

79. Tsai, M. D., Takata, S., Inui, M., Kimura, F. and Sata, T., 1990, Prediction of chatter vibration by means of a model-based cutting simulation system. *Annals of the CIRP*, **39** (1), pp. 447 - 450.

80. Vidakovic, B. and Guy, B., 2000, Statistical modeling by wavelets. *Society for Industrial and Applied Mathematics Review*, **42** (3), pp. 522 - 523.

81. Vnaherck, P., 1967, Increasing milling machine productivity by use of cutter with non-constant cutting-edge pitch. *Proceeding of the 6th MTDE Conference*, Vol. 9, pp. 947 - 960.

82. Walker, J. S., 1999, *A Primer on Wavelets and Their Scientific Applications*, (Boca Raton: Chapman & Hall/CRC).

83. Wang, J. H. and Lee, K. N., 1996, Suppression of chatter vibration of a CNC machine centre - An example. *Mechanical Systems and Signal Processing*, **10** (5), pp. 551 - 560.

84. Weck, M., Altintas, Y. and Beer, C., 1994, CAD assisted chatter-free NC tool path generation in milling. *International Journal of Machine Tools and Manufacturing*, **34** (6), pp. 879 - 891.

85. Weck, M., Verhang, E. and Gather, M., 1975, Adaptive control for face-milling operations with strategies for avoiding chatter vibration and for automatic cut distribution. *Annals of the CIRP*, **24** (1), pp. 405 - 409.

86. Williams, H. R., 2003, *Probability, Statistics, and Random Processes for Engineers*, (Pacific Grove: Thomson Learning).

87. Hines, W. W., Montgomery, D. C., Goldsman, D. M. and Borror, C. M., 2003, *Probability and Statistics in Engineering*, 4th ed., (Hooboken: John Wiley & Sons, Inc.).

88. Wu, Y. and Du, R., 1996, Feature extraction and assessment using wavelet packets for monitoring of machining processes. *Journal of Mechanical Systems and Signal Processing*, **10** (1), pp. 29-53.

89. Xiao, M., Karube, S., Soutome, T. and Sato, K., 2002, Analysis of chatter suppression in vibration cutting. *International Journal of Machine Tools and Manufacturing*, **42** (15), pp. 1677 - 1685.

90. Xu, D., 2002, *A Fuzzy Logic Approach for Chatter Detection and Suppression in End Milling*, Master Thesis, Department of Mechanical Engineering, University of Ottawa, Canada.

91. Yamazaki, T., 1982, *An Improved Algorithm for a Self-Organizing Controller and its Experimental Analysis*, PhD thesis, Dept. of Electrical and Electronic Engineering, Queen Mary College, London.

92. Yang, F. L., Zhang, B. and Yu, J. Y., 1997, Chatter suppression through time-varying feedrate in cutting process. *Transaction of NAMRI/SME*, **25**, pp. 87 - 92.

93. Yilmaz, A., Al-Regib, E. and Ni, J., 2002, Machine-tool chatter suppression by multi-level random spindle speed variation. *ASME Journal of Manufacturing Science and Engineering*, **124** (2), pp. 208 - 216.

94. Zhang, H., Ni, J. and Shi, H., 1994, Machining chatter suppression by means of spindle speed variation - Part I: The numerical solution and Part II: Experimental investigation. *S.M. Wu Symposium on Manufacturing Science*, Vol. 1, Evanston, Illinois, May 27 -28, pp. 161 - 175.

95. Zhang, H. and Ni, J., 1995, Phase difference and its sensitivity analysis for a nonlinear difference-differential machining chatter model. *Transaction of the NAMRI/SME*, **23**, pp. 131 - 136.

96. Zhao, R., 1992, *Handbook for Machinists*, 3rd edition, (Shanghai: Shanghai Science and Technology Press).

Appendices

```
/*****************************************************************************/
/* LabWindows™/CVI 5.0 C Program for on-line chatter detection and suppression */
/* in CNC end milling process                                                */
/* Developed at Computer Integrated Manufacturing Lab at University of Ottawa */
/* Date: July 31, 2005                                                       */
/*****************************************************************************/

#include <dataacq.h>
#include <utility.h>
#include <cvirte.h>
#include <userint.h>
#include <ansi_c.h>
#include <analysis.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include <stdlib.h>
#include "lowlvlio.h"
#include "daq.h"
#include "NIDAQEX.H"
#include <formatio.h>
#include "wfbd.h"

#include "four1.c"
#include "realft.c"
#include "NIDAQEX.C"
#include "fuzzyeng.c"

#define OutOfMemErr     -20001
#define BUFF_SIZE       1024
#define TRUE            1
#define FALSE   0
#define PI      3.1416
#define LOOP_LEN        4
#define DATA_POINTS  2048
#define NUM_OF_ELEMENTS    512

double sign(double);
int round(double);
void denoising(intnum, floatnum*, double);
```

```
static int daqpanel;
static int childpanel;
double  FeedMaxVoltage = 5.0;   // maximal feed voltage corresponding to 150mm/min
double  FeedMinVoltage = 1.5;   // minimum feed voltage corresponding to 50mm/min
double  FeedOverride = 1.5;     // feed rate reading at 1.5volt = 0% feed override
double  SpeedMaxVoltage = 8.5;  // maximal speed voltage corresponding to 1500rpm
double  SpeedMinVoltage = 3.5;  // minimum speed voltage corresponding to 500rpm
double  SpeedOverride = 3.5;    // speed voltage reading at 3.5volt=500rpm override
double  E_r = 0.8;              // reference vibration energy level
double  E_min = 0.5;            // minimum vibration energy level
double  E_max = 5.5;            // maximum vibration energy level
unsigned int  CyclesPerSec = 14; // cycling rate 14, i.e., 14 control cycles per second
double speed_k, feed_k;         // control variables of spindle speed and feed rate
FIS     *fis;
FIS     *fis1;
FIS     *fis_1;
FIS     *fis1_1;
double datamatrix[3];
int fis_row_n, fis_col_n;
int fis_row_n_1, fis_col_n_1;
double *tmp = NULL;
double *tmp1 = NULL;
double *tmp2 = NULL;
double *tmp3 = NULL;
double **fisMatrix = NULL;
double **fisMatrix1 = NULL;
double **fisMatrix_1 = NULL;
double **fisMatrix1_1 = NULL;
char *fis_file = NULL;
char *fis_file1 = NULL;
char *fis_file_1 = NULL;
char *fis_file1_1 = NULL;
double Ke, delta_Ke;    // input scaling factors for energy error and change of energy error
double yita;            // target time constant
double T_s;             // sampling period
double GU_ss, GU_fr;    // output scaling factor in terms of speed or feed adjustment
short iNumMUXBrds = 0;
short iDAQstopped = 0;
unsigned long ulRetrieved = 0;
short iIgnoreWarning = 0;
double dGainAdjust = 1.0;
double dOffset = 0.0;
static short piBuffer[2048] = {0};  // set the size of double buffer for collecting sensor data
static short piHalfBuffer[1024] = {0};  // set the half size of double buffer
static double pdVoltBuffer[1024] = {0.0}; // set the size of half buffer after ADC conversion
static double datapoints_x[512] = {0.0};    // set the array size for holding the sample data
short iStatus = 0;                          // status check of DAQ
```

```
short iRetVal = 0;
short iDevice = 1;                          // device number
double dSampRate = 2000.0;                  // sampling rate
double dScanRate = 2000.0;                  // scan rate
short iSampTB = 3;                          // sampling time base
static unsigned short uSampInt = 0;         // time interval between samples
short iScanTB = 3;                          // scan time base
static unsigned short uScanInt = 0;         // time interval between scans
short iNumChans = 2;                        // number of channels
static short piChanVect[2] = {5, 6};        // channel array
static short piGainVect[2] = {100, 100};    // sensor signal gain
short iHalfReady;
unsigned long ulPtsTfr;
unsigned long ulCount = 1024;
short iUnits = 0;
int iDBmode=1;                              // set the double buffer mode
short iChan = 2;
short iGain = 1;                            // 1 corresponds to -5V ~ 5V signal range
intnum err,nx,ny0,ny1,nx0,ny00,ny01,ny000,ny001,ny0000,ny0001,nx00,nx000,nx0000;
floatnum *y0 = 0, *y1 = 0, *y00 = 0, *y01 = 0, *y000 = 0, *y001 = 0, *y0000 = 0,
        *y0001 = 0;
floatnum *y11 = 0, *y011 = 0, *y0011 = 0, *y00011 = 0;
floatnum *x = 0, *x0 = 0, *x00 = 0, *x000 = 0, *x0000 = 0;
floatnum *wave_cross_power = 0, *wave_cross_power1 = 0, *wave_cross_power2 = 0,
        *wave_cross_power3 = 0, *wave_cross_power4 = 0;
floatnum *win_wave_coeff = 0, *win_wave_coeff1 = 0, *win_wave_coeff2 = 0,
        *win_wave_coeff3 = 0, *win_wave_coeff4 = 0;
FilterBankPtr anaptr = NULL;
FilterBankPtr synptr = NULL;
int control_ID = 1000;
int control_value = 0;
int toggle = FALSE;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/************************** Main Function Routine **************************/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int main (int argc, char *argv[])
{
  char *error_msg;

/* spindle speed and feed rate initilizations */
  AO_VWrite (1, 0, 7.0);          // the reading of 7.0V corresponds to 1000rpm speed
  AO_Configure (1, 0, 0, 0, 10.0, 0);
  error_msg = GetNIDAQErrorString (speed_k);
  speed_k = 7.0;
  AO_VWrite(1, 1, 3.71);
```

```
    feed_k = 3.71;          // the reading of 3.71V corresponds to feed rate=100 mm/min
    remove("C:\\WINDOWS\\Desktop\\chatter\\exprt_data.txt");

/* build up the fuzzy structure */
    fis_file = ("C:\\WINDOWS\\Desktop\\chatter\\995-88d.fis");
    fis_file1 = ("C:\\WINDOWS\\Desktop\\chatter\\995-88d1.fis");
    fis_file_1 = ("C:\\WINDOWS\\Desktop\\chatter\\995-88d-1.fis");
    fis_file1_1 = ("C:\\WINDOWS\\Desktop\\chatter\\995-88d1-1.fis");
    fisMatrix = returnFismatrix(fis_file, &fis_row_n, &fis_col_n);
    fisMatrix1 = returnFismatrix(fis_file1, &fis_row_n, &fis_col_n);
    fisMatrix_1 = returnFismatrix(fis_file_1, &fis_row_n_1, &fis_col_n_1);
    fisMatrix1_1 = returnFismatrix(fis_file1_1, &fis_row_n_1, &fis_col_n_1);
    fis = (FIS *)malloc(1*sizeof(FIS));
    fis1 = (FIS *)malloc(1*sizeof(FIS));
    fis_1 = (FIS *)malloc(1*sizeof(FIS));
    fis1_1 = (FIS *)malloc(1*sizeof(FIS));
    fisBuildFisNode(fis, fisMatrix, fis_col_n);
    fisBuildFisNode(fis1, fisMatrix1, fis_col_n);
    fisBuildFisNode(fis_1, fisMatrix_1, fis_col_n_1);
    fisBuildFisNode(fis1_1, fisMatrix1_1, fis_col_n_1);

/* set sampling rate and scan rate */
    iStatus = DAQ_Rate(dSampRate, iUnits, &iSampTB, &uSampInt);
    iStatus = DAQ_Rate(dScanRate, iUnits, &iScanTB, &uScanInt);
    iStatus = DAQ_DB_Config(iDevice, iDBmode);
    iStatus = SCAN_Setup(iDevice, iNumChans, piChanVect, piGainVect);
    iRetVal = NIDAQErrorHandler(iStatus, "SCAN_Setup", iIgnoreWarning);

    if (InitCVIRTE (0, argv, 0) == 0)       // needed if linking in external compiler
                return -1;                  // out of memory
    if ((daqpanel = LoadPanel (0, "daq.uir", DAQPANEL)) < 0)
                return -1;
    if ((childpanel = LoadPanel (daqpanel, "daq.uir", CHILDPANEL)) <0 )
                return -1;

    DisplayPanel (daqpanel);
    RunUserInterface ();
    return 0;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/******************** Initialize panel and start data acquisition ********************/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int CVICALLBACK AcquireData (int panel, int control, int event, void *callbackData, int
                                    eventData1, int eventData2)
```

```
{
        switch (event)
          {
          case EVENT_COMMIT:
                    GetCtrlVal (daqpanel, DAQPANEL_SAMPLERATE, &dSampRate);
                    GetCtrlVal (daqpanel, DAQPANEL_NUMPOINTS, &ulCount);

/* samples data acquisition from sensor starts */
                        uSampInt = 70;
                        uScanInt = 140;
                        iScanTB = 2;
                        iSampTB = 2;
                        iStatus = SCAN_Start(iDevice, piBuffer, DATA_POINTS, iSampTB,
                                                              uSampInt, iScanTB, uScanInt);
                        iRetVal = NIDAQErrorHandler(iStatus, "SCAN_Start",
                                                                    iIgnoreWarning);

/* dim the buttons on the parent panel */
                        SetCtrlAttribute (daqpanel, DAQPANEL_ACQUIRE,
                                                                    ATTR_DIMMED, 1);
                        SetCtrlAttribute (daqpanel, DAQPANEL_SAMPLERATE,
                                                                    ATTR_DIMMED, 1);
                        SetCtrlAttribute (daqpanel, DAQPANEL_NUMPOINTS,
                                                                    ATTR_DIMMED, 1);
                        SetCtrlAttribute (daqpanel, DAQPANEL_TIMER,
                                                                  ATTR_ENABLED, 1);
                        break;

                case EVENT_RIGHT_CLICK:
                        break;
                    }
        return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/******************** Chatter detection and suppression function ******************/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int CVICALLBACK ChatterDetection (int panel, int control, int event,void *callbackData,
                                                    int eventData1, int eventData2)
{
  floatnum *wtmm_tmp = 0, *wtmm_tmp1= 0, *wtmm_tmp2 = 0, *wtmm_tmp3 = 0,
           *wtmm_tmp4=0;
  floatnum *wtmm_cdf = 0, *wtmm_cdf1 = 0, *wtmm_cdf2 = 0, *wtmm_cdf3 = 0,
           *wtmm_cdf4 = 0;
  floatnum *wtmm = 0, *wtmm1 = 0, *wtmm2 = 0, *wtmm3 = 0, *wtmm4 = 0;
  int BUTTON[5]={0,0,0,0,0};
```

```
int i,j,fd;
double thr[5] = {0.0};              // wavelet de-noising threshold array
double std1[5] = {0.0};             // estimated standard deviation of noise
double slope[5] = {0.0};            // the chatter index value for each frequency band
double freq[256] = {0.0};

char path[] = "C:\\WINDOWS\\Desktop\\chatter\\COEF1.DAT"; // path to filter coeffs file
char *errstr;
int max_index, min_index;
int min_index1, max_index1;
double max_a0_band, min_a0_band;
double INDEX_REF = 0.65;                // reference threshold for chatter detection index
int w,t;
double mad;                             // median value of a sequence
double p = 0;
double mean_peak[5] = {0.0, 0.0, 0.0, 0.0, 0.0};
double mean_shape[5] = {0.0, 0.0, 0.0, 0.0, 0.0};
double mean_a0_band, mean_shape_para;       // mean value of mean height value a_0
                                            // and shape parameter
double max_shape_para, min_shape_para;// maximal and mimimal shape
                                      // parameter corresponding to some band
double max_shape_para1[5];
double J[5] = {0.0, 0.0, 0.0, 0.0, 0.0};
double max_J = 0.0;                     // maximal energy value in certain band
double min_J = 0.0;                     // minimum energy value in certain band
double wave_mean[5] = {0.0}; // mean value of wavelet coefficients after the de-noising
double wave_std[5] = {0.0};     // standard deviation of cleaned wavelet coefficients
double percent[5] = {0.0};              // energy distrivution percentage
double rms_sum;                         // total energy absorbed by band1
double rms_sum1;                        // total energy absorbed by band2
double rms_sum2;                        // total energy absorbed by band3
double rms_sum3;                        // total energy absorbed by band4
double rms_sum4;                        // total energy absorbed by band5
double tmp_rms[5][LOOP_LEN] = {0.0};
double copy_rms[5] = {0.0};
double data[1000], data1[1000], data2[1000], data3[1000], data4[1000];
double data_E[100];
double data_CE[100];
double energy_data[1000], energy_data1[1000], energy_data2[1000], energy_data3[1000],
       energy_data4[1000], energy_data5[1000];
double cov_sum, cov_sum1, cov_sum2, cov_sum3, cov_sum4;
int memory_size, memory_size1, memory_size2, memory_size3, memory_size4;
double covariance, covariance1, covariance2, covariance3, covariance4;
double sum_rms, sum_rms1, sum_rms2, sum_rms3, sum_rms4;

/* shape parameter of Weibull distribution */
double shape_para, shape_para1, shape_para2, shape_para3, shape_para4;
```

```
/* mean height value (a_0) in each band */
   double a0_band, a0_band1, a0_band2, a0_band3, a0_band4;
/* mean value of extracted wavelet transform modulus maxima */
   double wtmm_mean[5] = {0.0};
/* standard deviation of extracted wavelet transform modulus maxima */
   double wtmm_std[5] = {0.0};
/* plot for both chatter index and reference threshold */
   double plot[2], plot1[2], plot2[2], plot3[2], plot4[2];

   double cdf_mean[5] = {0.0};     // mean value of Weibull distribution function
   double cdf_std[5] = {0.0};      // standard deviation of Weibull distribution function
   double ww = 0.0;
   int pp = 0, nn = 1, mm = hh = gg = 0, tt = -1, bb = q =  k = v = c = cc = vv = 0, ttt;
   int button = 1;
   double Energy[2] = {0.0};
   double EE[100] = {0.0};
   double E, CE;          // fuzzified values for energy error and change of energy error
   int ctrl_signal_count = 0;                  // count the number of control being sent
   double FeedVoltageRange;                    // range for feed rate change
   double SpeedVoltageRange;                   // range for spindle speed change
   double feed_step;                           // adjustment step for spindle speed and feed rate
   double speed_step;
   double WI;                  // chatter detection index value from chatter-prone band
   double WI_previous = 0;
   double E_x, E_k;            // instant energy values
   double runtime;
   double e, ce;               // energy error and change of energy error
   double e_max = 4.5;         // maximal value of energy error
   double e_min = 0.0;         // minimum value of energy error
   double ce_max = 4.5;        // maximal value of change of energy error
   double ce_min = -4.5;       // minimum value of change of energy error
   double previous_e = 0;
   double current_e;
   double output_ss;           // spindle speed output calculated by fuzzy engine
   double output_fr;           // feed rate output calculated by fuzzy engine
   double *output_ss1 = NULL;
   double *output_fr1 = NULL;
   int OUTPUT_BUTTON = 0;
   int SELF_REG_BUTTON = 0;
   int DECREASE_BUTTON = 0;
   int SUPPRESSION_BUTTON = 0;
   double K1 = 0.001;          // K1 and K2 are output scaling factor coefficients
   double K2 = 0.1;
   double mse_current, mse_previous, current_sum_e, previous_sum_e;
   double G_p_ss, G_p_fr,      // adaptatin gain
   double p_n_ss, p_n_fr;      // performance measurement value
   int kk = 0;
```

```
  int N = 512;
  int delay_d;                          // delay time
  int SS_VALUE = 0;
  int FR_VALUE = 0;
  double man_speed_k, man_feed_k;
  double yita_min, yita_max;        // lower and upper bounds of target time constant
  double approx, approx_max;
  double output1[6];
  double MAX_OUTPUT_SS = 1.0;          // maximal fuzzy output for spindle speed
  double MAX_OUTPUT_FR = 1.0;          // maximal fuzzy output for feed rate
  double MAX_NOMINATOR_SS = -0.7;              // maximal value of p_n_ss
  double MAX_NOMINATOR_FR = -0.7;              // maximal value of p_n_fr
  double T_p = 0.08;
  double yita_p = 0.5;
  double FeedLowerLimit = 2.45; // limit for lowest feed at 2.45 volt = 50mm/min feed
  double FeedUpperLimit = 4.28; // limit for highest feed at 150.0mm/min
  double SpeedLowerLimit = 3.5; // limit for lowest speed at 500rpm, high gear mode
  double SpeedUpperLimit = 8.5; // limit for highest speed at 1500rpm, high gear mode

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* constraint input & output parameters */
  FeedVoltageRange = FeedMaxVoltage - FeedMinVoltage;
  SpeedVoltageRange = SpeedMaxVoltage - SpeedMinVoltage;
  GetCtrlVal (childpanel, CHILDPANEL_CYCLES_PERSEC, &CyclesPerSec);

/* calculate feed_step and speed_step */
  feed_step = FeedVoltageRange/CyclesPerSec;
  speed_step = SpeedVoltageRange/CyclesPerSec;

/* the program starts to execute */
  if (event == EVENT_TIMER_TICK) {
      anaptr = AllocCoeffWFBD();
      synptr = AllocCoeffWFBD();

/* read filter coefficients */
  err = ReadCoeffWFBD(path,anaptr,synptr);
  if(err) goto errend;

/* main control loop for detection and suppression */
  while ((iDAQstopped != 1) && (iStatus == 0)) {

      if (control_ID == DAQPANEL_TOGGLEBUTTON) break;
      if (control_ID == 900);
      if (control_ID == 2000);

/* check if the half buffer is ready for data transfer */
      iStatus = DAQ_DB_HalfReady(iDevice, &iHalfReady, &iDAQstopped);
```

```
        iRetVal = NIDAQErrorHandler(iStatus, "DAQ_DB_HalfReady", iIgnoreWarning);

        if (iHalfReady == 1) {
                iStatus = DAQ_DB_Transfer(iDevice, piHalfBuffer, &ulPtsTfr,
                                                                &iDAQstopped);
                iRetVal = NIDAQErrorHandler(iStatus, "DAQ_DB_Transfer",
                                                                iIgnoreWarning);
                iStatus = SCAN_Demux(piHalfBuffer, ulCount, iNumChans,
                                                                iNumMUXBrds);
                iRetVal = NIDAQErrorHandler(iStatus, "SCAN_Demux", iIgnoreWarning);
                iStatus = DAQ_VScale(iDevice, iChan, iGain, dGainAdjust, dOffset, ulCount,
                                                        piHalfBuffer, pdVoltBuffer);
                iRetVal = NIDAQErrorHandler(iStatus, "DAQ_VScale", iIgnoreWarning);

                for (i = 0; i < BUFF_SIZE; i+=2)
                        datapoints_x[i/2]=pdVoltBuffer[i];

                PlotStripChart (daqpanel, DAQPANEL_CHANNEL0, datapoints_x, 512, 0, 0,
                                                                VAL_DOUBLE);

/* write the data into file */
                ArrayToFile ("exprt_data.txt", datapoints_x, VAL_DOUBLE, 512, 1,
                                VAL_GROUPS_TOGETHER, VAL_GROUPS_AS_COLUMNS,
                                        VAL_CONST_WIDTH, 10, VAL_ASCII, VAL_APPEND);

/* Calculation of mean vibration energy */
                E_x = 0;
                for (i = 0; i < N; i++)
                        E_x = E_x + fabs(datapoints_x[i]);
                        E_x = E_x/N;
                        E_k = E_x;
                if(control_ID == 2000 && control_value == 1) {
                        EE[ctrl_signal_count] = E_k;
                        ctrl_signal_count++;
                        }
                Energy[0] = E_k;
                Energy[1] = E_r;
                PlotStripChart (daqpanel, DAQPANEL_STRIPCHART, &Energy, 2, 0, 0,
                                                                VAL_DOUBLE);

/* check if the energy value exceeds the maximal limit*/
                if (E_k >= E_max) {
                        SetCtrlVal (daqpanel, DAQPANEL_ALARM, 1);// alarm turns on
                        AO_VWrite(1, 0, FeedMinVoltage);    // reset feed rate
                        AO_VWrite(1, 1, SpeedMinVoltage);  // reset spindle speed
                        FeedOverride = FeedMinVoltage;
                        SetCtrlVal (childpanel, CHILDPANEL_FEEDKNOB, FeedOverride);
```

```
                    SpeedOverride = SpeedMinVoltage;
                    SetCtrlVal (childpanel, CHILDPANEL_SPINDLEKNOB,
                                                        SpeedOverride);
                    feed_k = FeedMinVoltage;
                    speed_k = SpeedMinVoltage;
                }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* D4 discrete wavelet transform and decomposition level is set to 4 */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* level 1 wavelet decomposition */
            NUM_OF_ELEMENTS = ulCount/iNumChans;
            nx = (long)NUM_OF_ELEMENTS;
            x = (floatnum*)malloc(nx*sizeof(floatnum));
            if(!x) {
               err = OutOfMemErr;
               goto errend;
               }
            Copy1D (datapoints_x, NUM_OF_ELEMENTS, x);
            ny0 = ceil(0.5*(nx+anaptr->nh-1));

/* allocate memory for 1st stage analysis lowpass filter output */
            y0 = (floatnum*)malloc(ny0*sizeof(floatnum));
            if(!y0) {
               err = OutOfMemErr;
               goto errend;
               }
            ny1 = ceil(0.5*(nx+anaptr->nl-1));

/* allocate memory for 1st stage analysis highpass filter output */
            y1 = (floatnum*)malloc(ny1*sizeof(floatnum));
            y11 = (floatnum*)malloc(ny1*sizeof(floatnum));
            if(!y1) {
               err = OutOfMemErr;
               goto errend;
               }

/* compute the outputs of 1st stage analysis filter bank */
            err = AnalysisFilterBank (x, nx, anaptr, 1, y0, ny0, y1, ny1);
            if(err) goto errend;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* level 2 wavelet decomposition */
            ny00 = ceil(0.5*(ny0+anaptr->nh-1));
/* allocate memory for 2nd stage analysis lowpass filter output */
            y00 = (floatnum*)malloc(ny00*sizeof(floatnum));
```

```
            if(!y00) {
               err = OutOfMemErr;
               goto errend;
               }
            ny01 = ceil(0.5*(ny0+anaptr->nl-1));

/* allocate memory for 2nd stage analysis highpass filter output */
            y01 = (floatnum*)malloc(ny01*sizeof(floatnum));
            y011 = (floatnum*)malloc(ny01*sizeof(floatnum));
            if(!y01) {
               err = OutOfMemErr;
               goto errend;
               }

/* compute the outputs of 2nd stage analysis filter bank */
            err=AnalysisFilterBank(y0,ny0,anaptr,0,y00,ny00,y01,ny01);
            if(err) goto errend;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* level 3 wavelet decomposition */
            ny000 = ceil(0.5*(ny00+anaptr->nh-1));

/* allocate memory for 3rd stage analysis lowpass filter output */
            y000 = (floatnum*)malloc(ny000*sizeof(floatnum));
            if(!y000) {
               err = OutOfMemErr;
               goto errend;
               }
            ny001 = ceil(0.5*(ny00+anaptr->nl-1));

/* allocate memory for 3rd stage analysis highpass filter output */
            y001 = (floatnum*)malloc(ny001*sizeof(floatnum));
            y0011 = (floatnum*)malloc(ny001*sizeof(floatnum));
            if(!y001) {
               err = OutOfMemErr;
               goto errend;
               }

/* compute the outputs of 3rd stage analysis filter bank */
            err=AnalysisFilterBank(y00,ny00,anaptr,0,y000,ny000,y001,ny001);
            if(err) goto errend;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* level 4 wavelet decomposition */
            ny0000 = ceil(0.5*(ny000+anaptr->nh-1));
/* allocate memory for 4th stage analysis lowpass filter output */
            y0000 = (floatnum*)malloc(ny0000*sizeof(floatnum));
```

```
            if(!y0000) {
               err = OutOfMemErr;
               goto errend;
               }
            ny0001 = ceil(0.5*(ny000+anaptr->nl-1));

/* allocate memory for 4th stage analysis highpass filter output */
            y0001 = (floatnum*)malloc(ny0001*sizeof(floatnum));
            y00011 = (floatnum*)malloc(ny0001*sizeof(floatnum));
            if(!y001) {
               err = OutOfMemErr;
               goto errend;
               }

/* compute the outputs of 4th stage analysis filter bank */
            err=AnalysisFilterBank(y000,ny000,anaptr,0,y0000,ny0000,y0001,ny0001);
            if(err) goto errend;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* wavelet denoising with a custermized thresholding function and universal threshold rule*/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* level 1 noise reductin for detail wavelet coefficients */
            Copy1D (y1, ny1, y11);
            Abs1D (y11, ny1, y11);
            Sort (y11, ny1, 0, y11);
            mad =  y11[ny1/2];  // calculate the median value of wavelet coefficients
            std1[0] = mad/0.6745; // estimate the standard deviation of noise level
            thr[0]= std1[0]*sqrt(2*log(256));      // calculate the threshold value
            denoising(ny1, y1, thr[0]);            // call de-noising function

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* level 2 noise reduction for detail wavelet coefficients */
            Copy1D (y01, ny01, y011);
            Abs1D (y011, ny01, y011);
            Sort (y011, ny01, 0, y011);

/* calculate the median value of wavelet coefficients */
            mad = (y011[ny01/2] + y011[(ny01/2)+1])/2;
            std1[1] = mad/0.6745;        // estimate the standard deviation of noise level
            thr[1]= std1[1]*sqrt(2*log(131));      // calculate the threshold value
            denoising(ny01, y01, thr[1]);          // call de-noising function

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* level 3 noise reduction for detail wavelet coefficients */
            Copy1D (y001, ny001, y0011);
            Abs1D (y0011, ny001, y0011);
```

```
            Sort (y0011, ny001, 0, y0011);

/* calculate the median value of wavelet coefficients */
            mad = (y0011[ny001/2] + y0011[(ny001/2)+1])/2;
            std1[2] = mad/0.6745;        // estimate the standard deviation of noise level
            thr[2]= std1[2]*sqrt(2*log(67));        // calculate the threshold value
            denoising(ny001, y001, thr[2]);        // call de-noising function

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* level 4 noise reduction for detail wavelet coefficients */
            Copy1D (y0001, ny0001, y00011);
            Abs1D (y00011, ny0001, y00011);
            Sort (y00011, ny0001, 0, y00011);

/* calculate the median value of wavelet coefficients */
            mad = (y00011[ny0001/2] + y00011[(ny0001/2)+1])/2;
            std1[3] = mad/0.6745;        // estimate the standard deviation of noise level
            thr[3]= std1[3]*sqrt(2*log(35));        // calculate the threshold value
            denoising(ny0001, y0001, thr[3]);        // call de-noising function

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* wavelet reconstruction of the underlying signal */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* level 1 reconstruction */
            nx0000 = ny0000*2-synptr->nl+1;
            x0000 = (floatnum*)malloc(nx0000*sizeof(floatnum));
            if(!x0000) {
               err = OutOfMemErr;
               goto errend;
               }
            err =SynthesisFilterBank(y0000,ny0000,y0001,ny0001,synptr,x0000,nx0000);
            if(err) goto errend;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* level 2 reconstruction */
            nx000 = ny000*2-synptr->nl+1;
            x000 = (floatnum*)malloc(nx000*sizeof(floatnum));
            if(!x000) {
               err = OutOfMemErr;
               goto errend;
               }
            err = SynthesisFilterBank(x0000,nx0000,y001,ny001,synptr,x000,nx000);
            if(err) goto errend;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* level 3 reconstruction */
```

```
nx00 = ny00*2-synptr->nl+1;
x00 = (floatnum*)malloc((nx00+1)*sizeof(floatnum));
if(!x00) {
   err = OutOfMemErr;
   goto errend;
   }
err = SynthesisFilterBank(x000,nx000,y01,ny01,synptr,x00,nx00);
if(err) goto errend;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* level 4 reconstruction and the original signal is approximately reconstructed */
nx0 = nx00*2-synptr->nl+1;
x0 = (floatnum*)malloc(nx0*sizeof(floatnum));
if(!x0) {
   err = OutOfMemErr;
   goto errend;
   }
err = SynthesisFilterBank(x00,nx00,y1,ny1,synptr,x0,nx0);
if(err) goto errend;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* apply Fourier transform to the cleaned signal to observe chatter frequency */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

realft(x0, 512, 1);              // call fast Fourier transform function
t=2;
for (q = 2; q < 511; q+=2) {
      if (t>510) break;
      freq[0] = x0[0];
      freq[t/2] = sqrt(x0[q]*x0[q]+x0[q+1]*x0[q+1]);
      t = t+2;
      }
PlotY (daqpanel, DAQPANEL_FREQ_PLOT3, freq, 256, VAL_DOUBLE,
VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* calculate the signal's energy in terms of wavelet coefficients */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

wave_cross_power  = (floatnum*)malloc(ny1*sizeof(floatnum));
wave_cross_power1 = (floatnum*)malloc(ny01*sizeof(floatnum));
wave_cross_power2 = (floatnum*)malloc(ny001*sizeof(floatnum));
wave_cross_power3 = (floatnum*)malloc(ny0001*sizeof(floatnum));
wave_cross_power4 = (floatnum*)malloc(ny0000*sizeof(floatnum));

if(button == 1) {
      win_wave_coeff =
```

```
                    (floatnum*)malloc(ny1*LOOP_LEN*sizeof(floatnum));
            win_wave_coeff1 =
                    (floatnum*)malloc(ny01*LOOP_LEN*sizeof(floatnum));
            win_wave_coeff2 =
                    (floatnum*)malloc(ny001*LOOP_LEN*sizeof(floatnum));
            win_wave_coeff3 =
                    (floatnum*)malloc(ny0001*LOOP_LEN*sizeof(floatnum));
            win_wave_coeff4 =
                    (floatnum*)malloc(ny0000*LOOP_LEN*sizeof(floatnum));
            }
        button = 0;

/* Calculate wavelet power spectrum */
        Mul1D (y1, y1, ny1, wave_cross_power);
        Mul1D (y01, y01, ny01, wave_cross_power1);
        Mul1D (y001, y001, ny001, wave_cross_power2);
        Mul1D (y0001, y0001, ny0001, wave_cross_power3);
        Mul1D (y0000, y0000, ny0000, wave_cross_power4);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* implement a moving average window with specified size */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

        if(cc < LOOP_LEN) {
            cc++;
            for(w = ny1*(cc-1); w < ny1*cc; w++)
                win_wave_coeff[w] = y1[w-ny1*(cc-1)];
            for(w = ny01*(cc-1); w < ny01*cc; w++)
                win_wave_coeff1[w] = y01[w-ny01*(cc-1)];
            for(w = ny001*(cc-1); w < ny001*cc; w++)
                win_wave_coeff2[w] = y001[w-ny001*(cc-1)];
            for(w = ny0001*(cc-1); w < ny0001*cc; w++)
                win_wave_coeff3[w] = y0001[w-ny0001*(cc-1)];
            for(w = ny0000*(cc-1); w < ny0000*cc; w++)
                win_wave_coeff4[w] = y0000[w-ny0000*(cc-1)];
            }

        if(cc >= LOOP_LEN) {
            if(cc > LOOP_LEN) {
                for(w = 0; w < ny1*(LOOP_LEN-1); w++)
                    win_wave_coeff[w] = win_wave_coeff[w+ny1];
                for(w = 0; w < ny01*(LOOP_LEN-1); w++)
                    win_wave_coeff1[w] = win_wave_coeff1[w+ny01];
                for(w = 0; w < ny001*(LOOP_LEN-1); w++)
                    win_wave_coeff2[w] = win_wave_coeff2[w+ny001];
                for(w = 0; w < ny0001*(LOOP_LEN-1); w++)
                    win_wave_coeff3[w] = win_wave_coeff3[w+ny0001];
```

```
                for(w = 0; w < ny0000*(LOOP_LEN-1); w++)
                        win_wave_coeff4[w] = win_wave_coeff4[w+ny0000];
                for(w = ny1*(LOOP_LEN-1); w < ny1*LOOP_LEN; w++)
                        win_wave_coeff[w] = y1[w-ny1*(LOOP_LEN-1)];
                for(w = ny01*(LOOP_LEN-1); w < ny01*LOOP_LEN; w++)
                        win_wave_coeff1[w] = y01[w-ny01*(LOOP_LEN-1)];
                for(w = ny001*(LOOP_LEN-1); w < ny001*LOOP_LEN;
                                                                w++)
                        win_wave_coeff2[w] = y001[w-ny001*
                                                        (LOOP_LEN-1)];
                for(w = ny0001*(LOOP_LEN-1); w < ny0001*LOOP_LEN;
                                                                w++)
                        win_wave_coeff3[w] = y0001[w-ny0001*
                                                        (LOOP_LEN-1)];
                for(w = ny0000*(LOOP_LEN-1); w < ny0000*LOOP_LEN;
                                                                w++)
                        win_wave_coeff4[w] = y0000[w-ny0000*
                                                        (LOOP_LEN-1)];
                }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* calculate the band standard deviations and shift the wavelet coefficients of each band with
   respet to their mean value in order to obtain zero mean */
                StdDev (win_wave_coeff, ny1*LOOP_LEN, &wave_mean[0],
                                                        &wave_std[0]);
                for (w = 0; w < ny1*LOOP_LEN; w++)
                        win_wave_coeff[w] = win_wave_coeff[w] - wave_mean[0];
                StdDev (win_wave_coeff1, ny01*LOOP_LEN, &wave_mean[1],
                                                        &wave_std[1]);
                for (w = 0; w < ny01*LOOP_LEN; w++)
                        win_wave_coeff1[w] = win_wave_coeff1[w] - wave_mean[1];
                StdDev (win_wave_coeff2, ny001*LOOP_LEN, &wave_mean[2],
                                                        &wave_std[2]);
                for (w = 0; w < ny001*LOOP_LEN; w++)
                        win_wave_coeff2[w] = win_wave_coeff2[w] - wave_mean[2];
                StdDev (win_wave_coeff3, ny0001*LOOP_LEN, &wave_mean[3],
                                                        &wave_std[3]);
                for (w = 0; w < ny0001*LOOP_LEN; w++)
                        win_wave_coeff3[w] = win_wave_coeff3[w] - wave_mean[3];
                StdDev (win_wave_coeff4, ny0000*LOOP_LEN, &wave_mean[4],
                                                        &wave_std[4]);
                for (w = 0; w < ny0000*LOOP_LEN; w++)
                        win_wave_coeff4[w] = win_wave_coeff4[w] - wave_mean[4];

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* extract WTMM from the corresponding wavelet coefficients*/ of each band */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
/* WTMM extraction from sub-band 1 */
                    i = 0;
                    memory_size = 0;
                    wtmm = (floatnum*)malloc(1*sizeof(floatnum));
                    while(i < ny1*LOOP_LEN) {
                        if(i == 0 && win_wave_coeff[0] >= 0) {
                            wtmm[memory_size] = win_wave_coeff[i];
                            memory_size++;
                            i++;
                            continue;
                            }
                        if(i = =ny1*LOOP_LEN-1 &&
                                win_wave_coeff[ny1*LOOP_LEN-1] > 0) {
                            memory_size++;
                            wtmm = (floatnum*)realloc(wtmm,
                                        (memory_size)*sizeof(floatnum));
                            wtmm[memory_size-1] = win_wave_coeff[i];
                            break;
                            }
                        else {
                            if (i == ny1*LOOP_LEN-1 &&
                                win_wave_coeff[ny1*LOOP_LEN-1] <= 0)
                                break;
                            if (win_wave_coeff[i] <= 0.000000000000001) {
                                i++;
                                continue;
                                }
                            if(win_wave_coeff[i] > win_wave_coeff[i+1] &&
                                  win_wave_coeff[i] > win_wave_coeff[i-1]) {
                                memory_size++;
                                wtmm = (floatnum*)realloc(wtmm,
                                        (memory_size)*sizeof(floatnum));
                                wtmm[memory_size-1] = win_wave_coeff[i];
                                }
                            } /* end of else */
                        i++;
                    } /* end of while(i<ny1*LOOP_LEN) */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* WTMM extraction from sub-band 2 */
                    i = 0;
                    memory_size1 = 0;
                    wtmm1 = (floatnum*)malloc(1*sizeof(floatnum));
                    while(i <= ny01*LOOP_LEN) {
                        if(i == 0 && win_wave_coeff1[0] >= 0) {
                            wtmm1[memory_size1] = win_wave_coeff1[i];
                            memory_size1++;
```

```
                    i++;
                    continue;
                    }
                if(i == ny01*LOOP_LEN-1 &&
                        win_wave_coeff1[ny01*LOOP_LEN-1] > 0) {
                    memory_size1++;
                    wtmm1 = (floatnum*)realloc(wtmm1,
                                (memory_size1)*sizeof(floatnum));
                    wtmm1[memory_size1-1] = win_wave_coeff1[i];
                    break;
                    }
                else {
                    if (i == ny01*LOOP_LEN-1 &&
                        win_wave_coeff1[ny01*LOOP_LEN-1] <= 0)
                        break;
                    if (win_wave_coeff1[i]<=0.000000000000001) {
                        i++;
                        continue;
                        }
                    if(win_wave_coeff1[i] > win_wave_coeff1[i+1] &&
                        win_wave_coeff1[i] > win_wave_coeff1[i-1]) {
                        memory_size1++;
                        wtmm1 = (floatnum*)realloc(wtmm1,
                                (memory_size1)*sizeof(floatnum));
                        wtmm1[memory_size1-1] =
                                                win_wave_coeff1[i];
                        }
                    } /* end of else */
                i++;
            } /* end of while(i<ny01*LOOP_LEN) */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* WTMM extraction from sub-band 3 */
            i = 0;
            memory_size2 = 0;
            wtmm2 = (floatnum*)malloc(1*sizeof(floatnum));
            while(i <= ny001*LOOP_LEN) {
                if(i == 0 && win_wave_coeff2[0] >= 0) {
                    wtmm2[memory_size2] = win_wave_coeff2[i];
                    memory_size2++;
                    i++;
                    continue;
                    }
                if(i == ny001*LOOP_LEN-1 &&
                        win_wave_coeff2[ny001*LOOP_LEN-1] > 0) {
                    memory_size2++;
                    wtmm2 = (floatnum*)realloc(wtmm2,
```

```
                                        (memory_size2)*sizeof(floatnum));
                        wtmm2[memory_size2-1] = win_wave_coeff2[i];
                        break;
                        }
                else {
                        if (i == ny001*LOOP_LEN-1 &&
                                win_wave_coeff2[ny001*LOOP_LEN-1] <= 0)
                                break;
                        if (win_wave_coeff2[i] <= 0.000000000000001) {
                                i++;
                                continue;
                                }
                        if(win_wave_coeff2[i] > win_wave_coeff2[i+1] &&
                                win_wave_coeff2[i] > win_wave_coeff2[i-1]) {
                                memory_size2++;
                                wtmm2 = (floatnum*)realloc(wtmm2,
                                        (memory_size2)*sizeof(floatnum));
                                wtmm2[memory_size2-1] =
                                                win_wave_coeff2[i];
                                }
                        } /* end of else */
                i++;
        } /* end of while(i<ny001*LOOP_LEN) */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* WTMM extraction from sub-band 4 */
        i = 0;
        memory_size3 = 0;
        wtmm3 = (floatnum*)malloc(1*sizeof(floatnum));
        while(i <= ny0001*LOOP_LEN) {
                if(i == 0 && win_wave_coeff3[0] >= 0) {
                        wtmm3[memory_size3] = win_wave_coeff3[i];
                        memory_size3++;
                        i++;
                        continue;
                        }
                if(i == ny0001*LOOP_LEN-1 &&
                                win_wave_coeff3[ny0001*LOOP_LEN-1] > 0) {
                        memory_size3++;
                        wtmm3 = (floatnum*)realloc(wtmm3,
                                        (memory_size3)*sizeof(floatnum));
                        wtmm3[memory_size3-1] = win_wave_coeff3[i];
                        break;
                        }
                else {
                        if(i == ny0001*LOOP_LEN-1 &&
                                win_wave_coeff3[ny0001*LOOP_LEN-1] <= 0)
```

```
                        break;
                    if (win_wave_coeff3[i] <= 0.000000000000001) {
                        i++;
                        continue;
                        }
                    if(win_wave_coeff3[i] > win_wave_coeff3[i+1] &&
                        win_wave_coeff3[i] > win_wave_coeff3[i-1]) {
                        memory_size3++;
                        wtmm3 = (floatnum*)realloc(wtmm3,
                                (memory_size3)*sizeof(floatnum));
                        wtmm3[memory_size3-1] =
                                                win_wave_coeff3[i];
                        }
                    } /* end of else */
                i++;
            } /* end of while(i<ny0001*LOOP_LEN) */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* WTMM extraction from sub-band 5 */
            i = 0;
            memory_size4 = 0;
            wtmm4 = (floatnum*)malloc(1*sizeof(floatnum));
            while(i <= ny0000*LOOP_LEN) {
                if(i == 0 && win_wave_coeff4[0] >= 0) {
                    wtmm4[memory_size4] = win_wave_coeff4[i];
                    memory_size4++;
                    i++;
                    continue;
                    }
                if(i == ny0000*LOOP_LEN-1 &&
                        win_wave_coeff4[ny0000*LOOP_LEN-1] > 0) {
                    memory_size4++;
                    wtmm4 = (floatnum*)realloc(wtmm4,
                                (memory_size4)*sizeof(floatnum));
                    wtmm4[memory_size4-1] = win_wave_coeff4[i];
                    break;
                    }
                else {
                    if(i == ny0000*LOOP_LEN-1 &&
                        win_wave_coeff4[ny0000*LOOP_LEN-1] <= 0)
                        break;
                    if (win_wave_coeff4[i] <= 0.000000000000001) {
                        i++;
                        continue;
                        }
                    if(win_wave_coeff4[i] > win_wave_coeff4[i+1] &&
                        win_wave_coeff4[i] > win_wave_coeff4[i-1]) {
```

```
                                        memory_size4++;
                                        wtmm4 = (floatnum*)realloc(wtmm4,
                                                (memory_size4)*sizeof(floatnum));
                                        wtmm4[memory_size4-1] =
                                                                win_wave_coeff4[i];
                                        }
                                } /* end of else */
                        i++;
                } /* end of while(i<ny0000*LOOP_LEN) */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* estimate Weibull shpae parameter (gamma) using regression technique */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* estimate shape parameter for sub-band 1 */
                wtmm_cdf = (floatnum*)malloc(memory_size*sizeof(floatnum));
                wtmm_tmp = (floatnum*)malloc(memory_size*sizeof(floatnum));
                Copy1D (wtmm, memory_size, wtmm_tmp);
                Sort (wtmm_tmp, memory_size, 1, wtmm_tmp);
                for(w = 0; w < memory_size; w++) {
                        wtmm_cdf[w] = log(-(log(w+1)-log(memory_size+1)));
                        wtmm_tmp[w] = log(wtmm_tmp[w]);
                        }
                StdDev (wtmm_tmp, memory_size, &wtmm_mean[0],
                                                                &wtmm_std[0]);
                StdDev (wtmm_cdf, memory_size, &cdf_mean[0], &cdf_std[0]);
                cov_sum = 0;
                for (w = 0; w < memory_size; w++)
                        cov_sum = cov_sum + (wtmm_tmp[w]-
                                wtmm_mean[0])*(wtmm_cdf[w] - cdf_mean[0]);

/* calculate the line's covariance value after the logarithmic transformation */
                covariance = cov_sum/(memory_size-1);

/* calculate the shape parameter based on covariance and variance */
                shape_para = covariance/(wtmm_std[0]*wtmm_std[0]);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* estimate shape parameter for sub-band 2 */
                wtmm_cdf1 = (floatnum*)malloc(memory_size1*sizeof(floatnum));
                wtmm_tmp1 = (floatnum*)malloc(memory_size1*sizeof(floatnum));
                Copy1D (wtmm1, memory_size1, wtmm_tmp1);
                Sort (wtmm_tmp1, memory_size1, 1, wtmm_tmp1);
                for(w = 0; w < memory_size1; w++) {
                        wtmm_cdf1[w] = log(-(log(w+1)-log(memory_size1+1)));
                        wtmm_tmp1[w] = log(wtmm_tmp1[w]);
                        }
```

```
                    StdDev (wtmm_tmp1, memory_size1, &wtmm_mean[1],
                                                        &wtmm_std[1]);
                    StdDev (wtmm_cdf1, memory_size1, &cdf_mean[1], &cdf_std[1]);
                    cov_sum1 = 0;
                    for (w = 0; w < memory_size1; w++)
                        cov_sum1 = cov_sum1 + (wtmm_tmp1[w]-
                                wtmm_mean[1])*(wtmm_cdf1[w] - cdf_mean[1]);

/* calculate the line's covariance value after the logarithmic transformation */
                        covariance1 = cov_sum1/(memory_size1-1);

/* calculate the shape parameter based on covariance and variance */
                        shape_para1 = covariance1/(wtmm_std[1]*wtmm_std[1]);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* estimate shape parameter for sub-band 3 */

                    wtmm_cdf2 = (floatnum*)malloc(memory_size2*sizeof(floatnum));
                    wtmm_tmp2 = (floatnum*)malloc(memory_size2*sizeof(floatnum));
                    Copy1D (wtmm2, memory_size2, wtmm_tmp2);
                    Sort (wtmm_tmp2, memory_size2, 1, wtmm_tmp2);
                    for(w = 0; w < memory_size2; w++) {
                        wtmm_cdf2[w] = log(-(log(w+1)-log(memory_size2+1)));
                        wtmm_tmp2[w] = log(wtmm_tmp2[w]);
                        }
                    StdDev (wtmm_tmp2, memory_size2, &wtmm_mean[2],
                                                        &wtmm_std[2]);
                    StdDev (wtmm_cdf2, memory_size2, &cdf_mean[2], &cdf_std[2]);
                    cov_sum2 = 0;
                    for (w = 0; w < memory_size2; w++)
                        cov_sum2 = cov_sum2 + (wtmm_tmp2[w]-
                                wtmm_mean[2])*(wtmm_cdf2[w] - cdf_mean[2]);

/* calculate the line's covariance value after the logarithmic transformation */
                    covariance2 = cov_sum2/(memory_size2-1);

/* calculate the shape parameter based on covariance and variance */
                    shape_para2 = covariance2/(wtmm_std[2]*wtmm_std[2]);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* estimate shape parameter for sub-band 4 */
                    wtmm_cdf3 = (floatnum*)malloc(memory_size3*sizeof(floatnum));
                    wtmm_tmp3 = (floatnum*)malloc(memory_size3*sizeof(floatnum));
                    Copy1D (wtmm3, memory_size3, wtmm_tmp3);
                    Sort (wtmm_tmp3, memory_size3, 1, wtmm_tmp3);
                    for(w = 0; w < memory_size3; w++) {
                        wtmm_cdf3[w] = log(-(log(w+1)-log(memory_size3+1)));
```

```
                wtmm_tmp3[w] = log(wtmm_tmp3[w]);
                }
        StdDev (wtmm_tmp3, memory_size3, &wtmm_mean[3],
                                                &wtmm_std[3]);
        StdDev (wtmm_cdf3, memory_size3, &cdf_mean[3], &cdf_std[3]);
        cov_sum3 = 0;
        for (w = 0; w < memory_size3; w++)
                cov_sum3 = cov_sum3 + (wtmm_tmp3[w]-
                        wtmm_mean[3])*(wtmm_cdf3[w] - cdf_mean[3]);

/* calculate the line's covariance value after the logarithmic transformation */
        covariance3 = cov_sum3/(memory_size3-1);

/* calculate the shape parameter based on covariance and variance */
        shape_para3 = covariance3/(wtmm_std[3]*wtmm_std[3]);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* estimate shape parameter for sub-band 5 */
        wtmm_cdf4 = (floatnum*)malloc(memory_size4*sizeof(floatnum));
        wtmm_tmp4 = (floatnum*)malloc(memory_size4*sizeof(floatnum));
        Copy1D (wtmm4, memory_size4, wtmm_tmp4);
        Sort (wtmm_tmp4, memory_size4, 1, wtmm_tmp4);
        for(w = 0; w < memory_size4; w++) {
                wtmm_cdf4[w] = log(-(log(w+1)-log(memory_size4+1)));
                wtmm_tmp4[w] = log(wtmm_tmp4[w]);
                }
        StdDev (wtmm_tmp4, memory_size4, &wtmm_mean[4],
                                                &wtmm_std[4]);
        StdDev (wtmm_cdf4, memory_size4, &cdf_mean[4], &cdf_std[4]);
        cov_sum4 = 0;
        for (w = 0; w < memory_size4; w++)
                cov_sum4 = cov_sum4 + (wtmm_tmp4[w]-
                        wtmm_mean[4])*(wtmm_cdf4[w] - cdf_mean[4]);

/* calculate the line's covariance value after the logarithmic transformation */
        covariance4 = cov_sum4/(memory_size4-1);

/* calculate the shape parameter based on covariance and variance */
        shape_para4 = covariance4/(wtmm_std[4]*wtmm_std[4]);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* calculate mean peak height a0 for each band */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

        a0_band = exp((log(log(2))-(cdf_mean[0]-
                                shape_para*wtmm_mean[0]))/shape_para);
```

```
a0_band1 = exp((log(log(2))-(cdf_mean[1]-
                        shape_para1*wtmm_mean[1]))/shape_para1);
a0_band2 = exp((log(log(2))-(cdf_mean[2]-
                        shape_para2*wtmm_mean[2]))/shape_para2);
a0_band3 = exp((log(log(2))-(cdf_mean[3]-
                        shape_para3*wtmm_mean[3]))/shape_para3);
a0_band4 = exp((log(log(2))-(cdf_mean[4]-
                        shape_para4*wtmm_mean[4]))/shape_para4);

mean_peak[0] = a0_band;
mean_peak[1] = a0_band1;
mean_peak[2] = a0_band2;
mean_peak[3] = a0_band3;
mean_peak[4] = a0_band4;

mean_shape[0] = shape_para;
mean_shape[1] = shape_para1;
mean_shape[2] = shape_para2;
mean_shape[3] = shape_para3;
mean_shape[4] = shape_para4;

MaxMin1D (mean_peak, 5, &max_a0_band, &max_index,
                                    &min_a0_band, &min_index);
Mean (mean_peak, 5, &mean_a0_band);
                        Mean (mean_shape, 5, &mean_shape_para);
MaxMin1D (mean_shape, 5, &max_shape_para, &max_index,
                                    &min_shape_para, &min_index);

if (BUTTON[0] == 0) max_shape_para1[0] = max_shape_para;
if (BUTTON[1] == 0) max_shape_para1[1] = max_shape_para;
if (BUTTON[2] == 0) max_shape_para1[2] = max_shape_para;
if (BUTTON[3] == 0) max_shape_para1[3] = max_shape_para;
if (BUTTON[4] == 0) max_shape_para1[4] = max_shape_para;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* Weibull probability distribution calculation i.e.,calculation of chatter detection index */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

for(w = 0; w < memory_size; w++)
        wtmm_tmp[w] = 1-exp(-log(2)*pow(wtmm[w]/mean_a0_band,
                                        max_shape_para1[0]));
for(w = 0; w < memory_size1; w++)
        wtmm_tmp1[w] = 1-exp(-
                        log(2)*pow(wtmm1[w]/mean_a0_band,
                                        max_shape_para1[1]));
for(w = 0; w < memory_size2; w++)
        wtmm_tmp2[w] = 1-exp(-
```

```
                                        log(2)*pow(wtmm2[w]/mean_a0_band,
                                                        max_shape_para1[2]));
for(w = 0; w < memory_size3; w++)
        wtmm_tmp3[w] = 1-exp(-
                                        log(2)*pow(wtmm3[w]/mean_a0_band,
                                                        max_shape_para1[3]));
for(w = 0; w < memory_size4; w++)
        wtmm_tmp4[w] = 1-exp(-
                                        log(2)*pow(wtmm4[w]/mean_a0_band,
                                                        max_shape_para1[4]));

Mean (wtmm_tmp,  memory_size,  &slope[0]);
Mean (wtmm_tmp1, memory_size1, &slope[1]);
Mean (wtmm_tmp2, memory_size2, &slope[2]);
Mean (wtmm_tmp3, memory_size3, &slope[3]);
Mean (wtmm_tmp4, memory_size4, &slope[4]);

data[mm] = slope[0];
data1[mm] = slope[1];
data2[mm] = slope[2];
data3[mm] = slope[3];
data4[mm] = slope[4];

mm++;

plot[0] = slope[0];
plot[1] = INDEX_REF;
plot1[0] = slope[1];
plot1[1] = INDEX_REF;
plot2[0] = slope[2];
plot2[1] = INDEX_REF;
plot3[0] = slope[3];
plot3[1] = INDEX_REF;
plot4[0] = slope[4];
plot4[1] = INDEX_REF;

PlotStripChart (daqpanel, DAQPANEL_COHENCHART1, &plot, 2,
                                                        0, 0, VAL_DOUBLE);
PlotStripChart (daqpanel, DAQPANEL_COHENCHART2, &plot1, 2,
                                                        0, 0, VAL_DOUBLE);
PlotStripChart (daqpanel, DAQPANEL_COHENCHART3, &plot2, 2,
                                                        0, 0, VAL_DOUBLE);
PlotStripChart (daqpanel, DAQPANEL_COHENCHART4, &plot3, 2,
                                                        0, 0, VAL_DOUBLE);
PlotStripChart (daqpanel, DAQPANEL_COHENCHART5, &plot4, 2,
                                                        0, 0, VAL_DOUBLE);
```

```
                cc++;

                free(wtmm);
                free(wtmm1);
                free(wtmm2);
                free(wtmm3);
                free(wtmm4);
                free(wtmm_cdf);
                free(wtmm_cdf1);
                free(wtmm_cdf2);
                free(wtmm_cdf3);
                free(wtmm_cdf4);

                } /* end of if(cc >= LOOP_LEN) */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* target the suspicious sub-band using a percentage power criterion */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* calculate signal's power with the use of wavelet coefficents */
                rms_sum = 0;
                for (w = 0; w < ny1; w++)
                        rms_sum = rms_sum + wave_cross_power[w];  // band1 energy
                rms_sum1 = 0;
                for (w = 0; w < ny01; w++)
                        rms_sum1 = rms_sum1 + wave_cross_power1[w]; // band2 energy
                rms_sum2 = 0;
                for (w = 0; w < ny001; w++)
                        rms_sum2 = rms_sum2 + wave_cross_power2[w];// band3 energy
                rms_sum3 = 0;
                for (w = 0; w < ny0001; w++)
                        rms_sum3 = rms_sum3 + wave_cross_power3[w];// band4 energy
                rms_sum4 = 0;
                for (w = 0; w < ny0000; w++)
                        rms_sum4 = rms_sum4 + wave_cross_power4[w];// band5 energy

                copy_rms[0] = rms_sum/ny1;          // normalize the band1's energy
                copy_rms[1] = rms_sum1/ny01;        // normalize the band2's energy
                copy_rms[2] = rms_sum2/ny001;       // normalize the band3's energy
                copy_rms[3] = rms_sum3/ny0001;      // normalize the band4's energy
                copy_rms[4] = rms_sum4/ny0000;      // normalize the band5's energy

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* consider a moving window with specified size */
                sum_rms = 0;
                sum_rms1 = 0;
                sum_rms2 = 0;
```

```
sum_rms3 = 0;
sum_rms4 = 0;
tt++;
if(tt < LOOP_LEN) {
        for(bb = 0; bb < 5; bb++)
                tmp_rms[bb][tt] = copy_rms[bb];
        }
if(tt >= LOOP_LEN-1) {
        if(tt > LOOP_LEN-1) {
                for(w = 0; w <5 ; w++)
                        for(v = 0; v < LOOP_LEN-1; v++)
                                tmp_rms[w][v] = tmp_rms[w][v+1];
                for(w = 0; w < 5; w++)
                        tmp_rms[w][LOOP_LEN-1] = copy_rms[w];
                }
        for(v = 0; v < LOOP_LEN; v++) {
                sum_rms = sum_rms + tmp_rms[0][v];
                sum_rms1 = sum_rms1 + tmp_rms[1][v];
                sum_rms2 = sum_rms2 + tmp_rms[2][v];
                sum_rms3 = sum_rms3 + tmp_rms[3][v];
                sum_rms4 = sum_rms4 + tmp_rms[4][v];
                }

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* calculate the energy percentage of each band */
                percent[0] =
                  sum_rms/(sum_rms+sum_rms1+sum_rms2+sum_rms3+sum_rms4);
                percent[1] =
                 sum_rms1/(sum_rms+sum_rms1+sum_rms2+sum_rms3+sum_rms4);
                percent[2] =
                 sum_rms2/(sum_rms+sum_rms1+sum_rms2+sum_rms3+sum_rms4);
                percent[3] =
                 sum_rms3/(sum_rms+sum_rms1+sum_rms2+sum_rms3+sum_rms4);
                percent[4] =
                 sum_rms4/(sum_rms+sum_rms1+sum_rms2+sum_rms3+sum_rms4);

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* select the sub-band with maximal percentage of power */
                for(w = 0; w < 5; w++) J[w] = percent[w];
                MaxMin1D (J, 5, &max_J, &max_index1, &min_J, &min_index1);

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* chatter occurrence decision making by comparing index value with reference threshold */
                if ((slope[max_index1] > 0.5) && (slope[max_index1] <
                                                        INDEX_REF)) {
                        CanvasDrawText (daqpanel, DAQPANEL_CANVAS,
```

```
                                                    "Moderate Oscillation!",
                                        VAL_EDITOR_META_FONT, MakeRect (10, 0,
                                                        VAL_KEEP_SAME_SIZE,
                                        VAL_KEEP_SAME_SIZE), VAL_CENTER);
                        SetCtrlAttribute (daqpanel, DAQPANEL_CHATTER_LED,
                                                        ATTR_ON_COLOR, VAL_GREEN);
                        control_ID = 900;
                        }
                if (slope[max_index1] < 0.5) {
                        CanvasDrawText (daqpanel, DAQPANEL_CANVAS,
                            "Normal-Cut Condition!", VAL_EDITOR_META_FONT,
                                        MakeRect (10, 0, VAL_KEEP_SAME_SIZE,
                                        VAL_KEEP_SAME_SIZE), AL_CENTER);
                        SetCtrlAttribute (daqpanel, DAQPANEL_CHATTER_LED,
                                                        ATTR_ON_COLOR, VAL_GREEN);
                        control_ID = 900;
                        }
                if(slope[max_index1] > INDEX_REF) {
                        CanvasDrawText (daqpanel, DAQPANEL_CANVAS,
                                                        "Chatter-Fault Occurs!!",
                                        VAL_EDITOR_META_FONT, MakeRect (10,
                                                        0, VAL_KEEP_SAME_SIZE,
                                        VAL_KEEP_SAME_SIZE), VAL_CENTER);
                        WI = slope[max_index1];
                        control_ID = 2000;
                        }
                energy_data4[gg] = slope[max_index1];
                gg++;
                } /* end of if(tt >= LOOP_LEN-1) */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* release all the allocated memory */
                free(wtmm_tmp);
                free(wtmm_tmp1);
                free(wtmm_tmp2);
                free(wtmm_tmp3);
                free(wtmm_tmp4);
                free(wave_cross_power);
                free(wave_cross_power1);
                free(wave_cross_power2);
                free(wave_cross_power3);
                free(wave_cross_power4);

/* check the memory allocation error */
        errend:
                        if(y0) free(y0);
                        if(y1) free(y1);
```

```
                if(y00) free(y00);
                if(y01) free(y01);
                if(y000) free(y000);
                if(y001) free(y001);
                if(y0000) free(y0000);
                if(y0001) free(y0001);
                if(x)  free(x);
                if(x0) free(x0);
                if(x00) free(x00);
                if(x000) free(x000);
                if(x0000) free(x0000);
                free(y11);
                free(y011);
                free(y0011);
                free(y00011);

        if(err < 0) {
                errstr = GetAnalysisErrorString (err);
                MessagePopup ("error message", errstr);
                }
              ProcessSystemEvents ();

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* chatter is detected and suppression module starts to activate */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* obtain the control_ID from the panel button */
            if (control_ID == 2000) {
                    control_value = 1;
                    SetCtrlAttribute (childpanel, CHILDPANEL_CONTROLSWITCH,
                                                        ATTR_CTRL_VAL, control_value);
                    }

/* alarm button turns on when chatter is detected (for the use of chatter detection only) */
                if (control_ID == 2000 && control_value == 0) {
                        SetCtrlAttribute (daqpanel, DAQPANEL_CHATTER_LED,
                                                        ATTR_ON_COLOR, VAL_RED);
                        control_ID = 900;
                        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* Chatter suppression control starts */
                if (control_ID == 2000 && control_value == 1) {
                        SUPPRESSION_BUTTON = 1;
                        c++;
                        if (c == 1) Beep();
                        c = 0;
```

```
                    SetCtrlAttribute (daqpanel, DAQPANEL_CHATTER_LED,
                                                    ATTR_ON_COLOR, VAL_RED);
                    DisplayPanel (childpanel);  // to show the child panel

/* display three outputs - Energy, Reference Energy and Chatter Index on the control chart*/
                    output1[0] = E_k;
                    output1[1] = E_r;
                    output1[2] = WI;
                    output1[3] = INDEX_REF;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* obtain the inputs for adpative fuzzy control system */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                    runtime = Timer();              // start timer to record control time
                    SetCtrlVal (childpanel, CHILDPANEL_RUNTIME, runtime);

/* calculate the inputs (energy error and change of energy error) of adaptive fuzzy control
system */
                    e = E_r - E_k;
                    energy_data[hh] = e;
                    current_e = e;
                    ce = current_e - previous_e;
                    energy_data1[hh] = ce;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* determine the input scaling and output scaling factors, respectively */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* calculate and set input scaling factors for the input variable (energy error) */
                    if (E_k > E_r)
                            Ke = 1/(e_max - E_r);
                    else
                            Ke = 1/(E_r - e_min);
                    SetCtrlVal (childpanel, CHILDPANEL_K_E, Ke);
                    E = Ke*e;
                    if(E>1)
                            E = 1;
                    if(E<-1)
                            E = -1;

/* calculate and set input scaling factor for the input variable (change of energy error) */
                    delta_Ke = 1/(ce_max - ce_min);
                    SetCtrlVal (childpanel, CHILDPANEL_DELTA_K_E, delta_Ke);
                    CE = delta_Ke*ce;
                    if(CE>1)
                            CE = 1;
```

```
                if(CE<-1)
                        CE = -1;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* calculate the output scaling factor whilw adjusting parameters of K1 and K2 according   */
/* MSE criterion                                                                           */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                if (kk == 0) {
                        previous_sum_e = current_e*current_e;
                        GU_ss = K1*fabs(e) + K2;
                        GU_fr = K1*fabs(e) + K2;
                        }
                if (kk > 0) {
                        current_sum_e = current_e*current_e + previous_sum_e;
                        mse_current = (1/(kk+1))*current_sum_e;
                        previous_sum_e = previous_e*previous_e + previous_sum_e;
                        mse_previous = (1/(kk+1))*previous_sum_e;
                        if (mse_current > mse_previous) {
                                K1 = K1 + 0.005;
                                K2 = K2 + 0.01;
                                }
                        previous_sum_e = current_sum_e;
                        GU_ss = K1*fabs(e) + K2;
                        SetCtrlVal (childpanel, CHILDPANEL_SPINDLESPEED,
                                                                        GU_ss);
                        GU_fr = K1*fabs(e) + K2;
                        SetCtrlVal (childpanel, CHILDPANEL_FEEDRATE, GU_fr);
                        }
                previous_e = current_e;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* calculation of output of fuzzy logic control system */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* calculate the fuzzy outputs in terms of spindle speed and feed rate for three inputs */
                tmp = (double *) malloc (1*sizeof(double));
                tmp1 = (double *) malloc (1*sizeof(double));
                if (OUTPUT_BUTTON == 0) {
                        output_ss1 = (double*) malloc ((kk+1)*sizeof(double));
                        output_fr1 = (double*) malloc ((kk+1)*sizeof(double));
                        OUTPUT_BUTTON = 1;
                        }
                datamatrix[0] = E;              // energy error fuzzy input
                datamatrix[1] = CE;             // change of energy error fuzzy input
                datamatrix[2] = WI;             // chatter index fuzzy input
                tmp2 = tmp3 = datamatrix;
```

```
/* two-way adjustment mechanism for spindle speed or feed rate */
                    if (pp == 0) {
                            getFisOutput(tmp2, fis, tmp);    // increase feed rate
                            getFisOutput(tmp3, fis1, tmp1);// increase speed
                            WI_previous = WI;
                            }
                    if (pp == 1) {
                            if (WI > WI_previous) {
                                    DECREASE_BUTTON = 1;
                                    getFisOutput(tmp2, fis_1, tmp); // decrease feed rate
                                    getFisOutput(tmp3, fis1_1, tmp1); // decrease speed
                                    }
                            else {
                                    getFisOutput(tmp2, fis, tmp);      // increase feed rate
                                    getFisOutput(tmp3, fis1, tmp1); // increase speed
                                    DECREASE_BUTTON = 0;
                                    }
                            WI_previous = WI;
                    if (pp >= 2) {
                            if (WI < WI_previous && DECREASE_BUTTON == 1) {
                                    getFisOutput(tmp2, fis_1, tmp);// decrease feed rate
                                    getFisOutput(tmp3, fis1_1, tmp1); // decrease speed
                                    }
                            else {
                                    if (WI > WI_previous && DECREASE_BUTTON==1)
                                            {
                                            getFisOutput(tmp2, fis, tmp);// increase feed
                                            getFisOutput(tmp3, fis1, tmp1); // speed up
                                            DECREASE_BUTTON = 0;
                                            }
                                    if (WI > WI_previous && DECREASE_BUTTON==0)
                                            {
                                            getFisOutput(tmp2, fis_1, tmp);// decrease feed
                                            getFisOutput(tmp3, fis1_1, tmp1); // speed low
                                            DECREASE_BUTTON = 1;
                                            }
                                    if (WI < WI_previous && DECREASE_BUTTON==0)
                                            {
                                            getFisOutput(tmp2, fis, tmp);// increase feed
                                            getFisOutput(tmp3, fis1, tmp1); // speed up
                                            }
                                    }
                            WI_previous = WI;
                            }
                    pp++;
                    output_ss = *tmp1;
                    output_ss1[kk] = output_ss;
```

```
                    energy_data2[hh] = output_ss1[kk];
                    output_fr = *tmp;
                    output_fr1[kk] = output_fr;
                    free(tmp);
                    free(tmp1);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* Self-regulating algorithm mechanism starts */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                    if (SELF_REG_BUTTON == 1) {

/* obtain the "optimal " controller parameters: yita and G_p according to the cost function */
                        yita_min = T_p;                 // T_p is dead time
                        yita_max = T_p + yita_p;        // yita_p is target time constant
                        approx_max = e + yita_min*ce;
                        yita = yita_min;
                        ww = yita_min;
                        for (vv = 0; vv < (yita_max-yita_min)/0.001; vv++) {
                            ww = ww +0.001;
                            if ((approx = e+ww*ce) < 0) {
                                if(approx_max < 0) {
                                    if(fabs(approx) > fabs(approx_max)) {
                                        approx_max = approx;
                                        yita = ww;
                                        }
                                    }
                                else {
                                    approx_max = approx;
                                    yita = ww;
                                    }
                                }
                            else {
                                if (approx_max > 0)
                                    if (fabs(approx_max) > fabs(approx)) {
                                        approx_max = approx;
                                        yita = ww;
                                        }
                                }
                        } /* end of for (vv = 0;.....) */

                        SetCtrlVal (childpanel, CHILDPANEL_YITA, yita);
                        GetCtrlVal (childpanel, CHILDPANEL_T_S, &T_s);

/* calculate the control reinforcement value p_n_ss and p_n_fr for speed and feed */
                        if ((approx_max = e+yita*ce) < 0 ) {
                            G_p_ss = (0.2*fabs(MAX_OUTPUT_SS))/
```

```
                                        (fabs(MAX_NOMINATOR_SS)*T_s);
                        G_p_fr = (0.2*fabs(MAX_OUTPUT_FR))/
                                        (fabs(MAX_NOMINATOR_FR)*T_s);
                        if (G_p_ss > 50) G_p_ss = 50;
                        if (G_p_fr > 50) G_p_fr = 50;
                        }
                if ((approx_max = e+yita*ce) > 0) {
                        G_p_ss = 2;
                        G_p_fr = 2;
                        }

                SetCtrlVal (childpanel, CHILDPANEL_G_P, G_p_ss);
                SetCtrlVal (childpanel, CHILDPANEL_G_P_1, G_p_fr);
                p_n_ss = G_p_ss*(e+yita*ce)*T_s;
                p_n_fr = G_p_fr*(e+yita*ce)*T_s;
                delay_d = 2;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* calculation of performance measurements "delay_d" is exceeded */
                if (fabs(e) > 0.10 && WI > INDEX_REF && kk > delay_d) {
                        ttt = kk - delay_d;
                        output_ss1[ttt] = (output_ss1[ttt] + p_n_ss)*GU_ss;
                        output_fr1[ttt] = (output_fr1[ttt] + p_n_fr)*GU_fr;
                        }
                else {
                        output_ss1[kk] = output_ss*GU_ss;
                        output_fr1[kk] = output_fr*GU_fr;
                        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* generate control signals (speed and feed) depending on the different time instant */
                if (kk == 0) {
                        feed_k = feed_step * output_fr + feed_k;
                        speed_k = speed_step * output_ss + speed_k;
                        }
                if (kk <= delay_d && kk > 0) {
                        if (DECREASE_BUTTON == 0) {
                                feed_k  = feed_step * output_fr1[kk] + feed_k;
                                speed_k = speed_step * output_ss1[kk] +
                                                                        speed_k;
                                }
                        if (DECREASE_BUTTON == 1) {
                                feed_k  = feed_k - feed_step * output_fr1[kk];
                                speed_k = speed_k - speed_step *
                                                                output_ss1[kk];
                                }
                        }
```

```
                    if (kk > delay_d) {
                        if (DECREASE_BUTTON == 0) {
                            feed_k  = feed_step * output_fr1[ttt] + feed_k;
                            speed_k = speed_step * output_ss1[ttt] +
                                                                    speed_k;
                            }
                        if (DECREASE_BUTTON == 1) {
                            feed_k  = feed_k - feed_step * output_fr1[ttt];
                            speed_k = speed_k - speed_step *
                                                                output_ss1[ttt];
                            }
                        }
                    kk++;

                    } /* end of if (SELF_REG_BUTTON == 1) */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* plain fuzzy starts to function with fuzzy control with self-regulating disabled */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                if (SELF_REG_BUTTON == 0) {
                    if (DECREASE_BUTTON == 0) {
                        feed_k  = feed_step * output_fr1[kk] + feed_k;
                        speed_k = speed_step * output_ss1[kk] + speed_k;
                        }
                    if (DECREASE_BUTTON == 1) {
                        feed_k  = feed_k - feed_step * output_fr1[kk];
                        speed_k = speed_k - speed_step * output_ss1[kk];
                        }
                    kk++;
                }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
                output_ss1 = realloc (output_ss1, (kk+1)*sizeof(double));
                output_fr1 = realloc (output_fr1, (kk+1)*sizeof(double));

/* D/A conversion safety check */
                if(feed_k < FeedLowerLimit)
                        feed_k = FeedLowerLimit;      // override min feed %
                if (feed_k > FeedUpperLimit)
                    feed_k = FeedUpperLimit;          // override max feed %
                if (speed_k <  SpeedLowerLimit)
                    speed_k = SpeedLowerLimit;        // override min RPM
                if (speed_k > SpeedUpperLimit)
                    speed_k = SpeedUpperLimit;        // override max RPM
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* output the digital control commands (spindle speed or feed rate) the machine */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                output1[4] = speed_k;
                energy_data3[hh] = output1[4];
                output1[5] = feed_k;
                energy_data5[hh] = output1[5];
                hh++;
                PlotStripChart(childpanel, CHILDPANEL_CONTROLCHART,
                                                &output1, 6, 0, 0, VAL_DOUBLE);
                AO_VWrite(1, 0, speed_k);// output speed voltage signal to machine
                AO_VWrite(1, 1, feed_k); // output feed voltage signal to machine

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* Manually manipulate the spindle speed and feed rate to suppress chatter */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                GetCtrlVal(childpanel, CHILDPANEL_SS_O_SWITCH,
                                                        &SS_VALUE);
                if(SS_VALUE == 1) {
                    GetCtrlVal (childpanel, CHILDPANEL_SPINDLEKNOB,
                                                        &man_speed_k);
                    PlotStripChart(childpanel,
                                CHILDPANEL_CONTROLCHART, &man_speed_k,
                                                    1, 0, 0, VAL_DOUBLE);
                    AO_VWrite(1, 1, man_speed_k); // speed control command
                    }

                GetCtrlVal(childpanel, CHILDPANEL_FR_O_SWITCH,
                                                        &FR_VALUE);
                if(FR_VALUE == 1) {
                    GetCtrlVal (childpanel, CHILDPANEL_FEEDKNOB,
                                                        &man_feed_k);
                    PlotStripChart(childpanel,
                            CHILDPANEL_CONTROLCHART, &man_feed_k, 1, 0,
                                                    0, VAL_DOUBLE);
                    AO_VWrite(1, 0, man_feed_k);    // feed control command
                    }

                } /* end of if(control_ID == 2000 && control_value == 1) */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* control objective is achieved when chatter is suppressed, i.e., the energy is falling within
   an allowable band and chatter detection index value is less than 0.65 */
                if (SUPPRESSION_BUTTON == 1){
                    if (E_k >= 0.5 && E_k <= 1.0) {
```

```
                        if (WI < INDEX_REF) {
                            control_ID = 900;
                            SetCtrlAttribute (daqpanel,
                                    DAQPANEL_CHATTER_LED,
                                ATTR_ON_COLOR, VAL_GREEN);
                            control_value = 0;
                            SetCtrlAttribute (childpanel,
                                CHILDPANEL_CONTROLSWITCH,
                                  ATTR_CTRL_VAL, control_value);
                            CanvasDrawText (daqpanel,
                                    DAQPANEL_CANVAS, "Chatter is
                             suppressed!!", VAL_EDITOR_META_FONT,
                              MakeRect (10, 0, VAL_KEEP_SAME_SIZE,
                                        VAL_KEEP_SAME_SIZE),
                                                VAL_CENTER);
                            }
                        }
                    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                }/* end of if(iHalfReady) */

            }/* end of while loop */

    }/* end of if(event) */

    return 0;

} /* end of main function! */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/************************ Wavelet de-noising function ************************/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void denoising (intnum nn, floatnum* yy, double thr)
{
    floatnum *tmp;
    floatnum *tmp1;
    int i, k;
    double para = -0.01;
    double para_best;                   // optimal parameter which minimizes the GCV function
    double gcv_min;
    double gcv;                         // generalized cross validation function
    double w_sum = 0.0;
    int N_zero = 0;         // number of wavelet coefficients whose values exceed the threshold
```

```
    double thr_old;

    tmp = (floatnum*)malloc(nn*sizeof(floatnum));
    Copy1D (yy, nn, tmp);
    thr_old = thr;

/* select the best approriate parameter value according to minimum GCV criterion */
    for (i = 0; i < (1/0.01)+1; i++) {
        tmp1 = (floatnum*)malloc(nn*sizeof(floatnum));
        Copy1D (yy, nn, tmp1);
        para = para + 0.01;
        thr = (1-para)*thr_old;

        for (k = 0; k < nn; k++) {
                if (fabs(tmp1[k] ) >= thr)
                        tmp1[k] = tmp1[k]-sign(tmp1[k])*thr;
                if (fabs(tmp1[k]) < thr) {
                        tmp1[k] = 0;
                        N_zero++;
                }
        w_sum = w_sum + (tmp1[k] - tmp[k])*(tmp1[k] - tmp[k]);

        } /* end of second for loop */

         free(tmp1);
         if (N_zero == 0)
         break;

         gcv = (w_sum*nn)/(N_zero*N_zero);
         if (i == 0) gcv_min = gcv;
         if (i != 0 && gcv < gcv_min) {
                gcv_min = gcv;
                para_best = para;
                }
         N_zero = 0;
         w_sum = 0;

    }/* end of first for loop */

/* start the custermized thresholding based on the obtained best approriate parameter */
    thr = (1-para_best)*thr_old;
    for (k = 0; k < nn; k++) {
         if (fabs(yy[k]) >= thr)
                yy[k] = yy[k]-sign(yy[k])*thr;
         if (fabs(yy[k]) < thr)
                yy[k] = 0;
        }
```

```
    para = -0.01;
    free(tmp);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/********************* Callback function for "EXIT" button ********************/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int CVICALLBACK ExitCallBack (int panel, int control, int event, void *callbackData, int
                                        eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        QuitUserInterface (0);
                        exit(1);
                        break;
                case EVENT_RIGHT_CLICK:
                        break;
                }
        return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/****************** Callback function for "STOP/RESUME" button ****************/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int CVICALLBACK ToggleCallBack (int panel, int control, int event, void *callbackData,
                                                int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_VAL_CHANGED:
                        if(toggle == FALSE) {
                                toggle = TRUE;
                                control_ID = DAQPANEL_TOGGLEBUTTON;
                                }
                        else {
                                toggle = FALSE;
                                control_ID = 1000;
                                }
                        break;
                case EVENT_RIGHT_CLICK:
                        break;
                }
        return 0;
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*************************** sgn(x) function ******************************/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

double sign (double a)
{
        if (a > 0) a = 1;
        if (a = 0) a = 0;
        if (a < 0) a = -1;
        return a;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/******************* Callback function for "CLOSE" button **********************/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int CVICALLBACK ClosePanel (int childpanel, int control, int event, void *callbackData,
                                        int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        HidePanel(childpanel);
                        break;
                case EVENT_VAL_CHANGED:
                        HidePanel(childpanel);
                        break;
                case EVENT_KEYPRESS:
                        HidePanel(childpanel);
                        break;
                case EVENT_GOT_FOCUS:
                        break;
                }
        return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/****************** Callback function for "Show chile panel" button ***************/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int CVICALLBACK ShowPanel (int panel, int control, int event, void *callbackData, int
                                        eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        DisplayPanel (childpanel);
```

```
                    break;
                case EVENT_VAL_CHANGED:
                    DisplayPanel (childpanel);
                    break;
                case EVENT_KEYPRESS:
                    DisplayPanel (childpanel);
                    break;
                case EVENT_GOT_FOCUS:
                    break;
                }
        return 0;
}
```