Application of the Wigner Distribution to Monitoring Cutting Tool Condition

Zheng, Kougen

Center for Micro-Engineering and Metrology,
Department of Engineering,
University of Warwick

This thesis is submitted for the degree of Doctor of Philosophy

31 May 1992
Acknowledgements

While I am indebted to many people for their help throughout the course of my research, I would like to thank, in particular, the following:

My supervisor, Prof. D.J. Whitehouse, for his continuous encouragement and constructive criticism, and originally suggesting the possible use of the ambiguity function, the Wigner distribution, and the Hough transform.

Prof. D.K. Bowen for his supervision while Prof. D.J. Whitehouse was on temporary sickleave.

Dr. D.G. Chetwynd, Dr. J.H. Rakels, Dr. S.T. Smith, and Mr. T.R. Judge, Mr. R. D. Dale-Jones, for proof reading this thesis and many useful discussions.

Mr. R.A. Bridgeland for his assistance with the use of Talyrond 200. Mr. P.R. Bellwood and Mr. D.J. Robinson for their help in building the instrument for vibration measurement.

Finally, but not least, I am very grateful to my family and particularly my wife. Without their support, this thesis would not have been completed.

Zheng, Kougen
Declaration

This thesis is presented in accordance with the regulations for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work described in this thesis has been done by myself except where stated otherwise.

Zheng, Kougen
To my mum!
Summary

This thesis is about the application of the Wigner distribution to cutting tool monitoring and control. After reviewing traditional methods, a new method is proposed. This is to regard the surface texture and geometric error of form of a machined workpiece as the fingerprint of a cutting process, to analyse it, and to extract cutting tool vibration information from it, which can then be used for cutting tool monitoring.

In order to analyse the surface texture effectively, three analysing tools, i.e. the Fourier transform, the ambiguity function, the Wigner distribution (WD), are examined and compared with each other, and it is concluded that the WD is best able to analyse both stationary and nonstationary signals. Furthermore, computer simulation of both chirp signals and frequency modulated signals is then carried out, and it is shown that the WD can be used to extract useful parameters successively.

In order to demonstrate the suitability of the WD for machine tool condition monitoring, first cutting tool vibration are measured directly by two linear variable differential transformers mounted on the cutting tool, and then these measured data about vibration are used to verify those parameters extracted from the surface of the machined workpiece by the WD. It is found that

- the extracted frequencies in both horizontal and vertical direction are within 10% of those measured,

- the extracted amplitudes in both horizontal and vertical direction are highly correlated with those measured.

This result confirms the feasibility of this technique. In spite of being an off-line process, this technique is simple, reliable, and can reveal the direct effect of cutting processes.
Nomenclature

Except where stated explicitly, the signal and its transform are regarded as being continuous.

$A_f(\omega, x)$ The ambiguity function for $f(x)$.

$A_{fa}(\omega, x)$ The ambiguity function for $fa(x)$.

$A_f(\omega, \eta)$ The discrete ambiguity function for $f[n]$.

$A_{fa}(\omega, \eta)$ The discrete ambiguity function for $fa[n]$.

$f(x)$ A real-valued or complex-valued signal.

$fa(x)$ An analytical signal for a real-valued signal $f(x)$.

$F(\omega)$ The Fourier transform for $f(x)$.

$F_a(\omega)$ The Fourier transform for $fa(x)$.

$f[n]$ A discrete, real-valued or complex-valued signal.


$F'(\theta)$ The discrete Fourier transform for $f[n]$.

$F_a(\theta)$ The discrete Fourier transform for $fa[n]$.

$m_f(x)$ The second-order local moment in frequency $\omega$ for $W_f(x, \omega)$.

$M_f(\omega)$ The second-order local moment in space $x$ for $W_f(x, \omega)$.

$m_f[n]$ The second-order local moment in frequency $\theta$ for $W_f(n, \theta)$.

$M_f(\theta)$ The second-order local moment in space $n$ for $W_f(n, \theta)$. 
$p_f(x)$ The zeroth-order local moment in frequency $\omega$ for $W_f(x, \omega)$.

$P_f(\omega)$ The zeroth-order local moment in space $x$ for $W_f(x, \omega)$.

$p_f(n)$ The zeroth-order local moment in frequency $\theta$ for $W_f(n, \theta)$.

$P_f(\theta)$ The zeroth-order local moment in space $n$ for $W_f(n, \theta)$.

$W_f(x, \omega)$ The Wigner distribution for $f(x)$.

$W_{fa}(x, \omega)$ The Wigner distribution for $f_a(x)$.

$W_f(n, \theta)$ The discrete Wigner distribution for $f[n]$.

$W_{fa}(n, \theta)$ The discrete Wigner distribution for $f_a[n]$.

$\Omega_f(x)$ The first-order local moment in frequency $\omega$ for $W_f(x, \omega)$.

$\Theta_f(n)$ The first-order local moment in frequency $\omega$ for $W_f(n, \theta)$. 
Contents

1 Review of Cutting Tool Monitoring and Control .......................... 1
  1.1 Introduction ........................................................................ 1
  1.2 Cutting tool failure and its monitoring ................................ 2
  1.3 Direct methods .................................................................... 4
    1.3.1 Optical methods .......................................................... 4
    1.3.2 Radioactive techniques ............................................... 5
  1.4 Indirect methods .................................................................. 6
    1.4.1 Cutting forces .............................................................. 6
    1.4.2 Drive motor current ..................................................... 8
    1.4.3 Temperature .............................................................. 9
    1.4.4 Vibration ................................................................. 11
    1.4.5 Sound ..................................................................... 13
    1.4.6 Acoustic emission ...................................................... 14
    1.4.7 Workpiece size changes ............................................. 15
  1.5 Surface analysis ................................................................... 16
    1.5.1 Statistical approach ................................................... 17
    1.5.2 Deterministic approach .............................................. 19
  1.6 Comparative tables of traditional methods ........................... 19
  1.7 An approach by the Wigner distribution ................................ 21

2 The Fourier Transform and the Ambiguity Function ................. 22
  2.1 Introduction ....................................................................... 22
  2.2 A few notes on mathematics ............................................... 23
## CONTENTS

2.2.1 The Lebesgue integration ........................................... 24
2.2.2 The Fourier transform ............................................. 24

2.3 The Fourier transform (FT) ......................................... 25
  2.3.1 Definition ....................................................... 25
  2.3.2 Simple properties ........................................... 26
  2.3.3 Application to signals ....................................... 30

2.4 The discrete Fourier transform (DFT) ............................. 30
  2.4.1 Definition ....................................................... 30
  2.4.2 Properties ....................................................... 31
  2.4.3 The fast Fourier transform .................................. 34
  2.4.4 Applications to signals ..................................... 34

2.5 The ambiguity function (AF) ....................................... 40
  2.5.1 Definition ....................................................... 40
  2.5.2 Simple properties ........................................... 41
  2.5.3 Application to signals ....................................... 44

2.6 The discrete ambiguity function (DAF) ........................... 45
  2.6.1 Definition ....................................................... 45
  2.6.2 Properties ....................................................... 46
  2.6.3 Computation of the DAF ..................................... 47
  2.6.4 Application to signals useful in manufacturing .......... 47
  2.6.5 Application to signals with noise ......................... 57

2.7 Conclusion of suitability of the FT and AF ..................... 66

3 The Wigner Distribution .............................................. 67
  3.1 Recapitulation of the problem .................................. 67
  3.2 Introduction ....................................................... 68
  3.3 The Wigner distribution (WD) .................................... 69
    3.3.1 Definition ..................................................... 69
    3.3.2 Simple properties .......................................... 71
    3.3.3 The WD for analytical signals ........................... 74
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.4</td>
<td>Moments</td>
<td>75</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Application to signals</td>
<td>80</td>
</tr>
<tr>
<td>3.4</td>
<td>The discrete Wigner distribution (DWD)</td>
<td>81</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Definition</td>
<td>81</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Basic properties</td>
<td>81</td>
</tr>
<tr>
<td>3.4.3</td>
<td>The DWD for analytical signals</td>
<td>85</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Moments</td>
<td>86</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Application to signals</td>
<td>89</td>
</tr>
<tr>
<td>3.4.6</td>
<td>Application to signals with noise</td>
<td>98</td>
</tr>
<tr>
<td>3.5</td>
<td>Relationship with the AF</td>
<td>107</td>
</tr>
<tr>
<td>4</td>
<td>Extraction of Parameters from the WD</td>
<td>110</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>110</td>
</tr>
<tr>
<td>4.2</td>
<td>The Hough transform</td>
<td>111</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Preliminary</td>
<td>111</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Fundamentals</td>
<td>111</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Examples</td>
<td>113</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Conclusion</td>
<td>116</td>
</tr>
<tr>
<td>4.3</td>
<td>Local moments in frequency</td>
<td>117</td>
</tr>
<tr>
<td>4.4</td>
<td>Complex-valued chirp signals</td>
<td>118</td>
</tr>
<tr>
<td>4.5</td>
<td>Real-valued, chirp signals</td>
<td>124</td>
</tr>
<tr>
<td>4.6</td>
<td>Complex-valued, FM Signals</td>
<td>130</td>
</tr>
<tr>
<td>4.7</td>
<td>Real-valued, FM signals</td>
<td>136</td>
</tr>
<tr>
<td>4.8</td>
<td>Conclusion</td>
<td>142</td>
</tr>
<tr>
<td>5</td>
<td>Direct Measurement of Cutting Tool Vibration</td>
<td>143</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>143</td>
</tr>
<tr>
<td>5.2</td>
<td>Elementary vibratory systems</td>
<td>144</td>
</tr>
<tr>
<td>5.2.1</td>
<td>The equation of systems with one-degree of freedom</td>
<td>144</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Stability</td>
<td>145</td>
</tr>
<tr>
<td>5.3</td>
<td>Free-vibrations</td>
<td>146</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Weak damping</td>
<td>147</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Critical damping</td>
<td>147</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Heavy damping</td>
<td>147</td>
</tr>
<tr>
<td>5.4</td>
<td>Forced-vibrations</td>
<td>147</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Vibration isolations</td>
<td>148</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Undamped dynamic vibration absorbers</td>
<td>148</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Lanchester vibration absorbers</td>
<td>149</td>
</tr>
<tr>
<td>5.5</td>
<td>Self-excited vibrations</td>
<td>149</td>
</tr>
<tr>
<td>5.6</td>
<td>The cantilever</td>
<td>150</td>
</tr>
<tr>
<td>5.6.1</td>
<td>The equation for a cantilever</td>
<td>151</td>
</tr>
<tr>
<td>5.6.2</td>
<td>The natural frequencies</td>
<td>152</td>
</tr>
<tr>
<td>5.7</td>
<td>Measurement of vibration</td>
<td>154</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Traditional method</td>
<td>154</td>
</tr>
<tr>
<td>5.8</td>
<td>The linear variable differential transformer</td>
<td>156</td>
</tr>
<tr>
<td>5.8.1</td>
<td>Operating Principle</td>
<td>156</td>
</tr>
<tr>
<td>5.8.2</td>
<td>Circuit Analysis</td>
<td>158</td>
</tr>
<tr>
<td>5.8.3</td>
<td>Modulation</td>
<td>160</td>
</tr>
<tr>
<td>5.8.4</td>
<td>Features of an LVDT and its justification for tool vibration application</td>
<td>160</td>
</tr>
<tr>
<td>5.9</td>
<td>Demodulation</td>
<td>161</td>
</tr>
<tr>
<td>5.10</td>
<td>A low-pass filter</td>
<td>162</td>
</tr>
<tr>
<td>5.11</td>
<td>Data display and further processing</td>
<td>164</td>
</tr>
<tr>
<td>5.12</td>
<td>Experiments</td>
<td>165</td>
</tr>
<tr>
<td>5.12.1</td>
<td>Equipments</td>
<td>165</td>
</tr>
<tr>
<td>5.12.2</td>
<td>Set up</td>
<td>166</td>
</tr>
<tr>
<td>5.12.3</td>
<td>Results</td>
<td>169</td>
</tr>
<tr>
<td>5.13</td>
<td>Discussion</td>
<td>170</td>
</tr>
</tbody>
</table>

6 Extraction of Cutting Tool Vibration by WD Analysis of Surface Texture 171
6.1 Introduction .................................................. 171
6.2 A mathematical model for cutting tool vibration ............... 172
  6.2.1 Without any vibration .................................. 174
  6.2.2 With z vibration only .................................. 174
  6.2.3 With z and y vibrations ............................... 174
  6.2.4 Vibrations in the x directions ....................... 175
6.3 Roundness measurement .................................... 175
6.4 Band pass filtering ....................................... 180
6.5 The WD and its local moments ............................ 184
  6.5.1 The WD ................................................. 184
  6.5.2 The local moments in frequency .................... 185
6.6 Extraction of vibration information ....................... 186
  6.6.1 a\z and f\z ........................................ 186
  6.6.2 a\y and f\y ........................................ 187
6.7 An example of extracting vibration data from roundness .... 188
6.8 Results .................................................. 190
6.9 Conclusions .............................................. 193
6.10 Simulation ............................................... 194

7 Conclusions .................................................. 197

References ..................................................... 201

A The Fast Fourier Transform .................................. 213

B The Ambiguity Function ...................................... 219

C The Wigner Distribution .................................... 225

D The Hough Transform ........................................ 231

E Extraction of Useful Parameters ............................. 243

F Extraction of Vibration by the WD ........................... 257
Chapter 1

Review of Cutting Tool Monitoring and Control

1.1 Introduction

The remarkable progress in every facet of computer science during the past 50 years has brought about rapid changes in every field. Manufacturing industry is no exception. The first contribution of computer technology was computer numerical control (CNC), then came computer aided design (CAD) and computer aided manufacturing (CAM). Today, flexible manufacturing system (FMS) and computer-integrated manufacturing (CIM) are emerging and promise to change manufacturing industry beyond recognition.

One crucial part of FMS and CIM is to be able to successfully and automatically monitor and control the machining process. This includes

1. monitoring the machine, in particular, spindle bearings, guideways, feed drives.

2. monitoring the cutting tool, such as tool wear and breakage, tool vibration.

3. checking the workpiece for its dimensions, shape and surface texture.

In this thesis, it is the monitoring of the cutting tool in turning that is investigated. This is because it is widely used, and the cutting tool has an important
influence on productivity as well as performance of workpieces (Whitehouse and Zheng 1992).

1.2 Cutting tool failure and its monitoring

There are many kinds of tool failure, the most common causes being

1. temperature failure, where the tool temperature becomes high enough to cause plastic deformation at the cutting edge,

2. fracturing, either a complete tool failure or small chipping of the cutting edge, caused by fatigue or excessive forces,

3. gradual wear, including both crater wear and flank wear,

4. tool vibration etc.

For both crater wear and flank wear, see Figure 1.1 and Figure 1.2

![Figure 1.1: Regions of tool wear (from Boothroyd 1975, p 109).](image-url)
1.2 Cutting tool failure and its monitoring

Figure 1.2: Features of single-point-tool wear (from Boothroyd 1975, p 112). KT is the crater depth measured at the deepest point of the crater, VB is the average wear land width in the central portion of the active cutting edge, VC is the width of the flank wear land at the tool corner, VN is the width of the wear land at the wear notch.
1.3 Direct methods

With regard to monitoring the cutting tool in turning, there are three types of methods (Cook 1980; Jones 1989; Kegg 1984; Li and Mathew 1990; Micheletti, Koenig and Vitor 1976; Tlusty and Andrews 1983; Tönshoff et al 1988):

1. direct methods, involving measurement of the actual tool wear and fracture from a cutting tool (such as optical techniques and radioactive methods).

2. indirect methods, involving measurement of parameters correlated with cutting processes (such as cutting forces, vibration, sound, power input, acoustic emission, cutting temperature)

3. surface analysis, based on the fact that the surface texture of a machined workpiece is the fingerprint of a cutting process.

In this chapter, the above methods will be examined. After this, a novel method of machine tool monitoring based on the Wigner distribution will be introduced briefly.

1.3 Direct methods

1.3.1 Optical methods

The lapping-comparator technique has been employed for investigating tool wear (Tsao et al 1986). First, a transparent thermoplastic mould of a tool crater was made, then lapped to show contour lines on an optical comparator. These contours were seen as clear, sharp lines of high contrast. The depth of the crater can be easily and accurately measured by reading the height of the mould on the optical comparator screen.

A fiber optic sensor for in-process measurement of tool flank wear has been developed (Giusti 1979). This was capable of giving information about tool flank wear conditions, and could scan different zones of the wear-land. It was simple and easy to implement. Another method for measurement of flank wear has been presented by Jeon and Kim (1988). This method was based on real time
vision technology in which the tool was illuminated by the beam of a laser and the wear zone was viewed by means of a Vidicon camera. The image was then converted into digital form and processed to detect the wear land width. It was reported that this method was capable of detecting tool wear to an accuracy of 0.1 mm within 1.7 second processing time. Pedersen (1990) has also developed a prototype experimental computer vision system for flank wear measurement. The wear measured conformed with the traditional three-stage wear pattern normally observed for cutting tools (initial, steady-state, and terminal wear.) The measurement program took about 7 seconds to execute.

The morphology of the cutting tool has been examined by a TV camera (Matsushima, Kawabata, and Sata 1979), and the wear pattern was extracted by an image processing technique and classified into five predetermined categories. When an undesirable morphology was found, the tool material or the tool geometry was changed according to a decision table which was constructed in advance by a learning algorithm. In the experiments, the system was able to fulfill the intended functions.

The optical method has been applied to a NC lathe (Giusti, Santochi and Tantussi 1984). The tool wear was sensed by a TV camera and then processed by a computer to extract the actual value of wear, which was then used to control the lathe.

Optical sensing can only be used between cutting cycles when the tool is removed from the workpiece, and as such is not strictly an in-process technique. However, it appears to be accurate and reliable, especially when machine vision is used. It has its share of problems though, in that there can be instances when it is difficult to detect tool wear if a built-up edge or metal deposit exists.

### 1.3.2 Radioactive techniques

A simple and safe method has been developed in order to provide information prior to tool replacement (Cook and Subramanian 1978). This method involved attaching a small quantity of radioactive material (less than $10^{-8}$ Curie) to the
1.4 Indirect methods

flank of the tool. At the end of each cutting cycle, the tool was quickly tested to determine whether the spot was still there, and so whether the tool had reached some predefined state of wear and whether the tool had to be replaced. This is in effect a "Go, No-go" gauge.

In practice, some measures must be taken to minimize the effects of radiation on the shop floor. Moreover it is inevitable that operators are somewhat prejudiced against using methods involving radioactivity no matter how small the level is.

1.4 Indirect methods

There are many theoretical models for indirect methods, for example (Bhattacharyya and Ham 1969; Koren 1978; Kramer and Suh 1980; Lenz, Katz and Ber 1976; Ramalingam, Peng and Watson 1978; Ramalingam and Watson 1978; Tlusty and Masood 1978; Usui and Hirota 1978; Wu 1964). These are interesting and useful, however there are more practical methods which have found significant use in industry. These methods will be explained below.

1.4.1 Cutting forces

Among the many parameters which can be used to characterise tool wear, one is the cutting force. This has the advantage that it is relatively easy to measure. It can be used to monitor the cutting processes because it changes as the tool wears. It usually get greater because of the increase in the coefficient of friction between tool and workpiece. Force sensing methods have been reported that are very sensitive, in fact in some cases more sensitive than vibration and power measurements (Martin et al 1986).

An early work (Micheletti, De Filippi and Ippolito 1968) has claimed that it is difficult to derive accurate information on tool wear based on measurements of cutting forces in turning. It was shown that significant increases in force occurs only at the moment of tool failure. Therefore this technique was said to be
limited to the detection of tool failure only and not wear in its many forms. In spite of this, cutting forces have been widely employed by many research workers for monitoring cutting processes.

Some theoretical models based on force measurements for on-line tool wear and breakage sensing have been presented and verified by experiments (Akgerman and Frisch 1971; Danai and Ulsoy 1987; Koren and Lenz 1970; Shiliam 1971; Usui and Hirota 1978). One (Koren 1986) is based on the fact that as a tool wears, the tool face force decreases with increasing crater wear while the tool clearance force increases with increasing wearland. From measuring these two forces, tool wear can be estimated.

It has been experimentally shown that the relationship between the feed force and the feed per revolution is strongly influenced by the tool wear. Based on this, tool wear can be detected by the cutting force measurement (Uehara, Kiyosawa and Takeshita 1979). A special NC program consisting of a flank wear detection procedure and crater wear detection element was inserted at the beginning of the NC tape for every workpiece. In the tests, both VB and KT (see Figure 1.2) were determined quantitatively. The limit of the method was 0.15 mm for VB and 20 \( \mu m \) for KT. The method was also effective in detecting chipping of cutting tools.

With a large rake angle, the feed force at steady levels was negative, but achieved relatively high positive levels for short periods of time at both the beginning and the end of the cut (Colwell 1971). The level and duration of these positive excursions increased rapidly with even small amounts of tool wear, even though it was also influenced by the cutting conditions. A phase shift between the main cutting force and feed force with a zero rake angle cutting tool was considered to be due to the rubbing effect of the initial cut. Both phenomena offered the possibility of monitoring tool wear in initial rough cuts.

Cutting forces have been sensed during turning cuts carried out on an NC lathe (Colwell 1975). It was found that the influence of the tool was much more evident during the dwell and set-up periods than in steady state cutting. During the dwell period, the main cutting force \( F_c \) and the feed force \( F_n \) dropped to equal
each other with a sharp tool while the value of $F_c$ dropped to a level greater or less than $F_n$ according to the flank wear. Therefore the feed force, or even better, the ratio of the feed component to the cutting component was a sensitive means for monitoring tool wear.

A method of wear estimation for carbide tools, using a function of the cutting forces, was presented by Mackinnon, Wilson and Wilkinson in 1986. In this, a wear index was proposed to categorize the state of wear on a tool as a percentage. For perfect tools the value was defined as 0% while for extremely badly damaged tools it was given a value of 100%. Tests demonstrated that the strategy was reliable. Although the system could not prevent breakage, by anticipating it, time-wasting due to the continuation of machining following breakage was avoided. Moreover, the strategy could halt cutting when a tool became severely affected by plastic deformation since this also resulted in a large increase in the value of the force ratio.

It should be stated in conclusion that although the cutting force method is one of the most commonly used techniques for the detection of tool failure, it can not be generalised because tool wear and failure have a complex relationship with cutting forces and results can be different for sensing tool wear in similar studies.

### 1.4.2 Drive motor current

During the cutting process, the current consumed by the spindle motor or feed motor is related to the output torque of the motor and therefore the tool load. Hence this current can be used for monitoring machine tool just like cutting forces.

The current from the spindle motor of an NC lathe has been measured (Matsushima, Bertok and Sata 1982) by a current transformer, then rectified, amplified, and further low pass filtered. The resulting signal was found to drop instantaneously and soon to recover to a certain level when tool breakage occurred. With the aid of the statistical quality control concept, a lower limit was
generated from the normal process data. The breakage of a cutting tool was
detected when a data point intersected the lower limit threshold. Furthermore,
it was found that under the constant spindle speed cutting conditions, the per­
centage increase of the current from the beginning until the end of a tool’s useful
life was approximately constant if the same material was used. This result was
employed to develop a method by which to judge whether a tool reached its ser­
vice life expectancy. A disadvantage was the small bandwidth of its frequency
response.

Meter’kov and Liberman (1989) have proposed a method to use the current
from a spindle motor — by isolating the frequency part related to tool breakage
by means of band pass filters. Third order filters were suggested as an ideal
choice.

A model based on the current from a feed motor has been presented and
verified experimentally on a lathe (Stein et al 1984).

The current measurement system is relatively simpler and cheaper, more
durable and flexible than a dynamometer. It has been found to be reliable in
monitoring tool breakage at medium and heavy cuts (Novak and Ossbahr 1986).
However, it is less sensitive for tool wear sensing when compared to force sensing
and vibration measurement (Martin et al 1986).

1.4.3 Temperature

During the metal cutting, heat is generated in the region of tool cutting edge
(Figure 1.3), therefore temperatures are high. These temperatures have a signifi­
cant influence on the tool wear. As a result, the cutting temperature can be used
to monitor tool failure.

However, in a study by Boothroyd, Eagle, and Chisholm (1967), it is suggested
that for a variety of reasons, reliability of the cutting temperature for monitoring
tool wear is doubtful:

1. the thermal voltage (from the thermocouple used for measurement of the
1.4 Indirect methods

Figure 1.3: Heat generation in metal cutting (Boothroyd 1975, p93)

cutting temperature), and hence the cutting temperature, were sensitive to the cutting conditions.

2. the thermal voltage, and hence the cutting temperature, tended to stabilize after an initial increase.

3. the noise signal was about 60 percent of the signal’s level due to tool wear, which complicated the task of obtaining the trend data from the signal.

4. the increase in the signal due to tool wear in either absolute terms or as a percentage of the initial signal level, did not bear a consistent relation with the cutting variables.

Zakaria and El Gomayel (1975) have also doubted the reliability.

In spite of the doubt, many methods based on temperature have been developed.

The temperature at a position on the cutting tool remote from the cutting edge has been measured by a thermocouple (Groover, Karpovich and Levy 1977). It was found that the temperature measured had a strong correlation with tool wear. Thus,

\[ TW = W_0 + WR(T - T_0) \]  

(1.1)
where $TW$ (mm²) is tool wear, $W_0$ (mm²) is the initial interface area between tool and chip before wear begins, $T, T_0$ (°C) are the temperature measured during cutting and at the beginning of cutting, $WR$ (mm²/°C) is the wear rate. Under various conditions, the correlation coefficient was between 0.837 and 0.978. At the initial cutting, the error was the largest. This method is limited because the work-tool thermocouple method gives no indication of the distribution of temperature along the rake face.

Another method is to employ the tool-work thermocouple technique. In this technique, the emf between the tool and workpiece is taken as a measure of the mean temperature of chip-tool interface. Using this method, the tool and workpiece should be electrically isolated from the machine tool structure. Akgerman and Frisch (1971) have used this method for tracking tool failure and controlling the cutting process.

The temperature at the cutting edge of a tool in machining glass fiber reinforced plastic (GFRP) has been measured by a special thermocouple method (Sakuma and Seto 1981). In this method, two insulated wires were set in a hole drilled in the workpiece, and fixed by bonding. When these wires were cut together with the workpiece, the hot junction of the thermocouple was established, the temperature at the cutting edge could be measured and used to estimate the flank wear. However, the widespread use of this method is unlikely.

### 1.4.4 Vibration

As metal is being cut, the workpiece and the chips rub against the worn tool and produce vibration which can be measured. This information may then be used in various ways for tool failure control.

According to the investigation by Martin, Mutel and Drapier (1974), the vertical tool vibrations in the course of stable machining were almost sinusoidal, with frequency equal to the natural frequency of the tool. In the experiments conducted, the power of the acceleration signal obtained by spectral analysis was a linear function of the cutting speed and of the tool wear, and varied in the ratio
1.4 Indirect methods

of 1:10 between a new tool and a worn tool. These experiments correlated with theory.

The inter-relationship between tool wear and the power spectrum of the flexural vibrations of the tool during cutting has been investigated (Del Taglia, Portuna and Toni 1976). It was discovered that

1. In the frequency range up to 2.5 kHz, a very small percentage of the total power of the acceleration signal varied in a typical way with wear. It increased up to seven times, while tool wear increased from a very small value to about 1.3-1.5 mm. A further increase of wear beyond this point caused the power contained in this frequency range to fall rapidly to the values found for small amounts of wear.

2. The cross power spectra, obtained from each recording and the previous recording in the range of 0-2.5 kHz, showed a trend similar to that of the power spectra with increasing of wear.

3. The mean coherence of a recording and the previous one in the same range lessened in a fairly predictable manner with wear up to a certain degree of wear and then increased again.

The sensitivity found in the fixed frequency range of 0-2.5 kHz was fairly satisfactory, at least for a certain class of machine tools.

The Data Dependent System (DDS) has been developed (Pandit 1977, 1978; Pandit and Kashou 1982) to measure tool wear by using the signal from an accelerometer mounted on the tool holder at a safe distance away from the cutting process. The DDS picked the mode of vibration most sensitive to tool wear and gave its power contribution, which decreased with increasing wear at the beginning, reached a minimum when the critical point of tool wear was reached, and increased again, much the same way as a rate of wear curve. The trend remained unchanged under different cutting conditions. Therefore the minimum actual power contribution or minimum tool acceleration could be used on-line to monitor tool wear.
1.4 Indirect methods

By comparison of relative vibration spectra of the machine tool (an experimental spectrum and a specified basic spectrum), the state of a machine tool can be assessed (Selezneva 1987). This can be used to find and eliminate vibration sources.

In summary, vibration signals vary with tool failure in certain frequency ranges and are widely employed in tool condition monitoring. With the growing sophistication of transducers and instrumentation used in vibration measurement and analysis, this technique will become more practical and cost-effective.

1.4.5 Sound

Sound produced during the metal-cutting process contains all sorts of information, some components of which have been used to monitor the condition of the cutting edge.

As early as 1968, Weller, Schrier and Weichbrodt had built an electronic-mechanical system which utilized sonic signals to detect the degree of cutting edge wear in cutting and automatically trigger a cutting edge change.

Machining noise has been found to exhibit a characteristic frequency at around 4–6 kHz for a large variety of workpiece-material combinations and operating conditions (Lee 1986). At the characteristic frequency, the SPL (sound pressure level) appeared to be distinctively higher than free running and showed a good correlation with tool wear. The drop of SPL before a rapid increase in the maximum flank wear showed that the tertiary wear zone has been reached. This could be used to predict the onset of tool failure. However due to the high ambient noise level in factory environments (typically around 90 dB), this method is perhaps impractical.

Tool flank wear has been detected by measuring the emitted low frequency noise from the rubbing action of the tool and workpiece (Sadat and Raman 1987). A significant increase in the noise level in the frequency range 2.75–3.50 kHz was observed from comparisons of the noise spectra of a sharp tool with a worn tool, and this increase was high during the initial stages of tool wear and then
tended to saturate. However, this noise spectrum was influenced by other cutting conditions.

1.4.6 Acoustic emission

This method is based on acoustic emission from the cutting process. During the cutting process, deformation and fracture will occur which causes the strain energies stored in the solid material to be released. This results in acoustic emission (AE).

A quantitative model relating the peak value of the RMS AE signals, to both the fractured area and the resultant cutting force at tool fracture, has been developed (Diei 1987). With certain approximations, it can be written as

\[ V_p = C F (\Delta A)^{1.5} \]

where \( V_p \) is the peak AE RMS voltage at tool fracture, \( F \) is the resultant cutting force, \( \Delta A \) is the area of the fracture surface, \( C \) is a constant related to the material and the geometry. It was pointed out that the theoretical prediction agreed with the experimental results.

Acoustic emission has been detected at the end of tool shank and processed in a number of different ways (Iwata and Moriwaki 1976). The frequency spectrum increased as the tool wear increased, but tended to saturate after further increase of tool wear. The total count of acoustic emission over certain time period was also taken, and had good correlation with tool wear, which could be used for in-process sensing of tool wear.

Significant bursts of AE were generated at the moment of tool breakage (Lan and Dornfeld 1984). Furthermore, it was found that the amplitude of the RMS energy of burst AE events was larger if the tool fracture area was larger.

A technique based on spectral analysis and pattern recognition of AE signals has been developed (Emel and Kannatey-Asibu 1988), and applied to sample sets of data obtained under fixed cutting conditions. It was found that the reliable
frequency range was between 100 kHz and 1 MHz. Progressive tool wear was found to be associated with increasing power within the 400 to 700 kHz band, while catastrophic tool failure had a power spectrum spanning a much wider frequency range. The reliability of detecting tool failure was between 84% to 94%.

Dalpiaz and Remondi (1988) have investigated the relationship between the AE signal on one hand and both tool deterioration phenomena and cutting conditions on the other, by performing turning tests under practical conditions. Many parameters for characterizing AE were considered. It was found that the AE burst frequency matched the chip breaking frequency, thus confirming that chip breakage is the source of the signal bursts.

In conclusion, AE sensing techniques appear to have a quick response time and consistency. It seems to be more sensitive to tool fracture than cutting force measurements and tool vibration analysis, although no convincing experimental evidence of the relative sensitivity has yet been demonstrated.

### 1.4.7 Workpiece size changes

A device for sensing the change of the workpiece diameter during turning operations has been developed in order to measure tool wear (El Gomayel and Bregger 1986). The change of the workpiece diameter was sensed by electromagnetic sensors which gave a voltage directly related to the gap between the sensor and the workpiece. Two sensors operating in differential modes were used so as to compensate for deflections and vibrations. Small amounts of wear could be detected with this device. For example, using a K21 carbide insert at a cutting speed of 1.79 m/s (350 feet/minute), a feed of 0.528 mm (0.0208 inch), a depth of cut 2.54 mm (0.100 inch), a change in flank wear from 0.206 to 0.241 mm (0.0081 to 0.0095 inch) was measured. At the same time the nose wear remained constant at 15.4 µm (0.0006 inch). The diameter increase was found to be 7.62 µm (0.0003 inch).

During straight turning, tool wear can also be detected by measuring the
change in distance between a tool holder and work surface using a stylus which is mounted on the tool holder (Suzuki and Weinmann 1985). The motion of the stylus was sensed by a displacement transducer, such as an eddy current type. The result of the experiments was found satisfactory. This tool wear sensor is inexpensive because of its simplicity. The drawbacks are that inaccurate slideway motion, changes in the feed force, and the large distance between tool tip and stylus can introduce errors, and that it cannot be readily applied to contour cutting.

Pneumatic gauges have been used for in-process correction of tool wear by monitoring the distance between the tool post or tool and workpiece surface (Bath and Sharp 1968; Stoferle and Bellmann 1975).

Although these measuring systems can detect tool wear by diameter increases, they are not able to diagnose tool failure modes. For example, it is not possible to distinguish between nose wear and flank wear using this technique. Moreover, errors can also be introduced by thermo-expansion of the workpiece and by inaccurate movements of the machine tool.

1.5 Surface analysis

As pointed out by Whitehouse (1978), the surface and its measurement provide a link between the manufacturing workpiece and its function.

As Figure 1.4 shows, the surface can be used to predict how well the parts will function. However, the question arises of how it can also be used to monitor the manufacturing process. The surface texture is, in fact, the fingerprint of the whole machining process, therefore it contains much information about the process, particularly tool wear and vibration. So through analysing the surface, it is possible to estimate the severity of vibration, which enables control of the process. For this method, there are two approaches: one is based on probability and stochastic theory, and the other is deterministic.
1.5 Surface analysis

1.5.1 Statistical approach

Parameters for characterising the surface

In order to measure the surface, there are quite a few quantities to choose from. Here $R_a$, $R_q$, $\Delta_q$, and $\lambda_q$ are used. Their definitions are given as

$$R_a = \frac{1}{l} \int_0^l |y(x)| \, dx$$  \hspace{1cm} (1.2)

$$R_q = \sqrt{\frac{1}{l} \int_0^l y^2(x) \, dx}$$  \hspace{1cm} (1.3)

$$\Delta_q = \sqrt{\frac{1}{l} \int_0^l \left( \frac{dy(x)}{dx} \right)^2 \, dx}$$  \hspace{1cm} (1.4)

$$\lambda_q = 2\pi \frac{R_q}{\Delta_q}$$  \hspace{1cm} (1.5)

where $y(x)$ is the profile with respect to the reference line (the reference line is chosen to represent the form of the profile usually such that the areas enclosed above and below the line are equal), $x$ is the distance along a profile, and $l$ is the evaluation length.

To measure these parameters, there are many instruments: optical and non-optical. For non-optical, there are the stylus instrument, electron microscope, scanning tunnelling microscope, and so on. Among them, the stylus instrument,
such as Talysurf VI, is most widely used and it provides all national standard parameters despite the fact that it is slow, not in-process, only provides information for a profile section of surface not an area of surface, and may cause undesirable permanent damage to the surface.

For optical instruments, they can be further divided into those which can record the topographic structure of surface, such as the Foucault knife probe, or interference microscopy, and those which gives parametric values such as $R_q$.

**Monitoring**

Since $R_q$, $\Delta_q$, and $\lambda_q$ are determined by the cutting tool and cutting condition, we can use them for evaluating the tool wear and monitoring cutting processes.

An optical fibre transducer has been used for in-process indication of surface roughness during a finish turning process (Spurgeon 1974). The transducer was used to trace the same path as the cutting tool, the reflectivity of light from a newly turned surface varied inversely as the roughness of that surface. With the surface having $1\mu m < R_a < 3\mu m$ ($R_a$ is the arithmetical mean deviation), such an optical transducer gave a good indication of surface roughness and thus, indirectly tool wear (for tool nose radius wear has a direct correlation with surface roughness).

Based on diffraction, Rakels and Hingle (1986) have developed an optical instrument capable of in situ measurement of information about the surface roughness and therefore the behaviour of the machine tool which produced the component.

Rau and Huebner (1986) has constructed an optoelectronic device for evaluating the distribution of the scattered light reflected from the surface. A characteristic can be derived from the optical signal in such a way that the results correspond to the frequency analysis of the testing machine. It was claimed that economic 100% control of the shafts with regard to chatter marks was possible if this method applied.
1.5.2 Deterministic approach

Besides the approach on statistics, the deterministic approach can be used. This has been tried by many researchers.

Raja and Whitehouse (1983) have applied complex demodulation technique to surface analysis. It was shown that this technique enabled small changes in the manufacturing process to be identified quickly and positively via the surface profiles. It was predicted that this type of technique would become necessary in future for the control of the manufacturing and also for monitoring the condition of the machine and tool. Further work (Raja and Whitehouse 1984) has confirmed that it is a real possibility to use surface profiles for machine tool surveillance.

Hingle (1986) has studied tool wear monitoring by Fourier analysis of the surface of a component turned by an NC lathe. A computer simulation was performed and verified by experiments. It was found that the maximum value of the power spectral density reached a minimum value at the useful limit of tool life. Comparison of the time to reach the minimum value with tool life time showed good correlation.

The above methods all used Fourier analysis in one way or another to analyze the surface texture of a workpiece. This is not surprising, it is well known that Fourier analysis can be used to analyse stationary signals effectively. However, Fourier analysis is not satisfactory for nonstationary signals. (Fourier analysis will be studied in detail in Chapter 2.) This means that it cannot reveal all the information contained in a nonstationary signal because the surface texture is a very complicated spatial signal and contains nonstationary as well as stationary signals. As a result, the practical application of the above methods are limited.

1.6 Comparative tables of traditional methods

The major advantages and disadvantages of the aforementioned techniques are outlined in Table 1.1
### Table 1.1: Comparative tables of traditional methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Advantages and Disadvantages</th>
<th>General Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direct Methods</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optical methods</td>
<td>accurate, can be expensive, sometimes difficult to implement</td>
<td>straightforward, reliable, not in-process</td>
</tr>
<tr>
<td>Radioactive technique</td>
<td>problem of radiation, safety issue, qualitative results only</td>
<td></td>
</tr>
<tr>
<td>Cutting forces</td>
<td>commonly used, effective, sometimes difficult to measure at the tip</td>
<td></td>
</tr>
<tr>
<td>Motor currents</td>
<td>simpler, less sensitive, can have significant time delay, small bandwidth</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>sensitive, difficult to implement</td>
<td>in-process, influenced by cutting conditions, complicated signal processing sometimes required</td>
</tr>
<tr>
<td>Vibration</td>
<td>widely used, practical, cost-effective, complicated signal processing sometimes required</td>
<td></td>
</tr>
<tr>
<td>Sound</td>
<td>similar to vibration method, but less practical</td>
<td></td>
</tr>
<tr>
<td>Acoustic emission</td>
<td>quick, consistent, sensitive</td>
<td></td>
</tr>
<tr>
<td>Workpiece sizes</td>
<td>cheap, only for cylindrical workpieces, less sensitive</td>
<td></td>
</tr>
<tr>
<td><strong>Indirect Methods</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistical approaches</td>
<td>simple, reliable, giving average results only, slow</td>
<td>not in-process, control of manufacturing,</td>
</tr>
<tr>
<td>Deterministic approaches</td>
<td>accurate, reliable, limited if only the Fourier transform is used</td>
<td>prediction of performance</td>
</tr>
</tbody>
</table>
1.7 An approach by the Wigner distribution

Upon considering the comparative table in the previous section, it appears that the method of analysing surface texture by a deterministic approach seems more promising than others. This method is based on the fact:

1. that the surface texture is the fingerprint of the cutting process which generated the surface and contains all sorts of information about the process, and

2. that extracting this information enables us to control machining processes and predict the performance of the workpiece.

However, the wide use of this method has so far been limited. This is because the Fourier transform, which is only suitable for stationary signals, has been used.

In order to analyse all kinds of surface texture effectively, three mathematical analysing tools, i.e. the Fourier transform (FT), the ambiguity function (AF), the Wigner distribution (WD) will be examined in some detail, and their suitability for the task will be assessed. It emerges that the WD is most useful (Chapter 2-3) (Whitehouse and Zheng 1992.)

The rest of the thesis is devoted to validating the technique both by simulated and practical examples (Chapter 4, 5, and 6) (Zheng and Whitehouse 1992).
Chapter 2

The Fourier Transform and the Ambiguity Function

2.1 Introduction

In engineering, there are various types of signals. The type of signal to be analysed has a very important influence on both the type of analysis to be carried out and the choice of analysis parameters. Table 2.1 shows the basic types of signals.

<table>
<thead>
<tr>
<th>A signal $f(x)$ where $x$ may be distance, time, or other variable</th>
<th>stationary</th>
<th>non-stationary</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic</td>
<td>random</td>
<td>deterministic</td>
</tr>
<tr>
<td>random</td>
<td>determinant</td>
<td>random</td>
</tr>
</tbody>
</table>

The most fundamental division is into stationary and nonstationary signals. The stationary signal $f(x)$ is interpreted as being those whose average properties do not vary with $x$ and is thus independent of the particular sample record used to determine them. This applies to both deterministic and random signals. Stationary deterministic signals $f(x)$ are usually regarded as those whose Fourier spectrum do not vary with $x$ while stationary random signals are treated as those whose statistics are independent of $x$. Nonstationary signals are of course
understood as those that are not stationary.

In order to analyse signals, the Fourier transform (FT) is usually employed in practice. Now the question is whether the FT is able to analyse both stationary and nonstationary signals. If not, is there any function which can do so? To answer these questions, the FT, the ambiguity function (AF), and the Wigner distribution (WD) will be examined and their suitability for analysis of signals will be assessed.

For the FT, most of its properties are well known. Therefore most of its proofs are omitted, which can be found in any standard textbook, for example, see (Papoulis 1962).

Because of its practical usage, the discrete Fourier transform is also discussed. The AF and WD, in both continuous and discrete forms, are similarly examined (Whitehouse and Zheng 1992.)

In order to be systematic and emphasize the similar nature of the variables, the following symbols are chosen to represent distance and frequency variable in continuous and discrete cases signals.

<table>
<thead>
<tr>
<th>continuous signals</th>
<th>discrete signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>distance (continuous)</td>
<td>frequency</td>
</tr>
<tr>
<td>distance (discrete)</td>
<td>frequency</td>
</tr>
<tr>
<td>$x$</td>
<td>$\omega$</td>
</tr>
<tr>
<td>$\chi$</td>
<td>$\omega$</td>
</tr>
</tbody>
</table>

### 2.2 A few notes on mathematics

In this chapter and this thesis as a whole, there is a good deal of mathematics. In order to be precise and clear, a few important points about the mathematics used need to be mentioned.
2.2 A few notes on mathematics

2.2.1 The Lebesgue integration

In engineering, the Riemann integral (Rudin 1976, Chapter 6) is usually used. The Riemann-integrable functions are subject to rather stringent continuity conditions. Moreover, many limit operations cannot be carried out nicely. For example, limits of Riemann-integrable functions may fail to be Riemann-integrable. However, the Lebesgue integral (Rudin 1976, Chapter 11) does not have these problems. The set of Lebesgue-integrable functions is larger than the set of Riemann-integrable functions. In fact, every Riemann-integrable function is also Lebesgue-integrable. Besides this, in Lebesgue theory, many limiting operations can be handled very well. In a word, Lebesgue integration is complete while Riemann integration is not. This is analogous to the fact that the real number system is complete while the rational number system is not (because of irrational numbers). As a result, Lebesgue integration rather than Riemann integration is chosen here.

Summation is treated as a special case of Lebesgue integration. Therefore, many of the theorems about integration can be readily applied to summation (Rudin 1986).

2.2.2 The Fourier transform

In this thesis, the FT and the inverse FT are widely used. For a signal \( f(x) \), its FT is not always defined. In order to be valid, \( f(x) \) must belong to some set of functions. Here, \( f(x) \) is assumed to be in \( L^2 \) where

\[
L^2 = \{ f(x) \mid \left( \int_{-\infty}^{\infty} |f(x)|^2 dx \right)^{\frac{1}{2}} < \infty \} \tag{2.1}
\]

In engineering, this set is the set of functions which have finite energy.

If \( f(x) \) is not in \( L^2 \), it is then extended to \( L^1 \)

\[
L^1 = \{ f(x) \mid \int_{-\infty}^{\infty} |f(x)| dx < \infty \} \tag{2.2}
\]
Usually, this is the set of absolutely integrable functions. Note that $L^2$ is a proper subset of $L^1$. For more, see (Rudin 1986).

If $f(x)$ does not even belong to $L^1$, then $f(x)$ is extended to the generalized function or distribution. For more, see (Rudin 1973).

If a signal $f(x)$ is in $L^2$, or $L^1$, or generalized functions, all the theorems about the FT will be valid (Rudin 1986, Chapter 9; Rudin 1973, Chapter 7).

### 2.3 The Fourier transform (FT)

The Fourier transform (mainly due to the French engineer J.B.J. Fourier) was developed about two hundred years ago. Since then, it has found wide applications in every area of science and technology (Brigham 1988; Oppenheim 1983).

#### 2.3.1 Definition

Let $f(x)$ be a complex-valued signal of a real variable $x$, then its *Fourier transform* (FT) $F(\omega)$ is defined (Papoulis 1962) as

$$ F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-j\omega x} dx \tag{2.3} $$

where $x$ is normally a time or spatial variable, and $\omega$ is its corresponding frequency variable.

From $F(\omega)$, its *inverse Fourier transform* (IFT) $f(x)$ can be constructed by

$$ f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega x} d\omega \tag{2.4} $$

Eqn 2.4 is usually called the *inversion formula* or *inversion theorem*. 
Since \( f(x) \) and \( F(\omega) \) are closely related, they are termed a *Fourier transform pair*, this relationship can be represented by the notation

\[
f(x) \overset{\mathcal{F}}{\leftrightarrow} F(\omega)
\]  

(2.5)

### 2.3.2 Simple properties

There are many useful properties about the Fourier transform, particularly those outlined below (Papoulis 1962; Rudin 1986). In order to avoid repetition, some Fourier transform pairs that will be used in this thesis are listed below:

\[
\begin{align*}
  &f(x) \overset{\mathcal{F}}{\leftrightarrow} F(\omega) \\
  &f_1(x) \overset{\mathcal{F}}{\leftrightarrow} F_1(\omega) \\
  &f_2(x) \overset{\mathcal{F}}{\leftrightarrow} F_2(\omega) \\
  &g(x) \overset{\mathcal{F}}{\leftrightarrow} G(\omega) \\
  &h(x) \overset{\mathcal{F}}{\leftrightarrow} H(\omega) \\
  &m(x) \overset{\mathcal{F}}{\leftrightarrow} M(\omega)
\end{align*}
\]

**Symmetry**

\[
F(x) \overset{\mathcal{F}}{\leftrightarrow} 2\pi f(-\omega)
\]

**Sum formula**

Let \( a_1 \) and \( a_2 \) be two arbitrary constants, then

\[
a_1 f_1(x) + a_2 f_2(x) \overset{\mathcal{F}}{\leftrightarrow} a_1 F_1(\omega) + a_2 F_2(\omega)
\]  

(2.6)
2.3 The Fourier transform (FT)

Spatial scaling

If \( a \neq 0 \) is a real constant, then

\[
f(ax) \leftrightarrow \frac{1}{|a|} F\left(\frac{\omega}{a}\right)
\]

(2.7)

Spatial shifting

If \( f(x) \) is shifted by a constant \( x_0 \), then its FT remains the same, but a linear term \(-x_0\omega\) is added to its phase angle, i.e.,

\[
f(x - x_0) \leftrightarrow F(\omega)e^{-j\omega x_0}
\]

(2.8)

Frequency shifting

If \( \omega_0 \) is a real constant, then

\[
f(x)e^{j\omega_0 x} \leftrightarrow F(\omega - \omega_0)
\]

(2.9)

Spatial differentiation

\[
\frac{d^n f(x)}{dx^n} \leftrightarrow F(\omega)(j\omega)^n
\]

(2.10)

Frequency differentiation

\[
f(x)(-j x)^n \leftrightarrow \frac{d^n F(\omega)}{d\omega^n}
\]

(2.11)

Moment theorem

This means that

\[
m_n = j^n \frac{d^n F(0)}{d\omega^n}
\]

(2.12)

where \( m_n \) is called the \( n \)th moment of \( f(x) \), defined by

\[
m_n = \int_{-\infty}^{\infty} f(x)x^n dx
\]
2.3 The Fourier transform (FT)

Proof: Because of

\[ F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-j\omega x} \, dx \]
\[ = \int_{-\infty}^{\infty} f(x) \left( \sum_{n=0}^{\infty} \frac{(-j\omega x)^n}{n!} \right) \, dx \]
\[ = \sum_{n=0}^{\infty} \left( \int_{-\infty}^{\infty} f(x) x^n \, dx \right) (-j)^n \frac{\omega^n}{n!} \]
\[ = \sum_{n=0}^{\infty} m_n (-j)^n \frac{\omega^n}{n!} \]

and

\[ F(\omega) = \sum_{n=0}^{\infty} \frac{d^n F(0)}{d\omega^n} \frac{\omega^n}{n!} \]

The proof is obtained by equating coefficients of equal powers of \( \omega \) in the two equations above.

Convolution

If \( g(x) \) is the convolution of \( f(x) \) and \( h(x) \), i.e.

\[ g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(x) h(x-x) \, dx \] (2.13)

then

\[ G(\omega) = F(\omega)H(\omega) \] (2.14)

Proof:

\[ G(\omega) = \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} f(x) h(x-x) \, dx \right) e^{-j\omega x} \, dx \]
\[ = \int_{-\infty}^{\infty} f(x) \left( \int_{-\infty}^{\infty} h(x-x) e^{-j\omega x} \, dx \right) \, dx \]
\[ = \int_{-\infty}^{\infty} f(x) H(\omega) e^{-j\omega x} \, dx \]
\[ = F(\omega)H(\omega) \]
Modulation

If
\[ g(x) = f(x)h(x) \]  \hspace{1cm} (2.15)

then
\[ G(\omega) = \frac{1}{2\pi} F(\omega) \ast H(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)H(\omega - \omega) d\omega \] \hspace{1cm} (2.16)

Proof:

\[
g(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} G(\omega)e^{i\omega x} d\omega
\]
\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} \left( \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)H(\omega - \omega) d\omega \right) e^{i\omega x} d\omega
\]
\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) \left( \frac{1}{2\pi} \int_{-\infty}^{\infty} H(\omega - \omega) e^{i\omega \tau} d\tau \right) d\omega
\]
\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)h(x)e^{i\omega x} d\omega
\]
\[
= f(x)h(x)
\]

Parseval’s formula

For any \( f(x) \),
\[
\int_{-\infty}^{\infty} |f(x)|^2 dx = \frac{1}{2\pi} \int_{-\infty}^{\infty} |F(\omega)|^2 d\omega \] \hspace{1cm} (2.17)

Proof: From
\[
f(x) \xrightarrow{\mathcal{F}} F(\omega)
\]
\[
f^*(x) \xrightarrow{\mathcal{F}} F^*(-\omega)
\]

and Eqn 2.16, it follows that
\[
\int_{-\infty}^{\infty} |f(x)|^2 e^{-i\omega x} dx = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)F^*(-(\omega - \omega)) d\omega
\]

Let \( \omega = 0 \), Eqn 2.17 is then obtained.
2.3.3 Application to signals

Periodic signals

For $f(x) = ae^{j\omega_0 x}$ where $a$ is a constant, then

$$F(\omega) = a \cdot 2\pi \delta(\omega - \omega_0)$$

which shows the spectrum of the signal very clearly.

Chirp signals

For a chirp signal $f(x) = ae^{j\frac{\alpha}{2}x^2}$ where $a$ is a constant, then

$$F(\omega) = a \cdot \left( \sqrt{\frac{\pi}{4\alpha}} \left( \cos \frac{\omega^2}{2\alpha} + \sin \frac{\omega^2}{2\alpha} \right) + j \sqrt{\frac{\pi}{4\alpha}} \left( \cos \frac{\omega^2}{2\alpha} - \sin \frac{\omega^2}{2\alpha} \right) \right)$$

which is not helpful for revealing the varying spectrum of the chirp signal.

2.4 The discrete Fourier transform (DFT)

2.4.1 Definition

Let $f[n]$ be a complex-valued signal where $n = \cdots, -2, -1, 0, 1, 2, \cdots$, then its discrete Fourier transform (DFT) $F(\theta)$ is defined as (Oppenheim 1983).

$$F(\theta) = \sum_{n=-\infty}^{\infty} f(n)e^{-j\theta n} \quad (2.18)$$

From $F(\theta)$, $f(n)$ can be obtained via

$$f[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\theta)e^{j\theta n}d\theta \quad (2.19)$$
The same notation as employed in the continuous case will be used to describe the relationship between $f(n)$ and $F(\theta)$.

$$f[n] \leftrightarrow F(\theta) \quad (2.20)$$

### 2.4.2 Properties

The properties are similar to those of the continuous FT, however there are some differences. For completeness and by virtue of the wide usage of the DFT, its properties are listed below. For more, see (Oppenheim 1983).

#### Periodicity

For any $f[n]$, its DFT $F(\theta)$ is periodic with period $2\pi$.

$$F(\theta) = F(\theta + 2\pi) \quad (2.21)$$

#### Sum formula

Let $a$ and $b$ be any arbitrary constants, then

$$af_1[n] + bf_2[n] \leftrightarrow aF_1(\theta) + bF_2(\theta) \quad (2.22)$$

#### Spatial shifting

$$f[n - n_0] \leftrightarrow F(\theta)e^{-jn_0} \quad (2.23)$$

#### Frequency shifting

$$f[n]e^{jn_0} \leftrightarrow F(\theta - \theta_0) \quad (2.24)$$

#### Scaling

For any positive integer $k$,

$$f(k)[n] \leftrightarrow F(k\theta) \quad (2.25)$$
2.4 The discrete Fourier transform (DFT)

where

\[ f_{(k)}[n] = \begin{cases} 
  f[n/k], & \text{if } n \text{ is a multiple of } k \\
  0, & \text{if } n \text{ is not a multiple of } k 
\end{cases} \]  \hspace{1cm} (2.26)

Proof:

\[ \sum_{n=-\infty}^{\infty} f_{(k)}[n] e^{-j\theta n} = \sum_{m=-\infty}^{\infty} f_{(k)}[mk] e^{-j\theta m k} = \sum_{m=-\infty}^{\infty} f[m] e^{-j\theta m} = F(k\theta) \]

Differentiation in frequency

\[ n f[n] \leftrightarrow j \frac{dF(\theta)}{d\theta} \]  \hspace{1cm} (2.27)

Parseval's relation

\[ \sum_{n=-\infty}^{\infty} |f^2[n]| = \frac{1}{2\pi} \int_{-\pi}^{\pi} |F(\theta)|^2 d\theta \]  \hspace{1cm} (2.28)

Proof:

\[ \sum_{n=-\infty}^{\infty} |f^2[n]| = \sum_{n=-\infty}^{\infty} f[n] f^*[n] = \sum_{n=-\infty}^{\infty} f[n] \left( \frac{1}{2\pi} \int_{-\pi}^{\pi} F^*(\theta) e^{-j\theta n} d\theta \right) = \frac{1}{2\pi} \int_{-\pi}^{\pi} F^*(\theta) \left( \sum_{n=-\infty}^{\infty} f[n] e^{-j\theta n} \right) d\theta = \frac{1}{2\pi} \int_{-\pi}^{\pi} |F(\theta)|^2 d\theta \]
2.4 The discrete Fourier transform (DFT)

Convolution

Let $g[n]$ be the convolution of $f[n]$ and $h[n]$, i.e.

$$g[n] = f[n] \ast h[n] = \sum_{\eta=-\infty}^{\infty} f[\eta] h[n - \eta] \quad (2.29)$$

then

$$G(\theta) = F(\theta) H(\theta) \quad (2.30)$$

Proof:

$$

g[n] = \sum_{n=-\infty}^{\infty} \sum_{\eta=-\infty}^{\infty} (f[\eta] h[n - \eta]) e^{-j\theta n} \\
= \sum_{\eta=-\infty}^{\infty} f[\eta] \left( \sum_{n=-\infty}^{\infty} h[n - \eta] e^{-j\theta n} \right) \\
= \sum_{\eta=-\infty}^{\infty} f[\eta] (H(\theta) e^{-j\theta \eta}) \\
= \sum_{\eta=-\infty}^{\infty} f[\eta] e^{-j\theta \eta} H(\theta) \\
= F(\theta) H(\theta)
$$

Modulation

If $g[n] = f[n] h[n]$, then

$$G(\theta) = \frac{1}{2\pi} F(\theta) * H(\theta) = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\vartheta) H(\theta - \vartheta) d\vartheta \quad (2.31)$$

Proof:

$$

G(\theta) \\
= \sum_{n=-\infty}^{\infty} f[n] h[n] e^{-j\theta n} \\
= \sum_{n=-\infty}^{\infty} h[n] \left( \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\vartheta) e^{j\vartheta n} d\vartheta \right) e^{-j\theta n}
$$
2.4 The discrete Fourier transform (DFT)

\[
= \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\theta) \left( \sum_{n=-\infty}^{\infty} h[n] e^{-j(\theta-\varphi)n} \right) d\theta \\
= \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\theta) H(\theta-\varphi) d\theta
\]

2.4.3 The fast Fourier transform

In theoretical work, the continuous FT is normally used. However, the DFT is used in practice because there is an efficient algorithm called the fast Fourier transform (FFT) which can compute the DFT very efficiently (Cooley and Tukey 1965).

Let \( f[n] \) be restricted to \([0, N]\).

\[
F(\theta) = \sum_{n=0}^{N-1} f(n) e^{-j\theta n}
\]  \hspace{1cm} (2.32)

When \( \theta = \frac{2\pi k}{N} \)

\[
F[k] = F(\frac{2\pi k}{N}) = \sum_{n=0}^{N-1} f(n) e^{-j(2\pi/N)nk}
\]  \hspace{1cm} (2.33)

The complexity of this algorithm is \( O(N^2) \). This algorithm is very slow for large \( N \).

However, the complexity of the FFT is \( O(N \log N) \). As a result, this algorithm is very efficient even for large \( N \). As a result of this development, applications of the DFT has spread quickly. For the algorithm itself, see Appendix A.

2.4.4 Applications to signals

As a measure of its usefulness, the FT will be applied to a number of signals often found in manufacturing engineering. Some examples are given in Fig 2.1 to Fig 2.4.

In Figure 2.1, \( f[n] \) is

\[
f[n] = \begin{cases} 
  e^{j\theta_0 n} & \text{when } 0 \leq n \leq N - 1 \\
  0 & \text{otherwise}
\end{cases}
\]  \hspace{1cm} (2.34)
2.4 The discrete Fourier transform (DFT)

where $\theta_0 = \frac{\pi}{4}$ and $N = 256$.

In Figure 2.2, $f[n]$ is the sum of three terms which are of the type:

$$f[n] = \begin{cases} 
\sum_{i=1}^{3} A_i e^{j\theta_i n} & \text{when } 0 \leq n \leq N - 1 \\
0 & \text{otherwise}
\end{cases}$$ (2.35)

where $A_1 = 0.5, \theta_1 = \frac{\pi}{8}; A_2 = 1.0, \theta_2 = \frac{\pi}{4};$ and $A_3 = 0.25, \theta_3 = \frac{3\pi}{8}$ and $N = 256$.

It is clearly seen that the Fourier transform can analyse stationary signals very effectively from Figure 2.1 and Figure 2.2,

In Figure 2.3, $f[n]$ is where

$$f[n] = \begin{cases} 
\cos\left(\frac{\pi}{2N}n^2\right) & \text{when } 0 \leq n \leq N - 1 \\
0 & \text{otherwise}
\end{cases}$$ (2.36)

where $\alpha = \frac{\pi}{2N}$ and $N = 256$. This $f[n]$ is normally called as a chirp signal, a typically nonstationary signal.

In Figure 2.4,

$$f[n] = \begin{cases} 
e^{j(\theta_0 n + b\sin(\theta_m n))} & \text{when } 0 \leq n \leq N - 1 \\
0 & \text{otherwise}
\end{cases}$$ (2.37)

where $\theta_0 = \frac{\pi}{4}, \theta_m = \frac{\pi}{32},$ and $b = 3.0$. This is a frequency-modulated (FM) signal.

From Fig 2.3 and Fig 2.4, it can be seen that the Fourier transform can not reveal how the local spectrum varies with space or time. It can be concluded that the Fourier transform is not very helpful for analysing nonstationary signals.

To overcome this, other type of transform is evidently needed if the nonstationary properties are to be easily revealed. A transform which encompasses both space (or time) and frequency simultaneously seems an obvious solution. One such possibility is the ambiguity function, which is investigated next.
Figure 2.1: A stationary signal \( f[n] \) and its DFT \( |F(\theta)| \).
Figure 2.2: Another stationary signal $f[n]$ and its DFT $|F(\theta)|$. 

2.4 The discrete Fourier transform (DFT)
2.4 The discrete Fourier transform (DFT)

Figure 2.3: A chirp signal $f[n]$ and its DFT $|F(\theta)|$. 
Figure 2.4: A frequency modulated signal $f[n]$ and its DFT $|F(\theta)|$. 
2.5 The ambiguity function (AF)

The ambiguity function was originally introduced by Woodward (1953) into Radar for detection of the distance and velocity of aeroplanes. Since then it has been applied to optical signal processing (Bruck and Sodin 1979; Guigay 1978; Lee et al. 1980; Marks, Walkup, and Krile 1977; Marks and Hall 1979; Papoulis 1974; Said and Cooper 1973). As a tool for analysing signals, the ambiguity function has been studied in a great detail by many researchers (Papoulis 1974; Papoulis 1984; Reis 1962; Siebert 1958; Stutt 1959; Stutt 1964; Sussman 1962; Titlebaum and DeClaris 1966).

2.5.1 Definition

For any complex-valued signal $f(x)$ where $x$ is an independent continuous variable, then its ambiguity function (AF), denoted as $A_f(\omega, \chi)$, is defined\(^1\) as

$$ A_f(\omega, \chi) = \int_{-\infty}^{\infty} f(x + \frac{\chi}{2}) f^*(x - \frac{\chi}{2}) e^{-j\omega x} dx $$

(2.39)

The AF $A_f(\omega, \chi)$ can also be defined as

$$ A_f(\omega, \chi) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega + \frac{\omega}{2}) F^*(\omega - \frac{\omega}{2}) e^{j\omega \chi} d\omega $$

(2.40)

In fact, from the change-of-variables theorem (Rudin 1976, p 252; or Rudin 1986, p 153), it follows that

$$ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x + \frac{\chi}{2}) f^*(x - \frac{\chi}{2}) e^{-j\omega x} e^{-j\omega \chi} dx d\chi $$

$$ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t) f^*(\tau) e^{-j\omega (t+\tau)/2} e^{-j\omega (t-\tau)} \cdot dt d\tau $$

\(^1\)Besides the definition by Eqn 2.39, there is another definition which is also often used:

$$ A_f(\omega, \chi) = \int_{-\infty}^{\infty} f(x) f^*(x - \chi) e^{-j\omega x} dx $$

(2.38)
2.5 The ambiguity function (AF)

\[ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t) f^*(\tau) e^{-j(\omega + \frac{\pi}{2})t} e^{-j(\frac{\pi}{2} - \omega)\tau} dt d\tau \]
\[ = F(\omega + \frac{\pi}{2}) F^*(-(\frac{\omega}{2} - \omega)) \]
\[ = F(\omega + \frac{\pi}{2}) F^*(\omega - \frac{\omega}{2}) \]

where \( x \) and \( \chi \) are substituted by \( \frac{1 + x}{2} \) and \( t - \tau \) respectively. Hence,

\[ \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega + \frac{\pi}{2}) F^*(\omega - \frac{\omega}{2}) e^{j\omega \chi} d\omega = \int_{-\infty}^{\infty} f(x + \frac{\chi}{2}) f^*(x - \frac{\chi}{2}) e^{-j\omega \chi} dx \quad (2.41) \]

If the \( A_f(\omega, \chi) \) is known, then we can reconstruct the original signal \( f(x) \) within a constant factor. In fact, it can be proved that (Papoulis 1984, p. 287)

\[ f(x) \cdot f^*(0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} A_f(\omega, \chi) e^{j\omega \chi} d\omega \quad (2.42) \]

Proof: From Eqn 2.39,

\[ \frac{1}{2\pi} \int_{-\infty}^{\infty} A_f(\omega, \chi) e^{j\omega \chi} d\omega = f(x + \frac{\chi}{2}) f^*(x - \frac{\chi}{2}) \quad (2.43) \]

Let \( x = \frac{\chi}{2} \),

\[ f(x) f^*(0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} A_f(\omega, \chi) e^{j\omega \chi} d\omega \quad (2.44) \]

Just like the Fourier transform pair, we will express \( f(x) \) and \( A_f(\omega, \chi) \) together as

\[ f(x) \xrightarrow{A} A_f(\omega, \chi) \]

2.5.2 Simple properties

For more detail, see (Papoulis 1984).

Spatial shifting

\[ f(x - x_0) \xrightarrow{A} A_f(\omega, \chi) e^{-j\omega x_0} \quad (2.45) \]
2.5 The ambiguity function (AF)

Frequency shifting

\[ f(x)e^{j\omega x} \overset{A}{\leftrightarrow} A_f(\omega, x)e^{j\omega x} \]  \hspace{1cm} (2.46)

Spatial limited signals

If \( f(x) \) is restricted to \([x_a, x_b]\), then \( A_f(\omega, x) \) is restricted to \([-\omega_b-\omega_a, \omega_b-\omega_a]\) with respect to \( x \).

Frequency limited signals

If \( f(x) \) is band-limited to \([\omega_a, \omega_b]\), then \( A_f(\omega, x) \) is limited to \([-\omega_b-\omega_a, \omega_b-\omega_a]\) in terms of \( \omega \).

Concentration of energy

For any \( f(x) \),

\[ |A_f(\omega, x)| \leq \int_{-\infty}^{\infty} |f(x)|^2 dx = A_f(0, 0) \]

Proof: Because of

\[ |A_f(\omega, x)| \leq \int_{-\infty}^{\infty} |f(x + \frac{x}{2})f^*(x - \frac{x}{2})| dx \]  \hspace{1cm} (2.47)

and

\[ \int_{-\infty}^{\infty} |f(x + \frac{x}{2})f^*(x - \frac{x}{2})| dx \leq \left( \int_{-\infty}^{\infty} |f(x + \frac{x}{2})| dx \right)^{\frac{1}{2}} \cdot \left( \int_{-\infty}^{\infty} |f(x - \frac{x}{2})| dx \right)^{\frac{1}{2}} \]  \hspace{1cm} (2.48)

from the Schwarz inequality (Rudin 1986, Theorem 3.5), it follows

\[ |A_f(\omega, x)| \]
\[ \leq \left( \int_{-\infty}^{\infty} |f(x + \frac{x}{2})| dx \right)^{\frac{1}{2}} \cdot \left( \int_{-\infty}^{\infty} |f(x - \frac{x}{2})| dx \right)^{\frac{1}{2}} \]
\[ = \int_{-\infty}^{\infty} |f(x)|^2 dx \]
\[ = \int_{-\infty}^{\infty} f(x)f^*(x) dx \]
\[ = A_f(0, 0) \]
2.5 The ambiguity function (AF)

Total energy

\[
\frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |A_f(\omega, x)|^2 d\omega dx = \|f(x)\|^4
\]

Proof: From Eqn 2.39 and Parseval's formula (Eqn 2.17)

\[
\frac{1}{2\pi} \int_{-\infty}^{\infty} |A_f(\omega, x)|^2 d\omega = \int_{-\infty}^{\infty} |f(x + \frac{x}{2})f^*(x - \frac{x}{2})|^2 dx
\]

Hence,

\[
\frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |A_f(\omega, x)|^2 d\omega dx = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |f(x + \frac{x}{2})f^*(x - \frac{x}{2})|^2 dx dx
\]

But from Theorem 10.9 (Rudin 1976),

\[
\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |f(x + \frac{x}{2})f^*(x - \frac{x}{2})|^2 dx dx = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |f(x_1)f^*(x_2)|^2 dx_1 dx_2
\]

Combining the two equations above, we obtain the proof.

Convolution

If \( g(x) = f(x) * h(x) \), then

\[
A_g(\omega, x) = \int_{-\infty}^{\infty} A_f(\omega, x) A_h(\omega, x - x) dx \quad (2.49)
\]

Proof:

\[
A_g(\omega, x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} G(\omega + \frac{x}{2})G^*(\omega - \frac{x}{2}) e^{i\omega x} d\omega
\]

\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega + \frac{x}{2})F^*(\omega - \frac{x}{2}) \cdot H(\omega + \frac{x}{2})H^*(\omega - \frac{x}{2}) e^{i\omega x} d\omega
\]

\[
= \int_{-\infty}^{\infty} A_f(\omega, x) A_h(\omega, x - x) dx
\]
The last equation is obtained by regarding $\chi$ and $\omega$ as variables, treating $\varpi$ as a fixed parameter, and applying Eqn 2.13.

Modulation

If $g(x) = f(x)m(x)$, then

$$A_g(\omega, x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} A_f(\omega, x) A_h(\omega - \omega, x) d\omega \quad (2.50)$$

Proof:

$$A_g(\omega, x) = \int_{-\infty}^{\infty} g(x + \frac{\chi}{2})g^*(x - \frac{\chi}{2})e^{-j\omega x} dx$$

$$= \int_{-\infty}^{\infty} f(x + \frac{\chi}{2})f^*(x - \frac{\chi}{2})h(x + \frac{\chi}{2})h^*(x - \frac{\chi}{2})e^{-j\omega x} dx$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} A_f(\omega, x) A_h(\omega - \omega, x) d\omega$$

The last equation is from Eqn 2.16.

2.5.3 Application to signals

Periodic signals

Let $f(x) = ae^{j\omega x}$ where $a$ is a constant,

$$a_f(\omega, x) = |a|^2 \cdot e^{j\omega x} \cdot 2\pi \delta(\omega) \quad (2.51)$$

Chirp signals

Let $f(x) = ae^{j\alpha x^2}$ where $a$ is a constant,

$$a_f(\omega, x) = |a|^2 \cdot 2\pi \delta(\omega - \alpha x) \quad (2.52)$$
2.6 The discrete ambiguity function (DAF)

Although the AF has been studied in great detail, the discrete ambiguity function has hardly been investigated. In fact, the literature about it is very limited. Here, the way it is defined as Eqn 2.53 is to make the comparison with the discrete Wigner distribution (see Section 3.3) easier.

2.6.1 Definition

For a complex-valued signal \( f[n] \) where \( n = \ldots, -2, -1, 0, 1, 2, \ldots \), its discrete Ambiguity function (DAF), denoted as \( A_f(\vartheta, \eta) \), can be defined as

\[
A_f(\vartheta, \eta) = \sum_{n=-\infty}^{\infty} 2f[n + \eta]f^*[n - \eta]e^{-j2\vartheta n}
\]

The DAF can also be defined as

\[
A_f(\vartheta, \eta) = \frac{1}{\pi} \int_{-\pi}^{\pi} F(\theta + \vartheta)F^*(\theta - \vartheta)e^{j2\vartheta d\theta}
\]

where \( F(\vartheta) \) is the DFT of \( f[n] \).

In fact,

\[
A_f(\vartheta, \eta) = \sum_{n=-\infty}^{\infty} 2f[n + \eta]f^*[n - \eta]e^{-j2\vartheta n}
\]

\[
= 2 \sum_{n=-\infty}^{\infty} f[n + \eta]e^{-j2\vartheta n} \cdot f^*[-(\eta - n)]
\]

\[
= \frac{1}{\pi} \int_{-\pi}^{\pi} F(\theta + 2\vartheta)e^{j(\theta + 2\vartheta)n} \cdot F^*(\theta)e^{j\vartheta n}d\theta
\]

\[
= \frac{1}{\pi} \int_{-\pi}^{\pi} F(\theta + \vartheta + \vartheta)F^*(\theta + \vartheta - \vartheta)e^{j2(\theta + \vartheta)n}d\theta
\]

\[
= \frac{1}{\pi} \int_{-\pi}^{\pi} F(\theta + \vartheta)F^*(\theta - \vartheta)e^{j2\vartheta n}d\theta
\]

Just like the continuous AF, the following notation will be used

\[
f[n] \xrightarrow{A} A_f(\vartheta, \eta)
\]

(2.55)
to represent the relationship between $f[n]$ and $A_f(\vartheta, \eta)$.

### 2.6.2 Properties

**Periodicity**

$$A_f(\vartheta, \eta) = A_f(\vartheta + \pi, \eta) \quad (2.56)$$

**Spatial shifting**

$$f[n - n_0] \xrightarrow{A} A_f(\vartheta, \eta) \cdot e^{-j2\theta n_0} \quad (2.57)$$

**Frequency shifting**

$$f[n]e^{j\theta_0 n} \xrightarrow{A} A_f(\vartheta, \eta) \cdot e^{j2\theta_0 \eta} \quad (2.58)$$

**Spatial-limited signals**

If $f[n]$ is restricted to $[n_a, n_b]$, then $A_f(\vartheta, \eta)$ is restricted to $[-n_c, n_c]$ where $n_c$ is the quotient of $n_b - n_a$ divided by 2.

**Frequency-limited signals**

If $F(\vartheta)$ (the DFT of $f[n]$) is restricted to $[\vartheta_a, \vartheta_b]$, then $A_f(\vartheta, \eta)$ is restricted to $[-(\vartheta_b - \vartheta_a), \vartheta_b - \vartheta_a]$.

**Concentration of energy**

$$|A_f(\vartheta, \eta)| \leq 2 \sum_{n=-\infty}^{\infty} |f[n]|^2 = A_f(0, 0) \quad (2.59)$$

**Proof:** From Eqn 2.53,

$$|A_f(\vartheta, \eta)|$$

$$\leq 2 \sum_{n=-\infty}^{\infty} |f[n + \eta]f^*[n - \eta]|$$

$$\leq 2 \sum_{n=-\infty}^{\infty} |f[n]|^2 \quad \text{(the Schwarz inequality)}$$

$$= A_f(0, 0)$$
2.6.3 Computation of the DAF

When $f[n]$ is restricted to $[0, N - 1]$ where $N$ is even\(^2\), computation of the DAF is straightforward. This is achieved as follows

1. Compute $r[\eta, n] = 2f[n + \eta]f^*[n - \eta]$ for $n = 0, 1, \ldots, N - 1$ and $\eta = -M, \ldots, 0, \ldots, M$ where $M = \frac{N}{2} - 1$.

2. By means of the FFT, compute $\sum_{n=0}^{N-1} r[\eta, n] e^{-j\frac{2\pi nm}{N}}$ for $\eta = -M, \ldots, 0, \ldots, M$.

3. $A_f(\omega, \eta)$ is then obtained for $\omega = \frac{kn}{N}$ where $k = 0, 1, \ldots, N - 1$ and $\eta = -M, \ldots, 0, \ldots, M$. Because of $A_f(\omega, \eta)$ is restricted in $\eta$ and periodic in $\omega$, so $A_f[k, \eta] = A_f(\frac{kn}{N}, \eta)$ is obtained for $k, \eta = -\ldots, -2, -1, 0, 1, 2, \ldots$.

2.6.4 Application to signals useful in manufacturing

Figure 2.6.4, 2.6, 2.7 and 2.8 show four DAFs computed for the same signals as in Section 2.4.4.

In addition to three-dimensional (3D) plots, both density plots and contour plots are also added. In a density plot, the height of the function at each point is shown by shading: the higher the lighter; the lower the darker. In a contour plot, the contour lines are just like those in a standard topographical map, they join points on the same height. Because of its nature, when Mathematica makes a contour plot, it tries to include only the "interesting" parts of the plot. If a function increases very rapidly, or has singularities, the parts where it gets too large will be cut off while the parts with small values may be plot out. Therefore, it is necessary to concentrate on the main feature of a contour plot. For 3D plots and density plots, there is no such a problem for the plot range is chosen to be large enough to display the whole range.

\(^2\)For odd $N$, it is similar provided that it can be represented by $N = N_1 \cdot N_2 \cdots N_k$
2.6 The discrete ambiguity function (DAF)

For the two stationary signals, the DAFs are mainly concentrated on the \( \theta = 0 \). In other words, the DAF can not distinguish the high frequency stationary signals from the low frequency ones. This means that the AF is not suitable for stationary signals. However, for two nonstationary signals, the DAFs do display changes of instantaneous frequencies.

It can be concluded that although the DAF is good for analysing nonstationary signals, it is not so for stationary signals from these figures.
Figure 2.5: A stationary signal $f[n]$ and its DAF.
A stationary signal $f[n]$ and its DAF (continued).
2.6 The discrete ambiguity function (DAF)

Figure 2.6: Another stationary signal $f[n]$ and its DAF.
Another stationary signal $f[n]$ and its DAF (continued).
2.6 The discrete ambiguity function (DAF)

Figure 2.7: A chirp signal \( f[n] \) and its DAF.
2.6 The discrete ambiguity function (DAF)

A chirp signal $f[n]$ and its DAF (continued).
2.6 The discrete ambiguity function (DAF)

Figure 2.8: A frequency modulated signal $f[n]$ and its DAF.
A frequency-modulated signal $f[n]$ and its DAF (continued).
2.6.5 Application to signals with noise

Figure 2.9, 2.10, 2.11 and 2.12 show four DAFs computed for the same signals as in Section 2.4.4, but with noise. The noise is of type

\[ n(x) = n_1(x) + jn_2(x) \]

where \( n_1(x) \) and \( n_2(x) \) are random noise with the range as \([-0.1, 0.1]\), about 10% percent of the original signal, \( j = \sqrt{-1} \).

From these figures, it can still be concluded that although the DAF is good for analysing nonstationary signals, it is not so for stationary signals from these figures.
2.6 The discrete ambiguity function (DAF)

Three-dimensional plot of $|A_f(\vartheta, \eta)|$

Figure 2.9: A stationary signal with noise and its DAF.
2.6 The discrete ambiguity function (DAF)

A stationary signal with noise and its DAF (continued).
2.6 The discrete ambiguity function (DAF)

Figure 2.10: Another stationary signal with noise and its DAF.
2.6 The discrete ambiguity function (DAF)

Another stationary signal with noise and its DAF (continued).
2.6 The discrete ambiguity function (DAF)

Figure 2.11: A chirp signal with noise and its DAF.
A chirp signal with noise and its DAF (continued).
2.6 The discrete ambiguity function (DAF)

Figure 2.12: A frequency modulated signal with noise and its DAF.
A frequency-modulated signal with noise and its DAF (continued).
2.7 Conclusion of suitability of the FT and AF

Now it can be concluded that ON ONE HAND THE FT IS A VERY USEFUL FOR ANALYSING STATIONARY SIGNALS, BUT NOT FOR NONSTATIONARY SIGNALS; ON THE OTHER HAND THE AF IS EFFECTIVE FOR ANALYSING NONSTATIONARY SIGNALS, BUT NOT FOR STATIONARY SIGNALS. In other words, neither of the FT and AF is good for analysing both stationary and nonstationary signals. In order to overcome this problem, a new tool for analysis of signals is required. This can be met by the Wigner distribution, which will be discussed in the next chapter.
Chapter 3

The Wigner Distribution

3.1 Recapitulation of the problem

From the previous chapter, it is known that the Fourier transform (FT) is capable of analysing stationary signals, but becomes unsatisfactory for analysing nonstationary signals. Therefore, the usage of the FT is limited for there are many nonstationary signals in practice. One example is a piece of music (De Bruijn 1967).

"..., if $f$ represents a piece of music, then the composer does not produce $f$ itself; he does not even define it. He may try to prescribe the exact frequency and the exact time interval of a note (although the uncertainty principle says that he can never be completely successful in this effort), but he does not try to prescribe the phase. The composer does not deal with $f$; it is only the gramophone company which produces and sells an $f$. On the other hand, the composer certainly does not want to describe the Fourier transform. This Fourier transform is very useful for solving mathematical and physical problems, but it gives an absolutely unreadable picture of the given piece of music."
What the composer really does, or thinks he does, or should think he does, is something entirely different from describing either \( f \) or \( \mathcal{F} \). Instead, he constructs a function of two variables. The variables are the time and the frequency, the function describes the intensity of the sound. He describes the function by a complicated set of dots on score paper. His way of describing time is slightly different from what a mathematician would do, but certainly vertical lines denote constant time, and horizontal lines denote constant frequency. 

In order to be able to analyse such nonstationary signals, short-time Fourier transforms are often used (Allen and Rabiner 1977). This is based on the assumption that some signals can be regarded as stationary signals on a short-time basis. Despite the wide usage of this technique, it has an important drawback, i.e. the length of the assumed short-time stationarity determines the frequency resolution which can be obtained. To increase the frequency resolution, a longer range has to be taken, which means that nonstationarities will be made vague in both time and frequency. Therefore, to overcome this problem, a mixed time-frequency representation of signals is required.

The ambiguity function is a kind of time-frequency representation of a signal as shown in the last chapter. It enables the analysis of nonstationary signals to be made but not stationary signals, which means that it is not an ideal tool for analysing general signals.

Unlike both the short-time Fourier transform and ambiguity function, the Wigner distribution can be used to analyse both stationary and nonstationary signals because it has some important properties which make it suitable for signal processing which will be explained in the rest of this chapter.

3.2 Introduction

The concept of the Wigner distribution was originally introduced in the context of quantum mechanics by Wigner (1932), while attempting to formulate mathemat-
ical tools for solving quantum problems which involved Heisenberg’s Uncertainty Principle. Then in 1948, Ville reintroduced it for signal processing, although it did not receive wide attention at that time.

However, recently the Wigner distribution has become more and more popular because it is potentially an ideal tool for space-frequency (or time-frequency) representation of signals.

The Wigner distribution has been studied by many researchers (Bamler and Glunder 1983; Boudreaux-Bartels and Parks 1986; Boashash and Black 1987; Boashash 1988; Classen and Mecklenbrauker 1980, 1983; De Bruijn 1973; Kumar, Neuman, and Deros 1986; Peyrin and Prost 1986; Szu and Blodgett 1981; White and Boashash 1988; Yu and Cheng 1987) and applied to many areas (Bastiaans 1978, 1979, 1980; Frensley 1987 Kaluzynski 1989; Zhu, Peix and Babot 1990).

For the mathematical background of the Wigner distribution refer to (De Bruijn 1973); while for a more practical reference see (Classen and Mecklenbrauker 1980).

3.3 The Wigner distribution (WD)

3.3.1 Definition

For any complex-valued signal $f(x)$ where $x$ is a continuous variable such as distance, time and etc., its Wigner distribution (WD) is defined as

$$W_f(x, \omega) = \int_{-\infty}^{\infty} f(x + \frac{x}{2})f^*(x - \frac{x}{2})e^{-j\omega x}dx \tag{3.1}$$

The WD can also be defined as

$$W_f(x, \omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega + \frac{\omega}{2})F^*(\omega - \frac{\omega}{2})e^{i\omega x}d\omega \tag{3.2}$$
3.3 The Wigner distribution (WD)

where $F(\omega)$ is the Fourier transform of $f(x)$.

In fact, from
\[ f(x) \overset{\mathcal{F}}{\rightarrow} F(\omega) \]
it follows that
\[ f(x)f^*(x) \overset{\mathcal{F}}{\rightarrow} F(\omega)F^*(-\omega) \]

Hence,
\[ f(x + \frac{x}{2})f^*(x - \frac{x}{2}) \overset{\mathcal{F}}{\rightarrow} F(\omega + \frac{\omega}{2})F^*(\omega - \frac{\omega}{2}) \]

Therefore, Eqn 3.2 is obtained by using the Fourier transform.

Just like the Fourier transform pair and the ambiguity function pair, we will express the Wigner distribution pair $f(x)$ and $W_f(x, \omega)$ as
\[ f(x) \overset{w}{\rightarrow} W_f(x, \omega) \quad (3.3) \]

Compared with the definition of the ambiguity function

The definitions of the WD and AF are very similar. For the differences, see Table 3.1.

Table 3.1: The definitions of the WD and AF for signal $f(x)$

<table>
<thead>
<tr>
<th></th>
<th>space domain</th>
<th>frequency domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_f(x, \omega)$</td>
<td>$\int_{-\infty}^{\infty} f(x + \frac{x}{2})f^*(x - \frac{x}{2})e^{-j\omega x}dx$</td>
<td>$\frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega + \frac{\omega}{2})F^*(\omega - \frac{\omega}{2}) e^{j\omega x}d\omega$</td>
</tr>
<tr>
<td>$A_f(\omega, x)$</td>
<td>$\int_{-\infty}^{\infty} f(x + \frac{x}{2})f^*(x - \frac{x}{2})e^{-j\omega x}dx$</td>
<td>$\frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega + \frac{\omega}{2})F^*(\omega - \frac{\omega}{2}) e^{j\omega x}d\omega$</td>
</tr>
</tbody>
</table>
The cross Wigner distribution for two signals

For two signals \( f_1(x) \) and \( f_2(x) \), their cross Wigner distribution can be defined as

\[
W_{f_1, f_2}(x, \omega) = \int_{-\infty}^{\infty} f_1(x + \frac{x}{2}) f_2^*(x - \frac{x}{2}) e^{-j\omega x} dx
\]  
(3.4)

The WD for multidimensional signals

For a complex-valued signal \( f(x) \) where \( x \in \mathbb{R}^n \), its WD is defined as

\[
\int_{-\infty}^{\infty} f(x + \frac{x}{2}) f^*(x - \frac{x}{2}) e^{-j\omega \cdot x} dx = W_f(x, \omega)
\]  
(3.5)

where \( \omega \in \mathbb{C}^n \)

### 3.3.2 Simple properties

**Symmetry**

For any real-valued signal, the WD is an even function of the frequency:

\[
W_f(x, \omega) = W_f(x, -\omega)
\]  
(3.6)

**Realness**

For any complex-valued signal, the WD is real-valued:

\[
W_f(x, \omega) = (W_f(x, \omega))^*
\]  
(3.7)

**Spatial-shifting**

A spatial shift in \( f(x) \) corresponds to a same shift in its WD, i.e.

\[
f(x + x_0) \xrightarrow{W} W_f(x + x_0, \omega)
\]  
(3.8)
3.3 The Wigner distribution (WD)

Frequency-shifting

Just like spatial-shift, a frequency shift in $f(x)$ corresponds to a same shift in its WD:

$$f(x)e^{j\omega_0 x} \xrightarrow{\mathcal{W}} W_f(x, \omega - \omega_0)$$  \hspace{1cm} (3.9)

Sum formula

For $f(x) = f_1(x) + f_2(x)$,

$$W_f(x, \omega) = W_{f_1}(x, \omega) + W_{f_2}(x, \omega) + 2\text{Re}W_{f_1, f_2}(x, \omega)$$  \hspace{1cm} (3.10)

Spatial-limited signals

If $f(x)$ is restricted to $[x_a, x_b]$, so is $W_f(x, \omega)$.

Frequency-limited signals

If $f(x)$ is band-limited to $[\omega_a, \omega_b]$, so is $W_f(x, \omega)$.

Spatial-energy

This is that the integral of the WD in $\omega$ at a fixed $x$ is equal to the instantaneous energy at that $x$, i.e.

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} W_f(x, \omega) d\omega = |f(x)|^2$$  \hspace{1cm} (3.11)

Frequency-energy

The integral of the WD over $x$ at a certain $\omega$ gives the energy spectrum at that $\omega$, i.e.

$$\int_{-\infty}^{\infty} W_f(x, \omega) dx = |F(\omega)|^2$$  \hspace{1cm} (3.12)

Total energy

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_f(x, \omega) dx d\omega = \|f(x)\|^2$$  \hspace{1cm} (3.13)
This means that the integral of the WD over the whole plane \((x, \omega)\) is equal to the total energy of \(f(x)\).

**Moyal's formula**

\[
\frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_f^2(x, \omega) dx d\omega = \|f(x)\|^4 \tag{3.14}
\]

This means that the integral of the squared WD over the whole plane \((x, \omega)\) is equal to the square of the total energy of \(f(x)\). This is similar to the Parseval's formula about the FT.

**Convolution**

Let \(g(x)\) be the convolution of \(f(x)\) and \(h(x)\), i.e.

\[
g(x) = f(x) \ast h(x) \tag{3.15}
\]

then

\[
W_g(x, \omega) = \int_{-\infty}^{\infty} W_f(x, \omega) W_h(x - x, \omega) dx \tag{3.16}
\]

**Proof:**

\[
W_g(x, \omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\omega + \frac{\omega}{2}) H(\omega + \frac{\omega}{2}) F^*(\omega - \frac{\omega}{2}) H^*(\omega - \frac{\omega}{2}) e^{i\omega x} d\omega
\]

\[
= \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\omega + \frac{\omega}{2}) F^*(\omega - \frac{\omega}{2}) H(\omega + \frac{\omega}{2}) H^*(\omega - \frac{\omega}{2}) e^{i\omega x} d\omega
\]

\[
= \int_{-\infty}^{\infty} W_f(x, \omega) W_h(x - x, \omega) dx
\]

From Eqn 3.16, it follows that the WD of the convolution is the convolution of the WD provided that \(\omega\) is regarded as a fixed parameter in the WD.

**Modulation**

If

\[
g(x) = f(x) m(x)
\]
then
\[ W_g(x, \omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} W_f(x, \omega) W_m(x, \omega - \omega) d\omega \] (3.17)

Proof:

\[ W_g(x, \omega) = \int_{-\infty}^{\infty} f(x + \frac{x}{2}) m(x + \frac{x}{2}) f^*(x - \frac{x}{2}) m^*(x - \frac{x}{2}) e^{-j\omega x} dx \]
\[ = \int_{-\infty}^{\infty} f(x) f^*(x) m(x) m^*(x - \omega) e^{-j\omega x} dx \]
\[ = \frac{1}{2\pi} \int_{-\infty}^{\infty} W_f(x, \omega) W_m(x, \omega - \omega) d\omega \]

### 3.3.3 The WD for analytical signals

One point which needs to be highlighted before the WD can be applied widely, is that the WD should be applied to analytical signals, i.e. those which have no redundant spectrum.

This is achieved for any real-valued signal \( f(x) \) by constructing its analytical signal \( f_a(x) \) as

\[ f_a(x) = f(x) + j\hat{f}(x) \] (3.18)

where \( \hat{f}(x) \) is the Hilbert transform of \( f(x) \):

\[ \hat{f}(x) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{f(x)}{x-x} dx \] (3.19)

If \( F(\omega) \) and \( F_a(\omega) \) are the FTs of \( f(x) \) and \( f_a(x) \), then

\[ F_a(\omega) = \begin{cases} 
2F(\omega) & \omega > 0 \\
F(0) & \omega = 0 \\
0 & \omega < 0 
\end{cases} \] (3.20)

which means that the analytical signal does not contain redundant spectrum which in turn means that neither does the WD.

\[ W_{f_a}(x, \omega) = 0 \quad \omega < 0 \] (3.21)
This trick considerably reduces aliasing problems.

Furthermore,

\[
W_{f_a}(x, \omega) = \begin{cases} 
\frac{4}{\pi} \int_{-\infty}^{\infty} W_f(x - x, \omega) \sin \frac{2\omega x}{x} dx & \omega \geq 0 \\
0 & \omega < 0 
\end{cases}
\] (3.22)

Proof: For \( \omega \geq 0 \)

\[
W_{f_a}(x, \omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F_a(\omega + \frac{\omega}{2}) F_a^*(\omega - \frac{\omega}{2}) e^{j\pi \omega} d\omega
\]

\[
= \frac{2}{\pi} \int_{-2\omega}^{2\omega} F(\omega + \frac{\omega}{2}) F^*(\omega - \frac{\omega}{2}) e^{j\pi \omega} d\omega
\]

\[
= \frac{2}{\pi} \int_{-2\omega}^{2\omega} \left( \int_{-\infty}^{\infty} W_f(x, \omega) e^{-j\pi x} dx \right) e^{j\pi \omega} d\omega
\]

\[
= \frac{2}{\pi} \int_{-\infty}^{\infty} W_f(x, \omega) \left( \int_{-2\omega}^{2\omega} e^{-j\omega(x-x)} d\omega \right) dx
\]

\[
= \frac{4}{\pi} \int_{-\infty}^{\infty} W_f(x, \omega) \sin \frac{2\omega(x - x)}{x - x} dx
\]

\[
= \frac{4}{\pi} \int_{-\infty}^{\infty} W_f(x - x, \omega) \frac{\sin 2\omega x}{x} dx
\]

3.3.4 Moments

From the properties of the WD, specially those related to the energy distribution, it can be seen that the WD could be interpreted as the energy distribution in the mixed space-frequency plane. Therefore, in order to get an idea of how the energy in a signal is distributed in space and frequency, it is only necessary to compute the WD of the signal. Moreover, moments can be introduced to characterise the distribution rather than using the values of the WD over the whole plane, thereby condensing the amount of information considerably. There are two types of moments: local moments in frequency or space and global moments over the whole space-frequency plane. These moments can be used to specify averages and the spread of the WD and turn out to be very useful for characterising instantaneous properties of the signal.
Local moments in frequency

The *0th-order moment in frequency* $p_f(x)$ is defined as

$$p_f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} W_f(x, \omega) d\omega \quad (3.23)$$

From Eqn 3.11,

$$p_f(x) = |f(x)|^2 \quad (3.24)$$

which means that $p_f(x)$ is the instantaneous power of $f(x)$, which is non-negative.

The *1st-order moment in frequency* $\Omega_f(x)$ is defined as

$$\Omega_f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \omega W_f(x, \omega) d\omega / p_f(x) \quad (3.25)$$

It can be proved that

$$\Omega_f(x) = \text{Im} \frac{f'(x)}{f(x)} \quad (3.26)$$

Proof: From Eqn 3.1 and the inverse Fourier transform,

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} W_f(x, \omega) e^{i\omega x} d\omega = f(x + \frac{\chi}{2}) f^*(x - \frac{\chi}{2})$$

Differentiating with respect to $x$,

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} j\omega W_f(x, \omega) e^{i\omega x} d\omega = \frac{1}{2} \left( f'(x + \frac{\chi}{2}) f^*(x - \frac{\chi}{2}) - f(x + \frac{\chi}{2}) (f')^*(x - \frac{\chi}{2}) \right) \quad (3.27)$$

i.e.

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \omega W_f(x, x) e^{-j\omega x} d\omega = \frac{1}{2j} \left( f'(x + \frac{\chi}{2}) f^*(x - \frac{\chi}{2}) - f(x + \frac{\chi}{2}) (f')^*(x - \frac{\chi}{2}) \right)$$

When $\chi = 0$, it follows that

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \omega W_f(x, x) d\omega = \frac{1}{2j} (f'(x) f^*(x) - f(x)(f')^*(x)) = \text{Im} f'(x) f^*(x)$$
3.3 The Wigner distribution (WD)

Divided by $f(x)f^*(x)$, the result is obtained.

The 2nd-order moment in frequency $m_f(x)$ is defined as

$$m_f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} (\omega - \Omega_f(x))^2 W_f(x, \omega) d\omega / p_f(x)$$  \hspace{1cm} (3.28)

For $m_f(x)$,

$$m_f(x) = -\frac{1}{2} \text{Re} \frac{d}{dx} f(x)$$ \hspace{1cm} (3.29)

Proof: Differentiating Eqn 3.27 with respect to $x$,

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} (-j\omega)^2 W_f(x, \omega) e^{j\omega x} d\omega$$

$$= \frac{1}{4} f''(x + \frac{x}{2}) f^*(x - \frac{x}{2}) - \frac{1}{2} f'(x + \frac{x}{2}) (f')^*(x - \frac{x}{2})$$

$$+ \frac{1}{4} f(x + \frac{x}{2}) (f'')^*(x - \frac{x}{2})$$

Let $\chi = 0$,

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \omega^2 W_f(x, \omega) d\omega = -\frac{1}{4} f''(x)f^*(x) + \frac{1}{2} f'(x)(f')^*(x) - \frac{1}{4} f(x)(f'')^*(x)$$

Combined with the 0th-order moment and the 1st-order moment,

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} (\omega - \Omega_f(x))^2 W_f(x, \omega) d\omega$$

$$= -\frac{1}{4} f''(x)f^*(x) + \frac{1}{2} f'(x)(f')^*(x) - \frac{1}{4} f(x)(f'')^*(x)$$

$$- \Omega^2_f(x) \cdot p_f(x)$$

$$= -\frac{1}{4} f''(x)f^*(x) + \frac{1}{2} f'(x)(f')^*(x) - \frac{1}{4} f(x)(f'')^*(x)$$

$$+ \frac{1}{4} (f'(x)f^*(x) - f(x)(f')^*(x))^2$$

$$= -\frac{1}{4} f''(x)f^*(x) - \frac{1}{4} f(x)(f'')^*(x) + \frac{1}{2} f'(x)(f')^*(x)$$

$$+ \frac{1}{4} \frac{(f')^2(x)f^*(x)}{f(x)} + \frac{1}{4} \frac{f(x)((f')^2)^*(x)}{f^*(x)} - \frac{1}{2} f'(x)(f')^*(x)$$

$$= -\frac{1}{4} f''(x)f^*(x) - \frac{1}{4} f(x)(f'')^*(x)$$
3.3 The Wigner distribution (WD)

\[
\frac{1}{4} \left( \frac{(f')^2(x)f^*(x)}{f(x)} + \frac{f(x)((f')^2)^*(x)}{f^*(x)} \right)
\]

Divided by \( f(x)f^*(x) \),

\[
\frac{1}{2\pi} \int_{-\infty}^{\infty} (\omega - \Omega_f(x))^2 W_f(x, \omega) d\omega / p_f(x)
\]

\[
= -\frac{1}{4} \frac{f''(x)}{f(x)} - \frac{1}{4} \frac{(f'')^*(x)}{f^*(x)}
\]

\[
+ \frac{1}{4} \frac{(f')^2(x)}{f^2(x)} + \frac{1}{4} \frac{((f^*)')^2(x)}{(f^*)^2(x)}
\]

\[
= -\frac{1}{2} \text{Re} \left( \frac{f''(x)}{f(x)} - \frac{(f')^2(x)}{f^2(x)} \right)
\]

Hence, the proof is obtained.

For a signal \( f(x) = A(x)e^{j\phi(x)} \), then its 0th-order, 1st-order, 2nd-order moment is

\[
p_f(x) = |A(x)|^2
\]

\[
\Omega_f(x) = \phi'(x)
\]

\[
m_f(x) = -\frac{1}{2} \frac{d}{dx} \frac{A'(x)}{A(x)}
\]

From the above three equations, it can be seen that the 0th-order moment \( p_f(x) \) is the instantaneous power of the signal, the 1st-order moment \( \Omega_f(x) \) is the instantaneous frequency of the signal, and the 2nd-order moment \( m_f(x) \) is dependent on the envelope of the signal. For a real-valued signal, a similar conclusion can be reached if its analytical signal is used.

**Local moments in space**

The 0th-order moment in space \( P_f(\omega) \) is defined as

\[
P_f(\omega) = \int_{-\infty}^{\infty} W_f(x, \omega) dx
\]
From Eqn 3.12,

$$P_f(\omega) = |F(\omega)|^2 \tag{3.34}$$

The 1st-order moment in space $X_f(\omega)$ is defined as

$$X_f(\omega) = \int_{-\infty}^{\infty} xW_f(x, \omega)dx/P_f(\omega) \tag{3.35}$$

In fact,

$$X_f(\omega) = -\text{Im} \frac{F'(\omega)}{F(\omega)} \tag{3.36}$$

The 2nd-order moment in space $M_f(\omega)$ is defined as

$$M_f(\omega) = \int_{-\infty}^{\infty} (x - X_f(\omega))^2W_f(x, \omega)dx/P_f(\omega) \tag{3.37}$$

It can be proved that

$$M_f(\omega) = -\frac{1}{2} \text{Re} \frac{d}{d\omega} \frac{F'(\omega)}{F(\omega)} \tag{3.38}$$

For a signal $f(x)$ whose FT is $F(\omega) = A(\omega)e^{j\phi(\omega)}$, its moments in space are

$$P_f(\omega) = |A(\omega)|^2 \tag{3.39}$$

$$T_f(\omega) = -\phi'(\omega) \tag{3.40}$$

$$M_f(\omega) = -\frac{1}{2} \frac{d}{d\omega} \frac{A'(\omega)}{A(\omega)} \tag{3.41}$$

Global moments

Besides local moments, global moments can also be used. Some of them are listed here.

For the 0th-order global moment of the WD,

$$\overline{P_f} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_f(x, \omega)dx d\omega = \|f\|^2 \tag{3.42}$$
For the 1st-order global moment with respect to frequency of the WD,

\[
\overline{\Omega_f} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \omega W_f(x,\omega)dxd\omega/P_f
\]  
(3.43)

For the 1st-order global moment with respect to space of the WD,

\[
\overline{X_f} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x W_f(x,\omega)dxd\omega/P_f
\]  
(3.44)

3.3.5 Application to signals

For \( f(x) = ae^{j\omega_0 x} \) where \( a \) is a constant, then

\[
W_f(x,\omega) = |a|^2 \cdot 2\pi \cdot \delta(\omega - \omega_0)
\]  
(3.45)

Just like the FT, the WD is also concentrated at \( \omega = \omega_0 \) for \( e^{j\omega_0 x} \).

For \( f(x) = ae^{j\frac{\alpha}{2} x^2} \) where \( a \) is a constant,

\[
W_f(x,\omega) = |a|^2 \cdot 2\pi \cdot \delta(\omega - \alpha x)
\]  
(3.46)

This means that the WD for a chirp signal is concentrated around the instantaneous frequency so that the chirp effect is clearly visible and measurable from its WD.
3.4 The discrete Wigner distribution (DWD)

3.4.1 Definition

Let \( f[n] \) be a complex-valued signal where \( n = \ldots, -2, -1, 0, 1, 2, \ldots \), then its discrete Wigner distribution (DWD) is defined as

\[
W_f(n, \theta) = \sum_{\eta=-\infty}^{\infty} 2f[n + \eta]f^*[n - \eta]e^{-j2\theta\eta} \tag{3.47}
\]

The DWD can also be defined as

\[
W_f(n, \theta) = \frac{1}{\pi} \int_{-\pi}^{\pi} F(\theta + \vartheta)F^*(\theta - \vartheta)e^{j2\theta\eta}d\vartheta \tag{3.48}
\]

where \( F(\theta) \) is the DFT of \( f[n] \).

In fact

\[
\sum_{\eta=-\infty}^{\infty} 2f[n + \eta]f^*[n - \eta]e^{-j2\theta\eta} = \sum_{\eta=-\infty}^{\infty} 2f[n + \eta]e^{-j2\theta\eta} \cdot f^*[n - \eta] = \frac{1}{\pi} \int_{-\pi}^{\pi} F(\vartheta) e^{j(\vartheta + 2\theta)\eta}F^*(-\vartheta) e^{j\vartheta\eta}d\vartheta = \frac{1}{\pi} \int_{-\pi}^{\pi} F(\theta + \vartheta + \theta)F^*(\theta - (\vartheta + \theta)) e^{2j\vartheta(\vartheta + \theta)}d\vartheta
\]

3.4.2 Basic properties

Some useful properties are listed here\(^1\).

**Periodicity**

\[
W_f(n, \theta) = W_f(n, \theta + \pi) \tag{3.49}
\]

\(^1\)The proofs for Eqn 3.57, Eqn 3.62 and Eqn 3.64, as shown here, are given by myself.
3.4 The discrete Wigner distribution (DWD)

Symmetry

For a real-valued $f[n],

$$W_f(n, \theta) = W_f(n, -\theta)$$ \hfill (3.50)

Realness

For any signal $f[n],

$$W_f(n, \theta) = W_f^*(n, \theta)$$ \hfill (3.51)

Spatial shifting

Let $n_0$ be a fixed integer,

$$f[n - n_0] \xrightarrow{W} W_f(n - n_0, \theta)$$ \hfill (3.52)

Frequency shifting

Let $\theta_0$ be a fixed constant,

$$f[n]e^{j\theta_0 n} \xrightarrow{W} W_f(n, \theta - \theta_0)$$ \hfill (3.53)

Spatial-limited signals

If $f[n]$ is restricted to $[n_a, n_b]$, so is its DWD.

Band-limited signals

If $f(n)$ is band-limited to $[\theta_a, \theta_b]$, so is its DWD provided that $0 < \theta_b - \theta_a < \pi$.

Sum formula

For $f[n] = f_1[n] + f_2[n],

$$W_f(n, \theta) = W_{f_1}(n, \theta) + W_{f_2}(n, \theta) + 2\text{Re}W_{f_1,f_2}(n, \theta)$$ \hfill (3.54)
3.4 The discrete Wigner distribution (DWD)

where \( W_{f_1,f_2}(n, \theta) \) is the cross Wigner distribution of \( f_1 \) and \( f_2 \):

\[
W_{f_1,f_2}(n, \theta) = \sum_{\eta=-\infty}^{\infty} 2f_1[n + \eta]f_2^*[n - \eta]e^{-j2\theta\eta} \tag{3.55}
\]

**Spatial-energy**

\[
\sum_{n=-\infty}^{\infty} W_f(n, \theta) = \frac{1}{2}(|F(\theta)|^2 + |F(\theta + \pi)|^2) \tag{3.56}
\]

**Proof:** From the definition of the DWD, it follows that

\[
\sum_{n=-\infty}^{\infty} W_f(n, \theta)e^{-j2\theta n}
= \sum_{n=-\infty}^{\infty} \left( \frac{1}{\pi} \int_{-\pi}^{\pi} F(\theta + \vartheta_1)F^*(\theta - \vartheta_1)e^{j2\vartheta_1 n}d\vartheta_1 \right) e^{-j2\theta n}
= \frac{1}{\pi} \int_{-\pi}^{\pi} F(\theta + \vartheta_1)F^*(\theta - \vartheta_1) \left( \sum_{n=-\infty}^{\infty} e^{-j2(\theta-\vartheta_1)n} \right) d\vartheta_1
= \frac{1}{\pi} \int_{-\pi}^{\pi} F(\theta + \vartheta_1)F^*(\theta - \vartheta_1) \left( \sum_{k=-\infty}^{\infty} 2\pi \delta(2\theta - 2\vartheta_1 - 2\pi k) \right) d\vartheta_1
= \int_{-\pi}^{\pi} F(\theta + \vartheta_1)F^*(\theta - \vartheta_1) \left( \sum_{k=-\infty}^{\infty} \delta(\vartheta_1 - \theta + \pi k) \right) d\vartheta_1
\]

Therefore,

\[
\sum_{n=-\infty}^{\infty} W_f(n, \theta)e^{-j2\theta n} = F(\theta + \vartheta)F^*(\theta - \vartheta) + F(\theta + \vartheta + \pi)F^*(\theta - \vartheta + \pi) \tag{3.57}
\]

Eqn 3.57 is very useful for proving other theorems. Eqn 3.56 is then obtained by letting \( \vartheta = 0 \).

For any analytical signal \( f_1 \) or over-sampled signal\(^2\), Eqn 3.56 becomes

\[
\sum_{n=-\infty}^{\infty} W_f(n, \theta) = F(\theta)F^*(\theta) = |F(\theta)|^2 \tag{3.58}
\]

which is similar to its continuous case.

\(^2\)Let a signal \( f(x) \) be band-limited to \([-\omega_c, \omega_c]\), the over-sampled signal \( f_1[n] \) is obtained if the sampling frequency is larger then \( 4\omega_c \).
3.4 The discrete Wigner distribution (DWD)

Frequency-energy

\[ \frac{1}{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} W_f(n, \theta) d\theta = |f[n]|^2 \] (3.59)

Total-energy

\[ \frac{1}{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sum_{n=-\infty}^{\infty} W_f(n, \theta) d\theta = \frac{1}{2\pi} \int_{-\pi}^{\pi} |F(\theta)|^2 d\theta = \sum_{n=-\infty}^{\infty} |f[n]|^2 \] (3.60)

Moyal's formula

For a analytical (or over-sampled) signal,

\[ \frac{1}{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sum_{n=-\infty}^{\infty} W_f^2(n, \theta) d\theta = \|f[n]\|^4 \] (3.61)

Convolution

Let

\[ g[n] = f[n] * h[n] \]
\[ g^0[n] = f[n] * (h[n]e^{i\pi n}) \]

then

\[ W_g(n, \theta) + W_{g^0}(n, \theta) = \sum_{\eta=-\infty}^{\infty} W_f(\eta, \theta)W_h(n - \eta, \theta) \] (3.62)

Proof: Eqn 3.62 is from the inverse Fourier transform, Eqn 3.57, and

\[
\begin{align*}
F(\theta + \frac{\vartheta}{2})H(\theta + \frac{\vartheta}{2})F^*(\theta - \frac{\vartheta}{2})H^*(\theta - \frac{\vartheta}{2}) \\
+ F(\theta + \frac{\vartheta}{2} + \pi)H(\theta + \frac{\vartheta}{2} + \pi)F^*(\theta - \frac{\vartheta}{2} + \pi)H^*(\theta - \frac{\vartheta}{2} + \pi) \\
+ F(\theta + \frac{\vartheta}{2})H(\theta + \frac{\vartheta}{2} + \pi)F^*(\theta - \frac{\vartheta}{2} + \pi)H^*(\theta - \frac{\vartheta}{2} + \pi) \\
+ F(\theta + \frac{\vartheta}{2} + \pi)H(\theta + \frac{\vartheta}{2})F^*(\theta - \frac{\vartheta}{2} + \pi)H^*(\theta - \frac{\vartheta}{2} + \pi) \\
= \\
\left( F(\theta + \frac{\vartheta}{2})F^*(\theta - \frac{\vartheta}{2}) + F(\theta + \frac{\vartheta}{2} + \pi)F^*(\theta - \frac{\vartheta}{2} + \pi) \right)
\end{align*}
\]
3.4 The discrete Wigner distribution (DWD)

\[
\left( H(\theta + \frac{\vartheta}{2})H^\ast(\theta - \frac{\vartheta}{2}) + H(\theta + \frac{\vartheta}{2} + \pi)H^\ast(\theta - \frac{\vartheta}{2} + \pi) \right)
\]

If both \( f \) and \( h \) are both band-limited to \([\varpi, \frac{\varpi}{2}]\), then \( g^o \) is zero, therefore

\[
W_g(n, \theta) = \sum_{\eta = -\infty}^{\infty} W_f(\eta, \theta)W_h(n - \eta, \theta)
\]

(3.63)

Modulation

If

\[
g[n] = f[n]h[n]
\]

then

\[
g[n] \leftrightarrow W \frac{1}{2\pi} \int_{-\frac{\varpi}{2}}^{\frac{\varpi}{2}} W_f(n, \vartheta)W_h(n, \theta - \vartheta)d\vartheta
\]

(3.64)

Proof:

\[
W_g(n, \theta) = \sum_{\eta = -\infty}^{\infty} 2f[n + \eta]h[n + \eta] \cdot f^\ast[n - \eta]h^\ast[n - \eta]e^{-i2\pi\eta}
\]

\[
= \frac{1}{2} \cdot \sum_{\eta = -\infty}^{\infty} 2f[n + \eta]f^\ast[n - \eta] \cdot 2h[n + \eta]h^\ast[n - \eta]e^{-i2\pi\eta}
\]

\[
= \frac{1}{4\pi} \int_{-\pi}^{\pi} W_f(n, \frac{\vartheta}{2})W_h(n, \frac{\theta - \vartheta}{2})d\vartheta
\]

\[
= \frac{1}{2\pi} \int_{-\frac{\varpi}{2}}^{\frac{\varpi}{2}} W_f(n, \vartheta)W_h(n, \theta - \vartheta)d\vartheta
\]

3.4.3 The DWD for analytical signals

For a signal \( f[n] \), its analytical signal \( f_a[n] \) is of the form (Oppenheim and Schafer 1975)

\[
f_a[n] = f[n] + j\hat{f}[n]
\]

(3.65)

where \( \hat{f}[n] \) is the discrete Hilbert transform of \( f[n] \), defined by

\[
\hat{f}[n] = (\mathcal{H}_d f)[n] = \sum_{\eta = -\infty}^{\infty} f[\eta] \frac{2\sin^2(\pi(n-\eta))}{\pi (n-\eta)}
\]

(3.66)
The discrete Wigner distribution (DWD)

The DFT of \( f_a \) and \( f \) is related as

\[
F_a(\vartheta) = \begin{cases} 
2F(\vartheta) & 0 < \vartheta < \pi \\
F(0) & \vartheta = 0 \\
0 & -\pi \leq \vartheta < 0 
\end{cases} \tag{3.67}
\]

Since \( f_a \) is band-limited to \([0, \pi]\), its DWD does not have the aliasing effect.

3.4.4 Moments

For any signal \( f[n] \), both local moments and global moments can also be used to simplify the usage of the DWD. They are listed below.

Local moments in frequency

The **0th-order moment in frequency** \( p_f[n] \) is defined as

\[
p_f[n] = \frac{1}{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} W_f(n, \theta) d\theta \tag{3.68}
\]

From Eqn 3.59,

\[
p_f[n] = |f[n]|^2 \tag{3.69}
\]

which means that the 0th-order moment in frequency is equal to the instantaneous power of the signal.

The **1st-order moment in frequency** \( \Theta_f[n] \):

\[
\Theta_f[n] = \frac{1}{2} \arg \left( \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} e^{i2\theta} W_f(n, \theta) d\theta \right) \tag{3.70}
\]

then

\[
\Theta_f[n] = \frac{1}{2} \arg(f[n + 1]f^*[n - 1]) \tag{3.71}
\]
For \( f[n] = v[n]e^{j\phi[n]} \), then

\[
\Theta_f[n] = \frac{\phi[n + 1] - \phi[n - 1]}{2} \mod \pi \tag{3.72}
\]

which means that the 1st-order moment in frequency can be used to represent the instantaneous frequency.

The 2nd-order moment in frequency \( m_f[n] \):

\[
m_f[n] = \frac{p_f[n] - \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j2\theta} W_f(n, \theta) d\theta}{p_f[n] + \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j2\theta} W_f(n, \theta) d\theta} \tag{3.73}
\]

\( m_f[n] \) can be expressed as

\[
m_f[n] = \frac{|f[n]|^2 - |f[n + 1]|f^*[n - 1]|}{|f[n]|^2 + |f[n + 1]|f^*[n - 1]|} \tag{3.74}
\]

Local moments in space

The 0th-order moment in space \( P_f(\theta) \):

\[
P_f(\theta) = \sum_{n=-\infty}^{\infty} W_f(n, \theta) = |F(\theta)|^2 + |F(\theta + \pi)|^2 \tag{3.75}
\]

The 1st-order moment in space \( X_f(\theta) \):

\[
X_f(\theta) = \frac{\sum_{n=-\infty}^{\infty} W_f(n, \theta)n}{P_f(\theta)} \tag{3.76}
\]

For over-sampled or analytical signals, \( X_f \) can be expressed by

\[
X_f(\theta) = -\text{Im} \frac{d \ln F(\theta)}{d\theta} \tag{3.77}
\]

Proof: Differentiating Eqn 3.57 with respect to \( \theta \) obtains

\[
\sum_{n=-\infty}^{\infty} W_f(\theta, n)ne^{-j2\theta n}
\]
\[
\frac{\hat{f}(\omega + \vartheta + \pi)}{2} F^*(\theta - \vartheta + \pi) - \frac{j}{2} F(\theta + \vartheta + \pi)(F^*)'(\theta - \vartheta + \pi)
\]

then

\[
\sum_{n=-\infty}^{\infty} W_f(\theta, n)n = \frac{j}{2} F'(\theta) F^*(\theta) - \frac{j}{2} F(\theta)(F^*)'(\theta)
\]

Because of \( f \) is over-sampled or analytical,

\[
\sum_{n=-\infty}^{\infty} W_f(\theta, n)n = \frac{j}{2} F'(\theta) F^*(\theta) - \frac{j}{2} F(\theta)(F^*)'(\theta)
\]

Therefore

\[
X_f(\theta) = \frac{\frac{j}{2} F'(\theta) F^*(\theta) - \frac{j}{2} F(\theta)(F^*)'(\theta)}{F(\theta) F^*(\theta)}
\]

\[
= \frac{\frac{j}{2} \left( \frac{F'(\theta)}{F(\theta)} - \left( \frac{F'(\theta)}{F(\theta)} \right)^* \right)}{F(\theta)}
\]

\[
= -\text{Im} \left( \frac{F'(\theta)}{F(\theta)} \right)
\]

The 2nd-order moment in space \( M_f(\theta) \):

\[
M_f(\theta) = \frac{\sum_{n=-\infty}^{\infty} [n - T_f(\theta)]^2 W_f(n, \theta)}{P_f(\theta)}
\]

(3.78)

For over-sampled or analytical signals, \( M_f \) becomes

\[
M_f(\theta) = -\frac{1}{2} \text{Re} \left( \frac{d}{d\theta} \frac{F'(\theta)}{F(\theta)} \right)
\]

(3.79)
Global moments

The 0th-order global moment of the DWD:

\[ P_f = \frac{1}{2\pi} \int_{-\pi}^{\pi} \sum_{n=-\infty}^{\infty} W_f(n, \theta) d\theta = \| f \|^2 = \| F \|^2 \]  

(3.80)

The 1st-order global moment with respect to frequency of the DWD:

\[ \Theta_f = \frac{1}{2} \arg \left[ \int_{-\pi}^{\pi} \sum_{n=-\infty}^{\infty} e^{j2\theta} W_f(n, \theta) d\theta \right] \]  

(3.81)

The 1st-order global moment with respect to space of the DWD:

\[ X_f = \sum_{n=-\infty}^{\infty} \frac{1}{2\pi} \int_{-\pi}^{\pi} nW_f(n, \theta) d\theta / P_f \]  

(3.82)

3.4.5 Application to signals

Figure 3.1, 3.2, 3.3, and 3.4 show four DWDs computed for the same signals as in Section 2.3.

For the two stationary signals, the DWDs are helpful. In fact, the power-spectrum can be obtained by simply integrating the DWD along \( n \) at a fixed \( \theta \) provided that the original signal is over-sampled or analytical.

For the two nonstationary signals, the DWDs are mainly concentrated around at the instantaneous frequency and shows the trend of local spectrum.

It can be concluded that the DWD is good for analysing both stationary and nonstationary signals from these figures.
Figure 3.1: A stationary signal $f[n]$ and its DWD
3.4 The discrete Wigner distribution (DWD)

Density Plot of $|W_f(n, \theta)|$

Contour Plot of $|W_f(n, \theta)|$

A stationary signal $f[n]$ and its DWD (continued)
3.4 The discrete Wigner distribution (DWD)

Figure 3.2: Another stationary signal $f[n]$ and its DWD
3.4 The discrete Wigner distribution (DWD)

Another stationary signal $f[n]$ and its DWD (continued)
3.4 The discrete Wigner distribution (DWD)

Figure 3.3: A chirp signal $f[n]$ and its DWD
3.4 The discrete Wigner distribution (DWD)

A chirp signal $f[n]$ and its DWD (continued)
3.4 The discrete Wigner distribution (DWD)

Figure 3.4: A frequency modulated signal $f[n]$ and its DWD
A frequency-modulated signal $f[n]$ and its DWD (continued)
3.4.6 Application to signals with noise

Figure 3.5, 3.6, 3.7, and 3.8 show four DWDs computed for the same signals as in Section 2.4.4, but with noise. The noise is of type

\[ n(x) = n_1(x) + in_2(x) \]

where \( n_1(x) \) and \( n_2(x) \) are random noise with the range as \([-0.1, 0.1]\), about 10% percent of the original signal.

From these figures, the same conclusion can be reached that the DWD is good for analysing both stationary and nonstationary signals from these figures.
3.4 The discrete Wigner distribution (DWD)

Figure 3.5: A stationary signal with noise and its DWD
A stationary signal with noise and its DWD (continued)
Three-dimensional plot of $W_f(n, \theta)$

Figure 3.6: Another stationary signal with noise and its DWD
Another stationary signal with noise and its DWD (continued)
3.4 The discrete Wigner distribution (DWD)

Figure 3.7: A chirp signal with noise and its DWD
A chirp signal with noise and its DWD (continued)
3.4 The discrete Wigner distribution (DWD)

Figure 3.8: A frequency modulated signal with noise and its DWD
A frequency-modulated signal with noise and its DWD (continued)
3.5 Relationship with the AF

Both the WD and AF belong to mixed distance-frequency or time-frequency representations of signals, and are very closely related to each other.

For a signal \( f(x) \) with the FT as \( F(\omega) \), let

\[
\gamma_f(x, \chi) = f(x + \frac{\chi}{2})^* f(x - \frac{\chi}{2})
\]

\[
\Gamma_f(\omega, \omega) = F(\omega + \frac{\omega}{2})^* F(\omega - \frac{\omega}{2})
\]

so

\[
\Gamma_f(\omega, \omega) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \gamma_f(x, \chi) e^{-j\omega x} e^{-j\omega x} dx d\chi \tag{3.83}
\]

\[
\gamma_f(x, \chi) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Gamma_f(\omega, \omega) e^{j\omega x} e^{j\omega x} d\omega d\omega \tag{3.84}
\]

then the WD and AF are defined as

\[
W_f(x, \omega) = \int_{-\infty}^{\infty} \gamma_f(x, \chi) e^{-j\omega x} dx \tag{3.85}
\]

\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} \Gamma_f(\omega, \omega) e^{j\omega x} d\omega \tag{3.86}
\]

\[
A_f(\omega, \chi) = \int_{-\infty}^{\infty} \gamma_f(x, \chi) e^{-j\omega x} dx \tag{3.87}
\]

\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} \Gamma_f(\omega, \omega) e^{j\omega x} d\omega \tag{3.88}
\]

The above equations can be represented in Figure 3.9

From Figure 3.9, it follows that

\[
A_f(\omega, \chi) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_f(x, \omega) e^{-j\omega x} e^{j\omega x} dx d\omega \tag{3.89}
\]

\[
W_f(x, \omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} A_f(\omega, \chi) e^{j\omega x} e^{-j\omega x} d\omega d\chi \tag{3.90}
\]

The differences between the WD and the AF can be seen in Table 3.1. Shifting in spatial or frequency domain leads to a corresponding shifting in the WD; the effect on the AF is only a phase factor. Moreover, for a limited signal in
the spatial domain the WD is spatially limited and for a limited signal in the frequency domain the WD is frequency limited; while the AF does not have these properties. For the WD, the interpretation of the spatial and frequency variable corresponds to that of the original signal, while it does not for the AF. From these, it could be concluded that the WD is better than the AF in terms of analyzing both stationary and nonstationary signals. However, in order to verify this, both theoretical and practical application are investigated in Chapter 4 and Chapter 6.
<table>
<thead>
<tr>
<th></th>
<th>( f(x) )</th>
<th>( F(\omega) )</th>
<th>( A_f(\omega, \chi) )</th>
<th>( W_f(x, \omega) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spatial Shifting</strong></td>
<td>( f(x - x_0) )</td>
<td>( F(\omega) e^{-j\omega x_0} )</td>
<td>( A_f(\omega, \chi) e^{-j\omega x_0} )</td>
<td>( W_f(x - x_0, \omega) )</td>
</tr>
<tr>
<td><strong>Frequency Shifting</strong></td>
<td>( f(x) e^{j\omega_0 x} )</td>
<td>( F(\omega - \omega_0) )</td>
<td>( A_f(\omega, \chi) e^{j\omega_0 \chi} )</td>
<td>( W_f(x, \omega - \omega_0) )</td>
</tr>
<tr>
<td><strong>Spatial Limited</strong></td>
<td>([x_a, x_b]) for ( x )</td>
<td>([-\infty, \infty]) for ( \omega )</td>
<td>([- (x_b - x_a), x_b - x_a]) for ( \chi )</td>
<td>([x_a, x_b]) for ( x )</td>
</tr>
<tr>
<td><strong>Frequency Limited</strong></td>
<td>([-\infty, \infty]) for ( x )</td>
<td>([\omega_a, \omega_b]) for ( \omega )</td>
<td>([- (\omega_b - \omega_a), \omega_b - \omega_a]) for ( \chi )</td>
<td>([\omega_a, \omega_b]) for ( \omega )</td>
</tr>
<tr>
<td><strong>Convolution</strong></td>
<td>( \int_{-\infty}^{\infty} f(\chi) h(x - \chi) d\chi )</td>
<td>( F(\omega) H(\omega) )</td>
<td>( \int_{-\infty}^{\infty} A_f(\omega, x) A_h(\omega, x - x) dx )</td>
<td>( \int_{-\infty}^{\infty} W_f(\chi, \omega) W_h(\chi, x - x, \omega) d\chi )</td>
</tr>
<tr>
<td><strong>Modulation</strong></td>
<td>( f(x) m(x) )</td>
<td>( \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) M(\omega - \omega) d\omega )</td>
<td>( \frac{1}{2\pi} \int_{-\infty}^{\infty} A_f(\omega, x) A_m(\omega - \omega, \chi) d\omega )</td>
<td>( \frac{1}{2\pi} \int_{-\infty}^{\infty} W_f(x, \omega) W_m(x, \omega - \omega) d\omega )</td>
</tr>
<tr>
<td><strong>Typical Stationary</strong></td>
<td>( Ae^{j\omega_0 x} )</td>
<td>( A 2\pi \delta(\omega - \omega_0) )</td>
<td>(</td>
<td>A</td>
</tr>
<tr>
<td><strong>Typical Non-stationary</strong></td>
<td>( Ae^{j\frac{\alpha}{2} x^2} )</td>
<td>( A^2 e^{j\frac{\alpha}{2} x^2} e^{-j\frac{\alpha^2}{4\omega} x^2} )</td>
<td>(</td>
<td>A</td>
</tr>
</tbody>
</table>
Chapter 4

Extraction of Parameters from the WD

4.1 Introduction

In the previous chapter, the Wigner distribution, in both continuous and discrete forms, was studied, and it was suggested that the WD could be a realistic tool for representation and analysis of signals, especially non-stationary signals.

In this chapter, the WD is employed to analyse non-stationary signals and to extract relevant parameters. Two methods of extracting parameters are discussed. One is to employ the Hough transform; the other is to utilise local moments in frequency. More emphasis is put on the latter because it is simpler and easier to use, and more widely applicable.

Both chirp and frequency-modulated signals are tested here. These two types of signals are commonly found in engineering. Furthermore, frequency-modulated signals are in fact contained in the surface texture of a workpiece machined with the presence of tool vibration as will be shown later.
4.2 The Hough transform

4.2.1 Preliminary

The Hough transform deals with the detection of specific structural relationships between pixels in an image (therefore it can be used to extract valuable information from the WD). The Hough transform was first proposed by Hough (1962), later popularised by Duda and Hart (1972). A generalisation of the Hough transform for detecting arbitrary shapes has been proposed by Ballard (1981). Methods for reducing computational complexity have been investigated by Merlin and Farber (1975), and Davis (1982).

For an introduction to the Hough transform, suppose that given $N$ points in an image, it is required to find subsets of these points that lie on straight lines. One method is to use the least-square fitting. However, this only works for very simple case, i.e. all the points are very close on one single line. Another method is to find all $N(N-1)/2$ possible lines first, then to perform $(N-2)N(N-1)/2$ comparison of all points to each line. The complexity of this algorithm is $O(N^3)$ or at least $O(N^2)$. As a result, this method is computationally prohibitive in all but the most trivial cases.

4.2.2 Fundamentals

This problem for detection lines can be solved efficiently by the Hough transform. For a fixed point $(x_i, y_i)$ in $xy$ plane (the image plane), there is an infinite number of lines that pass through $(x_i, y_i)$ and they all have the line equation in slope-intercept form as

$$y_i = ax_i + b$$

for various values of $a$ and $b$, or

$$b = -ax_i + y_i$$
4.2 The Hough transform

which is the equation of a single line in the \( ab \) plane (the parameter space). In other words, each point in the \( xy \) plane corresponds to a straight line in the \( ab \) plane. It can be deduced that

1. the set of lines in the \( ab \) plane, having the same values for \( a \) or \( b \), means that those corresponding points in the \( xy \) plane have the same slope or intercept, respectively.

2. each intersection of these lines in the \( ab \) plane means that those corresponding points in the \( xy \) plane are on the same line.

Therefore, to detect lines in the \( xy \) plane, it is only necessary to find those intersections of lines in the \( ab \) plane.

The algorithm can be described as:

1. Let \( (a_{\text{min}}, a_{\text{max}}) \) and \( (b_{\text{min}}, b_{\text{max}}) \) be the expected ranges of slope and intercept values, then quantise \( (a_{\text{min}}, a_{\text{max}}) \) and \( (b_{\text{min}}, b_{\text{max}}) \) into \( KL \) squares. Create a two-dimensional array \( C[K][L] \) (This is in C language). and initialise \( C \) to zero.

2. Then for every point \( (x_i, y_i) \) in the image plane, let the parameter \( a \) equal each of the allowed subdivision values \( a_k \) on the axis and solve for the corresponding \( b \)'s by using the equation \( b = -x_i a + y_i \). The resulting \( b \)'s are then rounded off to the nearest allowed values \( b_l \) in the \( b \) axis. Then \( C[k, l] \) is increased by 1.

3. At the end of this procedure, a value of \( P \) in \( C[k][l] \) corresponds to \( P \) points in the \( xy \) plane lying on the line \( y = a_k x + b_l \). To find a line passing most of points in the \( xy \) plane is only necessary to find the maximum value of \( P \) in \( C[k][l] \).

For \( N \) points, there are \( NK \) values of \( b \) to compute, therefore, the complexity is \( O(N) \) provided that \( K \) is small.
4.2.3 Examples

Take for example a chirp signal. The result is good because the parameter $\alpha$ can be extracted quite simply from the graph. See Figure 4.1 and Figure 4.2.
4.2 The Hough transform

Three-dimensional plot of $W_f(n, \theta)$

Figure 4.1: The signal $f[n]$ and its DWD
4.2 The Hough transform

Figure 4.2: The Hough transform. True: \( \alpha = \frac{0.25\pi}{N} \); extracted: \( \alpha = \frac{0.245\pi}{N} \)
4.2.4 Conclusion

Although the Hough transform is effective in extracting information from the DWD for chirp signals, its usage is limited for more complicated signals, such as frequency-modulated (FM) signals, this is because

1. For FM signals, the instantaneous frequency is not linearly increasing any more, in fact, it is changing sinusoidally. Therefore, to describe it, the curve equation should be

\[ y = a + b \sin(\omega x + \phi) \]

which has four parameters rather than two. This results in more complicated parameter spaces which can have computation problems.

2. For FM signals, at a fixed distance or time, the local spectrum is not only concentrating on the instantaneous frequency, but also fluctuates at nearby. This complicates image spaces.

3. Some tricks (such as low pass filters) can be used to mask out those fluctuation so that the HT can still be used. However, this is messy and a bit complicated.

To overcome these problems, a new method is adopted instead. This involves the usage of the local moments in frequency and yields good results for various nonstationary signals. This will be described in the rest of this chapter.
4.3 Local moments in frequency

For a complex-valued signal $f(x) = a(x)e^{j\phi(x)}$ where $a(x)$ and $\phi(x)$ are real-valued, let $W_f(x, \omega)$ be its WD. Let $p_f(x)$ and $\Omega_f(x)$ be the 0th- and 1st-order local moments in frequency of the WD, then

\begin{align}
    p_f(x) &= a^2(x) \quad (4.1) \\
    \Omega_f(x) &= \phi'(x) \quad (4.2)
\end{align}

which means that the 0th-order moment in frequency is the instantaneous power and the 1st-order moment in frequency is the instantaneous frequency.

For a chirp signal $f(x) = ae^{j\frac{\alpha}{2}x^2}$,

\begin{align}
    p_f(x) &= a^2 \quad (4.3) \\
    \Omega_f(x) &= \alpha x \quad (4.4)
\end{align}

and a FM signal $f(x) = Ae^{j(\omega_0 x + \phi_0 + b \sin(\omega_m x + \phi_m))}$,

\begin{align}
    p_f(x) &= A^2 \quad (4.5) \\
    \Omega_f(x) &= \omega_0 + b\omega_m \cos(\omega_m x + \phi_m) \quad (4.6)
\end{align}

If both $p_f(x)$ and $\Omega_f(x)$ are known, then it is trivial to extract relevant information about both chirp and FM signals ($a$, $\alpha$; $a$, $b$, $\omega_0$, $\omega_m$). Since $p_f(x)$ and $\Omega_f(x)$ can be easily computed from the WD, so are the parameters. Furthermore, this method can be applied to other types of nonstationary signals as well.

Although it is straightforward to extract parameters for the complex-valued signal, it is the real-valued signal which is often encountered in engineering and for which the moment method can not be applied directly. However, if its corresponding analytical signals are used, then this method can still be used.
4.4 Complex-valued chirp signals

Fig 4.3, 4.7 show five cases to extract parameters from complex-valued chirp signals.

The tested signals are of form

\[
f[n] = \left\{ \begin{array}{ll}
                a e^{j \frac{\pi}{N} n^2} & 0 \leq n < N \\
                0 & \text{otherwise}
                \end{array} \right.
\]

(4.7)

where \( N = 256 \) and \( n = \cdots, -2, -1, 0, 1, 2, \cdots \). Its moments are

\[
p_f[n] = a^2 \quad (4.8)
\]

\[
\Theta_f[n] = \alpha n \mod \pi \quad (4.9)
\]

where \( 0 \leq n < N \).

Note that the extracted values for \( a \) are the same as original values within three significant digits. Both extracted and true values for \( \alpha \) are listed in Table 4.1

<table>
<thead>
<tr>
<th>true value ( (\frac{x}{N}) )</th>
<th>0.100</th>
<th>0.200</th>
<th>0.300</th>
<th>0.400</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>extracted value ( (\frac{x}{N}) )</td>
<td>0.098</td>
<td>0.195</td>
<td>0.293</td>
<td>0.391</td>
<td>0.488</td>
</tr>
<tr>
<td>error (%)</td>
<td>2.0</td>
<td>2.5</td>
<td>2.3</td>
<td>2.3</td>
<td>2.4</td>
</tr>
</tbody>
</table>
4.4 Complex-valued chirp signals

Figure 4.3: true: $a = 1.000$, $\alpha = \frac{0.100\pi}{N}$, Extracted: $a = 1.000$, $\alpha = \frac{0.098\pi}{N}$. 
Figure 4.4: True: $a = 1.000$, $\alpha = \frac{0.200\pi}{N}$, Extracted: $a = 1.000$, $\alpha = \frac{0.195\pi}{N}$. 
Figure 4.5: True: \( a = 1.000, \alpha = \frac{0.300\pi}{N} \), Extracted: \( a = 1.000, \quad \alpha = \frac{0.293\pi}{N} \).
4.4 Complex-valued chirp signals

Figure 4.6: True: $a = 1.000$, $\alpha = \frac{0.400\pi}{N}$, Extracted: $a = 1.000$, $\alpha = \frac{0.391\pi}{N}$. 
4.4 Complex-valued chirp signals

From the above two tables, it is clear that the results are good although they are not exact. Therefore, it is necessary to use the WD.

Both extracted values are real-valued, it is necessary to use the WD.

4.5 Real-valued chirp signals

The second chirp signal

where \( N = 256 \).

The WD.

Both extracted values are real-valued, it is necessary to use the WD.

\[ f[n] = a 

\]

\[ \omega[n] = \frac{0.500\pi}{N} \]

\[ \alpha[n] = \frac{0.488\pi}{N} \]

\[ \theta[n] \]

\[ P[n] \]

\[ \omega[n] \]

\[ \theta[n] \]

Figure 4.7: True: \( a = 1.000, \alpha = \frac{0.500\pi}{N}, \) Extracted: \( a = 1.000, \alpha = \frac{0.488\pi}{N} \).
4.5 Real-valued, chirp signals

Fig 4.8, ⋯, 4.12 show five cases to extract parameters from real-valued chirp signals.

The tested signals are of form

\[ f[n] = \begin{cases} 
    A \cos(\frac{\pi}{2} n^2) & 0 \leq n < N \\
    0 & \text{otherwise}
\end{cases} \]  \hspace{1cm} (4.10)

where \( N = 256 \) and \( n = \cdots, -2, -1, 0, 1, 2, \cdots \). Because these signals are real-valued, it is necessary to convert them to their analytical signals before applying the WD.

Both extracted and true values for \( a \) are listed in Table 4.2

<table>
<thead>
<tr>
<th>true</th>
<th>1.000</th>
<th>1.000</th>
<th>1.000</th>
<th>1.000</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>extracted</td>
<td>1.038</td>
<td>1.024</td>
<td>1.025</td>
<td>1.016</td>
<td>1.014</td>
</tr>
<tr>
<td>error (%)</td>
<td>3.8</td>
<td>2.4</td>
<td>2.5</td>
<td>1.6</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Both extracted and true values for \( \alpha \) are listed in Table 4.3

<table>
<thead>
<tr>
<th>true (( \frac{\pi}{N} ))</th>
<th>0.100</th>
<th>0.200</th>
<th>0.300</th>
<th>0.400</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>extracted (( \frac{\pi}{N} ))</td>
<td>0.085</td>
<td>0.191</td>
<td>0.284</td>
<td>0.366</td>
<td>0.485</td>
</tr>
<tr>
<td>error (%)</td>
<td>15.0</td>
<td>4.5</td>
<td>5.3</td>
<td>8.5</td>
<td>3.0</td>
</tr>
</tbody>
</table>

From the above two tables, it can be seen that the results are fairly good although they are not as good as for complex-valued signals.
4.5 Real-valued, chirp signals

Figure 4.8: True: $a = 1.000$, $\alpha = \frac{0.100\pi}{N}$, Extracted: $a = 1.038$, $\alpha = \frac{0.085\pi}{N}$. 
Figure 4.9: True: $a = 1.000$, $\alpha = \frac{0.200 \pi}{N}$, Extracted: $a = 1.024$, $\alpha = \frac{0.191 \pi}{N}$.
4.5 Real-valued, chirp signals

Figure 4.10: True: $a = 1.000$, $\alpha = \frac{0.300\pi}{N}$, Extracted: $a = 1.025$, $\alpha = \frac{0.284\pi}{N}$. 
4.5 Real-valued, chirp signals

Figure 4.11: True: $a = 1.000$, $\alpha = \frac{0.400\pi}{N}$, Extracted: $a = 1.016$, $\alpha = \frac{0.360\pi}{N}$. 
4.5 Real-valued, chirp signals

4.6 Complex-valued FM signals

Here the sine-Gaussian signal

\[
\begin{align*}
p[n] &= A \sin(\pi (n - \nu)^2/\nu^2) \cos(2\pi \nu n / N) \\
\theta[n] &= \sin(\pi (n - \nu)^2/\nu^2) \sin(2\pi \nu n / N)
\end{align*}
\]

where \( N = 256 \) and \( \nu = \frac{1}{2} - 1.0 \). Figure 4.12 shows the agreement is already at about 6 of 9 steps, 0.26% within three significant decimal places. The reason why the agreement is already at about 0.26% within three significant decimal places is that the agreement is already at about 0.26% within three significant decimal places. The reason why the agreement is already at about 0.26% within three significant decimal places is that the agreement is already at about 0.26% within three significant decimal places.

Figure 4.12: True: \( a = 1.000 \), \( \alpha = \frac{0.500\pi}{N} \), Extracted: \( a = 1.014 \), \( \alpha = \frac{0.485\pi}{N} \).
4.6 Complex-valued, FM Signals

Fig 4.13, ..., 4.17, show five cases of parameter extraction for complex-valued FM signals.

Here the tested signals are of form

\[ f[n] = \begin{cases} a e^{i(\theta_a n + \phi_a + b \sin(\theta_m n + \phi_m))} & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \quad (4.11) \]

where \( N = 256 \) and \( n = \cdots, -2, -1, 0, 1, 2, \cdots \). Its moments are

\[ p_f[n] = a^2 \quad (4.12) \]
\[ \Theta_f[n] = \theta_a + b \sin \theta_m \cos(\theta_m n + \phi_m) \mod \pi \quad (4.13) \]

when \( 0 \leq n < N \).

Note that the extracted values for \( k_o \) and \( k_m \) are the same as original values within three significant digits (except for \( k_m \) in Fig 4.17). The error of extraction for \( a \) is about 0.2%. The extracted values for \( b \) are the same as original values within three significant digits.

The reason why the agreement is not exact, is probably because of numerical rounding off errors in the computer evaluation. However, because the agreement is already so high, steps to improve the correlation have not been deemed necessary.
4.6 Complex-valued, FM Signals

Figure 4.13: True: $a = 1.000, \theta_0 = 32.0 \frac{2\pi}{N}, b = 1.000,$ $\theta_m = 4.0 \frac{2\pi}{N},$ Extracted: $a = 0.998, \theta_0 = 32.0 \frac{2\pi}{N}, b = 1.000,$ $\theta_m = 4.0 \frac{2\pi}{N}.$
4.6 Complex-valued, FM Signals

Figure 4.14: True: \(a = 1.000, \theta_o = 32.0 \frac{2\pi}{N}, b = 2.000, \theta_m = 4.0 \frac{2\pi}{N}\), Extracted: \(a = 0.998, \theta_o = 32.0 \frac{2\pi}{N}, b = 2.000, \theta_m = 4.0 \frac{2\pi}{N}\).
Figure 4.15: True: $a = 1.000, \theta_o = 32.0 \frac{2\pi}{N}, b = 4.000, \theta_m = 4.0 \frac{2\pi}{N}$, Extracted: $a = 0.998, \theta_o = 32.0 \frac{2\pi}{N}, b = 4.000, \theta_m = 4.0 \frac{2\pi}{N}$. 

4.6 Complex-valued, FM Signals
4.6 Complex-valued, FM Signals

Figure 4.16: True: $a = 1.000, \theta_o = 32.0 \frac{2\pi}{N}, b = 6.000, \theta_m = 4.0 \frac{2\pi}{N}$, Extracted: $a = 0.998, \theta_o = 32.0 \frac{2\pi}{N}, b = 6.000, \theta_m = 4.0 \frac{2\pi}{N}$. 
4.6 Complex-valued, FM Signals

4.7 FM Signals

Fig 4.16, ..., some plots exist for complex-valued FM signals.

\[ f[n] = P[n] \cos(\theta[n] + \phi[n]) + \sin(\theta[n] + \phi[n]) \]

(4.14)

\[ f[n] = \frac{2}{N} \sum_{k=0}^{N-1} \cos(\theta_k) \cos(\phi_k) - \sin(\theta_k) \sin(\phi_k) \]

where \( N = 256 \).

Note that the highest values within three signal.

The results of the experiment are in Table 4.5.

Figure 4.17: True: \( a = 1.000, \theta_o = 32.0 \frac{2\pi}{N}, b = 8.000, \theta_m = 4.0 \frac{2\pi}{N} \), Extracted: \( a = 0.998, \theta_o = 31.0 \frac{2\pi}{N}, b = 8.000, \theta_m = 4.0 \frac{2\pi}{N} \).
4.7 Real-valued, FM signals

Fig 4.18, ..., 4.22 show five cases of parameter extraction for complex-valued FM signals.

Here the tested signals are of form

\[ f[n] = \begin{cases} 
  a \cos(\theta_0 n + \phi_0) + b \sin(\theta_m n + \phi_m) & 0 \leq n < N \\
  0 & \text{otherwise}
\end{cases} \] (4.14)

where \( N = 256 \) and \( n = \cdots, -2, -1, 0, 1, 2, \cdots \).

Note that the extracted values for \( k_0 \) and \( k_m \) are the same as original values within three significant digits.

The results of extraction for both \( a \) and \( b \) are listed in Table 4.4 and Table 4.5.

Table 4.4: Extraction results for \( a \)

<table>
<thead>
<tr>
<th>original</th>
<th>1.000</th>
<th>1.000</th>
<th>1.000</th>
<th>1.000</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>extracted</td>
<td>0.998</td>
<td>0.998</td>
<td>0.998</td>
<td>1.008</td>
<td>1.128</td>
</tr>
<tr>
<td>error (%)</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.8</td>
<td>12.8</td>
</tr>
</tbody>
</table>

Table 4.5: Extraction results for \( b \)

<table>
<thead>
<tr>
<th>original</th>
<th>1.000</th>
<th>2.000</th>
<th>4.000</th>
<th>6.000</th>
<th>8.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>extracted</td>
<td>1.000</td>
<td>2.000</td>
<td>4.000</td>
<td>6.010</td>
<td>8.012</td>
</tr>
<tr>
<td>error (%)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Figure 4.18: True: $a = 1.000$, $\theta_o = 32.0\frac{2\pi}{N}$, $b = 1.000$, $\theta_m = 4.0\frac{2\pi}{N}$. Extracted: $a = 0.998$, $\theta_o = 32.0\frac{2\pi}{N}$, $b = 1.000$, $\theta_m = 4.0\frac{2\pi}{N}$.
4.7 Real-valued, FM signals

Figure 4.19: True: $a = 1.000$, $\theta_o = 32.0 \frac{2\pi}{N}$, $b = 2.000$, $\theta_m = 4.0 \frac{2\pi}{N}$, Extracted: $a = 0.998$, $\theta_o = 32.0 \frac{2\pi}{N}$, $b = 2.000$, $\theta_m = 4.0 \frac{2\pi}{N}$. 

---

---
4.7 Real-valued, FM signals

Figure 4.20: True: $a = 1.000$, $\theta_\omega = 32.0 \frac{\pi}{N}$, $b = 4.000$, $\theta_m = 4.0 \frac{\pi}{N}$, Extracted: $a = 0.998$, $\theta_\omega = 32.0 \frac{\pi}{N}$, $b = 4.000$, $\theta_m = 4.0 \frac{\pi}{N}$.
Figure 4.21: True: \( a = 1.000, \theta_o = 32.0 \frac{2\pi}{N}, b = 6.000, \theta_m = 4.0 \frac{2\pi}{N} \), Extracted: \( a = 1.008, \theta_o = 32.0 \frac{2\pi}{N}, b = 6.010, \theta_m = 4.0 \frac{2\pi}{N} \).
4.7 Real-valued, FM signals

4.8 Conclusions

The results from the previous four sections show that over a wide range of algorithms based on the parameters do not seem to depend on the specific value of $N$. Furthermore, this technique can be easily extended to higher-order real-valued signals. Having established a means of extracting the parameters of a signal, the next described in the

Figure 4.22: True: $a = 1.000, \theta_o = 32.0\frac{2\pi}{N}, b = 8.000, \theta_m = 4.0\frac{2\pi}{N}$. Extracted: $a = 1.128, \theta_o = 32.0\frac{2\pi}{N}, b = 8.012, \theta_m = 4.0\frac{2\pi}{N}$.
4.8 Conclusion

The results from the previous four sections show that over a wide range the algorithms based on the moments of the WD can detect non-stationary parameters successfully and accurately. More spectacularly the extraction of the parameters does not seem to depend on the severity of nonstationary. This seems to be true for all cases likely to be met with in manufacturing. Furthermore, this technique can be easily extended for general signals.

Having established a potential tool for non-stationary and stationary evaluation of a signal, the next step is to apply it to practical signals. This will be described in the next two chapters.
Chapter 5

Direct Measurement of Cutting Tool Vibration

In the previous chapter, the WD and its moments were successfully applied to simulated signals. However, this thesis is about the application of the WD to machine tool monitoring, in particular, to use the WD to extract vibration information from surface texture of a machined workpiece and use it for machine tool monitoring and control. In order to demonstrate the feasibility of this technique, it is necessary to review certain aspects of cutting tool vibration characteristics and more importantly how to measure it. This is done in this chapter. In the next chapter this practical vibration data will be used to validate the WD approach for machine tool monitoring (Chapter 6) (Zheng and Whitehouse 1992).

5.1 Introduction

This chapter consists of two parts: a discussion of cutting tool vibration and its measurement.

Some fundamentals of the vibration need to be examined, for it is the vibration that is going to be employed to monitor the machine-tool. Although much work has gone into understanding the mechanics of vibration, due to the very complicated dynamic characteristics of the machine tool, cutting tool and work-
piece, it is still not fully explained. Therefore, this chapter attempts to clarify some of the relevant fundamentals of cutting tool vibration. In Sections 5.2 to 5.6, an analysis of vibration is presented, following Tobias (1965) and Sweeney (1971).

The remainder of this chapter describes the instrument for cutting tool vibration measurement, developed by the author. To measure vibration, a piezoelectric accelerometer is usually employed, but it is not suitable for cutting tool vibration because 1) the dynamic range is not large enough for the strong vibration of a cutting tool and 2) it is difficult to mount an accelerometer on a cutting tool. Therefore, an instrument has had to be built especially for vibration measurement. Its construction and performance will be described in detail in the second part of this chapter.

5.2 Elementary vibratory systems

5.2.1 The equation of systems with one-degree of freedom

The simplest vibratory system is an idealised one-degree (1-D) of freedom, vibratory system as shown in Figure 5.1. It consists of a spring $c$, a mass $m$ and a dashpot $\rho$ (Tobias 1965, p 5).

If the mass $m$ is displaced from its position of rest, it will be acted upon by the following forces: the weight of mass $m$, the restoring force of the spring $c$, the
damping force of the dashpot $\rho$ and the external exciting force $F(t)$.

From Newton's second law of motion,

$$m\ddot{x} = F(t) - cx - \rho \dot{x}$$  \hspace{1cm} (5.1)

where $x$ is the displacement from the rest position (therefore the weight of the mass is balanced by the initial movement of the spring).

Rewriting,

$$m\ddot{x} + \rho \dot{x} + cx = F(t)$$ \hspace{1cm} (5.2)

which is the basic equation for a system with 1-D of freedom. Let

$$\delta = \frac{\rho}{2m}$$

$$\omega_0^2 = \frac{c}{m}$$

$$D = \frac{\delta}{\omega_0}$$

then Eqn 5.2 can be written as

$$\ddot{x} + 2\delta \dot{x} + \omega_0^2 x = \frac{F(t)}{m}$$ \hspace{1cm} (5.3)

or

$$\ddot{x} + 2D\omega_0 \dot{x} + \omega_0^2 x = \frac{F(t)}{m}$$ \hspace{1cm} (5.4)

- When $F(t) = 0$, it is called free vibration.

- When $F(t) \neq 0$, it is called forced vibration.

5.2.2 Stability

In order to operate properly, a machine-tool has to be stable. There are two kinds of stability: static and dynamic.
5.3 Free-vibrations

Static stability

A system is said to be *statically stable* only if it will return to its original equilibrium after the cessation of a static disturbance. This means that there must exist a force (so called *restoring force*) which tends to restore the system to its equilibrium condition. For a 1-D vibratory system,

- if $c > 0$, it is statically stable;

- otherwise it is not.

It is an intrinsic requirement that the machine-tool should be statically stable.

Dynamic stability

After an impact or other disturbances, a system in a steady-state condition of uniform motion may vibrate. If the vibration dies away and the uniform motion is thus restored, then it is said to be *dynamically stable*. However, it can also happen that the vibration will increase continuously. This type of system is termed as dynamically unstable. For a 1-D vibratory system with $c > 0$,

- if $\delta > 0$, it is dynamically stable;

- otherwise, it is not.

For machine-tool vibrations, some are dynamically stable, others are not, such as the case of self-induced vibration.

5.3 Free-vibrations

For free-vibrations, Eqn 5.4 turns out to be

$$\ddot{x} + 2D\omega_0 \dot{x} + \omega_0^2 x = 0$$

with the initial condition as $x(0) = x_0$ and $\dot{x}(0) = \dot{x}_0$. 
5.3.1 Weak damping

For weak damping, with the damping ratio $D$ in the interval $(-1, 1)$, the solution for Eqn 5.5 is

$$x = e^{-\delta t}(C_1 \cos \omega_d t + C_2 \sin \omega_d t)$$ (5.6)

where $\omega_d = \sqrt{\omega_0^2 - \delta^2}$, $C_1 = x_0$ and $C_2 = \frac{\dot{x}_0 + \delta \dot{x}_0}{\omega_d}$

5.3.2 Critical damping

When $|D| = 1$,

$$x = e^{-\delta t}(C_1 + C_2 t)$$ (5.7)

where $C_1$ and $C_2$ are determined by the initial conditions of displacement and velocity.

5.3.3 Heavy damping

When $|D| > 1$,

$$x = C_1 e^{(-\delta + \sqrt{\delta^2 - \omega_0^2})t} + C_2 e^{(-\delta - \sqrt{\delta^2 - \omega_0^2})t}$$ (5.8)

where $C_1$ and $C_2$ are determined by the initial conditions of displacement and velocity.

For a machine tool, free-vibrations can be induced by a shock or similar means.

5.4 Forced-vibrations

Because the disturbing force can be resolved into a finite or infinite number of components varying harmonically with time, it is therefore sufficient to solve

$$\ddot{x} + 2\delta \dot{x} + \omega_0^2 x = \frac{F_n}{m} \cos(\omega_n t + \phi_n)$$ (5.9)
The general solution for other types of $F(t)$ can then be constructed because Eqn 5.4 is linear.

For Eqn 5.9, the solution (for weak damping) is

$$x = e^{-\xi t}(C_1 \cos \omega_d t + C_2 \cos \omega_d t)$$

$$+ \frac{F_n}{m} \frac{1}{\sqrt{(\omega_0^2 - \omega_n^2)^2 + 4\delta_n^2 \omega_n^2}} \cos(\omega_n t - \phi_n)$$

where $C_1$ and $C_2$ are determined by the initial conditions of displacement and velocity.

For a machine-tool, forced vibration is usually caused by an out-of-balanced force associated with a component integral with, or external to the machine tool. There are four methods of tackling machine tool forced vibration:

1. by eliminating the exciting force at source,

2. by vibration isolation,

3. by an undamped dynamic vibration absorber,

4. or by a Lanchester vibration absorber.

5.4.1 Vibration isolations

There are two methods for isolating the machine-tool from the disturbing vibration. The first is by preventing the vibration passing from the source into the machine foundation. The second is by preventing vibration transmitted through the ground from entering the machine-tool. The former is termed active isolation and the latter passive isolation.

5.4.2 Undamped dynamic vibration absorbers

This involves attaching an additional mass to the original system through a spring. The secondary system is so designed that it supplies a force at the dis-
5.5 Self-excited vibrations

turbing frequency which will be against the disturbing force.

5.4.3 Lanchester vibration absorbers

Instead of a spring, a damper can be used to couple a mass to the machine tool structure. This increases the damping and suppresses the amplitude at resonance. This absorber is normally called a *Lanchester vibration absorber*.

Of course, it is possible to connect the absorber mass to the main mass through both a spring and damper to give a damped dynamic vibration absorber, which offers certain advantages over the undamped type.

Theoretically, both free and forced vibrations can be eliminated provided that the causes responsible for inducing it have been identified (however, it is not always possible that the remedy can be practically realised). Moreover, vibrations of these two types are usually detected in the maker’s works, therefore necessary counter-measures can be devised while the machine is still in the development stage.

There exists another type of vibration called *self-excited vibration* or *chatter*, which is far more difficult to cope with than free and forced vibrations. This will be discussed more in the next section.

5.5 Self-excited vibrations

Self-excited vibration is spontaneous. It is not induced by external periodic forces, but the forces generated in the vibratory processes itself. For a 1-D vibratory system, if it is dynamically unstable, then its vibration is self-induced. For the machine tool, the self-excited vibration is the result of interaction between the cutting force dynamics and the machine tool structure dynamics. It draws the energy for its maintenance from the tool or workpiece drive.

In machining, self-induced vibrations can occur in the following two ways:

1. Under certain conditions, the cutting process is basically, dynamically un-
stable. As a result, any slight displacement of the tool relative to the workpiece can rapidly build into a large-amplitude vibration.

2. The second way in which self-induced vibrations can occur is more complicated but more common. In this case, the cutting process is basically dynamically stable. However, because of overlapping cuts, the forced vibration resulting from the machining of the wavy surface from the previous stroke or revolution of the workpiece or tool can amplify the previous vibration. This type of vibration is called regenerative chatter.

5.6 The cantilever

As a first approximation, the cutting tool can be treated as a cantilever. A cantilever is a member fixed at one end and subjected to loads applied transverse to the long dimension causing the member to bend. See Figure 5.2.

Figure 5.2: The cantilever.
5.6.1 The equation for a cantilever

To derive the differential equation for a cantilever, consider the forces and moments acting on an element shown in Figure 5.3.

\[ p(x) \text{ represents the loading per unit length of the beam.} \]

\[ V(x) \text{ and } M(x) \text{ are shear and bending moments at } x, \text{ respectively.} \]

The forces in the \( y \) direction satisfy

\[ V + p(x) \, dx - (V + dV) = 0 \]  \hspace{1cm} (5.10)

from which

\[ \frac{dV}{dx} = p(x) \]  \hspace{1cm} (5.11)

For the moments about any point on the right face of the element,

\[ M + dM - M - V \, dx - \frac{1}{2} p(x) (dx)^2 = 0 \]  \hspace{1cm} (5.12)

In the limiting process, it follows that

\[ \frac{dM}{dx} = V(x) \]  \hspace{1cm} (5.13)
5.6 The cantilever

Let \( y(x) \) be the curve of a cantilever, then (Higdon et al 1985, p 358)

\[
EI\frac{d^2y}{dx^2} = M(x)
\]

(5.14)

where \( E \) is Young's Modulus, and \( I(x) \) the moment of inertia — the second moment of the cross-sectional area with respect to the centroidal axis.

Combining 5.11, 5.13 and 5.14, it follows that

\[
EI\frac{d^4y}{dx^4} = p(x)
\]

(5.15)

where \( I(x) \) is assumed to be independent of \( x \).

Since \( p(x) \) is the force of inertia acting on the element per unit length, it then follows

\[
p(x) = -\frac{\sigma A}{g} \frac{\partial^2 y}{\partial t^2}
\]

(5.16)

where \( \sigma \) is the weight per unit volume of the beam, \( A \) is the beam cross-section, and \( g \) is the acceleration due to gravity.

Combining Eqn 5.15 and Eqn 5.16, we then have

\[
\frac{\partial^2 y}{\partial t^2} + a^2 \frac{\partial^4 y}{\partial x^4} = 0
\]

(5.17)

where \( a = \frac{EIg}{A\sigma} \)

Eqn 5.17 is the basic differential equation for the cantilever, from which natural frequencies can be calculated.

5.6.2 The natural frequencies

Eqn 5.17 is a partial differential equation, so it can be assumed that

\[
y = X(x)T(t)
\]

(5.18)
Substituting this with Eqn 5.17 gives

\[
\frac{d^2T(t)}{dt^2} + \omega^2 T(t) = 0 \tag{5.19}
\]

\[
\frac{d^4X(x)}{dx^4} - \frac{\omega^2}{a^2} X(x) = 0 \tag{5.20}
\]

with a constant \( \omega \), whose solutions are given by

\[
T(t) = B_1 \cos \omega t + B_2 \sin \omega t \tag{5.21}
\]

\[
X(x) = C_1 \sin kx + C_2 \cos kx + C_3 \sin hx + C_4 \cosh kx \tag{5.22}
\]

with constants \( B_1 \) and \( B_2 \) determined by initial conditions, constants \( C_1, C_2, C_3, \) and \( C_4 \) determined by the end conditions, and \( k^4 = \frac{\omega^2}{a^2} = \frac{\omega^2 A\sigma}{EIg} \).

For the cantilever, at the clamped end

\[ y = 0 \quad \frac{dy}{dx} = 0 \]

at the free end

\[ \frac{d^2y}{dx^2} = 0 \quad \frac{d^3y}{dx^3} = 0 \]

Eqn 5.22 has the solutions as

\[
k_1 l \quad k_2 l \quad k_3 l \quad k_4 l \quad \cdots \tag{5.23}
\]

\[
1.875 \quad 4.694 \quad 7.855 \quad 10.966 \quad \cdots
\]

Since \( k^4 = \frac{\omega^2}{a^2} = \frac{\omega^2 A\sigma}{EIg} \), it follows that

\[
f_n = \frac{k_n^2}{2\pi} \sqrt{\frac{EIg}{A\sigma}}
\]

Moreover if the cross-section of the cantilever is rectangular with \( b \) and \( h \) as the width and height respectively then

\[
f_n = \frac{k_n^2}{2\pi} \sqrt{\frac{Eg bh^3}{A\sigma 12}}
\]
For a cantilever of steel with \( b = h = 1 \text{ cm}, \ l = 6.5 \text{ cm}, \) and let \( E = 2.0 \times 10^7 \text{ N/cm}^2, \ g = 980 \text{ cm/s}^2, \ \sigma = 0.0764 \text{ N/cm}^3, \) then

\[
f_1 = 1.9 \text{kHz}.
\]

which is close to cutting tool vibration frequency (about 1.8 kHz) in practice.

## 5.7 Measurement of vibration

### 5.7.1 Traditional method

For measurement of vibration, a vibration transducer is required to convert the mechanical vibration signal into an electrical form. There are various types of vibration transducers, such as proximity probes to sense displacement (mechanical lever, eddy current or proximity probe), velocity probes to sense velocity, and accelerometers to sense acceleration. Because displacement, velocity, and acceleration are interrelated in theory, any type of transducers can be used. However, in practice, the accelerometer is usually employed.

**Properties of accelerometers**

The accelerometer has the following advantages:

1. no moving parts, no wear;

2. compact, often low weight;

3. rugged;

4. very large dynamic range;

5. wide frequency range.

Displacement and velocity can be obtained from it by a simple analogue electronic integration technique. The disadvantage of an accelerometer is its high output impedance and no true DC response.
Application of accelerometers

As explained in the last section, an accelerometer was originally chosen to measure the vibration of the cutting tool, See Figure 5.4. The accelerometer selected was of type 8307 from Bruel and Kjaer, the output signal from which was then further amplified by a charge amplifier, and displayed on an oscilloscope.

It was found that the dynamic range of the above accelerometer was not large enough for the violent vibration of a cutting tool. For a typical case, the displacement amplitude was about 100 µm, the frequency was around 2 kHz, so the acceleration was between \(-1500 \, g\) and \(1500 \, g\) (where \(g\) is the acceleration of gravity). Furthermore, mounting the accelerometer on the cutting tool was difficult. As a result, this standard method for measuring vibration did not work very well for the cutting tool vibration.

Another approach

After considering many other possible methods, an instrument based on the linear variable differential transformer (LVDT) was developed. This instrument is able to get rid of some of the disadvantages mentioned above. See Figure 5.5 for its schematic diagram.

In the rest of this chapter, the components of this instrument are explained.
5.8 The linear variable differential transformer

5.8.1 Operating Principle

The linear variable differential transformer (LVDT) is a mechanical displacement transducer. Its construction is simple, including only two parts:

1. a primary coil and two identical secondary coils on a common former,

2. a movable magnetic armature or core.

Figure 5.8.1 shows a schematic diagram for the LVDT.

Figure 5.6: The schematic diagram of a LVDT (Doebelin 1983, p 240).
5.8 The linear variable differential transformer

Figure 5.7: The circuit diagram of a LVDT (Doebelin 1983, p240).
To operate the LVDT, the primary coil is normally excited with a sinusoidal voltage of 3 to 15 V rms amplitude and frequency of 60 to 20,000 Hz. Through the mutual coupling, the sinusoidal voltages in the secondary coils are induced, which have the same frequency as the excitation, but whose amplitudes vary with the position of the iron core. There is a position, called null position for the core, at which the output $e_o$ is essentially zero, for the secondary coils are connected in series opposition. As the core moves away from the null position, the mutual inductances of the secondary coils are varying: one increases as the other decreases. As a result, $e_o$ varies (in fact, linearly) with the displacement of the iron core. See Figure 5.8.1

5.8.2 Circuit Analysis

For completeness, a condensed analysis of the LVDT is given here.

\[ R_p, L_p \] are the resistor and inductance of the primary coil, respectively,

\[ R_s, L_s \] are the total resistor and inductance of the secondary coils,

\[ M_1, M_2 \] are the two mutual inductances between the secondary coils and the primary coil,

\[ R_m \] is the load,
$i_p, i_s$ are the currents in the primary coil and the secondary coils

e_{ex}$ is the active sinusoidal voltage source,

e_o$ is the output voltage, which is varying as the core moves in and out.

Applying Kirchhoff's voltage law, it is obtained from Figure 5.8.2 that

$$\begin{cases} 
i_p R_p + L_p \frac{di_p}{dt} + (M_1 - M_2) \frac{di_s}{dt} = e_{ex} \\
i_s (R_s + R_m) + L_s \frac{di_s}{dt} + (M_1 - M_2) \frac{di_p}{dt} = 0 \\
-i_s R_m = e_o \end{cases}$$

(5.24)

If let $I_p, I_s, E_{ex}, E_o$ are the Fourier transform of $i_p, i_s, e_{ex}, e_o$ respectively, then Fourier transforming the above equations leads to

$$\begin{cases} 
I_p R_p + L_p j \omega I_p + (M_1 - M_2) j \omega I_s = E_{ex} \\
I_s (R_s + R_m) + L_s j \omega I_s + (M_1 - M_2) j \omega I_p = 0 \\
-I_s R_s = E_o \end{cases}$$

(5.25)

Solving for $E_o$,

$$E_o = \frac{R_m (M_1 - M_2) j \omega \cdot E_{ex}}{[L_p L_s - (M_1 - M_2)^2 \omega^2 + [L_p (R_s + R_m) + L_s R_p] j \omega + (R_s + R_m) R_p]}$$

(5.26)

Because $R_m$ is very large,

$$E_o \approx \frac{(M_1 - M_2) \cdot E_{ex}}{L_p + \frac{R_p}{j \omega}}$$

(5.27)

i.e.

$$e_o \approx \frac{(M_1 - M_2) \cdot e_{ex}}{L_p + \frac{R_p}{j \omega}}$$

(5.28)
5.8.3 Modulation

According to electric circuit theory, \( M_1(t) - M_2(t) \) is proportional to the position of the core, \( x_i(t) \), provided that both take the value 0 simultaneously. So,

\[
M_1(t) - M_2(t) = C_m \cdot x_i(t) \quad (5.29)
\]

where \( C_m \) is a constant, which mainly depends on the winding of coils and the iron core.

From the above equation and \( e_{ex} = A \cos(\omega_c t) \) where \( \omega_c = 2\pi f_c \), then Eqn 5.28 becomes

\[
e_{ex}(t) = C \cdot x_i(t) \cdot \cos(\omega_c t) \quad (5.30)
\]

where \( C = \frac{C_m A}{L_p + j\omega} \).

Eqn 5.30 means that output from LVDT is actually amplitude modulated signal with the displacement of the core \( x_i(t) \) as the modulating signal.

5.8.4 Features of an LVDT and its justification for tool vibration application

The LVDTs used are of type D5/25 from RDP Electronics Ltd, Wolverhampton, England. For a typical LVDT calibrated under temperature 20 °C, with energising supply as 5 V rms, 5 kHz,

- **linear range** is ±635 \( \mu \)m, which is large enough for machine tool vibration whose amplitude is around 100 \( \mu \)m.
- **linearity** is 0.27%.
- **sensitivity** is 65.66 mV/V/mm.

If the LVDT is energised by a sinusoidal voltage of frequency as 15 kHz, this frequency will be large enough for the cutting tool vibration whose frequency is of 0.5 to 5 kHz. In the application, the LVDT can measure the vibration amplitudes
with an absolute error of ±0.25μm and measure the vibration frequencies with an absolute error of ±7 Hz.

### 5.9 Demodulation

As shown in the last section, the output from the LVDT is a modulated electrical signal:

\[
g(t) = f(t) \cos(\omega_c t)
\]

(5.31)

where \( f(t) = Cx_i(t) \). Let \( G(\omega) \) and \( F(\omega) \) be the FTs of \( g(t) \) and \( f(t) \) respectively, then

\[
G(\omega) = \frac{1}{2} F(\omega + \omega_c) + \frac{1}{2} F(\omega - \omega_c)
\]

(5.32)

which means that \( g(t) \) is simply a translation of the frequency spectrum of \( f(t) \) by ±\( \omega_c \) without changing the shape.

In order to make use of \( g(t) \), \( f(t) \) has to be extracted from it. This is achieved by a demodulation technique in communication theory. It is done by using \( g(t) \) to modulate \( \cos(\omega_c t) \) again, then passing through a low pass filter.

Let

\[
h(t) = g(t) \cos(\omega_c t)
\]

(5.33)

then

\[
H(\omega) = \frac{1}{2} F(\omega) + \frac{1}{4} F(\omega - 2\omega_c) + \frac{1}{4} F(\omega + 2\omega_c)
\]

where \( H(\omega) \) is the FT of \( h(t) \). This means that \( h(t) \) has two components, one is \( f(t) \) (low frequency), the other is \( \frac{1}{4} (f(t)e^{j2\omega_c} + f(t)e^{-j2\omega_c}) \) (high frequency). Therefore, \( f(t) \) is obtained by passing \( h(t) \) through a low-pass filter. NOTE: during the above discussion, \( f(t) \) must vary slowly with respect to \( \omega_c \).

Instead of the demodulation using the same waveform carrier as the original carrier, any other periodic signal can be used as carrier provided that its period is \( \omega_c \). Let \( \sum_{n=-\infty}^{\infty} c_n e^{-j\omega_c n x} \) be any carrier, then its FT can be written as
The signal from the demodulator normally has high frequency ripples. In order to reduce these ripples, a low-pass filter is used. A typical configuration is shown in shown Figure 5.9

\[ \sum_{n=-\infty}^{\infty} c_n \delta(\omega - \omega_c n), \)

so

\[ H(\omega) = \int_{-\infty}^{\infty} G(\omega - \varpi) \sum_{n=-\infty}^{\infty} c_n (\varpi - \omega_c n) d\varpi \]

\[ = \sum_{n=-\infty}^{\infty} c_n G(\omega - \omega_c n) \]

\[ = \sum_{n=-\infty}^{\infty} c'_n F(\omega - \omega_c n) \]

After passing a low-pass filter, \( f(t) \) can be extracted.

In practice, in the instrument, the sinusoidal signal is used in the LVDT while the square wave is used during demodulation.

In the equipment, the sinusoidal signal is generated from the oscillator, so is the square wave. The demodulation and lowpass filtering is done by the demodulator. Both oscillators and demodulators are purchased from RS (Radio Spares).

5.10 A low-pass filter

The signal from the demodulator normally has high frequency ripples. In order to reduce these ripples, a low-pass filter is used. A typical configuration is shown in shown Figure 5.9
This consists of two second-order low-pass filters which are of the type in­
vented by Sallen and Key (Hilburn and Johnson 1973).

For the first part, we have

\[ V_b = V \frac{R_3}{R_3 + R_4} \]

\[ V_a = V_b (1 + j \omega C_1 R_2) \]

\[ \frac{V_1 - V_a}{R_1} = (V_a - V) j \omega C + V_b j \omega C_1 \]

from which

\[ \frac{V}{V_1} = \frac{R_3 + R_4}{-C C_1 R_1 R_2 R_3 \omega^2 + C_1 R_1 R_3 j \omega + C_1 R_2 R_3 j \omega - C R_1 R_4 j \omega + R_3} \]

\[ (5.34) \]

i.e.

\[ \frac{V(\omega)}{V_1(\omega)} = \frac{K_1}{-\omega^2 + j a_1 \omega + b_1} \]

\[ (5.35) \]

where \( K_1 = \frac{R_3 + R_4}{R_1 R_2 R_3 C C_1} \), \( a_1 = \frac{1}{R_1 C} + \frac{1}{R_2 R_3 C C_1} \), and \( b_1 = \frac{1}{R_1 R_2 C C_1} \).

For the second part,

\[ \frac{V_2(\omega)}{V(\omega)} = \frac{K_2}{-\omega^2 + j a_2 \omega + b_1} \]

\[ (5.36) \]

where \( K_2 = \frac{R_7 + R_8}{R_5 R_6 R_7 C C_1} \), \( a_2 = \frac{1}{R_5 C} + \frac{1}{R_6 R_7 C C_1} \), and \( b_2 = \frac{1}{R_5 R_6 C C_1} \).

Combining the last two equations, and choosing suitable values for the resisters
and capacitances, a Butterworth filter is then obtained

\[ \frac{V_2(\omega)}{V_1(\omega)} = \frac{K}{\sqrt{1 + \left( \frac{\omega}{\omega_c} \right)^4}} \]

\[ (5.37) \]

where \( \omega_c \) is the cut-off frequency. Even though its cut-off characteristics are
not sharp compared to other types of filters, such as the Chebychev filter, the
Butterworth filter has a maximally flat, monotonic frequency response in the
passband, which is very important for cutting tool vibration. Figure 5.10 shows
the frequency response for a typical filter.

After this 4th-Order low-pass filter, the high frequency ripples are greatly
reduced. Therefore there is no need to use higher order low-pass filters.
5.11 Data display and further processing

The two outputs from the low-pass filters were then displayed on the oscilloscope. Here, a Digital Storage Oscilloscope 1421 from Gould was used to monitor vibration signals in real-time, to record them, and later on to replay them.

The recorded data from the Gould 1421 was also sent to the MINC (Modular
INstrument Computer). Despite its many disadvantage (mainly the memory), the MINC was easy to interface. Here, the MINC was used to store the data on a disk for future use and compute amplitudes and frequencies of cutting tool vibration. These were obtained through spectrum analysis, including the following stages:

1. the signal was weighted by a Hanning window, so that leakage was reduced;
2. its power spectrum was then computed by the FFT (For the program, see Appendix A);
3. Amplitudes and frequencies were then obtained by finding out the peak of the power spectrum.

The program for carrying out these calculations was written in BASIC for this was the only available language on the MINC.

5.12 Experiments

5.12.1 Equipments

In the experiments, the following equipment was used:

- the machine was a Hardinge CNC Lathe, Model HNC;
- all the cutting tools were from Sandvik Coromant,
  
  3 \( S10K - STFCR09 \)
  
  1 \( S12M - STFCR09 \)
  
  1 \( S12M - STFCR11 \)

- The inserts were from Sandvik Coroman, TCMT090208UR Grad H13A;
The material of workpiece was EN24T which contains (in percentage) the following elements,

\[0.36 \sim 0.44 \quad 0.10 \sim 0.35 \quad 0.45 \sim 0.77 \quad 1.30 \sim 1.70 \quad 1.00 \sim 1.40 \quad 0.20 \sim 0.35\]

Carbon  Silicon  Manganese  Nickel  Chromium  Molybdenum

### 5.12.2 Set up

The two LVDTs were located at the end of a tutting tool as shown in Figure 5.12 by the two thick lines on the cutting tool.

![Figure 5.12: The two LVDTs for measuring the tool vibration.](image)
5.12 Experiments

Photo 1: The two LVDTs and the lathe.

Photo 2: Two sets of modulation and demodulation and filtering.
Photo 3: Gould 1421 for displaying vibration signals.

Photo 4: The MINC.
5.12.3 Results

Table 5.1 and Table 5.2 are two typical sets of values measured. The cutting conditions were: feed = 200 μm, RPM = 935; overhang = 6.5 cm; radius of workpiece = 32 cm, radius of a insert tip = 800 μm.

The positions where horizontal and vertical vibration were measured were 0.5 cm and 1.5 cm respectively away from the cutting tool tip. For $a_y$ and $a_z$, the absolute error is less than ±0.25 μm; for $f_y$ and $f_z$, the absolute error is less than ±7 Hz. Note the measured vibration amplitudes have been corrected. This correction is based on

$$y(x) = \frac{P}{EI} \left( \frac{x^2L}{2} - \frac{x^3}{6} \right)$$

where $y(x)$ is the deflection along the cutting tool $x$, $L$ is the overhang, $P$ is the point force at the end of the cutting tool, $I$ is the the moment of inertia, and $E$ is Young’s Modulus.

Table 5.1: Group 1

<table>
<thead>
<tr>
<th></th>
<th>horizontal</th>
<th>vertical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>amplitude</td>
<td>frequency</td>
</tr>
<tr>
<td></td>
<td>μm</td>
<td>kHz</td>
</tr>
<tr>
<td>1</td>
<td>12.5</td>
<td>1.7</td>
</tr>
<tr>
<td>2</td>
<td>11.9</td>
<td>1.7</td>
</tr>
<tr>
<td>3</td>
<td>13.1</td>
<td>1.7</td>
</tr>
<tr>
<td>4</td>
<td>11.5</td>
<td>1.7</td>
</tr>
</tbody>
</table>

Table 5.2: Group 2

<table>
<thead>
<tr>
<th></th>
<th>horizontal</th>
<th>vertical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>amplitude</td>
<td>frequency</td>
</tr>
<tr>
<td></td>
<td>μm</td>
<td>kHz</td>
</tr>
<tr>
<td>1</td>
<td>9.4</td>
<td>1.7</td>
</tr>
<tr>
<td>2</td>
<td>7.0</td>
<td>1.7</td>
</tr>
<tr>
<td>3</td>
<td>10.8</td>
<td>1.7</td>
</tr>
<tr>
<td>4</td>
<td>9.3</td>
<td>1.7</td>
</tr>
</tbody>
</table>
5.13 Discussion

Because the vibration was measured away from the tool tip, it is different from the vibration at the tool tip. The measured amplitudes are much smaller compared with those at the tool tip although the measured frequencies are the same as those at the tool tip.

On one hand, it is difficult to measure the vibration of a cutting tool tip directly. On the other hand, analysing the surface texture by the WD can reveal the direct effect of tool vibration, which makes this technique more attractive. This will be discussed in detail in the next chapter and the extracted results will be compared with those in Table 5.1 and Table 5.2.
Chapter 6

Extraction of Cutting Tool Vibration by WD Analysis of Surface Texture

6.1 Introduction

The vibration of a cutting tool can be measured by using instruments such as the one discussed in the previous chapter, then used to monitor the cutting process. However, there are some drawbacks about this technique:

1. it needs specific and expensive equipments, such as accelerometers, amplifiers, and so on;

2. it requires mounting and dismounting equipments.

To overcome these problems, a new approach developed by the author is described here. This is to analyse the surface texture of a turned workpiece by the WD, and then to extract cutting tool vibration information, which can be used for machine tool monitoring.

Although the surface texture contains much information about the cutting process and its malfunction, it is not true that all the surface texture in every direction can be used. For instance, the surface profile along the axis of a cylinder
is not suitable. This is because the vibration in this direction (order of 10 \( \mu m \)) is dwarfed by the feed (order of 100 \( \mu m \)). Therefore it is necessary to choose the surface texture in the most sensitive direction. Here, the out of roundness is used.

The roundness of a workpiece is derived through a mathematical model for the tool vibration, with the conclusion that the roundness is just a typical nonstationary signal — a frequency-modulated signal which is amenable to be analysed by the WD. Next the roundness from a real workpiece is measured, processed, and analysed by the WD. Then through the local moments of the WD, the vibration data for the cutting process is extracted, which can be used to control cutting processes.

In addition to this practical application, a computer simulation is presented in Section 9 (Zheng and Whitehouse 1992).

6.2 A mathematical model for cutting tool vibration

In this section, the roundness of a workpiece turned in the presence of the vibration is investigated. The variables used in this section are briefly explained below.

\( f_m \) feed, i.e., the displacement of the tool relative to the workpiece, in the direction of feed motion, per revolution of the workpiece.

\( n_w \) rotational frequency of the workpiece.

\( r_w \) the radius of a workpiece.

\( r_c \) corner radius, i.e., the radius of a rounded tool corner.

\( a_y, f_y \) the amplitude and frequency of the cutting tool in vertical direction.

\( a_z, f_z \) the amplitude and frequency of the cutting tool in horizontal direction.
6.2 A mathematical model for cutting tool vibration

$r(\phi)$ the deviation of the actual radius of the workpiece from the ideal radius.

$\phi$ the angle about the workpiece axis of symmetry.

![Diagram of the fingerprint of turned workpiece in the presence of vibration.](image)

From Figure 6.1, it follows that

$$r(\phi) = r_\epsilon - \sqrt{r_\epsilon^2 - z^2(\phi)} = r_\epsilon \left(1 - \sqrt{1 - \left(\frac{z(\phi)}{r_\epsilon}\right)^2}\right) \quad (6.1)$$

where $-\pi \leq \phi < \pi$. Since $\left(\frac{z(\phi)}{r_\epsilon}\right)^2$ is relatively small for a typical situation, $z(\phi) = 100 \ \mu m$, then $r_\epsilon = 800 \ \mu m$, $\left(\frac{z(\phi)}{r_\epsilon}\right)^2 = \frac{1}{64}$,

$$r(\phi) \approx r_\epsilon \left(1 - \left(1 - \frac{1}{2} \left(\frac{z(\phi)}{r_\epsilon}\right)^2\right)\right) \quad (6.2)$$

i.e.

$$r(\phi) \approx \frac{1}{2r_\epsilon} \cdot z^2(\phi) \quad (6.3)$$
6.2 A mathematical model for cutting tool vibration

6.2.1 Without any vibration

Since the position of the cutting tool is uniquely determined by the feed rate.

\[ z(\phi) = -\frac{f_m \phi}{2\pi} \quad (6.4) \]

so

\[ r(\phi) = \frac{1}{2r_c} \left( \frac{f_m \phi}{2\pi} \right)^2 = \frac{f_m^2 \phi^2}{8\pi^2 r_c} \quad (6.5) \]

6.2.2 With z vibration only

An additional cosine term is added to Eqn 6.4 to represent the vibration,

\[ z(\phi) = -\left( a_z \cos \left( \frac{f_z \phi}{n_w} + \phi_z \right) + \frac{f_m \phi}{2\pi} \right) \quad (6.6) \]

so

\[ r(\phi) = \frac{1}{2r_c} \left( a_z \cos \left( \frac{f_z \phi}{n_w} + \phi_z \right) + \frac{f_m \phi}{2\pi} \right)^2 \]

\[ = \frac{a_z^2}{2r_c} \cos^2 \left( \frac{f_z \phi}{n_w} + \phi_z \right) + a_z f_m \phi \cos \left( \frac{f_z \phi}{n_w} + \phi_z \right) + \frac{f_m^2 \phi^2}{8\pi^2 r_c} \quad (6.7) \]

6.2.3 With z and y vibrations

From the point of view of theoretical mechanics, the situation with the cutting tool vibrating in both the z and y directions can be treated to be equivalent to the one with the cutting tool vibrating in the z direction and the workpiece (combined with the spindle) vibrating rotationally. So using Eqn 6.6 and allowing \( \phi \) to vibrate,

\[ z(\phi) = -\left( a_z \cos \left( \frac{f_z (\phi + \frac{a_x}{r_w} \sin (\frac{f_y \phi}{n_w} + \phi_y))}{n_w} + \phi_z \right) + \frac{f_m (\phi + \frac{a_y}{r_w} \sin (\frac{f_y \phi}{n_w} + \phi_y))}{2\pi} \right) \]

\[ = -\left( a_z \cos \left( \frac{f_z \phi}{n_w} + \frac{a_x}{r_w} \sin (\frac{f_y \phi}{n_w} + \phi_y) \right) + \phi_z \right) \]
6.3 Roundness measurement

\[ + \frac{f_m \phi}{2\pi} + \frac{f_m a_y}{2\pi r_w} \sin \left( \frac{f_y \phi}{n_w} + \phi_y \right) \]  \hspace{1cm} (6.8)

Since \( \frac{f_m a_y}{2\pi r_w} \) is small compared with \( a_z \), for a typical case, \( a_y = 60 \ \mu m, a_z = 10 \ \mu m, r_w = 30 \ mm = 30000 \ \mu m, f_m = 200 \ \mu m, \)

\[ \frac{f_m a_y}{2\pi r_w a_z} = \frac{200 \cdot 60}{2\pi \cdot 30000 \cdot 10} = \frac{1}{50\pi} \]

so

\[ z(\phi) \approx a_z \cos \left( \frac{f_x \phi}{n_w} + \frac{f_x a_y}{n_w r_w} \sin \left( \frac{f_y \phi}{n_w} + \phi_y \right) + \phi_z \right) + \frac{f_m \phi}{2\pi} \]

therefore

\[ r(\phi) = \frac{a_z^2}{2r_e} \cos^2 \left( \frac{f_x \phi}{n_w} + \frac{f_x a_y}{n_w r_w} \sin \left( \frac{f_y \phi}{n_w} + \phi_y \right) + \phi_z \right) \]

\[ + \frac{a_z f_m \phi}{2\pi r_e} \cos \left( \frac{f_x \phi}{n_w} + \frac{f_x a_y}{n_w r_w} \sin \left( \frac{f_y \phi}{n_w} + \phi_y \right) + \phi_z \right) + \frac{f_m^2 \phi^2}{8\pi^2 r_e} \]  \hspace{1cm} (6.9)

which will be discussed in the next two sections.

6.2.4 Vibrations in the x directions

The above equation does not include vibrations directly in x. Because of the large stiffness in this direction, the vibration in x is very small compared to those in y and z directions.

6.3 Roundness measurement

The roundness of a workpiece is measured by Talyrond 200 manufactured by Rank Taylor Hobson. See Figure 6.2.

A Talyrond 200 consists of the following three parts:

1. The turntable, on which the workpiece is placed. The turntable can be rotated manually as well as automatically.
Figure 6.2: Talyrond 200
2. The pick-up. As the workpiece rotates with the turntable, the pick-up will move radially in response to workpiece irregularities.

3. The processing unit. Radial motion is converted into an electrical signal, which is amplified, filtered, plotted and transferred to the MINC.

In order to achieve accuracy, attention must be paid to the following important points:

1. The base of the workpiece should be ground smooth so that it is stable during measurement.

2. The stylus arm should be placed so that stylus displacements are normal to the surface.

3. The filter should be set to 1-500 u.p.r.\(^1\) so that all undulations within the instrument bandwidth are recorded.

4. The magnification should be set as high as possible.

During the measurement, the magnification was set at 5000 or 10000, the filter was 1-500 u.p.r. For each revolution, \(L = 2048\) points are sampled. The sampled data were plotted out and also transferred to the MINC computer.

\(^1\)u.p.r. stands for undulations per revolution
6.3 Roundness measurement

Photo 5 and 6: The measurement of roundness.
Here is one of the typical plot of roundness for a turned workpiece.

![Figure 6.3: A typical roundness graph.](image)

After measurement, the roundness takes the following form

\[
 r[n] = \frac{a_z}{2r_\varepsilon} \cos^2 \left( \frac{f_z}{n_w} \phi[n] + \frac{f_z a_y}{n_w r_w} \sin \left( \frac{f_y}{n_w} \phi[n] + \phi_y \right) + \phi_z \right) \\
+ \frac{f_m a_z}{2\pi r_\varepsilon} \phi[n] \cos \left( \frac{f_z}{n_w} \phi[n] + \frac{f_z a_y}{n_w r_w} \sin \left( \frac{f_y}{n_w} \phi[n] + \phi_y \right) + \phi_z \right) \\
+ \frac{f_m^2}{8\pi^2 r_\varepsilon} (\phi[n])^2
\]

where \( \phi[n] = \frac{2\pi n}{L} \) and \(-\frac{L}{2} \leq n < \frac{L}{2}\).
6.4 Band pass filtering

Before applying the WD to analyse the roundness, there are a few preliminary things to do, in order to remove aspects of the signal that are unconcerned with the information concerning vibration, and to transform it into an analytical signal for the ease of computation. These include

1. choosing one section of the roundness for analysis, and if necessary more than one,
2. preprocessing the roundness data,
3. and transforming the roundness signal into its analytical form by means of the Hilbert transform.

Per revolution, there are $L = 2048$ sampled data points. In order to reduce computation time, one section of a revolution is chosen. It is found out that $\frac{1}{8}$ of a revolution is large enough for extraction of vibration information. The part of revolution is chosen as

$$f[n] = \frac{f_m^2}{8r_c} + \frac{a_z^2}{2r_c} \cos^2 \left( \frac{f_z}{n_w} \phi[n] + \phi_z + \frac{f_z a_y}{n_w r_w} \sin \left( \frac{f_y}{n_w} \phi[n] + \phi_y \right) \right)$$
$$+ \frac{f_m a_z}{2r_c} \cos \left( \frac{f_z}{n_w} \phi[n] + \phi_z + \frac{f_z a_y}{n_w r_w} \sin \left( \frac{f_y}{n_w} \phi[n] + \phi_y \right) \right)$$

(6.10)

where $\phi[n] = \frac{2\pi(L-N+n)}{L}$, $n = 0, 1, 2, \ldots, N - 1$, and $N = 256$.

Let

$$a = \frac{f_m a_z}{2r_c}$$
$$b = \frac{f_z a_y}{n_w r_w}$$

(6.11)
6.4 Band pass filtering

\[ k_o = \frac{f_z N}{n_w L} \]  
\[ k_m = \frac{f_y N}{n_w L} \]  
\[ \varphi_y = \frac{f_y \pi L - N}{n_w L} + \phi_y \]  
\[ \varphi_z = \frac{f_z \pi L - N}{n_w L} + \phi_z \]

From Eqn 6.17, it can be seen that the roundness of workpiece turned with vibration consists of 3 parts:

1. \( \frac{f_m^2}{8r_e} \)

2. \( \frac{a_z^2}{2r_e} \cos^2 \left( \frac{2\pi k_o n}{N} + \varphi_z + b \sin \left( \frac{2\pi k_m n}{N} + \varphi_y \right) \right) \)

3. \( a \cos \left( \frac{2\pi k_o n}{N} + \varphi_z + b \sin \left( \frac{2\pi k_m n}{N} + \varphi_y \right) \right) \)

Since the first part \( \frac{f_m^2}{8r_e} \) contains nothing about the vibration condition and is also slowly varying, therefore it can removed by using a high pass filter as shown in Figure 6.4.

Now, \( \frac{a_z^2}{2r_e} \) is relatively small compared with \( a = \frac{f_m a_z}{2r_e} \), for example, let \( a_z = 10 \mu m \) and \( f_m = 200 \mu m \), then

\[ \frac{\frac{a_z^2}{2r_e}}{\frac{f_m a_z}{2r_e}} = \frac{a_z}{f_m} = 0.05 \]
Therefore it can be concluded that the dominant part in the roundness of the workpiece after the high pass filtering is

\[ f[n] = a \cos\left(\frac{2\pi k_0 n}{N} + \varphi_z + b \sin\left(\frac{2\pi k_m n}{N} + \varphi_y\right)\right) \]

which is a FM signal. As explained in Chapter 3 and Chapter 4, this type of signal is amenable to be analysed by the WD.

However before analysing the roundness by the WD, it is necessary to convert it into its analytical form in order to remove the redundant spectrum in negative frequencies. This can be done by the Hilbert transform whose implementation requires a filter like the one shown in Figure 6.5.

The roundness may also contain high frequency noise, for example, from the instrument electronic circuits. To reduce this noise and (or) to emphasize the term of FM, a lowpass pass filter can be used, See Fig 6.6.

The above three linear operations can be combined into a single bandpass filter such as shown in Figure 6.7.
6.4 Band pass filtering

Figure 6.5: The filter $H[k]$ for converting a real signal to its analytical signal.

Figure 6.6: The low pass filter $H[k]$ for removing high frequency noises.

Figure 6.7: The combined bandpass filter $H[k]$. 
After this, the signal $f[n]$ would mainly contain

$$ae^{j\left(\frac{2\pi kn}{N} + \phi + \frac{\pi}{2} \sin\left(\frac{2\pi kn}{N} + \phi\right)\right)}$$  \hspace{1cm} (6.18)

### 6.5 The WD and its local moments

#### 6.5.1 The WD

Since the roundness after the operations discussed in the previous section, $f[n]$, is restricted to $[0, N - 1]$ (where $N = 256$), it is easy to compute its WD $W_f(n, \theta)$

$$W_f(n, \theta) = \sum_{m=-\infty}^{\infty} 2f[n + m]f^*[n - m]e^{-j2m\theta}$$

Because

$$\begin{cases} 
0 \leq n + m \leq N - 1 \\
0 \leq n - m \leq N - 1
\end{cases}$$

then

$$\begin{cases} 
0 \leq n + m \leq N - 1 \\
-(N - 1) \leq -n + m \leq 0 \\
-(N - 1) \leq 2m \leq N - 1 \\
-(\frac{N}{2} - 1) \leq m \leq \frac{N}{2} - 1 \\
-M \leq m \leq M
\end{cases}$$

where $M = \frac{N}{2} - 1$. Therefore

$$W_f(n, \theta) = \sum_{m=-M}^{M} 2f[n + m]f^*[n - m]e^{-j2m\theta}$$

$$= \sum_{m=0}^{N-1} 2f[n + m - M]f^*[n - m + M]e^{-j2(m-M)\theta}$$
Let $\theta = \frac{k\pi}{N}$, then

$$W_f(n, \frac{k\pi}{N}) = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} 2f[n + m - M] f^*[n - m + M] e^{-j2(m-M)\frac{mk\pi}{N}}$$

Therefore, to compute the WD is only necessary to compute

1. $2f[n + m - M] f^*[n - m + M]$ where $n, m = 0, 1, 2, \ldots, N - 1$.

2. $\frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} (2f[n + m - M] f^*[n - m + M]) e^{-j2m\frac{mk\pi}{N}}$ where $n, k = 0, 1, 2, \ldots, N - 1$

3. $W_f(n, \frac{k\pi}{N}) = \left(\frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} (2f[n + m - M] f^*[n - m + M])\right) \cdot \sqrt{N} e^{j2\frac{mk\pi}{N}}$ for $n, k = 0, 1, 2, \ldots, N - 1$.

### 6.5.2 The local moments in frequency

The 0th-order moment in frequency $p_f[n]$ is computed

$$p_f[n] = \frac{1}{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} W_f(n, \theta) d\theta$$

$$\approx \frac{1}{2N} \sum_{0}^{N-1} W_f(n, \frac{k\pi}{N})$$

The 1st-order moment in frequency $\Theta_f[n]$: $\quad$ 

$$\Theta_f[n] = \frac{1}{2} \arg \left(\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} e^{i2\theta} W_f(n, \theta) d\theta\right)$$

$$\approx \frac{1}{2} \arg \left(\sum_{0}^{N-1} e^{i2\frac{2k\pi}{N}} W_f(n, \frac{k\pi}{N})\right)$$
6.6 Extraction of vibration information

Since

\[ f[n] = ae^{i\left(\frac{2\pi kn}{N} + \phi_z + \frac{1}{2} \sin\left(\frac{2\pi km}{N} + \phi_y\right)\right)} \]  

(6.19)

where \( a = \frac{f_m a_z}{2r_e} \), \( b = \frac{f_x a_y}{n_w r_w} \), \( k_o = \frac{f_x N}{n_w L} \), and \( k_m = \frac{f_y N}{n_w L} \). So

\[ p_f(n) = a^2 \]  

(6.20)

\[ \Theta_f(n) = \frac{2\pi k_o}{N} + \frac{b}{2} \sin\left(\frac{2\pi k_m}{N}\right) \cos\left(\frac{2\pi k_m n}{N} + \phi_y\right) \]  

(6.21)

6.6.1 \( a_z \) and \( f_z \)

If we let \( \overline{p_f} \) and \( \overline{\Theta_f} \) be the mean value for \( p_f[n] \) and \( \Theta_f[n] \) respectively, i.e.

\[ \overline{p_f} = \frac{1}{N} \sum_{n=0}^{N-1} p_f[n] \]

\[ \overline{\Theta_f} = \frac{1}{N} \sum_{n=0}^{N-1} \Theta_f[n] \]

it follows that

\[ \overline{p_f} = a^2 = \left(\frac{f_m a_z}{2r_e}\right)^2 \]

\[ \overline{\Theta_f} = \frac{2\pi k_o}{N} = \frac{2\pi f_z}{n_w L} \]

from which

\[ a_z = \frac{2r_e \sqrt{\overline{p_f}}}{f_m} \]

\[ f_z = \frac{n_w L \overline{\Theta_f}}{2\pi} \]
6.6.2 \( a_y \) and \( f_y \)

By finding the local maximum of the DFT of \( \Theta_f[n] \) it is easy to obtain the values for \( \frac{b}{2} \sin \left( \frac{2\pi km}{N} \right) \) and \( k_m \). So \( a_y \) and \( f_y \) can be obtained through:

\[
\begin{align*}
  a_y &= \frac{n_u r_u b}{f_z} \\
  f_y &= \frac{n_u L k_m}{N}
\end{align*}
\]
6.7 An example of extracting vibration data from roundness

Figure 6.7 and Figure 6.9 present a complete example of extracting vibration data from a measured roundness.

Figure 6.8: The measured roundness and the section chosen for extraction vibration data.
Figure 6.9: Measured: \( a_z = 8.3, f_z = 3600.0, a_y = 22.6, f_y = 3600.0 \). Extracted: \( a_z = 10.8, f_z = 3615.3, a_y = 34.7, f_y = 3615.3 \).
6.8 Results

Here are three cases of extracting vibration parameters from roundness data via the WD. As for the measured $a_y$ and $a_z$, the absolute error is about $\pm 0.25\mu m$; while for the measured $f_y$ and $f_z$, the absolute error is about $\pm 7\ Hz$. Note that for $y$ direction the vibration was measured 1.5 cm away from the tip; for $z$ direction it was 0.5 cm away from the tip. Therefore the measured vibration amplitudes have been corrected, see Section 5.12.3.

Case 1 With the cutting condition as

cutting tool: S10K-STFCR09

overhang = 45 mm,

depth of cut = 200 $\mu m$, feed = 100 $\mu m$, rpm = 935,

radius of workpiece = 32 mm, radius of tool tip = 800 $\mu m$,

the results are listed in Table 6.1.

<table>
<thead>
<tr>
<th></th>
<th>$a_z$ $\mu m$</th>
<th>$f_z$ kHz</th>
<th>$a_y$ $\mu m$</th>
<th>$f_y$ kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>measured</td>
<td>6.6</td>
<td>3.60</td>
<td>11.4</td>
<td>3.60</td>
</tr>
<tr>
<td>extracted</td>
<td>10.2</td>
<td>3.49</td>
<td>31.1</td>
<td>3.49</td>
</tr>
<tr>
<td>extracted</td>
<td>10.7</td>
<td>3.62</td>
<td>35.9</td>
<td>3.49</td>
</tr>
<tr>
<td>extracted</td>
<td>11.0</td>
<td>3.49</td>
<td>28.0</td>
<td>3.49</td>
</tr>
<tr>
<td>extracted, measured, in average</td>
<td>1.60</td>
<td>0.98</td>
<td>2.80</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Case 2 With the cutting condition as

cutting tool: S10K-STFCR09
6.8 Results

overhang = 65 mm,

depth of cut = 50 \mu m, feed = 100 \mu m, rpm = 935,

radius of workpiece = 32 mm, radius of tool tip = 800 \mu m,

the results are listed in Table 6.2.

<table>
<thead>
<tr>
<th></th>
<th>(a_z)</th>
<th>(f_z)</th>
<th>(a_y)</th>
<th>(f_y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>measured</td>
<td>10.2</td>
<td>1.60</td>
<td>72.2</td>
<td>1.60</td>
</tr>
<tr>
<td>extracted</td>
<td>26.2</td>
<td>1.62</td>
<td>234.8</td>
<td>1.62</td>
</tr>
<tr>
<td>extracted</td>
<td>24.0</td>
<td>1.62</td>
<td>235.0</td>
<td>1.62</td>
</tr>
<tr>
<td>extracted</td>
<td>30.7</td>
<td>1.62</td>
<td>239.4</td>
<td>1.62</td>
</tr>
</tbody>
</table>

For Table 6.2, measured and extracted are the means.

From Table 6.1 to Table 6.3, a number of points emerge. One concerns the validity of the cantilever model for the cutting tool vibration. This says that the natural frequency even during cutting is given by \(f_1\)

\[
f_1 = \frac{k_1^2}{2\pi} \sqrt{\frac{Eg}{bh^3}} \frac{1}{A\sigma 12}
\]
Table 6.3: Experimental results: case 3

<table>
<thead>
<tr>
<th></th>
<th>$a_x$ $\mu m$</th>
<th>$f_x$ kHz</th>
<th>$a_y$ $\mu m$</th>
<th>$f_y$ kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>measured</td>
<td>9.4</td>
<td>1.70</td>
<td>39.8</td>
<td>1.70</td>
</tr>
<tr>
<td>extracted</td>
<td>39.7</td>
<td>1.75</td>
<td>153.7</td>
<td>1.75</td>
</tr>
<tr>
<td>measured</td>
<td>4.23</td>
<td>1.03</td>
<td>3.86</td>
<td>1.03</td>
</tr>
<tr>
<td>measured</td>
<td>7.0</td>
<td>1.70</td>
<td>38.6</td>
<td>1.70</td>
</tr>
<tr>
<td>extracted</td>
<td>33.5</td>
<td>1.75</td>
<td>130.1</td>
<td>1.75</td>
</tr>
<tr>
<td>extracted</td>
<td>4.8</td>
<td>1.03</td>
<td>3.37</td>
<td>1.03</td>
</tr>
<tr>
<td>measured</td>
<td>9.3</td>
<td>1.70</td>
<td>38.6</td>
<td>1.70</td>
</tr>
<tr>
<td>extracted</td>
<td>26.2</td>
<td>1.87</td>
<td>174.7</td>
<td>1.87</td>
</tr>
<tr>
<td>extracted</td>
<td>2.83</td>
<td>1.10</td>
<td>4.52</td>
<td>1.10</td>
</tr>
<tr>
<td>measured</td>
<td></td>
<td></td>
<td>3.95</td>
<td>1.05</td>
</tr>
</tbody>
</table>

$\text{extracted in average}$ $3.95$ $1.05$ $3.92$ $1.05$

where $k_1 = \frac{1.875}{l}$. For the Sandvic cutting used, $E = 2.0 \times 10^7$ N/cm$^2$, $g = 980$ cm/s$^2$, $\sigma = 0.0764$ N/cm$^2$, $b = h = 1$ cm, $A = 1$ cm$^2$, and $l$, the effective overhang of the tool which in the case of this experiment took two values 4.5 cm and 6.5 cm,

$$f_1 = \begin{cases} 
3.9\text{kHz} & l = 4.5\text{cm} \\
1.9\text{kHz} & l = 6.5\text{cm}
\end{cases} \quad (6.22)$$

These are compared well with the measured values of 3.6 kHz and 1.7 kHz, respectively. The small reduction is attributable to the damping of the cutting process. These results show that within the errors of this experience the model is satisfactory and that vibration due to the tool column is not significant when compared with that of the tool.

A comparison between the measured vibration frequencies and those picked out by the Wigner Moment Analysis of the roundness graphs is very encouraging. The relative errors are within a few percent only. This proves that the Wigner distribution can be used as an indirect measure of tool vibration.

That the amplitude of the measured vibration is small compared with those
determined by the WD is predictable. This is due to the fact the LVDT measuring the vibration cannot be located at the tool tip. It therefore measures a smaller movement, and if higher order vibration modes are present, there may be a large ratio factor of the measured amplitude to the tool tip amplitude; however, this ratio should be constant for a given cutting condition. In fact, it is exactly for this reason that the indirect method is a better alternative to detect vibration than the more direct method. Furthermore, it is vibration at the workpiece which is important not the vibration anywhere along the tool shank. Producing the workpiece after all is the reason for using the machine tool.

6.9 Conclusions

It has been demonstrated that by using the Wigner distribution, a potentially useful new tool for detecting cutting tool vibration prior to machining failure has been developed. In this preliminary study turning has been chosen as the process to be monitored because it is more easily understood than, say grinding. Furthermore, it has been possible to measure the vibration directly to verify the method. The practical results so far obtained are limited and more work needs to be carried out in this area both for turning and for other processes, but the results obtained are encouraging.

It also seems plausible that an optical on-line instrument could be developed to monitor the surface without contact. This will be the object of further work in the use of the Wigner distribution for monitoring machine tool conditions.
6.10 Simulation

In this section, the roundness is simulated according to the mathematical model discussed previously. The input for the simulation program is the parameters about tool vibration, they are the amplitudes \((a_z, a_y)\), the frequencies \((f_z, f_y)\), and the initial phases \((\phi_z, \phi_y)\) for both horizontal \((z)\) and vertical \((y)\) directions. The output is the extracted values for \(a_z, a_y, f_z\) and \(f_y\).

For an example, with the RPM 900, the feed 200 \(\mu m\), the radius of the work-piece 32 mm, and the radius of the insert tip 0.8 mm. The results are listed in Table 6.4 and Table 6.5.

<table>
<thead>
<tr>
<th></th>
<th>(a_z)</th>
<th>(f_z)</th>
<th>(a_y)</th>
<th>(f_y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit</td>
<td>(\mu m)</td>
<td>kHz</td>
<td>(\mu m)</td>
<td>kHz</td>
</tr>
<tr>
<td>true</td>
<td>10.0</td>
<td>1.80</td>
<td>100.0</td>
<td>1.80</td>
</tr>
<tr>
<td>extracted</td>
<td>9.9</td>
<td>1.80</td>
<td>112.2</td>
<td>1.80</td>
</tr>
<tr>
<td>error (%)</td>
<td>1.0</td>
<td>0</td>
<td>12.2</td>
<td>0</td>
</tr>
<tr>
<td>true</td>
<td>15.0</td>
<td>1.80</td>
<td>150.0</td>
<td>1.80</td>
</tr>
<tr>
<td>extracted</td>
<td>14.8</td>
<td>1.80</td>
<td>166.1</td>
<td>1.80</td>
</tr>
<tr>
<td>error (%)</td>
<td>0.1</td>
<td>0</td>
<td>10.7</td>
<td>0</td>
</tr>
<tr>
<td>true</td>
<td>20.0</td>
<td>1.80</td>
<td>200.0</td>
<td>1.80</td>
</tr>
<tr>
<td>extracted</td>
<td>19.4</td>
<td>1.80</td>
<td>217.4</td>
<td>1.80</td>
</tr>
<tr>
<td>error (%)</td>
<td>3.0</td>
<td>0</td>
<td>8.7</td>
<td>0</td>
</tr>
<tr>
<td>true</td>
<td>25.0</td>
<td>1.80</td>
<td>250.0</td>
<td>1.80</td>
</tr>
<tr>
<td>extracted</td>
<td>23.7</td>
<td>1.80</td>
<td>265.2</td>
<td>1.80</td>
</tr>
<tr>
<td>error (%)</td>
<td>0.5</td>
<td>0</td>
<td>6.1</td>
<td>0</td>
</tr>
<tr>
<td>true</td>
<td>30.0</td>
<td>1.80</td>
<td>300.0</td>
<td>1.80</td>
</tr>
<tr>
<td>extracted</td>
<td>27.6</td>
<td>1.80</td>
<td>308.7</td>
<td>1.80</td>
</tr>
<tr>
<td>error (%)</td>
<td>3.0</td>
<td>0</td>
<td>2.9</td>
<td>0</td>
</tr>
<tr>
<td>maximum error (%) for all</td>
<td>3.0</td>
<td>0</td>
<td>12.2</td>
<td>0</td>
</tr>
<tr>
<td>average error (%) for all</td>
<td>1.5</td>
<td>0</td>
<td>8.1</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 6.5: Simulation results of varying frequencies

<table>
<thead>
<tr>
<th></th>
<th>$a_x$</th>
<th>$f_z$</th>
<th>$a_y$</th>
<th>$f_y$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>unit</strong></td>
<td>$\mu$m</td>
<td>kHz</td>
<td>$\mu$m</td>
<td>kHz</td>
</tr>
<tr>
<td><strong>true</strong></td>
<td>10.0</td>
<td>1.50</td>
<td>100.0</td>
<td>1.50</td>
</tr>
<tr>
<td><strong>extracted</strong></td>
<td>9.9</td>
<td>1.56</td>
<td>110.3</td>
<td>1.56</td>
</tr>
<tr>
<td><strong>error (%)</strong></td>
<td>1.0</td>
<td>4.0</td>
<td>10.3</td>
<td>4.0</td>
</tr>
<tr>
<td><strong>true</strong></td>
<td>10.0</td>
<td>1.60</td>
<td>100.0</td>
<td>1.60</td>
</tr>
<tr>
<td><strong>extracted</strong></td>
<td>9.9</td>
<td>1.68</td>
<td>114.7</td>
<td>1.56</td>
</tr>
<tr>
<td><strong>error (%)</strong></td>
<td>1.0</td>
<td>5.0</td>
<td>14.7</td>
<td>4.0</td>
</tr>
<tr>
<td><strong>true</strong></td>
<td>10.0</td>
<td>1.70</td>
<td>100.0</td>
<td>1.70</td>
</tr>
<tr>
<td><strong>extracted</strong></td>
<td>9.9</td>
<td>1.68</td>
<td>118.1</td>
<td>1.68</td>
</tr>
<tr>
<td><strong>error (%)</strong></td>
<td>1.0</td>
<td>1.2</td>
<td>18.1</td>
<td>1.2</td>
</tr>
<tr>
<td><strong>true</strong></td>
<td>10.0</td>
<td>1.80</td>
<td>100.0</td>
<td>1.80</td>
</tr>
<tr>
<td><strong>extracted</strong></td>
<td>9.9</td>
<td>1.80</td>
<td>112.2</td>
<td>1.80</td>
</tr>
<tr>
<td><strong>error (%)</strong></td>
<td>1.0</td>
<td>0.0</td>
<td>12.2</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>true</strong></td>
<td>10.0</td>
<td>1.90</td>
<td>100.0</td>
<td>1.90</td>
</tr>
<tr>
<td><strong>extracted</strong></td>
<td>9.9</td>
<td>1.92</td>
<td>111.4</td>
<td>1.92</td>
</tr>
<tr>
<td><strong>error (%)</strong></td>
<td>1.0</td>
<td>1.1</td>
<td>11.4</td>
<td>1.1</td>
</tr>
<tr>
<td><strong>maximum</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>error (%)</strong></td>
<td>1.0</td>
<td>4.0</td>
<td>18.1</td>
<td>4.0</td>
</tr>
<tr>
<td><strong>for all</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>average</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>error (%)</strong></td>
<td>1.0</td>
<td>2.3</td>
<td>13.8</td>
<td>2.1</td>
</tr>
<tr>
<td><strong>for all</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Notice that the frequencies $f_z$ and $f_y$ can be extracted by the WD quite accurately, with the relative error less than 4.0 percent. The error for extraction of $a_y$ is usually less than 15 percent, this is due to the fact that preprocessing the roundness data has not completely removed the low and high frequency parts. If necessary, more refined bandpass filters can be used. Extraction of $a_z$ is successful, with its error less than 3 percent. This is rather encouraging for the deviation from roundness is mainly due to $a_z$ (with the feed) rather than $a_y$. 
Chapter 7

Conclusions

One of the major constituents of computer integrated manufacturing is the field of machine tool capability and machine tool monitoring. This is economically a necessity because of the high cost of failure of the machine tool and also of failure to meet the quality requirements of the parts produced. This need has led to much work being carried out to detect incipient failure due to chatter, tool wear or breakage, slideway and bearing error. In particular, the monitoring of the cutting tool itself is emerging as one of the most important areas because of the increased use of precision manufacturing in nanotechnology and other high technology areas.

Basically there are two methods: direct (such as the actual measurement of tool wear using optical or radioactive methods), and indirect (involving measurement of parameters associated with the cutting process such as cutting forces, vibration, sound, acoustic emission, cutting temperature and surface texture). In this thesis, the surface analysis method is investigated because this serves a dual purpose. The surface geometry has to be controlled and preferably known in order to meet the functional requirements of the workpiece. It seems logical therefore to use the surface geometry not only to predict function but also to control manufacture.

During the early days (Reason 1944), a single number such as $R_a$ was used to characterise the surface and predict the function of the workpiece. Despite
limited success, there were and are two problems associated with this approach. The first is that the change in the surface parameter did not necessarily identify the cause and the second is that this method was usually too slow to have stopped a large number of faulty components being made.

To overcome these problems, the concepts of random process analysis were introduced (Peklenik, 1968). This allowed a much better link between the geometry and the process to be made. By means of the power spectrum and the correlation function many aspects of manufacture could be identified.

However, even random process analysis as described has a conceptual limitation. This is due to the fact that it is presented either spatially (as a correlation function) or as a frequency plot (as the power spectrum). This is a disadvantage in some respects because it precludes the easy assessment of changes in the process. With the power spectrum or correlation function, it is difficult to measure the changes in the power spectrum in space i.e. over the surface of one workpiece. Therefore, what is needed is some function, probably based on Fourier and random process concepts, which is equally defined in the space and frequency domain; one which can characterise the surface geometry in space and frequency simultaneously and also preferably be realised instrumentally. There are at least two possibilities, the ambiguity function (AF) and the Wigner distribution (WD). After comparing these two functions in detail, the conclusion is reached that the Wigner distribution is the most likely to be useful as an analysing tool.

Although it is suitable for characterising both stationary and nonstationary signals, the WD contains too much information, which means data condensation is required. The Hough transform (HT) was tried. It turned out that the HT did not work very well despite its success with chirp signals. Next the method based on local moments in frequency for the WD was tested and found to be successful. This was verified for both simulated and real data (Zheng and Whitehouse 1992).

The technique of cutting tool monitoring by the WD has several advantages:
1. It is straightforward in spite of the fact that the WD is more complicated than the Fourier transform (FT).

2. It does not interfere with cutting processes.

3. It is indirect, but can reveal the direct effect of cutting condition, in particular, tool vibration.

4. Besides being able to extract tool vibration, it can be extended to extract other information such as tool wear and breakage.

5. It is possible to build an in-process integrated machine tool diagnosis system from this technique. This may be achieved by measuring the roundness optically, computing its WD and extracting parameters optically. It can also be realised by on-line measuring the roundness, and using a microcomputer for computation and control.

Despite the above advantages, this technique has its drawbacks:

- It is computationally more expensive than the FT because the WD is a function of two variables

- At present it is not in-process although it can be extended to in-process.

In conclusion, after comparing the FT, the AF, and the WD, it has been demonstrated that the WD is a very useful analysing tool which can be successfully used for cutting tool monitoring by extracting vibration data.

In addition to this application, a number of developments are possible in future. Potential usage of local spatial moments and global moments should be explored more fully. For example the first local spatial moment can be used as an indication of the position of freak behaviour on the surface, and the second local spatial moment can be used as a measure of the size of the freak. This has obvious implications for process control and flaw detection.
An in-process instrument based on the WD can be built. This is because the WD is in fact the Fourier transform of \( f(x + \frac{\pi}{2})f^*(x - \frac{\pi}{2}) \) provided that \( x \) is treated as a fixed parameter. Hence, the WD can be achieved through optical transformation, i.e. in an optical instrument.

Application to grinding should be explored. In principle, the WD will work as well for a statistical signal as for a deterministic signal, so the problem should be tractable. The problem of wear should also be tractable, since it can be divided into a combination of texture analysis; as the microscopic shape of the surface changes during wear, and flaw detection; as for example, the detection of pit formation. The WD also forms a rather complete description of surface texture. It may therefore be possible to relate this quantitatively to frictional properties of surfaces.

Its application is not restricted to surface analysis and indeed a project has recently commenced at Warwick University on its application to X-ray reflectivity data.

Further applications of the WD, such as flaw detection, wear, friction, should be explored.

Even from the limited amount of work carried out in this thesis it could be concluded that the WD provides powerful mathematical framework for application to machine tool condition monitoring.
References


Bulletin of Japan Society of Mechanical Engineers v 24 p 748-755.


De Bruijn, N. G. *Uncertainty principles in Fourier analysis.*


Reis, F. B. 1962. A linear transformation of the ambiguity function plane. IRE Trans. on Information Theory v 8 n 1 p 59.


Appendix A

The Fast Fourier Transform

The fast Fourier transform is an algorithm of efficiently computing

$$F[k] = \sum_{n=0}^{N-1} f[n] e^{\frac{2\pi i kn}{N}}$$  \hspace{1cm} (A.1)

For $N = 2^r$, this is derived here (Brigham 1988).

Let $k$ and $n$ be represented in binary form as

$$k = k_{r-1} 2^{r-1} + k_{r-2} 2^{r-2} + \cdots + k_0$$
$$n = n_{r-1} 2^{r-1} + n_{r-2} 2^{r-2} + \cdots + n_0$$

and $W = e^{\frac{2\pi i}{N}}$, then

$$F[k_{r-1}, k_{r-2}, \cdots, k_0] = \sum_{n_0=0}^{1} \sum_{n_1=0}^{1} \cdots \sum_{n_{r-1}=0}^{1} f[n_{r-1}, n_{r-2}, \cdots, n_0]$$
$$\times W^{k_0 n_{r-1} 2^{r-1}} \times W^{(k_1 2^1 + k_0) n_{r-2} 2^{r-2}}$$
$$\times \cdots$$
$$\times W^{(k_{r-1} 2^{r-1} + k_{r-2} 2^{r-2} + \cdots + k_0) n_0}$$
Performing each of the summations separately and labelling the intermediate results, we obtain

\[ F_0[n_{r-1}, n_{r-2}, \cdots, n_0] = f[n_{r-1}, n_{r-2}, \cdots, n_0] \]

\[ F_1[k_0, n_{r-2}, \cdots, n_0] = \sum_{n_{r-1}=0}^{1} F_0[n_{r-1}, n_{r-2}, \cdots, n_0] W^{k_0 n_{r-1} 2^{r-1}} \]

\[ F_2[k_0, k_1, \cdots, n_0] = \sum_{n_{r-2}=0}^{1} F_1[k_0, n_{r-2}, \cdots, n_0] W^{(k_1 2^{r} + k_0) n_{r-2} 2^{r-2}} \]

\[ \vdots \]

\[ F_r[k_0, k_1, \cdots, k_{r-1}] = \sum_{n_0=0}^{1} F_{r-1}[k_0, k_1, \cdots, n_0] W^{(k_{r-1} 2^{r-1} + k_{r-2} 2^{r-2} + \cdots + k_0) n_0} \]

\[ F[k_{r-1}, k_{r-2}, \cdots, k_0] = F_r[k_0, k_1, \cdots, k_{r-1}] \]

\( N \log_2(N) \) complex multiplications are required to compute Equation A.1 through the above equations indirectly, while \( N^2 \) complex multiplications are required to compute Equation A.1 directly. Therefore, the FFT is very efficient for large \( N \)
### The FFT

The FFT Makefile

<table>
<thead>
<tr>
<th>OBJ = dft.o input.o fft.o store_data.o</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAGS = -g</td>
</tr>
<tr>
<td>EXEC = dft</td>
</tr>
<tr>
<td>$(EXEC): $(OBJ)</td>
</tr>
<tr>
<td>cc $(FLAGS) $(OBJ) -o $(EXEC) -lm</td>
</tr>
<tr>
<td>dft.o: dft.h Makefile</td>
</tr>
<tr>
<td>cc $(FLAGS) -c dft.c -o dft.o</td>
</tr>
<tr>
<td>input.o: dft.h Makefile</td>
</tr>
<tr>
<td>cc $(FLAGS) -c input.c -o input.o</td>
</tr>
<tr>
<td>fft.o: dft.h Makefile</td>
</tr>
<tr>
<td>cc $(FLAGS) -c fft.c -o fft.o</td>
</tr>
<tr>
<td>store_data.o: dft.h Makefile</td>
</tr>
<tr>
<td>cc $(FLAGS) -c store_data.c -o store_data.o</td>
</tr>
</tbody>
</table>

The FFT dft.h

```c
#include <stdio.h>
#include <math.h>
#include <strings.h>

#define N 256
#define NU 8

extern void input();
extern void fft();
extern void store_data();
```
The FFT

dft.c

#include "dft.h"

main()
{
    double f[2][N];
    char type_of_signal[25];
    char file_name[250];

    input(f[0], f[1], type_of_signal);
    strcpy(file_name, "/home/hawk/eng/es036/work/fft/Data/");
    strcat(file_name, ".sig’’
    store_data(f[0], f[1], file_name);

    printf("Doing FFT...
    ");
    fft(f[0], f[1], N, NU);

    store_data(f[0], f[1], file_name);
}

The FFT

input.c

#include "dft.h"

void input(real, imag, type_of_signal)
{
    double real, imag;
    char type_of_signal[25];

    input(arg, int if;

    printf("Choose a signal from:
    ");
    printf("a sinusoidal signal
    ");
    printf("b sum of three sinusoidal signals
    ");
    printf("c chirp signal
    ");
    printf("d frequency-modulated signal
    ");
    scanf("%s", type_of_signal);

    if (type_of_signal[0] == ‘a’)
    {
        arg = 2.0*3.14159265*32.0/N;
        for (i=0; i < N; i++)
        {
            real[i] = cos(arg*1); 
            imag[i] = sin(arg*1);
        }
        strcpy(type_of_signal, "sinu1");
    }
    else if (type_of_signal[0] == ‘b’)
    {
        arg = 2.0*3.14159265/N;
        for (i=0; i < N; i++)
        {
            real[i] = 0.25*cos(arg*16.0*i) + 0.25*cos(arg*80.0*i);
            imag[i] = 0.25*sin(arg*16.0*i) + 0.25*sin(arg*80.0*i);
        }
        strcpy(type_of_signal, "sinu3");
    }
    else if (type_of_signal[0] == ‘c’)
    {
        arg = 3.14159265/(2.0*N);
        for (i=0; i < N; i++)
        {
            real[i] = cos(arg*0.5*i*1); 
            imag[i] = sin(arg*0.5*i*1);
        }
        strcpy(type_of_signal, "chirp");
    }
    else if (type_of_signal[0] == ‘d’)
    {
        arg = 2.0*3.14159265/N;
        for (i=0; i < N; i++)
        {
            real[i] = 1.0*cos(arg*32.0*i) + 3.0 * sin(arg*4.0*i);
            imag[i] = 1.0*sin(arg*32.0*i) + 3.0 * sin(arg*4.0*i);
        }
    }
The FFT

input.c

```c
#include <math.h>

int ibitr();

void fft(xreal, ximag, n, nu)

```
The FFT

fft.c

for (k=0; k<n; k++)
{
    i = ibitr(k, nu);
    if (i>k)
    {
        treal = xreal[k];
        timag = ximag[k];
        xreal[k] = xreal[i];
        ximag[k] = ximag[i];
        xreal[i] = treal;
        ximag[i] = timag;
    }
}

int ibitr(j, nu)
int j, nu;
{
    int i, j1, j2, k;
    j1 = j;
    k = 0;
    for (i=1; i<= nu; i++)
    {
        j2 = j1 / 2;
        k = k+2 + (j1- 2*j2);
        j1 = j2;
    }
    return (k);
}

store_data.c

#include "dft.h"

void store_data(real, imag, file_name)
double real[], imag[];
char file_name[];
{
    FILE *fp, *fopen();
    int i;
    fp = fopen(file_name, "w");
    for (i=0; i < N; i++)
    {
        fprintf(fp, "%12.4f %12.4f\n", real[i], imag[i]);
    }
    fclose(fp);
}
Appendix B

The Ambiguity Function

This is to demonstrate the AF for various signals.
The AF Makefile

OBJ =  ambiguity.o input.o store_signal.o af.o store_af.o misce.o
FLAGS = -g
EXEC = ambiguity

$(EXEC): $(OBJ)
   cc $(FLAGS) $(OBJ) -o $(EXEC) -lnag -lF77 -lI77 -lU77 -lm

ambiguity.o: ambiguity.h Makefile
   cc $(FLAGS) -c ambiguity.c -o ambiguity.o

input.o: ambiguity.h Makefile
   cc $(FLAGS) -c input.c -o input.o

store_signal.o: ambiguity.h Makefile
   cc $(FLAGS) -c store_signal.c -o store_signal.o

af.o: ambiguity.h Makefile
   cc $(FLAGS) -c af.c -o af.o

store_af.o: ambiguity.h Makefile
   cc $(FLAGS) -c store_af.c -o store_af.o

misce.o: ambiguity.h Makefile
   cc $(FLAGS) -c misce.c -o misce.o

The AF ambiguity.h

/****************************************/
/* ambiguity.h */
****************************************/
#include <math.h>
#include <stdio.h>
#include <strings.h>
#include <malloc.h>
#define N 256
#define N2 128
#define M 127
/****************************************/
extern void input();
extern void store_signal();
extern void af();
extern void store_af();
extern double **dmatrix();
extern void free_dmatrix();
extern void nrerror();
/****************************************/
The AF

#include "ambiguity.h"

void main()
{
    double f[2][N];
    double** a[2];
    char type_of_signal[25];
    char file_name[250];
    char noise[25];

    input(f[0], f[1], type_of_signal, noise);

    strcpy(file_name, "/home/hawk/eng/es036/work/ambiguity/Data/");
    if (noise[0] == 'y')
        strcat(file_name, "Noise.");
    strcat(file_name, type_of_signal);
    strcat(file_name, ".si
"f

    printf("Computing the AF...

    a[0] = dmatrix(-N2, M, 0, N-1);
    a[1] = dmatrix(-N2, M, 0, N-1);

    af(f[0], f[1], a[0], a[1]);

    strcpy(file_name, "/home/hawk/eng/es036/work/ambiguity/Data/");
    if (noise[0] == 'y')
        strcat(file_name, "Noise.");
    strcat(file_name, type_of_signal);
    strcat(file_name, ".af U
"

    printf("Computing the AF...

    af(f[0], f[1], a[0], a[1]);

    strcpy(file_name, "/home/hawk/eng/es036/work/ambiguity/Data/");
    if (noise[0] == 'y')
        strcat(file_name, "Noise.");
    strcat(file_name, type_of_signal);
    strcat(file_name, ".af U
"

    store_signal(f[0], f[1], file_name);
    store_af(a[0], a[1], file_name);

    free_dmatrix(a[0], -N2, M, 0, N-1);
    free_dmatrix(a[1], -N2, M, 0, N-1);
}

The AF

#include "ambiguity.h"

void input(real, imag, type_of_signal, noise)
{

double real[], imag[];
char type_of_signal[];
char noise[];

    double arg;
    int i;

    printf("Choose a signal from:

    a sinusoidal signal

    b sum of three sinusoidal signals

    c chirp signal

    d frequency-modulated signal\n")

    type_of_signal[0] = 'a';
    arg = 2.0*3.14159265*32.0/N;
    for (i=0; i < N; i++)
    {
        real[i] = cos(arg*i);
        imag[i] = sin(arg*i);
    }
    strcat(type_of_signal, "sinul");

    else if (type_of_signal[0] == 'b')
    {
        arg = 2.0*3.14159265/N;
        for (i=0; i < N; i++)
        {
            real[i] = 0.5*cos(arg*6.0*i) + 0.25*cos(arg*48.0*i);
            imag[i] = 0.5*sin(arg*16.0*i) + sin(arg*32.0*i) + 0.20*sin(arg*48.0*i);
        }
        strcat(type_of_signal, "sinul3");
    }

    else if (type_of_signal[0] == 'c')
    {
        arg = 3.14159265/(2.0*N);
        for (i=0; i < N; i++)
        {
            real[i] = 0.5*cos(arg*16.0*i) + cos(arg*32.0*i) + 0.25*cos(arg*48.0*i);
            imag[i] = 0.5*sin(arg*16.0*i) + sin(arg*32.0*i) + 0.20*sin(arg*48.0*i);
        }
        strcat(type_of_signal, "chirp");
    }

    else if (type_of_signal[0] == 'd')
    {
        arg = 2.0*3.14159265/N;
        for (i=0; i < N; i++)
        {
            real[i] = 1.0*cos(arg*32.0*i + 3.0 * sin(arg*4.0*i));
        }
        strcat(type_of_signal, "chirp");
    }

    printf("Computing the AF...

    af(f[0], f[1], a[0], a[1]);

    store_signal(f[0], f[1], file_name);
    store_af(a[0], a[1], file_name);

    free_dmatrix(a[0], -N2, M, 0, N-1);
    free_dmatrix(a[1], -N2, M, 0, N-1);"
The AF

`input.c`

```c
The AF
```}

```c
img[1] = 1.0*sin(arg*32.0*i + 3.0 * sin(arg*4.0*i));
}
strncpy(type_of_signal, "fm");
}
else nrerror("Wrong choice!!");
printf("n");
printf("Contain noise? (y/n) ");
scanf("%s", noise);
printf("n");
if (noise[0] == 'y')
for (i=0; i<N; i++)
{ real[i] += 0.2 * random()/2147483647.0 - 0.1;
imag[i] += 0.2 * random()/2147483647.0 - 0.1;
}
```

The AF

`af.c`

```c
#include "ambiguity.h"

void af(freal, fimag, afreal, afimag)
double freal[], fimag[];
double **afreal, **afimag;
{
  int n, nl, ml, m2;
  int i=N, ifail=0;
  for (nl=-N2; nl < N2; nl++)
    for (n=0; n<N; n++)
    { ml = n + nl;
m2 = -ml;
if (-1<ml && ml<N && -1<m2 && m2<N)
  { afreal[nl][n] = freal[nl] * freal[m2] + fimag[nl] * fimag[m2];
    afimag[nl][n] = fimag[nl] * freal[m2] - freal[nl] * fimag[m2];
    afreal[nl][n] *= 2.0;
    afimag[nl][n] *= 2.0;
  }
else
  { afreal[nl][n] = 0;
    afimag[nl][n] = 0;
  }
}
for (nl=-N2; nl < N2; nl++)
  c06ecf(afreal[nl], afimag[nl], &1, &ifail);
for (n=0; n<N; n++)
  { afreal[n][n] *= sqrt((double)N); /* nag */
    afimag[n][n] *= sqrt((double)N); /* nag */
  }
}
```
The AF

store_af.c

#include "ambiguity.h"

void store_af(real, imag, file_name)
double **real, **imag;
char file_name[];
{
    FILE *fp, *fopen();
    int nl, k;

    fp = fopen(file_name, "w");
    for (k=N2; k < N+N2; k += 4)
        for (nl=-N2; nl < N2; nl += 4)
            fprintf(fp, "%f
", real[k][k%N]*real[k][k%N] + imag[k][k%N]*imag[k][k%N]);

    fclose(fp);
}

The AF

store_signal.c

#include "ambiguity.h"

void store_signal(real, imag, file_name)
double real[], imag[];
char file_name[];
{
    FILE *fp, *fopen();
    int i;

    fp = fopen(file_name, "w");
    for (i=0; i < N; i++)
        fprintf(fp, "%12.4f %12.4f
", real[i], imag[i]);

    fclose(fp);
}
The AF misce.c

#include "ambiguity.h"
extern void nrerror();
double **dmatrix(int nrl, nrh, ncl, nch)
{
    int i;
    double **m;
    m = (double**) malloc((unsigned) (nrh-nrl+1)*sizeof(double*));
    if (!m) nrerror("Allocation failure 1 in dmatrix()");
    m -= nrl;
    for(i=nrl; i <= nrh; i++)
        { m[i] = (double*)malloc((unsigned) (nch-ncl+1)*sizeof(double));
          if (!m[i]) nrerror("Allocation failure 2 in dmatrix()");
          m[i] -= ncl;
        }
    return m;
}
void free_dmatrix(double **m, int nrl, nrh, ncl, nch)
{
    int i;
    for(i=nrh; i >= nrl; i--)
        { free((char*) (m[i]+ncl));
          free((char*) (m+nrl));
        }
}
void nrerror(char error_text[])
{
    void exit();
    fprintf(stderr, "ERROR: ");
    fprintf(stderr, "$s\n", error_text);
    exit(1);
}
Appendix C

The Wigner Distribution

To demonstrate the WDs for various signals.
**The WD**

**Makefile**

OBJ = wigner.o input.o store_signal.o wd.o store_wd.o misce.o

FLAGS = -g

EXEC = wigner

$(EXEC): $ (OBJ)
  cc $(FLAGS) $(OBJ) -o $(EXEC) -lnag -lF77 -lU77 -lm

wigner.o: wigner.h Makefile
  cc $(FLAGS) -c wigner.c -o wigner.o

input.o: wigner.h Makefile
  cc $(FLAGS) -c input.c -o input.o

store_signal.o: wigner.h Makefile
  cc $(FLAGS) -c store_signal.c -o store_signal.o

wd.o: wigner.h Makefile
  cc $(FLAGS) -c wd.c -o wd.o

store_wd.o: wigner.h Makefile
  cc $(FLAGS) -c store_wd.c -o store_wd.o

misce.o: wigner.h Makefile
  cc $(FLAGS) -c misce.c -o misce.o

---

**The WD**

**wigner.h**

```c
#include <math.h>
#include <stdio.h>
#include <malloc.h>

#define N 256
#define N2 128
#define M 127

extern void input();
extern void store_signal();
extern void wd();
extern void store_wd();
extern double **dmatrix();
extern void free_dmatrix();
extern void nrerror();
```

The WD

#include "wigner.h"

void main()
{
      double f[2][N];
      double* w[2];
      char type_of_signal[25];
      char file_name[250];
      char noise[25];
      input{f[0], f[l], type_of_signal, noise);
      strcpy(file_name, "./home/hawk/eng/es036/work/wigner/Data/");
      if (noise[0] == 'y')
            strcat(file_name, "Noise.");
      strcat(file_name, type_of_signal);
      strcat(file_name, ".sign");
      store_signal(f[0], f[1], file_name);

      printf("\n\nComputing the WD ....\n\n");
      w[0] = dmatrix(0, N-1, 0, N-1);
      w[1] = dmatrix(0, N-1, 0, N-1);
      wd(f[0], f[1], w[0], w[1]);
      strcpy(file_name, "./home/hawk/eng/es036/work/wigner/Data/");
      if (noise[0] == 'y')
            strcat(file_name, "Noise.");
      strcat(file_name, type_of_signal);
      strcat(file_name, ".wd");
      store_wd(w[0], w[1], file_name);
      free_dmatrix(0, N-1, 0, N-1);
      free_dmatrix(0, N-1, 0, N-1);
}

The WD

#include "wigner.h"

void main()
{
      double real[], imag[];
      char type_of_signal[];
      char noise[];
      
      double arg;
      int i;

      printf("\n\nChoose a signal from:
\n\n  a  sinusoidal signal
  b  sum of three sinusoidal signals
  c  chirp signal
  d  frequency-modulated signal
\n");
      scanf("%s", type_of_signal);
      if (type_of_signal[0] == 'a')
            { arg = 2.0*3.14159265*32.0/N;
              for (i=0; i < N; i++)
                    { real[i] = cos(arg*i);
                      imag[i] = sin(arg*i);
                    }
            }
      else if (type_of_signal[0] == 'b')
            { arg = 2.0*3.14159265/N;
              for (i=0; i < N; i++)
                    { real[i] = 0.5*cos((arg*16.0*i) + 0.25*cos((arg*48.0*i));
                      imag[i] = 0.5*sin((arg*16.0*i) + 0.25*sin((arg*48.0*i));
                    }
            }
      else if (type_of_signal[0] == 'c')
            { arg = 3.14159265/(2.0*N);
              for (i=0; i < N; i++)
                    { real[i] = cos(arg*0.5*i*i);
                      imag[i] = sin(arg*0.5*i*i);
                    }
            }
      else if (type_of_signal[0] == 'd')
            { arg = 2.0*3.14159265/N;
              for (i=0; i < N; i++)
                    { real[i] = 1.0*cos(arg*32.0*i + 3.0 * sin(arg*4.0*i));
                      imag[i] = sin(arg*4.0*i);
                    }
            }

      }
```c
#include <wigner.h>

void wd(double *freal, double *fimag, double *wdreal, double *wdimag, int n, int m, int M)
{
    double c, s, arg = 2.0 * M * 3.14159265358979323846;
    for (int k = 0; k < M; k++)
    {
        arg = 2.0 * k;
        c = cos(arg);
        s = sin(arg);
        for (int n = 0; n < M; n++)
        {
            real[n] = c06ecf(wdreal[n], wdimag[n], sqrt(double(M)), &1, &ifail);
        }
    }
    for (int n = 0; n < M; n++)
    {
        real[n] = sqrt(double(M)) * c06ecf(wdreal[n], wdimag[n], &1, &ifail);
    }
}
```

```c
/* The WD input.c */

```c
/* The WD wd.c */
```
The WD

`wd.c`

```c
wdreal[n][k] = treal * c - timag * s;
wdimag[n][k] = treal * s + timag * c;
```

The WD

`store_signal.c`

```c
#include "wigner.h"

void store_signal(real, imag, file_name)
double real[], imag[];
char file_name[];
{
    FILE *fp, *fopen();
    int i;

    fp = fopen(file_name, "w");
    for (i=0; i < N; i++)
    { fprintf(fp, "%12.4f %12.4f\n", real[i], imag[i]);
    }
    fclose(fp);
}
```
The WD

#include "wigner.h"

void store_wd(real, imag, file_name)
double **real, **imag;
char file_name[];
{
    FILE *fp, *fopen();
    int n, k;

    fp = fopen(file_name, "w");
    for (k=N2; k < N+N2; k += 4)
        for (n=0; n < N; n += 4)
        {
            fprintf(fp, "%f\n", real[n][k%N]);
        }
    fclose(fp);
}

The WD

#include "wigner.h"

extern void nrerror();

double **dmatrix(nrl, nrh, ncl, nch)
int nrl, nrh, ncl, nch;
{
    int i;
    double **m;

    m = (double**) malloc( (unsigned) (nrh-nrl+1)*sizeof(double*) );
    if (!m) nrerror("Allocation failure 1 in dmatrix()" );
    m -= nrl;
    for(i=nrl; i <= nrh; i++)
    {
        m[i] = (double*) malloc( (unsigned) (nch-ncl+1)*sizeof(double) );
        if (!m[i]) nrerror("Allocation failure 2 in dmatrix()" );
        m[i] -= ncl;
    }
    return m;
}

void free_dmatrix(m, nrl, nrh, ncl, nch)
{
    int i;
    for(i=nrh; i >= nrl; i--)
    {
        free((char*) (m[i]+ncl));
    }
    free((char*) (m+nrl));
}

void nrerror(error_text)
char error_text[];
{
    void exit();
    fprintf(stderr, "ERROR: ");
    fprintf(stderr, "%s\n", error_text);
    exit(1);
}
Appendix D

The Hough Transform

The extraction of useful parameters from chirp signals
The HT

Makefile

OBJ = Hough.o input.o wd.o \n    point_space.o para_space.o \n    extraction.o \n    suncore.o misc.o
FLAGS = -g
EXEC = Hough

$(EXEC): $(OBJ)
    cc $(FLAGS) $(OBJ) -o $(EXEC) \n        -lnag -lF77 -lU77 -lcore -lsunwindow -lpixrect -lm

Hough.o: Hough.h Makefile
    cc $(FLAGS) -c Hough.c -o Hough.o
input.o: Hough.h Makefile
    cc $(FLAGS) -c input.c -o input.o
wd.o: Hough.h Makefile
    cc $(FLAGS) -c wd.c -o wd.o
suncore.o: Hough.h Makefile
    cc $(FLAGS) -c suncore.c -o suncore.o
point_space.o: Hough.h Makefile
    cc $(FLAGS) -c point_space.c -o point_space.o
para_space.o: Hough.h Makefile
    cc $(FLAGS) -c para_space.c -o para_space.o
extraction.o: Hough.h Makefile
    cc $(FLAGS) -c extraction.c -o extraction.o
misc.o: Hough.h Makefile
    cc $(FLAGS) -c misc.c -o misc.o

Hough.h

/**************************************************************/
#include <math.h>
#include <stdio.h>
#include <strings.h>
#include <usercore.h>
#define N 256
#define M 127
/**************************************************************/
extern double a;
extern double frac;
extern double a1;
extern double frac1;
/**************************************************************/
extern void input();
extern void wd();
extern void point_space();
extern void para_space();
extern void extraction();
extern void store_sigr();
extern void store_sigm();
extern void store_simg();
extern void store_vec();
extern void plot_vec();
extern void store_wd();
extern void store_point_space();
extern void store_para_space();
extern void plot_wd();
extern void plot_para_space();
extern double **dmatix();
extern void free_dmatix();
extern void error_message();
/**************************************************************/
extern void set_up();
extern void clean_up();
extern char *io_by_keyboard();
extern int go_on_or_not();
extern char input_string();
extern int seg;
/**************************************************************/
extern void error_message();
The HT

Hough.h

/******************************************/

The HT

Hough.c

/*******************************/

#include "Hough.h"

main()
{

static double f[2][N];
static double w[2][N][M];
static double **xy;
static double **ab;
static double amin = 0.0, da = 0.005;
static double bmin = -50, db = 1;
static int precision = 100;
static char type_of_signal[25];
static int finished = 0;

set_up();

while( !finished )
{

input(f[0], f[1], type_of_signal);
store_sig_re(f[0], type_of_signal);
store_sig_im(f[1], type_of_signal);
set_viewport_2(0.0, 1.0, 0.65, 0.75);
plot_vector(f[0], N, "n", "Real part of f[n]!");
set_viewport_2(0.0, 1.0, 0.55, 0.65);
plot_vector(f[1], N, "n", "Imaginary part of f[n]!");

wd(f[0], f[1], w[0], w[1]);
store_wd(w[0], type_of_signal);
set_viewport_2(0.0, 1.0, 0.3, 0.55);
plot_wd(w[0], "n", "k", "W[n,k]!");

xy = dmatrix(0, N-1, 0, N-1);
point_space(w[0], xy);
store_point_space(xy, type_of_signal);

ab = dmatrix(0, precision, -precision/2, precision/2 -1);
para_space(xy, ab, amin, da, bmin, db, precision);
store_para_space(ab, precision, type_of_signal);
}
The HT Hough.c

set_viewport_2(0.0, 1.0, 0.05, 0.30);
plot_para_space(ab, precision, "a", "b", "Parameter Space");

extraction(ab, amin, da, bmin, db, precision);
free_dmatrix(xy, 0, N-1, 0, N-1);
free_dmatrix(ab, 0, precision, -precision/2, precision/2 -1);
finished = go_on_or_not("Go on ?");
}
clean_up();

The HT input.c

#include  "Hough.h"

double a, frac;

void input(real, imag, type_of_signal)
double real[], imag[];
char type_of_signal[];
{
    double coef, arg;
    int i;
    io_by_keyboard("Choose a, frac (Pi/N):", 0.2, 0.4);
    sscanf(input_string,"%.1f %.1f", &a, &frac);
    coef = 0.5 * 3.141596 * frac/N;
    for (i=0; i<N; i++)
    {
        arg = coef * i;
        real[i] = a * cos(arg);
        imag[i] = a * sin(arg);
    }
    sprintf(type_of_signal, "%f%.3f", a, frac);
```c
#include "Hough.h"

void plot_wd(w, xlabel, ylabel, zlabel)
double w[N][N];
char xlabel[], ylabel[], zlabel[];
{
    double min, max, scale, shift;
    int n, k;

    min = max = w[0][0];
    for (n=0; n<N; n++)
        for (k=0; k<N; k++)
        { min = (w[n][k]<min) ? w[n][k] : min;
          max = (w[n][k]>max) ? w[n][k] : max;
        }

    scale = 1.0 / (max - min);
    shift = (-min) * scale;

    set_window(-0.16, 1.7, -0.16, 1.7);
    create_retained_segment( +seg );
    move_abs -3 (0.0, 0.0, shift);
    line -abs -3 (1.0, 0.0, shift);
    move_abs -3 (0.0, 0.0, shift);
    line -abs -3 (0.0, 1.0, shift);
    move_abs -3 (0.0, 0.0, shift+max*scale);
    for (n=0; n < N; n += 5)
        for (k=0; k < N; k += 5)
        { move_abs 3(n/(double)N), 0.0, w[n][k]*scale+shift );
        }
    move_abs 3(1.05, 0.0, shift);
    text[xlabel];
    move_abs 3(0.0, 1.0, shift+max*scale*0.2);
    text[ylabel];
    move_abs 3(0.0, 0.0, max*scale + shift);
    text[zlabel];
    close_retained_segment();
```

```c
#include "Hough.h"

void plot_wd_wdreal[wreal[N][k] = frea1 + timag * s;}
    wdimag[N][k] = treal * s + timag * c;
}
```
The HT

**wd.c**

```c
/

/*-----------------------------*/

void store_wd(real, type_of_signal)
double real[N][N];
char type_of_signal[];
{
  FILE *fp, *fopen();
  int n, k;
  char filename[100];
  strcpy(filename, "Datal");
  strcat(filename, type_of_signal);
  strcat(filename, ".wd");

  fp = fopen(filename, "w");
  for (k=N2; k < N+N2; k += 4)
    for (n=0; n < N; n += 4)
      { fprintf(fp, "%f
", real[n][k%N]);
      }
  fclose(fp);
}
```

**point_space.c**

```c
#include "Hough.h"

void point_space(w, xy)
double w[N][N];
double **xy;
{
  double peak, valley, threshold;
  int i, j;
  peak = valley = w[O][0];
  for (i=0; i<N; i++)
    for (j=0; j<N; j++)
      { if (peak < w[i][j])
        peak = w[i][j];
        if (valley > w[i][j])
          valley = w[i][j];
      }
  threshold = (peak-valley)*0.25 + valley;
  for (i=0; i<N; i++)
    for (j=0; j<N; j++)
      { if (w[i][j] > threshold)
        xy[i][j] = 1;
        else
          xy[i][j] = 0;
      }
}

void store_point_space(xy, type_of_signal)
double **xy;
char type_of_signal[];
{
  FILE *fp, *fopen();
  int n, k;
  char filename[100];
  strcpy(filename, "Datal");
  strcat(filename, type_of_signal);
  strcat(filename, ".xy");

  fp = fopen(filename, "w");
  for (k=N2; k < N+N2; k += 4)
    { fprintf(fp, "%f
", real[n][k%N]);
    }
  fclose(fp);
}
```
The HT

point_space.c

for (n=0; n < N; n += 4)
{
    fprintf(fp, "d\n", (int) (xy[n][kN]));
}
fclose(fp);

para_space.c

#include "Hough.h"

void para_space(xy, ab, amin, da, bmin, db, P) double **xy;
double **ab;
double amin, da, bmin, db;
int P;
{
    int i, j, ia, jb;
    double a, b;
    int counter;
    for (i=0; i<P; i++)
    {
        for (j=-P/2; j<P/2; j++)
        {
            ab[i][j] = 0;
        }
    }
    for (i=0; i<P; i++)
    for (j=-P/2; j<P/2; j++)
    if (xy[i][j] > 0.5)
    for (ia=0; ia<P; ia++)
    {
        a = amin + ia * da;
        b = a * i + j;
        jb = (int) (b / db);
        ab[ia][jb] += 1;
    }
}

void plot_para_space(ab, P, xlabel, ylabel, zlabel)
double **ab;
int P;
char xlabel[], ylabel[], zlabel[];
{
    double min, max, scale, shift, yshift;
    int i, j;
    min = max = ab[0][-(P/2)];
    for (i=0; i<P; i++)
    for (j=-(P/2); j<P/2; j++)
    
    {
        min = (ab[i][j]<min) ? ab[i][j] : min;
        max = (ab[i][j]>max) ? ab[i][j] : max;
    }
```c
void double int char {
    scale = 1.0 / (max - min);
    shift = (-min) * scale;
    /*******************************/
    yshift = 0.5;
    set_window(-0.16, 1.7, -0.16, 1.7);
    create_retained_segment( ++seg );
    move_abs_3(0.0, 0.0+yshift, shift);
    line_abs_3(1.0, 0.0+yshift, shift);
    move_abs_3(0.0, 0.5+yshift, shift);
    line_abs_3(0.0, 0.0+yshift, shift);
    move_abs_3(0.0, 0.0+yshift, shift);
    line_abs_3(0.0, 0.0+yshift, shift+max*scale);
    for (i=0; i < P; i += 2)
    { move_abs_3(i/(double)P), -0.5+yshift, ab[i][-P/2]*scale+shift );
        for (j=-P/2; j<P/2; j += 2)
        { line_abs_3(i/(double)P),j/(double)P) + yshift, ab[i][j]*scale+shift );
    }
    move_abs_3(1.05, yshift, shift);
    text(xlabel);
    move_abs_3(0.0, 0.6+yshift, shift);
    text(ylabel);
    move_abs_3(0.0, yshift, max*scale + shift);
    text(zlabel);
    close_retained_segment();
    /**************-***************/
}
void store_para_space(ab, p, type_of_signal)
    double **ab;
    int p;
    char *type_of_signal;
{
    FILE *fp, *fopen();
    int i, j;
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type_of_signal);
    strcat(file_name, ".ab");

    fp = fopen(file_name, "w");
    for (j=-p/2; j < p/2; j += 2)
    { for (i=0; i < p; i += 2)
        { fprintf(fp, "%d ", (int) ab[i][j]);
        }
        fprintf(fp, "\n");
    }
    fclose(fp);
}
```
The HT

```
#include "Hough.h"

double a1, frac1;

void extraction(ab, amin, da, bmin, db, P)
{
    double **ab;
    double amin, da, bmin, db;
    int P;
    {
        int counter;
        int i, j, i0, j0;
        char str[100];
        counter = ab[0][0-
P/2];
        i0 = 0;
        j0 = -P/2;
        for (i=0; i<P; i++)
            for (j=-P/2; j<P/2; j++)
            {
                if (counter < ab[i][j])
                {
                    counter = ab[i][j];
                    i0 = i;
                    j0 = j;
                }
            }
        frac1 = amin + i0 * da;
        set_viewport(2(0.0, 1.0, 0.0, 0.05));
        set_window(-0.1, 1.1, -0.1, 1.1);
        create_retained_segment(**seg);
        move_abs(3(0.4, 0.0, 0.8);
        sprintf(str, "Original: frac%.3f", frac1);
        text(str);
        move_abs(3(0.4, 0.0, 0.4);
        sprintf(str, "Extracted: frac%.3f", frac1);
        text(str);
        close_retained_segment();
    }
}
```

The HT

```
#include "Hough.h"

extern int pixwindd();

struct vwsurf = DEFAULT_VWSURF(pixwindd);

int keyboard_number = 1;
char input_string[80];
int buffer_size = 80;
char initial_string[80] = {"enter:");
int initial_cursor_position = 7;
double echo_x, echo_y;
int echo_type = 1;
int length;

int seg = 0;

void set_up()
{
    initialize_core(DYNAMIC, SYNCHRONOUS, THREED);
    initialize_view_surface(&vwsurf, FALSE);
    select_view_surface(&vwsurf);
    set_view_reference_point(0.0, 0.0, 0.0);
    set_view_plane_normal(0.0, 1.0, 0.0);
    set_projection(PARALLEL, -1.0, 1.732, -1.0);
    set_view_up_3(0.0, 0.0, 1.0);
    initialize_device(KEYBOARD, keyboard_number);
    set_echo(KEYBOARD, keyboard_number, echo_type);
    set_echo_surface(KEYBOARD, keyboard_number, &vwsurf);
}

void clean_up()
{
    terminate_device(KEYBOARD, keyboard_number);
    deselect_view_surface(&vwsurf);
    terminate_core();
}
```
The HT suncore.c

```c
int io_by_keyboard(prompt, x, y)
  char* prompt;
  double x, y;
  {
    int MAXINT = 2147483647;
    char str[80], *p;
    strcpy(initial_string, prompt);
    initial_cursor_position = strlen(initial_string) + 5;
    echo_x = x; echo_y = y;
    set_echo_position(KEYBOARD, keyboard_number, echo_x, echo_y);
    set_keyboard(keyboard_number, buffer_size, initial_string, initial_cursor_position);
    await_keyboard(MAXINT, keyboard_number, input_string, &length);
    input_string[ strlen(input_string) - 1 ] = '\0';
    p = input_string;
    while ( *p == ' ' || *p == '\t' )
      { p++;
      }
    strcpy(str, p);
    strcpy(input_string, str);
    return input_string;
  }

int go_on_or_not(text)
  char text[];
  {
    if (io_by_keyboard("", 0.2, 0.4);
      while (seg > 0)
      { delete_retained_segment (seg--);
      }
    io_by_keyboard(text, 0.2, 0.4);
    if (input_string[0] == 'n')
      return 1;
    else if (input_string[0] == 'N')
      return 1;
    else
      return 0;
  }
```

The HT suncore.c

```c
/******************************************************************************/
```
The HT misce.c

```c
#include "Hough.h"

void store_sig_re(v, type_of_signal)
double v[];
char type_of_signal[];
{
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type_of_signal);
    store_vector(v, N, file_name);
}

void store_sig_im(v, type_of_signal)
double v[];
char type_of_signal[];
{
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type_of_signal);
    store_vector(v, N, file_name);
}

void store_vector(v, length, file_name)
double v[];
int length;
char file_name[];
{
    FILE *fp, *fopen();
    int i;
    fp = fopen(file_name, "w");
    for (i=0; i < length; i++)
    {
        fprintf(fp, "%f\n", v[i]);
    }
    fclose(fp);
}

void plot_vector(v, length, xlabel, ylabel)
double v[];
int length;
char xlabel[], ylabel[];
{
    double min, max;
    int i;
    min = max = v[0];
    for (i=0; i < length; i++)
    {
        if (min > v[i])
        {
            min = v[i];
        }
        if (max < v[i])
        {
            max = v[i];
        }
    }
    set_window(-0.1*length, 1.1*length,
               min-0.1*(max-min), max+0.1*(max-min));
    create_retained_segment(++seg);
    { move_abs_3(0.0, 0.0, 0.0);
      line_abs_3((double)length, 0.0, 0.0);
      move_abs_3(0.0, 0.0, max);
      move_abs_3(0.0, 0.0, v[0]);
      for (i=0; i < length; i++)
      {
        line_abs_3((double)i, 0.0, v[i]);
      }
      move_abs_3((double)length, 0.0, 0.0);
      text(xlabel);
      move_abs_3(0.0, 0.0, max+0.05*(max-min));
      text(ylabel);
    }
    close_retained_segment();
}

double **dmatrix(nrl, nrh, ncl, nch)
int nrl, nrh, ncl, nch;
{
    int i;
    double **m;
    m = (double**) malloc((unsigned) (nrh-nrl+1)*sizeof(double*));
    if (!m) error_message("Allocation failure 1 in dmatrix()");
    m -= nrl;
    for (i=nrl; i <= nrh; i++)
    { m[i] = (double*) malloc((unsigned) (ncl-nel+1)*sizeof(double));
      if (!m[i]) error_message("Allocation failure 2 in dmatrix()");
      m[i] -= ncl;
    }
}
```

return m;
}

void free_dmatrix(m, nrl, nrh, ncl, nch)
double **m;
int nrl, nrh, ncl, nch;
{
    int i;
    for(i=nrh; i >= nrl; i--)
    {
        free((char*) (m[i]+ncl));
    }
    free((char*) (m+nrl));
}

void error_message(text)
char text[];
{
    void exit();
    fprintf(stderr, "ERROR: ");
    fprintf(stderr, "%s\n", text);
    exit(1);
}
Appendix E

Extraction of Useful Parameters

The extraction of useful parameters from both chirp and FM signals.
Extraction

Makefile

OBJ = Extraction.o choose.o input.o hibert.o \ 
    moments.o wd.o extraction.o suncore.o misc.e.o
FLAGS = -q
EXEC = Extraction

$(EXEC): $(OBJ) $(EXEC) -o $(EXEC) \
      -lnag -lF77 -lI77 -lU77 \ 
      -lcore -lsunwindow -lpixrect -lm

Extraction.o: Extraction.h Makefile
  cc $(FLAGS) -c Extraction.c -o Extraction.o
choose.o: Extraction.h Makefile
  cc $(FLAGS) -c choose.c -o choose.o
input.o: Extraction.h Makefile
  cc $(FLAGS) -c input.c -o input.o
hilbert.o: Extraction.h Makefile
  cc $(FLAGS) -c hilbert.c -o hilbert.o
wd.o: Extraction.h Makefile
  cc $(FLAGS) -c wd.c -o wd.o
moments.o: Extraction.h Makefile
  cc $(FLAGS) -c moments.c -o moments.o
extraction.o: Extraction.h Makefile
  cc $(FLAGS) -c extraction.c -o extraction.o
suncore.o: Extraction.h Makefile
  cc $(FLAGS) -c suncore.c -o suncore.o
misc.e.o: Extraction.h Makefile
  cc $(FLAGS) -c misc.e.c -o misc.e.o

Extraction

Extraction.c

//=======================================================================
@WebServlet(/ajax)
Extraction Extraction.c

plot_vector(f[l], N, "n", "Imaginary part of f_a[n]");
store_analim(f[l], type_of_signal);
}

wd(f[0], f[l], w[0], w[l]);
set_viewport_2(0.0, 1.0, 0.2, 0.55);
plot_wd(w[0], "n", "k", "W[n,k]");
store_wd(w[0], type_of_signal);

moments(w[0], p, theta);
set_viewport_2(0.0, 1.0, 0.125, 0.2);
plot_vector(p, N, "n", "P");
store_mom_p(p, type_of_signal);
set_viewport_2(0.0, 1.0, 0.0, 0.05);
plot_vector(theta, N, "n", "theta");
extraction(p, theta, type_of_signal);

finished = go_on_or_not("Go on?");
#include "Extraction.h"

void choose(char *type_of_signal)
{
    type_of_signal[0] = 0;
    set_viewport(0.0, 1.0, 0.0, 0.75);
    set_window(0.0, 1.0, 0.0, 1.0);
    create_retained_segment(++seg);
    move_abs_3(0.25, 0.0, 0.5);
    text("Choose a signal from:");
    move_abs_3(0.25, 0.0, 0.45);
    text(" - cc complex-valued, chirp signals");
    move_abs_3(0.25, 0.0, 0.4);
    text(" - rc real-valued, chirp signals");
    move_abs_3(0.25, 0.0, 0.35);
    text(" - cf complex-valued, FM signals");
    move_abs_3(0.25, 0.0, 0.3);
    text(" - rf real-valued, FM signals");
}

close_retained_segment();

io_by_keyboard("Enter: ", 0.25, 0.2);
strncpy(type_of_signal, input_string, 5);

delete_retained_segment(seg--);
/include "Extraction.h"

double a, b;
double frac;
double k_o, phi_o, k_m, phi_m;

void input(real, imag, type_of_signal)

double real[], imag[];
char type_of_signal[];
{
    double coef, arg;
    int i;

    if (type_of_signal[1] == 'c')
    {
        io_by_keyboard("a, frac (Pi/N):", 0.2, 0.4);
        scanf(input_string, "%lf %lf", &a, &frac);
        coef = 0.5 * 3.141596 * frac/N;
        for (i=0; i<N; i++)
        {
            arg = coef * i;
            real[i] = a * cos(arg);
            if (type_of_signal[0] == 'c')
            {
                imag[i] = a * sin(arg);
                sprintf(type_of_signal, "cc%3f%3f", a, (int)k_o, b, (int)k_m);
            }
            else if (type_of_signal[0] == 'r')
            {
                imag[i] = 0;
                sprintf(type_of_signal, "rc%3f%3f", a, (int)k_o, b, (int)k_m);
            }
            else error_message("Wrong choice!!!");
        }
    }
    else if (type_of_signal[1] == 'f')
    {
        io_by_keyboard("a, k_o, phi_o (Deg):", 0.2, 0.4);
        scanf(input_string, "%lf %lf %lf", &a, &k_o, &phi_o);
        phi_o /= 180.0/3.141596;
        io_by_keyboard("b, k_m, phi_m (Deg):", 0.2, 0.4);
        scanf(input_string, "%lf %lf %lf", &b, &k_m, &phi_m);
        phi_m /= 180.0/3.141596;
        coef = 2.0 * 3.141596 /N;
        for (i=0; i<N; i++)
        {
            arg = coef * k_o * i + phi_o;
            arg += b * sin(coef * k_m * i + phi_m);
            real[i] = a * cos(arg);
            if (type_of_signal[0] == 'c')
            {...}
/* hilbert.c */

#include "Extraction.h"

void hilbert(real, imag)
  double real[], imag[];
{
  int I=N, ifail=0;
  int i;
  
  /***************************************************************************
   * ***************** FFT *****************
   * c06ecf(real, imag, &I, &ifail);  
   **************************************************************************/
  for (i=0; i<i; i++)
  {
    if ( i >= N2)
    { real[i] = 0.0;
      imag[i] = 0.0;
    } else
    { real[i] *= 2.0;
      imag[i] *= 2.0;
    }
  }

  /***************************************************************************
   * ***************** IFFT *****************
   * c06gcf(imag, &I, &ifail);  
   * c06ecf(real, imag, &I, &ifail);  
   * c06gcf(imag, &I, &ifail);  
   **************************************************************************/
}

/* wd.c */

#include "Extraction.h"

void wd(freal, fimag, wdreal, wdimag)
  double freal[N], fimag[N];
  double wdreal[N][N], wdimag[N][N];
{
  int n, m, m1, m2, k;
  int I=N, ifail=0;
  double treal, timag, arg, c, s;
  for (n=0; n<N; n++)
    for (m=0; m<N; m++)
    {
      m1 = n + m - M;
      m2 = n - m + M;
      if (-1<=m1 && m1<=N && -1<=m2 && m2<=N)
      { wdreal[n][m] = freal[m1] * freal[m2];
        + fimag[m1] * fimag[m2];
        wdimag[n][m] = fimag[m1] * freal[m2];
        - freal[m1] * fimag[m2];
        wdreal[n][m] *= 2.0;
        wdimag[n][m] *= 2.0;
      }
    else
      { wdreal[n][m] = 0;
        wdimag[n][m] = 0;
      }
  }

  /***************************************************************************
   * ***************** FFT *****************
   * c06ecf(wdreal[n], wdimag[n], &I, &ifail);  
   **************************************************************************/
  for (k=0; k<N; k++)
  {
    arg = 2.0 * M * 3.14159265358979323846;
    arg = k;
    arg *= M;
    c = cos(arg);
    s = sin(arg);
    for (n=0; n<N; n++)
    { treal = wdreal[n][k]*sqrt((double)N);  /* nag */
      timag = wdimag[n][k]*sqrt((double)N);  /* nag */
    }
  }

  /***************************************************************************
   * ***************** IFFT *****************
   * c06gcf(wdreal[n], wdimag[n], &I, &ifail);  
   **************************************************************************/
}
Extraction

```c
#include "Extraction.h"

void moments(w, p, theta)
double w[N][N];
double p[N];
double theta[N];
{
    double x[N], y[N], coef;
    int n, k;

    /************** p *****************/
    for (n=0; n<N; n++)
    {
        p[n] = 0;
        for (k=0; k<N; k++)
        {   
            p[n] += w[n][k];
        }
        p[n] /= 2.0*N;
    }
    p[0] = 0; /* for the sake of plot */

    /************** x ***************/
    coef = 2.0 * 3.141596 /N;
    for (n=0; n<N; n++)
    {
        x[n] = y[n] = 0.0;
        for (k=0; k<N; k++)
        {   
            x[n] += w[n][k] * cos(coef*k);
            y[n] += w[n][k] * sin(coef*k);
        }
    }
    for (n=0; n<N; n++)
    {
        if (y[n] == 0 && x[n] > 0)
            { theta[n] = 0; }
        else if (y[n] == 0 && x[n] < 0)
            { theta[n] = 3.141596; }
        else if (y[n] == 0 && x[n] == 0)
            { theta[n] = 0; }
        else if (x[n] < 0 )
            { theta[n] = atan(y[n]/x[n]) + 3.141596; }
        else if (x[n] > 0 )
            { theta[n] = 0; }
        if (y[n] > 0)
```
```c
double theta[n] = 3.141596/2.0;
else if (y[n] < 0)
    theta[n] = 3.0*3.141596/2.0;
else
    theta[n] = 0;
}
else
    theta[n] = atan(y[n]/x[n]);
theta[n] /= 2.0;
}

#include "Extraction.h"

double a_1, b_1;
double frac1;
double k_0_1, k_m, phi_0, phi_m;

void extraction(p, theta, type_of_signal)
    double p[N];
    double theta[N];
    char type_of_signal[];
    {
        extern void results_for_chirp();
        extern void extraction_for_chirp();
        extern void results_for_fm();
        extern void extraction_for_fm();

        char file_name[100];
        strcpy(file_name, "Data/");
        strcat(file_name, type_of_signal);  
        strcat(file_name, ".res.tex");

        if (type_of_signal[1] == 'c')
            extraction_for_chirp(p, theta);
            results_for_chirp(file_name);
        else
            extraction_for_fm(p, theta);
            results_for_fm(file_name);
    }

void extraction_for_chirp(p, theta)
    double p[N];
    double theta[N];
    {
        char str[100];
        double t[6];
        int n;

        //********** a_1 **********/
        a_1 = 0;
        for (n=1; n < N; n++)
            a_1 += p[n];
```
```c
void double double {
    a1 /= N-1.0;
    a1 = sqrt(a1);

    /******** frac1 ********/
    for (n=0; n<N; n++)
        { t[0] += n * n;
          t[1] += n;
          t[2] += theta[n] * n;
          t[3] += n;
          t[4] += 1;
          t[5] += theta[n];
        }
    frac1 = (t[2]*t[4] - t[1]*t[5])/t[0]*t[4] - t[1]*t[3];
    frac1 *= 256/3.141596;

    set_window(-0.1, 1.1, -0.1, 1.1);
    create_retained_segment(+seg);
        move_abs(3*0.30, 0.0, 0.8);
        sprintf(str,
"Original: a=%.3f, frac=\%.3f", a, frac);
        text(str);
        move_abs(3*0.30, 0.0, 0.8);
        sprintf(str,
"Extracted: a1=%.3f, frac1=%.3f", a1, frac1);
        text(str);
    close_retained_segment();
    /**************************/}

#endif
ek_ml = 2;
b1 = theta[2];
for (k=2; k<N/2; k++)
    { if (theta[k] > b1)
        { b1 = theta[k];
          k_ml = k;
        }
    }
/******** b1 **********/
```
```c
void char
{
    FILE *fp, *fopen();
    fp = fopen(file_name, "w");
    fprintf(fp, \caption{Original: a=%3.3f, k_o=%lf, b=%3f, k_m=%lf"
                , a, k_o, b, k_m);
    text(str);
    move_abs_3(0.25, 0, 0, 0.8);
    sprintf(str,
        Original: a=%3.3f, k_o=%lf, b=%3f, k_m=%lf"
        , a, k_o, b, k_m);
    text(str);
    move_abs_3(0.25, 0, 0.4);
    sprintf(str,
        Extracted: a1=%3.3f, k_o1=%lf, b1=%3f, k_m1=%lf"
        , a1, k_o1, b1, k_m1);
    text(str);
    close_retained_segment();
    /**********************/
}
```
```c
#include "Extraction.h"

extern int pixwindd();
struct vwsurf
vwsurf = DEFAULT_VWSURF(pixwindd);

int keyboard_number = 1;
char input_string[80];
buffer size = 80;
initial string[80] = ("enter:");
initial_cursor_position = 7;
double echo_x, echo_y;
int echo_type = 1;
int length;

int seg = 0;

void set_up()
{
    initialize_core(DYNAMICC, SYNCHRONOUS, THREED);
    initialize_view_surface(&vwsurf, FALSE);
    select_view_surface(&vwsurf);

    set_view_reference_point(0.0, 0.0, 0.0);
    set_view_plane_normal(0.0, 1.0, 0.0);
    set_projection(PARALLEL, -1.0, 1.732, -1.0);
    set_view_up_3(0.0, 0.0, 1.0);

    initialize_device(KEYBOARD, keyboard_number);
    set_echo(KEYBOARD, keyboard_number, echo_type);
    set_echo_surface(KEYBOARD, keyboard_number, &vwsurf);
}

void clean_up()
{
    terminate_device(KEYBOARD, keyboard_number);
    deselect_view_surface(&vwsurf);
    terminate_core();
}
```

```c
char* io_by_keyboard(prompt, x, y)
char* prompt;
double x, y;
{
    MAXINT = 2147483647;
    str = (prompt);
    char str[80], *p;
    strcpy(initial_string, prompt);
    initial_cursor_position = strlen(initial_string) + 5;
    echo_x = x; echo_y = y;

    set_echo_position(KEYBOARD, keyboard_number, echo_x, echo_y);
    set_keyboard(keyboard_number, buffer_size, initial_string, initial_cursor_position);
    await_keyboard(MAXINT, keyboard_number, input_string, &length);

    input_string[ strlen(input_string) - 1 ] = '\0';
    p = input_string;
    while( *p == ' ' || *p == '\t' )
    {
        p++;
    }
    strcpy(str, p);
    strcpy(input_string, str);
    return input_string;
}
```

```c
int go_on_or_not(text)
char* text[];
{
    io_by_keyboard("
    , 0.2, 0.4);
    while( seg > 0 )
    {
        delete_retained_segment(seg--);
    }
    io_by_keyboard(text, 0.2, 0.4);
    if (input_string[0] == 'n')
        return 1;
    else if (input_string[0] == 'N')
        return 1;
    else
        return 0;
}
```
#include "Extraction.h"

void store_sig_re(v[], type_of_signal)
  double v[];
  char type_of_signal[];
  {
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type_of_signal);
    store_vector(v, N, file_name);
  }

void store_sig_im(v[], type_of_signal)
  double v[];
  char type_of_signal[];
  {
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type_of_signal);
    strcat(file-name, ".sig.re");
    store_vector(v, N, file_name);
  }

void store_ana_re(v[], type_of_signal)
  double v[];
  char type_of_signal[];
  {
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type_of_signal);
    strcat(file-name, ".ana.re");
    store_vector(v, N, file_name);
  }

void store_ana_im(v[], type_of_signal)
  double v[];
  char type_of_signal[];
  {
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type_of_signal);
    strcat(file-name, ".ana.im");
    store_vector(v, N, file_name);
  }
```c
void store_mom_p(v, type_of_signal)
double v[];
char type_of_signal[];
{
char file_name[100];
strcpy(file_name, "Data/");
strcat(file_name, type_of_signal);
store_vector(v, type_of_signal);
}

void store_mom_t(v, type_of_signal)
double v[];
char type_of_signal[];
{
char file_name[100];
strcpy(file_name, "Data/");
strcat(file_name, type_of_signal);
store_vector(v, type_of_signal);
}

void store_wd(w, type_of_signal)
double w[N][N];
char type_of_signal[];
{
char file_name[100];
FILE *fp, *fopen();
int n, k;
strcpy(file_name, "Data/");
strcat(file_name, type_of_signal);
strcat(file_name, ".wd");
fp = fopen(file_name, "w");
for (k=N2; k < N+N2; k += 4)
for (n=0; n < N; n += 4)
{
fprintf(fp, "%f\n", w[n][kN]);
}
fclose(fp);
}

void plot_vector(v, length, xlabel, ylabel)
double v[];
int length;
char xlabel[], ylabel[];
{
double min, max;
int i;
min = max = v[0];
for (i=0; i<length; i++)
{
if (min > v[i])
{
min = v[i];
}
if (max < v[i])
{
max = v[i];
}
}
set_window(-0.1*length, 1.1*length,
min-0.1*(max-min), max+0.1*(max-min));
create_retained_segment(++seg);
line_abs_3(0.0, 0.0, 0.0);
move_abs_3(0.0, 0.0, min);
line_abs_3(0.0, 0.0, max);
move_abs_3(0.0, 0.0, v[0]);
for (i=0; i<length; i++)
{
line_abs_3( (double)i, 0.0, v[i]);
}
move_abs_3((double)length, 0.0, 0.0);
}
```
```c
Extraction

void plot_wd(w, xlabel, ylabel, zlabel)
char xlabel[], ylabel[], zlabel[];
{
  double min, max, scale, shift;
  int n, k;

  min = max - w[0][0];
  for (n=0; n<N; n++)
    for (k=0; k<N; k++)
      if (w[n][k] < min)
        min = w[n][k];
      if (w[n][k] > max)
        max = w[n][k];

  scale = 1.0 / (max - min);
  shift = -min * scale;
  net_window(-0.16, 1.7, -0.16, 1.7);
  create_retained_segment(+seg);
  move_abs_3(0.0, 0.0, shift);
  line_abs_3(1.0, 0.0, shift);
  move_abs_3(0.0, 0.0, shift);
  line_abs_3(0.0, 1.0, shift);
  move_abs_3(0.0, 0.0, shift);
  line_abs_3(0.0, 0.0, shift + max*scale);
  for (n=0; n < N; n += 5)
    for (k=0; k < N; k += 5)
      line_abs_3(n/((double)N), k/((double)N),
                 w[n][k]*scale + shift);
  move_abs_3(1.05, 0.0, shift);
  text(xlabel);
  move_abs_3(0.0, 1.0, shift + max*scale + shift);
  text(ylabel);
  move_abs_3(0.0, 0.0, max*scale + shift);
  text(zlabel);
}

void close_retained_segment();
};

void error_message(text)
char text[];
{
  void exit();
  fprintf(stderr, "ERROR: ");
  fprintf(stderr, "%s\n", text);
  exit(1);
}
```

Extraction
Appendix F

Extraction of Vibration by the WD

This program extracts vibration information from roundness signature, it covers both simulation and real cases.
Machinina Makefile

```
# Makefile

OBJ = turning.o choose.o \\
    real_parameters.o real_roundness.o plot_roundness.o cut_it_out.o \\
    simu_parameters.o simu_roundness.o \\
    level_data.o hilbert.o w.d.o moments.o extraction.o \\
    suncore.o misc.e.o

FLAGS = -g
EXEC = turning

$(EXEC): $(OBJ) $(OBJ) -o $(EXEC) \\
    -lnag -lF77 -lU77 -lU77 \\
    -lcore -lsunwindow -lpixrect -lm

turning.o: turning.h Makefile \\
    cc $(FLAGS) -c turning.c -o turning.o

choose.o: turning.h Makefile \\
    cc $(FLAGS) -c choose.c -o choose.o

real_parameters.o: turning.h Makefile \\
    cc $(FLAGS) -c real_parameters.c -o real_parameters.o

real_roundness.o: turning.h Makefile \\
    cc $(FLAGS) -c real_roundness.c -o real_roundness.o

cut_it_out.o: turning.h Makefile \\
    cc $(FLAGS) -c cut_it_out.c -o cut_it_out.o

plot_roundness.o: turning.h Makefile \\
    cc $(FLAGS) -c plot_roundness.c -o plot_roundness.o

simu_parameters.o: turning.h Makefile \\
    cc $(FLAGS) -c simu_parameters.c -o simu_parameters.o

simu_roundness.o: turning.h Makefile \\
    cc $(FLAGS) -c simu_roundness.c -o simu_roundness.o

hilbert.o: turning.h Makefile \\
    cc $(FLAGS) -c hilbert.c -o hilbert.o

level_data.o: turning.h Makefile \\
    cc $(FLAGS) -c level_data.c -o level_data.o

w.d.o: turning.h Makefile \\
    cc $(FLAGS) -c w.d.c -o w.d.o

moments.o: turning.h Makefile \\
    cc $(FLAGS) -c moments.c -o moments.o

extraction.o: turning.h Makefile \\
    cc $(FLAGS) -c extraction.c -o extraction.o

suncore.o: turning.h Makefile \\
    cc $(FLAGS) -c suncore.c -o suncore.o

misc.e.o: turning.h Makefile \\
    cc $(FLAGS) -c misc.e.c -o misc.e.o
```

Machinina Makefile
# include <stdio.h>
# include <math.h>
# include <strings.h>
# include <usercore.h>
#define L 2048
#define N 256
#define N2 128
#define M 127

extern double feed;
extern double rpm;
extern double r_W;
extern double r_T;
extern double a_y, f_y, phi_y;
extern double a_z, f_z, phi_z;
extern double a_o, phi_o;
extern double b, k_m, phi_m;
extern double a1, k_01;
extern double b1, k_01;
extern double a_yl, f_yl;
extern double a_zl, f_zl;

extern void choose();
extern void real_parameters();
extern void real_roundness();
extern void plot_roundness();
extern void cut_it_out();
extern void simu_parameters();
extern void simu_roundness();
extern void level_data();
extern void hilbert();
extern void wd();
extern void moments();
extern void extraction();
extern void store_wp();
extern void store_sig();

extern void set_up();
extern void clean_up();
extern char io_by_keyboard();
extern int go_on_or_not();
extern char input_string[];
extern struct vwsurf vwsurf;
extern int seg;

extern void store_re();
extern void store_mom_p();
extern void store_mom_t();
extern void store_wd();
extern void results();
extern void store_vector();
extern void plot_vector();

extern void store_wd();
#include "turning.h"

int main() {
    static double d[L];
    int middle;
    static double f[Z][N];
    static double w[Z][N][N];
    static double p[N];
    static double theta[N];
    char type_of_signal[100];
    double finished = 0;
    set_up();
    while( !finished )
    {
        set_viewport(0, 1.0, 0.55, 0.615);
        choose(type_of_signal);
        if (type_of_signal[0] == 'r')
        {
            real_parameters();
            real_roundness(type_of_signal, d);
            plot_roundness(d);
            cut_it_out(d, &middle, f);
            finished = go_on_or_not("Go Ahead?");
            if (finished) error_message("***Abort");
            store_wp(d, middle, type_of_signal);
        }
        else if (type_of_signal[0] == 's')
        {
            simu_parameters(type_of_signal);
            simu_roundness(f, type_of_signal);
        }
        else
        {
            error_message("***Wrong option!"\n);}
        level_data(f[0], N);
        set_viewport(0, 1.0, 0.68, 0.75);
        plot_vector(f[0], N, "n", "f_a[n]");
        store_ana_re(f[0], type_of_signal);
        set_viewport(0, 1.0, 0.55, 0.615);
        plot_vector(f[1], N, "n", "Imaginary part of f_a[n]");
        store_ana_im(f[1], type_of_signal);
        wd(f[0], f[1], w[0], w[1]);
        set_viewport(0, 1.0, 0.2, 0.55);
        plot_wd(w[0], "n", "k", "W[n,k]";)
        store_wd(w[0], type_of_signal);
        moments(w[0], p, theta);
        set_viewport(0, 1.0, 0.125, 0.2);
        plot_vector(p, N, "n", "p*";)
        store_mom_p(p, type_of_signal);
        set_viewport(0, 1.0, 0.05, 0.125);
        store_mom_t(theta, type_of_signal);
        set_viewport(0, 1.0, 0.0, 0.05);
        extraction(p, theta, type_of_signal);
        results(type_of_signal);
        finished = go_on_or_not("Do on ?");
        if (finished) clean_up();
    }
    go_on_or_not("Go Ahead?");
    clean_up();
}
#include "turning.h"

char type_of_signal[];
set_viewport(0.0, 1.0, 0.0, 0.75);
set_window(0.0, 1.0, 0.0, 1.0);
create_retained_segment(++seg);
move_abs_3(0.25, 0.0, 0.6);
text("Choose from:");
move_abs_3(0.25, 0.0, 0.5);
text("r----real cases");
move_abs_3(0.25, 0.0, 0.45);
text("s----simulation");
close_retained_segment();

io_by_keyboard("Enter: ", 0.25, 0.25);
strncpy(type_of_signal, input_string, 5);
delete_retained_segment(seg--);

FILE *fp, *fopen();
char infofilename[250];
strcpy(infofilename, "/home/hawk/eng/es036/source/vib/");
strcat(infofilename, input_string);
fp fopen(infofilename, "r");
fscanf(fp, "%lf", &feed);
fscanf(fp, "%lf", &rpm);
rpm /= 60.0;
fscanf(fp, "%lf", &r_w);
fscanf(fp, "%lf", &r_t);
fscanf(fp, "%lf %lf", &a_z, &f_z);
fscanf(fp, "%lf %lf", &a_y, &f_y);

a = feed * a_z / (2.0 * r_t);
b = (f_z * a_y) / (rpm * r_w);
k_o = (f_z / rpm) * 0.125;
k_m = (f_y / rpm) * 0.125;
fclose(fp);
Machining

#include "turning.h"

void real_roundness(type_of_signal, d)
char* type_of_signal;
double d[];
{
    FILE *fp, *fopen();
    char datafilename[200];
    int n;
    char str[200];
    io_by_keyboard("DataFileName:", 0.2, 0.4);
    strcpy(datafilename, "/home/hawk/eng/es036/source/vib/" why do not use the same file name?); *1
    strcat(datafilename, input_string);
    strcpy(type_of_signal, input_string);
    fp = fopen(datafilename, "r");
    fscanf(fp, "%d", str);
    for (n=0; n<; n++)
    {
        fscanf(fp, "%lf", &d[n]);
    }
    fclose(fp);
}

Machining

#include "turning.h"

void cut_it_out(d, p_middle, f)
double d[];
int *p_middle;
double f[2][N];
{
    int MAXINT = 2147483647;
    int locator_number = 1;
    int button_number = 1;
    int echo_type = 1;
    float x, y, x0, y0, x1, y1, arg;
    int i, j;
    char str[20];
    initialize_device(LOCATOR, locator_number);
    initialize_device(BUTTON, button_number);
    set_echo_surface(LOCATOR, locator_number, &vwsurf);
    set_echo_surface(BUTTON, button_number, &vwsurf);
    set_echo(LOCATOR, locator_number, echo_type);

    set_window(0.0, 1.0, 0.0, 0.75);
    create_retained_segment(++seg);
    await_any_button_get_locator_2(MAXINT, locator_number,
        &button_number, &x, &y);
    move_abs_3(x, 0.0, y);
    text("A");
    x0 = x;
    y0 = y;

    await_any_button_get_locator_2(MAXINT, locator_number,
        &button_number, &x, &y);
    move_abs_3(x, 0.0, y);
    text("B");
    x1 = x;
    y1 = y;

    x = (x0+x1)/2;
    y = (y0+y1)/2;
    move_abs_3(x, 0.0, y);
    text("C");
    move_abs_3(x0, 0.0, y0);
Machining

__cut_it_out.c__

```c
line_abs_3(x, 0.0, y);
line_abs_3(x1, 0.0, y1);
line_abs_3(x0, 0.0, y0);

x = (x1+x0)/2 - 0.5;
y = (y0+y1)/2 - 0.375;
if (x==0 && y==0) 
    arg = 0;
else if (x==0 && y<0) 
    arg = arg = -3.141596/2;
else if (x<0) 
    arg = atan(y/x) +3.141596;
else if (x<0 && y<0) 
    arg = atan(y/x) -3.141596;
else 
    arg = 0.0;

arg *= 180/3.141596;
sprintf(str, "%f", arg);
x = (x0+x1)/2;
y = (y0+y1)/2;
x = (x0.5)/2;
y = (y0.375)/2;

move_abs_3(x, 0.0, y);
text(str);
x = (x0+x1)/2;
y = (y0+y1)/2;
move_abs_3(x, 0.0, y);
line_abs_3(0.5, 0.0, 0.375);

f[0][0] = d1[i-128+i0] % L;
f[1][0] = 0;

close_retained_segment();

terminate_device(locator_number);
terminate_device(button_number);
```

Machining

__plot_roundness.c__

```c
#include "turning.h"

void plot_roundness(d)
    double d[];
{
    double low, high;
    double r, coef, x, y;
    int i;

    low = high = 0.0;
    for (i=0; i<128; i++)
    {
        low = (low < d[i]) ? low : d[i];
        high = (high > d[i]) ? high : d[i];
    }
    r = (high-low)*1.5;
    r = (r<low) ? -low : r;

    set_viewport(2.0, 0.875, 0.0, 0.75);
    set_window(-1.125*(r+high), 1.125*(r+high),
                     -1.125*(r-high), 1.125*(r+high));
    create_retained_segment(+seg);
    coef = 2.0 * 3.14159 / L ;
    x = (r + d[0]) * cos(coef*0);
    y = (r + d[0]) * sin(coef*0);
    move_abs_3(x, 0.0,y);
    for (i=0; i<L; i++)
    {
        x = (r + d[i]) * cos(coef*i);
        y = (r + d[i]) * sin(coef*i);
        line_abs_3(x, 0.0, y);
    }

    x = (r + low) * cos(coef*0);
    y = (r + low) * sin(coef*0);
    move_abs_3(x, 0.0, y);
    for (i=0; i<L; i++)
    {
        x = (r + low) * cos(coef*i);
        y = (r + low) * sin(coef*i);
        line_abs_3(x, 0.0, y);
    }

    x = r * cos(coef*0);
    y = r * sin(coef*0);
    move_abs_3(x, 0.0, y);
    for (i=0; i<L; i++)
    {
        x = r * cos(coef*i);
        y = r * sin(coef*i);
        line_abs_3(x, 0.0, y);
    }
```
Machining

```c
x = (r + high) * cos(coef);  
y = (r + high) * sin(coef);  
move_abs_3(x, 0.0, y);  
for (i=0; i<L; i+=8)  
    {  
        x = (r + high) * cos(coef);  
        y = (r + high) * sin(coef);  
        line_abs_3( x, 0.0, y);  
    }
move_abs_3(-0.1*r, 0.0, 0.0);  
text(input_string);
}
```

---

Machining

```c
#include "turning.h"

double feed, rpm, r_w, r_t;  
double f_z, a_z, phi_z;  
double f_y, a_y, phi_y;

double a, k_o, phi_o;  
double b, k_m, phi_m;

void simu_parameters(type_of_vibration)  
char *type_of_vibration;  
{
    io_by_keyboard("feed(um), rpm:", 0.2, 0.4);  
    scanf(input_string, "%lf %lf", &feed, &rpm);  
    rpm /= 60.0;
    io_by_keyboard("r_w(um), r_t(um):", 0.2, 0.4);  
    scanf(input_string, "%lf %lf", &r_w, &r_t);
    io_by_keyboard("a_y(um), f_y(Hz), phi_y(Deg):", 0.20, 0.42);  
    scanf(input_string, "%lf %lf %lf", &a_y, &f_y, &phi_y);  
    phi_y *= 3.141596/180.0;
    io_by_keyboard("a_z(um), f_z(Hz), phi_z(Deg):", 0.20, 0.38);  
    scanf(input_string, "%lf %lf %lf", &a_z, &f_z, &phi_z);  
    phi_z *= 3.141596/180.0;
    sprintf(type_of_vibration, "simu_%d_%d", (int)a_z, (int)a_y);
}
```
```c
#include "turning.h"

void simu_roundness(f, type_of_vibration)
  double f[2][N];
  char* type_of_vibration;
  double lowFrequencyPart, highFrequencyPart, mainPart;
  double theta, coef;
  int n;

  a = (feed * a_z) / (2.0 * r_t);
  b = (f_z * a_y) / (rpm * r_w);
  k_o = (f_z * (double) N) / (rpm * (double) L);
  k_m = (f_y * (double) N) / (rpm * (double) L);
  phi_o = (f_y * 3.141596 / rpm) * (double) (L-N) / (double)L;
  phi_m = (f_y * 3.141596 / rpm) * (double) (L-N) / (double)L;

  coef = (2.0 * 3.141596) / (double)N;
  for (n=0; n<N; n++)
    theta = k_o * coef * n + phi_o
            + b * sin(k_m * coef * n + phi_m);
            
    lowFrequencyPart = (feed * feed) / (8.0 * r_t);
    highFrequencyPart = (a_z * a_z) / (2 * r_t)
                         * cos(theta) * cos(theta);
    mainPart = a * cos(theta);
    f[0][n] = lowFrequencyPart + highFrequencyPart + mainPart;
    f[1][n] = 0.0;
```
/*********** FFT ***********/
c06ecf (real, imag, &1, &ifail);
/*****************************/
for (i=0; i<I; i++)
{
    f[i] = sqrt(real[i]*real[i] + imag[i]*imag[i]);
}
i_peak = 2;
peak = f[i_peak];
for (i=0; i<N2; i++)
{
    if (f[i] > peak)
    {
        peak = f[i];
        i_peak = i;
    }
}
i_low = i_peak /2;
i_high = i_peak + i_peak * 3 /2;
i_high = i_high < N2 ? i_high : N2;
for (i=0; i<N; i++)
{
    if (i < i_low || i > i_high)
    {
        real[i] = 0.0;
        imag[i] = 0.0;
    }
    else
    {
        real[i] *= 2.0;
        imag[i] *= 2.0;
    }
}
/*****************************/
/******* IFFT ***********/
c06gcf (imag, &1, &ifail);
c06ecf (real, imag, &1, &ifail);
c06gcf (imag, &1, &ifail);
/*****************************/
#include "turning.h"

void level_data(d, length)
  double d[];
  int length;
{  
  double slope, intercept, t[6];
  int n;
  
  for (n=0; n<length; n++)
  {  
    t[0] += n * n;
    t[1] += n;
    t[2] += d[n] * n;
    t[3] += n;
    t[4] += 1;
    t[5] += d[n];
  }
  slope = (t[2]*t[4] - t[1]*t[5])/(t[0]*t[4] - t[1]*t[3]);
  intercept = -(t[2]*t[3] - t[5]*t[0]) / (t[0]*t[4] - t[1]*t[3]);
  for (n=0; n<length; n++)
  {  
    d[n] = slope * n + intercept;
  }
}

void wd(freal, fimag, wreal, wimag)
  double freal[N], fimag[N];
  double wreal[N][N], wimag[N][N];
{  
  int n, m, ml, m2, k;
  int I=N, ifail=0;
  double treal, timag, arg, c, s;
  
  for (n=0; n<N; n++)
  {  
    ml = n + m -M;
    m2 = n - m +M;
    if (-1<m1 && m1<N && -1<m2 && m2<N)
    {  
      wreal[n][m] = freal[ml] * freal[m2];
      wimag[n][m] = fimag[ml] * fimag[m2];
      wreal[n][m] += freal[ml] * fimag[m2];
      wimag[n][m] += fimag[ml] * freal[m2];
    } else
    {  
      wreal[n][m] = 0;
      wimag[n][m] = 0;
    }
  }
  
  for (n=0; n<N; n++)
  {  
    c06ecf(wreal[n], wimag[n], &I, &ifail);
  }
  
  for (k=0; k<N; k++)
  {  
    arg = 2.0 * M * 3.14159265358979323846;
    arg -= k;
    arg /= N;
    c = cos(arg);
    s = sin(arg);
    for (n=0; n<N; n++)
    {  
      treal = wreal[n][k] * sqrt((double)N); /* nag */
      timag = wimag[n][k] * sqrt((double)N); /* nag */
      wreal[n][k] = treal * c - timag * s;
    }
  }

/*****************************/
/* wd.c */
/*****************************/

#include "turning.h"

void wd(freal, fimag, wreal, wimag)
  double freal[N], fimag[N];
  double wreal[N][N], wimag[N][N];
{  
  int n, m, ml, m2, k;
  int I=N, ifail=0;
  double treal, timag, arg, c, s;
  
  for (n=0; n<N; n++)
  {  
    ml = n + m -M;
    m2 = n - m +M;
    if (-1<m1 && m1<N && -1<m2 && m2<N)
    {  
      wreal[n][m] = freal[ml] * freal[m2];
      wimag[n][m] = fimag[ml] * fimag[m2];
      wreal[n][m] += freal[ml] * fimag[m2];
      wimag[n][m] += fimag[ml] * freal[m2];
    } else
    {  
      wreal[n][m] = 0;
      wimag[n][m] = 0;
    }
  }
  
  for (n=0; n<N; n++)
  {  
    c06ecf(wreal[n], wimag[n], &I, &ifail);
  }
  
  for (k=0; k<N; k++)
  {  
    arg = 2.0 * M * 3.14159265358979323846;
    arg -= k;
    arg /= N;
    c = cos(arg);
    s = sin(arg);
    for (n=0; n<N; n++)
    {  
      treal = wreal[n][k] * sqrt((double)N); /* nag */
      timag = wimag[n][k] * sqrt((double)N); /* nag */
      wreal[n][k] = treal * c - timag * s;
    }
  }

/*****************************/
/* wd.c */
/*****************************/
```c
#include "turning.h"

void moments(w, p, theta)
double w[N][N];
double p[N];
double theta[N];
{
    double x[N], y[N], coef;
    int n, k;
    
    /************************************************************************
    for (n=0; n<N; n++)
    {
        p[n] = 0;
        for (k=0; k<N; k++)
        {
            p[n] += w[n][k];
        }
        p[n] /= 2.0*N;
    }
    p[0] = 0; /* for the sake of plot */
    /************************************************************************
    coef = 2.0 * 3.141596 /N;
    for (n=0; n<N; n++)
    {
        x[n] = y[n] = 0.0;
        for (k=0; k<N; k++)
        {
            x[n] += w[n][k] * cos(coef*k);
            y[n] += w[n][k] * sin(coef*k);
        }
    }
    for (n=0; n<N; n++)
    {
        if (y[n] == 0 && x[n] > 0)
        {
            theta[n] = 0;
        } else if (y[n] == 0 && x[n] < 0)
        {
            theta[n] = 3.141596;
        } else if (y[n] == 0 && x[n] == 0)
        {
            theta[n] = 0;
        } else if (x[n] < 0)
        {
            theta[n] = atan(y[n]/x[n]) + 3.141596;
        } else if (x[n] == 0)
        {
            theta[n] = 0;
        }
    }
    return;
}
```
moments.c

```c
theta[n] = 3.141596/2.0;
else if (y[n] < 0)
    theta[n] = 3.0*3.141596/2.0;
else
    theta[n] = 0;
}
else
    theta[n] = atan(y[n]/x[n]);

theta[n] /= 2.0;
```

extraction.c

```c
#include "turning.h"

double a1, k_ol;
double b1, k_ml;

void extraction(p, theta, type_of_vibration)
p[N];
theta[N];
type_of_vibration[];

double y[N];
int I-N, ifail = 0;
int n, k;

double coef;
double tmp;
char str[120];

al = 0;
for (n=0; n<N; n++)
    al += p[n];
al /= N;

al = sqrt(al);

k_ol = 0.0;
for (n=0; n<N; n++)
    k_ol += theta[n];
k_ol /= (2.0 * 3.141596);
k_ol = (int)(k_ol + 0.5);

tmp = 0.0;
for (n=0; n<N; n++)
    tmp += theta[n];
tmp /= N;
```
Machinina

```c
{ 
    theta[n] -= tmp;
}

coef = 2.0 * 3.141596/N;
for (n=0; n<N; n++)
{ 
    theta[n] *= 0.5*(1-cos(coef*n));
    y[n] = 0.0;
}

/**********************************************************************
co6ecf_(theta, y, 4, 4, ifail);
coef = sqrt(N+0.0);
for (n=0; n<N; n++)
{ 
    theta[n] *= coef;
    y[n] *= coef;
}

for (k=0; k<N/2; k++)
{ 
    if (theta[k] > bl)
        bl = theta[k];
    k_m1 = k;
}

k_m1 = 2;
b1 = theta[2];
for (k=2; k<N/2; k++)
{ 
    k += theta[k-1] + theta[k-2] + theta[k+1] + theta[k+2];
    bl /= sin(2.0*3.141596*(k_m1+0.0)/(N+0.0));
    b1 *= 2.0/N;
    bl *= 2.0;
}

a_z = 2.0 * r * a / feed;
f_zl = k_ol * rpm * (double) L / (double) N;

a_yl = bl / (f_zl / (rpm * r_w));

set_window(-0.1, 1.1, -0.1, 1.1);
create_retained_segment(++)
move_abs(0.25, 0.0, 0.9); 
```
```c
#include "turning.h"

extern int pixwinfd();

struct vwsurf DEFAULT_VWSURF(pixwinfd);

int keyboard_number = 1;
char input_string[80];
int buffer_size = 80;
char initial_string[80] = {"enter:"};
int initial_cursor_position = 7;
double echo_x, echo_y;
int echo_type = 1;
int length;
int seg = 0;

void set_up()
{
    initialize_core(DYNAMICC, SYNCHRONOUS, THREED);
    initialize_view_surface(&vwsurf, FALSE);
    select_view_surface(&vwsurf);
    set_view_reference_point(0.0, 0.0, 0.0);
    set_view_plane_normal(0.0, 1.0, 0.0);
    set_projection(PARALLEL, 1.0, 1.732, -1.0);
    set_up_3D(0.0, 0.0, 0.1.0);
    initialize_device(KEYBOARD, keyboard_number);
    set_echo(KEYBOARD, keyboard_number, echo_type);
    set_echo_surface(KEYBOARD, keyboard_number, &vwsurf);
}

void clean_up()
{
    terminate_device(KEYBOARD, keyboard_number);
    deselect_view_surface(&vwsurf);
    terminate_core();
}
```

```c
int go_on_or_not(text)
{
    char* io_by_keyboard("", 0.2, 0.4);
    while( seg > 0 )
    { delete_retained_segment(seg--);
    }
    if io_by_keyboard(text, 0.2, 0.4); 
    
    io_by_keyboard(text, 0.2, 0.4); 
    if (input_string[0] == 'n')
        return 1;
    else if (input_string[0] == 'N')
        return 0;
    else
        return 0;
}
```
Machining

suncore.c

/*****************************/

Machining

misc.c

#include "turning.h"

void results(type)
char type[];
{
    FILE *fp, *fopen();
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type);
    strcat(file_name, ".res");
    fp = fopen(file_name, "w");
    fprintf(fp, "\caption{Original:\}
    fprintf(fp, "Sa = %.lf, f = %.lf, a, f_z); 
    fprintf(fp, "Sa_y = %.lf, f_y = %.lf, a_y, f_y); 
    fprintf(fp, "Extracted:\}
    fprintf(fp, "Sa_z %.lf, f_z %.lf, a_zl, f_zl); 
    fprintf(fp, "Sa_y %.lf, f_y %.lf, a_yl, f_yl); 
    fprintf(fp, "\}"
    fclose(fp);
}

void store_wp(d, middle, datafilename)
double d[];
int middle;
char datafilename[];
{
    FILE *fp, *fopen();
    char workpiece_filename[100];
    int n;
    strcpy(workpiece_filename, "Data/");
    strcat(workpiece_filename, datafilename);
    strcat(workpiece_filename, ".wp");
    fp = fopen(workpiece_filename, "w");
    fprintf(fp, "%12d\n", middle);
    for(n=0; n<=n++;)
       printf(fp, "%12.4f\n", d[n]);
}
Machinina

```c
void store_sig(v, type)
double v[];
char type[];
{
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type);
    strcat(file_name, ".sig");
    store_vector(v, N, file_name);
}

void store_ana_re(v, type)
double v[];
char type[];
{
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type);
    strcat(file-name, ".ana.re");
    store_vector(v, N, file_name);
}

void store_ana_im(v, type)
double v[];
char type[];
{
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type);
    strcat(file-name, ".ana.im");
    store_vector(v, N, file_name);
}

void store_mom_p(v, type)
double v[];
char type[];
{
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type);
    strcat(file-name, ".mom.p");
    store_vector(v, N, file_name);
}

void store_mom_t(v, type)
double v[];
char type[];
{
    char file_name[100];
    strcpy(file_name, "Data/");
    strcat(file_name, type);
    strcat(file-name, ".mom.t");
    store_vector(v, N, file_name);
}

void store_wd(w, type)
double w[N][N];
char type[];
{
    char file_name[100];
    FILE *fp = fopen();
    int n, k;
    strcpy(file_name, "Data/");
    strcat(file_name, type);
    strcat(file-name, ".wd");
    fp = fopen(file_name, "w");
    for (k=N; k < N+N2; k += 4)
        for (n=0; n < N; n += 4)
            fprintf(fp, "%f\n", w[n][k]);
    fclose(fp);
}
```

Machinina

```c
fclose(fp);
```
Machining misce.c

```c
void store_vector(v, length, file_name)
  double v[];
  int length;
  char file_name[];
  {
    FILE *fp, *fopen();
    int i;
    fp = fopen(file_name, "w");
    for (i=0; i < length; i++)
    { fprintf(fp, "%f\n", v[i]);
    }
    fclose(fp);
  }
void plot_vector(v, length, xlabel, ylabel)
  double v[];
  int length;
  char xlabel[], ylabel[];
  {
    double min, max;
    int i;
    min = max = v[0];
    for (i=0; i < length; i++)
    { if (min > v[i])
        { min = v[i];
        }
      if (max < v[i])
        { max = v[i];
        }
    }
    set_window(-0.1*length, 1.1*length,
               -0.1*(max-min), max+0.1*(max-min));
    create_retained_segment( ++seg );
    { move_abs_3((double)0.0, 0.0, 0.0);
      text(xlabel);
      move_abs_3(0.0, 0.0, max+0.05*(max-min));
      text(ylabel);
      close_retained_segment();
    }
    plot_vectors(w, xlabel, ylabel, zlabel)
      w[][N][N];
      xlabel[], ylabel[];
      double min, max, scale, shift;
      int n, k;
      min = max = w[0][0];
      for (n=0; n<N; n++)
      { for (k=0; k<N; k++)
        { min = (w[n][k]<min) ? w[n][k] : min;
          max = (w[n][k]>max) ? w[n][k] : max;
        }
      }
      scale = 1.0 / (max - min);
      shift = -(min) * scale;
      set_window(-0.16, 1.7,
                 -0.16, 1.7);
      create_retained_segment( ++seg );
      move_abs_3((double)0.0, 0.0, shift);
      line_abs_3(0.0, 0.0, shift);
      move_abs_3(0.0, 0.0, shift);
      line_abs_3(0.0, 1.0, shift);
      move_abs_3(0.0, 0.0, shift);
      line_abs_3(0.0, 0.0, shift*max*scale);
      for (n=0; n < N; n += 5)
      { move_abs_3((double)n/N), 0.0, w[n][0]*scale+shift ;
        for (k=0; k<N; k += 5)
        { line_abs_3((double)n/N),k/(double)N, w[n][k]*scale+shift; 
        }
      }
      move_abs_3(1.05, 0.0, shift);
      text(xlabel);
      move_abs_3(0.0, 1.0, shift+max*scale*0.2);
      text(ylabel);
      move_abs_3(0.0,0.0, max*scale + shift);
  }
```
Machinina

void char {
  text(zlabel);
  close_retained_segment();
  /**************************************************
}

void error_message(text)
char text[];
{
  void exit();
  fprintf(stderr, "ERROR: ");
  fprintf(stderr, "\e\n", text);
  exit(1);
}