

# TUTORIAL OF THE TOOLBOX

---

This file contains one document to help the user to understand the algorithms developed in this toolbox and how to use it.

The document is:

B. Boashash and A.P. Reilly, "**Algorithms for Time-Frequency Signal Analysis**", Chapter 7, pp. 163-181, in B. Boashash, editor, "Time Frequency Signal Analysis - Methods and Applications", Longman Cheshire, 1992.

A second document which can help the user is (not included):

B. Boashash, "**Getting Started With A Practical And Efficient Time-Frequency Toolbox TFSAP-7.0**", in B. Boashash, Ed, Time-Frequency Signal Analysis & Processing: A Comprehensive Reference 2nd Ed, chapter 17, pp. 967–988, Elsevier, Academic Press, 2015.

Published in Australia, New Zealand, the Pacific, Asia and Africa by  
Longman Cheshire Pty Limited  
Longman House  
Kings Gardens  
95 Coventry Street  
Melbourne 3205 Australia  
ISBN 0 582 71286 6

Offices in Sydney, Brisbane, Adelaide and Perth. Associated companies,  
branches and representatives throughout the world.

Copublished in the Western Hemisphere, United Kingdom  
and Europe by Halsted Press: an Imprint of  
John Wiley & Sons, Inc.  
New York Toronto Chichester

Copyright © Longman Cheshire 1992  
First published 1992

All rights reserved. Except under the conditions described in the Copyright Act  
1968 of Australia and subsequent amendments, no part of this publication may  
be reproduced, stored in a retrieval system or transmitted in any form or by any  
means, electronic, mechanical, photocopying, recording or otherwise, without  
the prior permission of the copyright owner.

Designed by Nadia Graziotto  
Printed in Hong Kong

National Library of Australia  
Cataloguing-in-Publication data

Time-frequency signal analysis.  
Bibliography.  
Includes index.  
ISBN 0-582-71286-6  
1. Signal processing. I. Boashash, Boualem.  
621.38223

Library of Congress Cataloguing-in-Publication data


Time-frequency signal analysis-methods and applications/edited by Boualem  
Boashash.  
p. cm.  
Includes bibliographical references and index.  
ISBN 0-470-21821-5  
1. Signal processing. I. Boashash, Boualem.  
YK5102.T585 1992  
621.382'2-dc20

91-32658  
CIP

# TIME - FREQUENCY SIGNAL ANALYSIS

## METHODS AND APPLICATIONS

Edited by Boualem Boashash

  
Longman Cheshire  

---

WILEY HALSTED  
PRESS

- [53] Wahba, G., "Practical Approximate Solutions to Linear Operator Equations when the Data are Noisy", *SIAM J. Numer. Anal.*, Volume 14, Number 4, September 1977, pp. 651-667.
- [54] Wahba, G., "Smoothing and Ill-Posed Problems", in M. A. Golberg, editor, *Solution Methods for Integral Equations*, Plenum Press, 1978, pp. 183-194.
- [55] West, G., "Eine schnelle Mellin-Transformation", *Computing*, Volume 33, pp. 237-245, 1984.
- [56] Whitehouse, H. J. and Speiser, J. M., "Algorithms and Architectures for Array Signal Processing", Presented at the *Indo-U.S. Workshop on Systems and Signal Processing*, Bangalore, India, January 8-12, 1988. The conference was sponsored by the U.S. Office of Naval Research, the Indian Department of Science and Technology, and the IEEE Bangalore chapter. A book of the extended abstracts of the conference papers is available, and a conference proceedings is planned.
- [57] Whitehouse, H. J. Boashash, B. and Speiser, J. M., "High Resolution Processing Techniques for Temporal and Spatial Signals", Presented at the Workshop on High Resolution Methods for Underwater Acoustics, organized by GRETSI, Juan les Pins, France, June 16, 1989. To appear as a chapter in a volume edited by M. Bouvet and G. Bienvenu, in the Springer-Verlag *Lecture Notes in Computer Science*.
- [58] Wiener, Norbert, *The Fourier Integral and Certain of its Applications*, Dover Publications Inc., New York, p.3.
- [59] Wiener, Norbert, *Generalized Harmonic Analysis and Tauberian Theorems*, The M.I.T. Press, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1964.
- [60] Wilcox, C. H. "The Synthesis Problem for Radar Ambiguity Functions", Mathematics Research Center, MRC Technical Summary Report No.157, The University of Wisconsin, April 1960.
- [61] Zwick, Philip E. and Imre Kiss, "A New Implementation of the Mellin Transform and its Application to Radar Classification of Ships" *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Volume PAMI-5, Number 2, pages 191-199, 1983.

## Chapter 7

### Algorithms for Time-Frequency Signal Analysis

Boualem Boashash and Andrew Reilly

**Keywords:** fast algorithms, efficient implementation, Fortran subroutine, optimizations, analytic signal, Cohen's class, Wigner-Ville distribution

#### 1 Introduction

This chapter presents algorithms which implement time-frequency signal analysis techniques on computer systems. Fortran code fragments are included.

A generalized framework for time-frequency distribution calculation is provided, and a number of speed-up optimizations are described. Algorithms are presented for the calculation of all of the time-frequency representations listed in table 7.1.

Most popular time-frequency representations can be expressed in terms of the general bilinear time-frequency distribution representation proposed by L. Cohen [6] (see also chapter 1 in this book). This is shown in equation 1 below:\*

$$\rho_z(t, f) = \iint \int e^{j2\pi v(u-t)} g(v, \tau) z(u + \frac{\tau}{2}) z^*(u - \frac{\tau}{2}) e^{-j2\pi f\tau} dv d\tau \quad (1)$$

Here  $z$  is an *analytic* signal, (see [11] i.e., a complex signal which contains no negative frequencies. The function  $g(v, \tau)$  determines the characteristics of the time-frequency distribution. For example if  $g(v, \tau) = 1$  then the distribution formed is the Wigner-Ville

distribution. If the integration with respect to  $v$  is performed, then the equation becomes:

$$\rho_z(t, f) = \iint G(t, \tau) z(u + \frac{\tau}{2}) z^*(u - \frac{\tau}{2}) e^{-j2\pi f\tau} du d\tau \quad (2)$$

This distribution ( $\rho_z$ ) is related by Fourier transforms to the time-lag representation  $R_z(t, \tau)$ , the Doppler-frequency representation  $r_z(v, f)$ , and the Doppler-delay representation,  $A_z(v, \tau)$  as shown in figure 7.1.

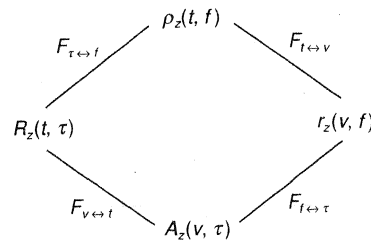


Figure 7.1. Relationships between continuous time representations.

The equivalent discrete time relationships, related by discrete Fourier transforms are  $R_z(n, m)$ ,  $\rho_z(n, k)$ ,  $r_z(l, k)$ ,  $A_z(l, m)$ , as shown in figure 7.2.

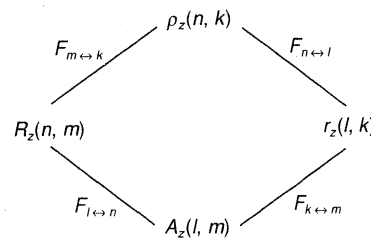


Figure 7.2. Relationships between discrete time representations.

The expression to be used for discrete time implementation of these transforms is the discrete time equivalent of equation 2, and is shown in equation 3.

$$\rho_z(n, k) = \sum_{m=-M}^M \sum_{p=-M}^M G(p, m) z(n+p+m) z^*(n+p-m) e^{-j4\pi mk/N} \quad (3)$$

This can also be expressed as:

$$\rho_z(n, k) = \mathcal{F}_{m \rightarrow k} [G(n, m) \underset{(n)}{*} K_z(n, m)] \quad (4)$$

where  $\underset{(n)}{*}$  denotes the discrete time convolution, and  $\mathcal{F}_{m \rightarrow k}$  denotes the discrete Fourier transform from time ( $n$ ) to frequency ( $k$ ).

## 2 Analytic signal calculation

In most practical cases, the signals to be analysed consist only of real values. In these cases it is necessary, as a first step towards signal analysis, to form the corresponding analytic signal, as explained in [3].

The direct method of producing the analytic signal is to use its definition: a signal with no negative frequency components. That is, form the Fourier transform of the signal, set the negative frequency values to zero, and perform the inverse Fourier transform. Unfortunately, finite data length effects (windowing) cause this method to produce undesirable ripples in the signal. Setting the negative frequency components to zero has the same effect in the time domain as any naive frequency domain filtering operation: the resulting response has sizable ripples for any non-harmonic frequencies.

Another method is to use a Hilbert transform filter to produce the required complex component of the signal, which when added to the real part produces the desired analytic signal. This is shown in equation 5.

$$z(n) = x(n) + jH[x(n)] \quad (5)$$

where  $H[\ ]$  is the Hilbert transform [11], and has the ideal impulse response shown in equation 6.

$$h(n) = \begin{cases} \frac{2 \sin^2(\pi n/2)}{\pi n} & \text{for } n \neq 0 \\ 0, & \text{for } n = 0 \end{cases} \quad (6)$$

The Hilbert transform can be implemented as a finite-impulse response (FIR) digital filter, which may be more efficient than the Fourier technique, depending on the length of the signal to be transformed, and the length of the filter impulse response used. Although an ideal Hilbert transform filter has an infinite impulse response length, in practise an FIR filter length of 79 samples has been shown to provide an adequate approximation[4].

This filter could be a rectangular-windowed version of the infinite length filter, or it could be an optimal implementation calculated using one of the filter design algorithms, such as the Remez-exchange algorithm. This produces a filter which is optimal in a minimax or Chebyshev sense [11].

### 3 General approach to computation of TFDs

The discrete-time definition of Cohen's class of time-frequency distributions given in equation 3 forms the basis of the general approach to implementation of time-frequency distributions. This approach can be expanded into three steps:

1. Form the bilinear product  $K_z(n, m) = z(n+m)z^*(n-m)$ . This is known as the 'bilinear kernel' or just 'kernel' where meaning is obvious.
2. Convolve the kernel with the desired determining function  $G(n, m)$  in the  $n$  (time) dimension.
3. Calculate the discrete Fourier transform of this result, to produce the time slice of the desired distribution.

These steps will now be elaborated on, and issues arising from implementation on computing systems mentioned. The next section, section 4, presents an example of some of the optimizations that can be applied to these techniques for a specific example of Cohen's class, the Wigner-Ville distribution.

#### 3.1 Calculation of the bilinear kernel

It can be easily shown that the bilinear kernel has Hermitian symmetry:

$$K_z(n, m) = \begin{cases} z(n+m)z^*(n-m) & \text{for } m \geq 0 \\ K_z^*(n, -m) & \text{for } m < 0 \end{cases} \quad (7)$$

This means that values of the kernel need only be calculated for positive time lags.

It is normally not necessary to store the calculated kernel values in an array, as each will only be used once. (Of course there are exceptions to this rule, such as when a sliding analysis window is used, with a large overlap between successive windows.) Since the kernel array would be rectangular, twice as long (in the  $n$  direction) as wide (the  $m$  direction), storage requirements for the kernel would increase with the square of the problem size. In a finite storage system this will reduce the useful analysis window length.

The alternative is to calculate the kernel values as they are required, and this is the way the code in section 3.4 works. This code includes the concurrent convolution with the kernel specification function,  $G(n, m)$ , which will be described in section 3.2 below, and so is best studied after that section has been read.

#### 3.2 Convolution in the $n$ (time) direction

This is simply the matrix multiplication of the weighting function values  $G(n, m)$  with the bilinear kernel values  $K_z(n, m)$ . This must be performed for each time instant for which output is required.

In an actual implementation, either the kernel matrix or the selection function matrix may be calculated at the point of use (in the convolution). This can save memory space and sometimes time. If the kernel values are only going to be used once, then a considerable memory saving can be made by not storing them at all (remember, they are complex, so would require a very large array to store). Conversely, if the selection function is trivial then this need not be stored either.

Since the selection function  $G(n, m)$  is  $\delta(n)$  for the Wigner-Ville distribution, the convolution step does not really occur at all.

Most common bilinear transforms exhibit symmetry in both the  $m$  and  $n$  directions, and this fact can be used to reduce both computation and storage requirements. In the code fragment shown in section 3.4, this symmetry is used by multiplying the same  $G(n, m)$  value with the required two bilinear kernel values at the same time. The delay axis symmetry of  $G$  and the corresponding kernel Hermitian symmetry allow the upper half of the result matrix  $R$  to be calculated from the lower half through conjugation, rather than direct calculation. Care must be taken that symmetry exists *around* the  $R(1)$  element, rather than including it. That is,  $R(\text{FFTLLEN}) = \text{conjg}(R(2))$ , not  $\text{conjg}(R(1))$ , where  $\text{FFTLLEN}$  is the data length of the Fourier transform to be used.

#### 3.3 Discrete Fourier transform

The discrete Fourier transform of each time-slice of the distribution is calculated using a fast-Fourier transform routine (FFT). Examples of this can be found in [11]. The most important point to note is that the use of one of the fast algorithms puts restrictions on the length of each data segment (usually to a power of two). This may require zero-padding of the filtered kernel before performing the transform. Another point to note is that these routines operate only on positive time and frequency values, rather than the positive and negative values expected by simple translation from the continuous domain. That is, rather than representing frequencies from  $-\frac{1}{2}f_s$  to  $\frac{1}{2}f_s$ , with the zero frequency value appearing in the middle of the array, frequencies from 0 to  $f_s$  are used, with the zero frequency value appearing as the first element in the array. This has absolutely no effect on the algorithms or calculations, due to the cyclic nature of the discrete Fourier transform, but does affect the way values are referred to. The 'negative' frequency values are stored at the 'top' of the array, and are usually referenced

with indices like: FFTLEN+1-i.

The FFT routine referred to in the code fragments in this chapter performs calculations in place, and requires both length and order parameters:

```

subroutine FFT(A,mf,FFTLEN)
c   A is an array of complex
c   mf is the calculation order, FFTLEN=2**mf
c   FFTLEN is the length of the array, a power of 2

```

The code to find the next power of two greater than or equal to the length of a signal is a simple loop:

```

c   calculate FFTLEN, the minimum (2**mf) .ge. lwin
mf=0
FFTLEN=1
6   if (FFTLEN.ge.lwin) goto 7
    mf=mf+1
    FFTLEN=FFTLEN+FFTLEN
    goto 6
7   continue

```

### 3.4 Arbitrary transform code fragment

This routine performs the convolution of the bilinear kernel with the distribution specification matrix G, for a signal window of length  $2 \times \text{hlf} + 1$ , centered on element 0 (see below). It then takes the Fourier transform of the result, producing one time-slice of the distribution.

G is known to be symmetrical in both time and lag dimensions, and so values are only used from the positive quadrant.

It is also known that the resulting spectrum is real, so the result of the convolution must have Hermitian symmetry. Thus half of the R() values are computed directly, the others from symmetry. (R is the array used to store the kernel products for different lags.)

Finally, it is known that the z array origin is not situated at the beginning of the signal array, but is at least  $2 \times \text{hlf}$  from either end. Thus negative array indices may be used, and a substantial complexity reduction may be realized, as no special cases are introduced at the beginning and end of the signal (see section 3.5).

```

subroutine transform(z,G,FFTLEN,mf,hlf,KERNMAX,R)
complex z(1), R(1)

```

```

real G(KERNMAX,KERNMAX)
integer FFTLEN,mf,hlf,KERNMAX,m,p

do 1 m=0,hlf
    R(m+1)=G(1,m+1)*z(m)*conjg(z(-m))
    do 2 p=1,hlf
        R(m+1)=R(m+1) +G(p+1,m+1)* (
1          z(p+m)*conjg(z(p-m)) + z(-p+m)*conjg(z(-p-m))
2          )
        if (m.gt.0) R(FFTLEN-m+1)=conjg(R(m+1))
1    continue

c make sure any points not containing calculated values are set
c to zero, as the FFT length may be greater than the window:

do 3 m=hlf+2,FFTLEN-hlf
3    R(m)=(0.0,0.0)

call FFT(R,mf,FFTLEN)
return
end

```

### 3.5 Notes on implementation techniques

The code presented to perform the convolution with the kernel, in section 3.4 used negative indices into the signal array z, which may seem strange, and may not work with a compiler that generates code to check array bounds, but is more efficient than the alternatives. The same effect can be achieved (a little less efficiently) by passing the whole of the signal (z) array, and also an index i to indicate the center of the analysis window, which must then be added into all references to z. That is, the first line of code inside the outer loop in section 3.4 would change from:

```

R(m+1)=G(1,m+1)*z(m)*conjg(z(-m))
to:
R(m+1)=G(1,m+1)*z(i+m)*conjg(z(i-m))

```

Both implementation techniques have a problem when the analysis window is not wholly contained within the extent of the signal, as in that case some of the values used in calculations must be zero. There are a number of ways to solve this problem:

- The obvious solution is to check each array index before accessing the value, and return zero if the access is outside the array bounds. This is inefficient, since it requires a lot of extra work in the inner loops of the convolution.
- Another technique is to move the range checking outside the loops, so that the indices never exceed the array bounds. This can be quite efficient, especially where the analysis window is very long. In that case the time required to do the checking, and the reduced symmetry available may be offset by the reduction in the number of computations required at the ends of the signal.
- The technique used here is to pad the signal in the signal array with sufficient zero values that the array will not be indexed out of bounds. Thus the correct windowing effect is achieved, without any bounds checking within the convolution loops. Here the reduced overhead in the 'working' loops makes up for the extra 'dead' iterations involving computation with zero values.

This approach is shown in the code fragment below, extracted from a Choi-Williams distribution program. It reads the data into the signal array, converts the signal to an analytic signal (with a call to SIGANA), and then produces `nplts` spectra using an analysis window of length `lwin` centered points with a time separation of `res` samples.

```

c      make hlf max (2*hlf+1 .le. lwin)
      hlf=(lwin+1)/2-1

c      read the signal
c      this is where the zero-padding buffers are built at either
c      end...
      do 10 i=1,LWINMAX-1
          z(i)=(0.0,0.0)
10      z(n+LWINMAX+i)=(0.0,0.0)
      do 5 i=LWINMAX,LWINMAX+n-1
          read (1,*) x
5          z(i)=cmplx(x,0.0)

c      form the analytic signal: (and get n1 as power of 2 .ge. n)
      call SIGANA(n,z(LWINMAX),.false.,n1)

      do 8 ii=0,nplts-1
          t=ii*res+hlf+LWINMAX
          call transform(z(t),G,FFTLEN,mf,hlf,KERNMAX,choi)
      do 9 i=1,FFTLEN

```

```

9          write(2,*) real(choi(i))
8          continue

```

**Table 7.1** Some TFDs and their determining functions  $G(n, m)$

Time-Frequency Representation	$G(n, m)$
Windowed Discrete WVD	$\delta(n) \quad m \in [-\frac{(M-1)}{2}, \frac{(M-1)}{2}]$ 0 otherwise
Smoothed WVD using a rectangular window of odd length P	$\frac{1}{P} \quad n \in [-\frac{(P-1)}{2}, \frac{(P-1)}{2}]$ 0 otherwise
Rihaczek-Margenau	$\frac{1}{2} [\delta(n+m) + \delta(n-m)]$
STFT using a Rectangular Window of odd length P.	$\frac{1}{P} \quad  m+n  \leq \frac{(P-1)}{2}$ 0 otherwise
Born-Jordan-Cohen	$\frac{1}{ m +1} \quad  m  \leq  n $ 0 otherwise
Choi-Williams (parameter $\sigma$ )	$\frac{\sqrt{\sigma/\pi}}{2m} e^{-\sigma n^2/4m^2}$

### 3.6 Code fragments to generate $G(n, m)$

This section shows example code fragments to generate the distribution specification matrix  $G(n, m)$  for the functions specified in table 7.1.

Note that for many of these distributions, this computation technique of convolving the bilinear kernel with the  $G(n, m)$  array is obviously inefficient, because most of the values of  $G(n, m)$  are zero, and the others are a constant (e.g., Wigner-Ville distribution, smoothed Wigner-Ville distribution, Rihaczek-Margenau, and STFT). For these distributions, algorithm improvements yield much higher performance. The Wigner-Ville distribution is the only one for which such an improvement will be demonstrated.

**G(n,m) for Choi-Williams distribution**

```

c      calculate the distribution specification matrix "G"
do 2 i=0,hlf
2      G(i+1,1)=0
      G(1,1)=1.0

      do 4 j=1,hlf
        wt=0.0
        do 3 i=0,hlf
          G(i+1,j+1)= exp(-(sigma*i*i)/(4*j*j))
3          wt= wt+2*G(i+1,j+1)
        wt=wt-G(1,j+1)
c      normalize array so that we know  $\sum_n G(n,j) = 1.0$ 
        do 4 i=0,hlf
4          G(i+1,j+1)=G(i+1,j+1)/wt

```

In this routine, the variable 'wt' is used to ensure that the sum along every constant 'j' (parallel to the time axis) is exactly one, a condition necessary to preserve the marginals of the distribution, as described in [2], Property 2-7. These are repeated here:

$$\sum_n \rho_z(n, k) = |Z(k)|^2$$

$$\sum_k \rho_z(n, k) = |z(n)|^2$$

This is obtained if

$$G(n, 0) = \delta(n) \text{ and } \sum_n G(n, m) = 1$$

This condition would not necessarily be met otherwise, due to finite precision in the calculations. It should be noted that the Choi-Williams distribution selection function:

$$G(n, m) = \frac{\sqrt{\sigma/\pi}}{2m} e^{-\sigma n^2/4m^2}$$

does not meet these criteria, and does not hold for  $m = 0$ , at which point the rule  $G(n, 0) = \delta(n)$  must be imposed.

**G(n,m) for Born-Jordan-Cohen distribution**

```

c      calculate the distribution specification matrix "G"
do 4 i=0,hlf

```

```

      wt=1.0/(real(i)+1.0)
      do 2 j=0,i
2          G(i+1,j+1)= wt
      do 3 j=i+1,hlf
3          G(i+1,j+1)= 0.0
4          continue

```

**G(n,m) for STFT**

```

c      calculate the distribution specification matrix "G"
wt=1.0/(real(2*hlf-1))
do 4 i=0,hlf
      do 2 j=0,hlf-i
2          G(i+1,j+1)= wt
      do 3 j=hlf-i+1,hlf
3          G(i+1,j+1)= 0.0
4          continue

```

**G(n,m) for Rihaczek-Margenau distribution**

```

c      calculate the distribution specification matrix "G"
do 2 i=0,hlf
      do 2 j=0,hlf
2          G(i+1,j+1)= 0.0
      do 3 i=1,hlf
3          G(i+1,i+1)= 0.5
      G(1,1)=1.0

```

**G(n,m) for smoothed WVD**

Note: p is half the length of the smoothing region, calculated in a similar manner to the calculation of hlf from lwin.

```

c      calculate the distribution specification matrix "G"
wt=1.0/real(2*p-1)
do 4 i=0,hlf
      do 2 j=0,p-1
2          G(i+1,j+1)= wt
      do 3 j=p,hlf
3          G(i+1,j+1)= 0.0
4          continue

```



### 3.7 Effect of windowing

Time-windowed distributions will often be calculated in preference to full-length distributions. Two reasons for this practice are:

- To reduce unnecessary computation, when it is known that the signals of interest have a limited time-extent within the data collected; or
- When the data stream is continuous (real-time analysis). For this case, it is impractical to collect all of the data before analysis, as results are required on early sections before later sections have been collected.

Instead of performing all of the operations on the entire signal, only sections of the signal are read into the data arrays at once, displacing previous values, and the calculations are repeated for each 'window'. If the windows are overlapped, then some data must be retained, and shifted forward within the data arrays by an amount equal to the time-displacement of the window.

## 4 Wigner-Ville distribution implementation

Although covered in conjunction with the other bilinear time-frequency distributions in the previous section, the implementation of the Wigner-Ville distribution will be presented again here, as an example of how to optimize the general calculation for special cases.

There are a number of optimizations applicable to the calculation of the Wigner-Ville distribution (some of these also apply to the calculation of other TFDs):

1. The array  $G(n, m)$  is trivial, and need not be stored or calculated. Instead, its effect can be incorporated into the algorithm.
2. The bilinear kernel has Hermitian symmetry, and so only values for positive lags need be calculated (This applies for all kernels.)
3. Two result spectra can be calculated by each Fourier transform. This is possible because the spectra are known to be real, the Fourier transform is known to be a linear operator, and the fast Fourier transform routine being used operates exclusively on complex data.

The best way to use this optimization is to multiply the second set of lags by  $j$ , so that the resulting spectrum-slice will appear in the imaginary part of the Fourier transform result. That is:

$$r(m) = r_1(m) + jr_2(m) \xrightarrow{\mathcal{F}_{m \leftrightarrow k}} R(k) = R_1(k) + jR_2(k)$$

4. Where the distribution will be calculated for every time value, there is another optimization, presented by Eilouti and Khadra [8]. Here a recursive technique is used to calculate the analytic signal for successive windows. This will not be described here.
5. The formulation of the discrete Wigner-Ville distribution (and all of the other distributions in Cohen's class) presented here produces a result with a frequency scaling of  $f = \frac{f_s k}{2M}$ , where other techniques, such as the short-time Fourier transform result in a scaling of  $f = \frac{f_s k}{M}$ . The consequence of this is that the resulting transforms are twice as long as one might expect. To overcome this Sun, Li, Sekhar and Sciabassi [12] have shown how to produce a transform with the usual length, and a saving of approximately half the computations. Since this technique requires the replacement of the fast Fourier transform with their own 'fast Fourier transform in part' (FFTP), it will not be described further here.

Only the code for the outer loop and the transform routine will be presented. It is assumed that the signal array 'z' and input and output files have been attended to by the surrounding code in a similar fashion to that presented in section 3.5.

```

nplts=n/res
c
c      since we're using the 'do two transforms at once'
c      optimization, nplts must be even, make it so (possibly
c      at the expense of the last plot:
c
nplts=(nplts/2)*2

c      Here is the outer loop of the actual distribution calculation:
c      Call the distribution routine for each two windows of the
c      signal. The signal array z() is passed with a time offset, t,
c      so that the transform2() routine can operate as though t was always
c      zero.

do 8 ii=0,nplts/2-1
    t=ii*res*2+hlf+LWINMAX
    call transform2 (z(t),z(t+res),FFTLEN,mf,hlf,wvd)
    do 9 i=1,FFTLEN
        write(2,*) real(wvd(i))
    do 11 i=1,FFTLEN
        write(2,*) aimag(wvd(i))
11
8      continue

```

```

close(2)

999 return
end

c -----
subroutine transform2(z1,z2,FFTLEN,mf,hlf,R)
complex z1(1), z2(1), R(1)
complex v1,v2
integer FFTLEN,mf,hlf,m

c This routine forms the bilinear lag array for two kernel
c time-slices at once, for lag values up to hlf. (i.e., using
c a window of length 2*hlf+1 centered on elements 0 and res.)
c
c The second set of lag values (those around res) are added
c into the lag array in an 'odd' fashion, as opposed to the
c 'even' fashion that the first set were inserted.
c
c Finally, the Fourier transform of the lag array is taken,
c forming the two time-slices at once, in the real and imaginary
c parts of the array.
c
c This is achieved by multiplying the second array of lags by
c j before performing the Fourier transform. Since this is a
c linear operator, this also rotates the resulting spectrum into
c the imaginary plane.
c
c Finally, it is known that the z array origin is not situated
c at the beginning of the signal array, but is at least 2*hlf
c from either end. Thus negative array indices may be used, and
c a substantial complexity reduction may be realized, as no
c special cases are introduced at the beginning and end of the
c signal (see section 3.5)

v1=z1(0)*conjg(z1(0))
v2=z2(0)*conjg(z2(0))
R(1)=v1+cmplx(-aimag(v2),real(v2))
do 1 m=1,hlf
  v1=z1(m)*conjg(z1(-m))
  v2=z2(m)*conjg(z2(-m))

```

```

R(m+1)=v1+cmplx(-aimag(v2),real(v2))
R(FFTLEN-m+1)=conjg(v1)+cmplx(aimag(v2),real(v2))
1 continue

c make sure any points not containing calculated values are set
c to zero, as the FFT length may be greater than the window:

do 3 m=hlf+2,FFTLEN-hlf
3 R(m)=(0.0,0.0)

call FFT(R,mf,FFTLEN)
return
end

```

## 5 Other methods

While the general approach presented will allow calculation of most time-frequency distributions, it does not cope with all possibilities, and for some popular cases it is not the most efficient approach. Optimizations of the method for the Wigner-Ville distribution have already been noted. If work is to center on one of these other techniques, then more computationally efficient approaches may be considered. Some techniques not covered by the general technique will be commented on briefly in this section, and references will be provided for sources of further information.

### 5.1 The wavelet transform

This is a linear transform, as opposed to the bilinear distributions discussed in this chapter, and has the interesting property of time resolution which is a function of frequency. See [7] for more information.

### 5.2 Cross-Wigner-Ville distribution

The most significant differences between the implementation techniques for the Cross-Wigner-Ville distribution and those already described are:

- the necessity to read and store two data streams;

- the necessity to provide storage for complex numbers in the output matrix;
- the loss of some of the symmetry optimizations that could be performed with the bilinear transforms.

Once these are taken into account, all of the presented techniques and program structure still apply. The cross Wigner-Ville distribution is discussed in the chapter by Boles in this volume [5].

### 5.3 Short-time Fourier transform

The short-time Fourier transform (STFT) is a special case because it can be efficiently calculated by taking the spectrum of windows of the data. The program structure will be somewhat similar, from the point of view of data input, output and windowing. Instead of calculation of the bilinear kernel, and convolution with the relevant determining function, all that is necessary is to form the Fourier transform of each window of data, and output the squared magnitude of each frequency point. Some care is necessary, if windowing is required. Unlike the bilinear transforms noted, this technique will produce both positive and negative frequency values (which will be zero for analytic signals), and so only half of the calculated frequency vector need be output.

Since the Fourier transform approach assumes signal stationarity over the period of the window, care must be taken to optimize the window length for signals that are not stationary. Where linear frequency modulated signals are involved, the optimum window length is  $(\frac{df_i}{dt})^{-\frac{1}{2}}$ . Windows shorter than this cause unnecessary spreading of the signal due to the width of the window in the frequency domain. Longer windows cause a smearing of the signal within the window due to the non-stationarity of the frequency law contained within it.

### 5.4 Parametric methods

All of the parametric spectrum estimation techniques can be used for time-frequency spectrum analysis in the same manner that the Fourier transform is used: by the assumption of short time stationarity. For a very detailed discussion of these methods, consult the books by Kay and Marple [9,10].

The higher resolution offered by these techniques can also be applied to the bilinear distributions, by replacing the final Fourier transform with a parametric spectrum calculation. This generally requires a much greater amount of computation. An algorithm is described by Whitehouse, Boashash and Speiser in [13].

### 5.5 The Q-distribution

A time-frequency technique that will not be covered in depth here is the Q-distribution of Altes [1], so called because it shares the constant-Q or proportional bandwidth property of the Wavelet transform, although it is based on the bilinear Wigner-Ville distribution, rather than a linear transform. The paper [1] describes how this may be done efficiently, after which the calculation would proceed as for the Wigner-Ville.

## 6 Description of the TFSA package

TFSA is the current name of a package of signal analysis tools built up over a period of time by B. Boashash and his students. Its primary function is to produce time-frequency representations and plots from time series, although it has a number of auxiliary functions. Currently the spectrum analysis tools include the short-time Fourier transform, the Wigner-Ville distribution, an auto-regressive (parametric) model based spectrum estimator, Wigner-Ville distribution modified to use an auto-regressive model spectrum estimator, Choi-Williams distribution, Born-Jordan-Cohen distribution and ZAM distribution [14].

It also has a suite of test signal generation routines with which a wide variety of test signals, including Gaussian white noise can be produced and manipulated.

The package consists of a large number of stand alone programs linked by common file formats and an interactive menu/form based front end, which simplifies and speeds the use of these programs dramatically. The code fragments presented in this chapter were extracted from this package, which is constantly being updated. It is available for a small cost from the authors, and may be ordered with the order form which appears at the back of the book.

## 7 Summary

In this chapter we have presented a general approach to the calculation of the time-frequency energy distributions based on Cohen's class of distributions. The principle steps of this approach are:

1. Produce the analytic signal from the real data sequence to be analysed (section 1).
2. Calculate the bilinear kernel values (section 3.1).

3. Convolve the bilinear kernel with the distribution specification function  $G(n, m)$ , (sections 3.2, 3.6).
4. Take the discrete Fourier transform with respect to lag ( $m$ ) for each time instant (section 3.3).

This process was demonstrated for the case of the Choi-Williams distribution in the code fragments in sections 3.4 and 3.5.

This algorithm can be simplified and optimized quite significantly for a number of common members of Cohen's class, due to extra properties of the kernel. The Wigner-Ville distribution was presented as an example of how the kernel selection function  $G(n, m)$  can be incorporated into the algorithm, rather than being stored or calculated (see section 4).

References were presented for a number of other popular time-frequency signal analysis techniques which do not fit into this framework, in section 5.

The final section (6) briefly described a signal analysis package written by the authors and their colleagues which utilizes most of the techniques presented.

## References

- [1] R. A. Altes, "Wideband, Proportional Bandwidth Wigner-Ville Analysis," *IEEE Trans. Acoust. Speech Signal. Process.*, 38, June 1990.
- [2] B. Boashash, "Time-Frequency Signal Analysis", in *Advances in Spectral Analysis and Array Processing* (Prentice-Hall Signal Processing Series), S. Haykin, editor, Englewood Cliffs, NJ, Prentice-Hall, 1990.
- [3] B. Boashash, "Note on the Use of the Wigner Distribution of Time-Frequency Signal Analysis", *IEEE Trans. Acoust. Speech Signal. Process.*, 36, pp. 1518-1521, September 1988.
- [4] B. Boashash and P. Black, "An Efficient Real Time Implementation of the Wigner-Ville Distribution", *IEEE Trans. Acoust. Speech Signal. Process.* ASSP 35, no. 11, pp. 1611-1618, November 1987, CRISSP Reference CSP 87/6.
- [5] P. Boles, "Application of the Cross-Wigner-Ville Distribution to Seismic Surveying" in *TFSA Methods and Applications*, B. Boashash, editor, Longman Cheshire, 1991.
- [6] L. Cohen, "Time-Frequency Distributions—A Review," *IEEE Proc.*, 77, no. 7, pp. 941-981, July 1989.
- [7] I. Daubechies, "The Wavelet Transform: a Method for Time-Frequency Localization" in *Advances in Spectral Analysis and*

- Array Processing* (Prentice-Hall Signal Processing Series), S. Haykin, editor, Englewood Cliffs, NJ, Prentice-Hall, 1990.
- [8] H. H. Eilouti and L. M. Khadra, "Optimized Implementation of Real-Time Discrete Wigner Distribution," *Electronics Letters*, 25, no. 11, pp. 706,707, 25 May 1989.
- [9] S. Kay, *Modern Spectral Analysis*, Englewood Cliffs, NJ, Prentice-Hall, 1989.
- [10] S. L. Marple, Jr, *Digital Spectral Analysis with Applications*, Englewood Cliffs, NJ, Prentice-Hall, 1987.
- [11] A. V. Oppenheim and R. W. Schaffer, *Discrete-Time Signal Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1989.
- [12] M. Sun, C. C., Li, L. N. Sekhar and R. J. Scabassi, "Efficient Computation of the Discrete Pseudo-Wigner Distribution," *IEEE Trans. Acoust. Speech Signal. Process.*, 37, no. 11, pp. 1735-1742, November 1989.
- [13] H. J. Whitehouse, B. Boashash and J. M. Speiser, "High Resolution Processing Techniques for Temporal and Spatial Signals," in *High Resolution Techniques in Underwater Acoustics* (Lecture Notes in Control and Information Series), New York-Heidelberg-Berlin, Springer-Verlag, 1989.
- [14] Y. Zhao, L. E. Atlas and R. J. Marks "The Use of Cone-Shaped Kernels for Generalized Time-Frequency Representations of Nonstationary Signals," *IEEE Trans. Acoust. Speech Signal. Process.*, 38, no. 7, July 1990.