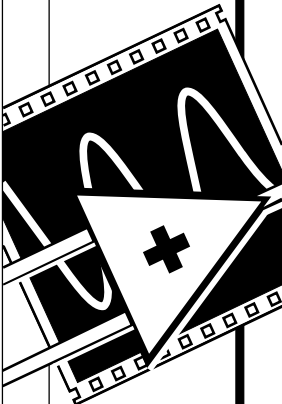


LabVIEW



Wavelet and Filter Bank Design Toolkit Reference Manual

January 1997 Edition
Part Number 321380A-01



Internet Support

support@natinst.com

E-mail: info@natinst.com

FTP Site: ftp.natinst.com

Web Address: <http://www.natinst.com>



Bulletin Board Support

BBS United States: (512) 794-5422

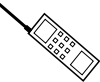
BBS United Kingdom: 01635 551422

BBS France: 01 48 65 15 59



Fax-on-Demand Support

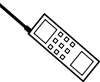
(512) 418-1111



Telephone Support (U.S.)

Tel: (512) 795-8248

Fax: (512) 794-5678



International Offices

Australia 02 9874 4100, Austria 0662 45 79 90 0, Belgium 02 757 00 20,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, Denmark 45 76 26 00,
Finland 09 527 2321, France 01 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186,
Israel 03 5734815, Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456,
Mexico 5 520 2635, Netherlands 0348 433466, Norway 32 84 84 00, Singapore 2265886,
Spain 91 640 0085, Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200,
U.K. 01635 523545

National Instruments Corporate Headquarters

6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: (512) 794-0100

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THEREOF PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

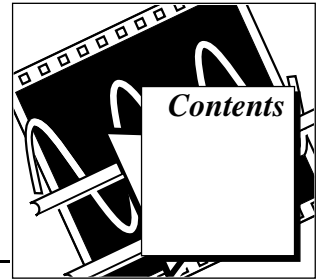
Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

LabVIEW®, National Instruments®, and natinst.com™ are trademarks of National Instruments Corporation. Product and company names listed are trademarks or trade names of their respective companies.

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.



About This Manual

Organization of This Manual	ix
Conventions Used in This Manual	x
Related Documentation	xi
Customer Communication	xi

Chapter 1

Introduction

Package Contents	1-1
Installation Procedure	1-1
Windows	1-2
Macintosh and Power Macintosh	1-2
Sun and HP-UX	1-2
WFBD Toolkit Applications	1-3

Chapter 2

Wavelet Analysis

History of Wavelet Analysis	2-1
Conventional Fourier Transform	2-1
Innovative Wavelet Analysis	2-3
Wavelet Analysis vs. Fourier Analysis	2-7
Applications of Wavelet Analysis	2-9
Discontinuity Detection	2-9
Multiscale Analysis	2-11
Detrend	2-12
Denoise	2-13
Performance Issues	2-13

Chapter 3 Digital Filter Banks

Two-Channel Perfect Reconstruction Filter Banks	3-1
Biorthogonal Filter Banks	3-4
Orthogonal Filter Banks	3-10
Two-Dimensional Signal Processing	3-13

Chapter 4 Using the Wavelet and Filter Bank Design Toolkit

Wavelet and Filter Bank Design	4-1
Design Panel	4-6
One-Dimensional Data Test	4-11
Two-Dimensional Data Test	4-14
Wavelets and Filters	4-16
Create Your Own Applications	4-18
Wavelet Packet Analysis	4-18
On-Line Testing Panel	4-20
Denoising	4-20

Chapter 5 Function Reference

LabVIEW VI Applications	5-1
2-Channel Analysis Filter Bank VI	5-1
2-Channel Synthesis Filter Bank VI	5-4
2D Analysis Filter Bank VI	5-5
2D Synthesis Filter Bank VI	5-6
Decimation Filter VI	5-7
Interpolation Filter VI	5-10
Mother Wavelet and Scaling Function VI	5-12
LabWindows/CVI Applications	5-13
Calling WFBD functions in LabWindows/CVI	5-13
WFBD Instrument Driver	5-13
AllocCoeffWFBD	5-14
Analysis2DArraySize	5-15
AnalysisFilterBank	5-17
AnalysisFilterBank2D	5-18
DecimationFilter	5-22
FreeCoeffWFBD	5-24
InterpolationFilter	5-24
ReadCoeffWFBD	5-27
Synthesis2DArraySize	5-27

SynthesisFilterBank	5-29
SynthesisFilterBank2D	5-31
Windows Applications.....	5-34

Appendix A References

Appendix B Error Codes

Appendix C Customer Communication

Glossary

Index

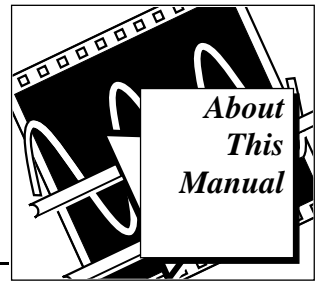
Figures

Figure 2-1.	Sum of Two Truncated Sine Waveforms	2-2
Figure 2-2.	Wavelet	2-4
Figure 2-3.	Wavelet Analysis	2-6
Figure 2-4.	Short-Time Fourier Transform Sampling Grid	2-7
Figure 2-5.	Wavelet Transform Sampling Grid	2-8
Figure 2-6.	Comparison of Transform Processes	2-9
Figure 2-7.	Detection of Discontinuity	2-10
Figure 2-8.	Multiscale Analysis	2-11
Figure 2-9.	Detrend	2-12
Figure 2-10.	Denoise	2-13
Figure 3-1.	Two-Channel Filter Bank	3-1
Figure 3-2.	Relationship of Two-Channel PR Filter Banks to Wavelet Transform ..	3-3
Figure 3-3.	Filter Bank and Wavelet Transform Coefficients	3-4
Figure 3-4.	Halfband Filter	3-7
Figure 3-5.	Zeros Distribution for $(1 - z^{-1})^6Q(z)$	3-8
Figure 3-6.	B-Spline Filter Bank	3-9
Figure 3-7.	Dual B-Spline Filter Bank	3-9
Figure 3-8.	Third Order Daubechies Filter Banks and Wavelets	3-12
Figure 3-9.	2D Signal Processing	3-13
Figure 3-10.	2D Image Decomposition	3-14

Figure 4-1.	Design Procedure for Wavelets and Filter Banks	4-2
Figure 4-2.	Non-Negative Equiripple Halfband Filter	4-3
Figure 4-3.	Minimum Phase Filter	4-5
Figure 4-4.	Linear Phase Filter	4-5
Figure 4-5.	Orthogonal Filter	4-5
Figure 4-6.	Design Panel	4-6
Figure 4-7.	Equiripple Filter	4-8
Figure 4-8.	1D_Test Panel	4-11
Figure 4-9.	DAQ Setup Panel	4-13
Figure 4-10.	Specifying a Path	4-14
Figure 4-11.	2D_Test Panel	4-15
Figure 4-12.	Wavelets and Filters	4-17
Figure 4-13.	Full Path of a Three Level PR Tree	4-19
Figure 4-14.	Wavelet Packet	4-19
Figure 4-15.	Implementation of a Wavelet Packet	4-20
Figure 5-1.	Zero Padding	5-3
Figure 5-2.	Symmetric Extension	5-3
Figure 5-3.	Filtering Operation	5-9
Figure 5-4.	Two-Channel Perfect Reconstruction System	5-10
Figure 5-5.	Signal Interpolated by 2	5-11

Tables

Table 4-1.	Filter Comparison	4-4
Table B-1.	LabVIEW VI Error Codes	B-1



The *LabVIEW Wavelet and Filter Bank Design Toolkit Reference Manual* describes the features, functions, and applications of wavelet analysis and filter bank design. In addition, this manual contains descriptions of LabVIEW virtual instruments (VIs) and LabWindows/CVI[®] functions you can use to develop your own wavelet and filter bank designs.

Organization of This Manual


The *Wavelet and Filter Bank Design Toolkit Reference Manual* is organized as follows:

- Chapter 1, *Introduction*, lists the contents of the Wavelet and Filter Bank Design (WFBD) Toolkit, describes the installation procedure, and provides an overview of the WFBD Toolkit.
- Chapter 2, *Wavelet Analysis*, describes the history of wavelet analysis, compares Fourier transformation and wavelet analysis, and describes some applications of wavelet analysis.
- Chapter 3, *Digital Filter Banks*, describes the design of two-channel perfect reconstruction filter banks and defines the types of filter banks used with wavelet analysis.
- Chapter 4, *Using the Wavelet and Filter Bank Design Toolkit*, describes the architecture of the WFBD Toolkit, lists the design procedures, and describes some applications you can create with the WFBD Toolkit.
- Chapter 5, *Function Reference*, describes the VIs in the WFBD Toolkit package, the instrument driver for LabWindows/CVI, and the functions in the DLLs.
- Appendix A, *References*, lists the reference material used to produce the Wavelet and Filter Bank Design Toolkit.
- Appendix B, *Error Codes*, lists the error codes returned by the LabVIEW VIs, including the error number and a description.
- Appendix C, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.

- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

bold	Bold text denotes a parameter, menu name, palette name, menu item, return value, function panel item, or dialog box button or option.
<i>italic</i>	Italic text denotes mathematical variables, emphasis, a cross reference, or an introduction to a key concept.
<i>bold italic</i>	Bold italic text denotes an activity objective, note, caution, or warning.
monospace	Text in this font denotes text or characters that you should literally enter from the keyboard. Sections of code, programming examples, and syntax examples also appear in this font. This font also is used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, variables, filenames, and extensions, and for statements and comments taken from program code.
<>	Angle brackets enclose the name of a key on the keyboard—for example, <PageDown>.
-	A hyphen between two or more key names enclosed in angle brackets denotes that you should simultaneously press the named keys—for example, <Control-Alt-Delete>.
<Control>	Key names are capitalized.
»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence File»Page Setup»Options»Substitute Fonts directs you to pull down the File menu, select the Page Setup item, select Options , and finally select the Substitute Fonts option from the last dialog box.
paths	Paths in this manual are denoted using backslashes (\) to separate drive names, directories, and files, as in C:\dir1name\dir2name\filename.
	This icon to the left of bold italicized text denotes a note, which alerts you to important information.

Abbreviations, acronyms, metric prefixes, mnemonics, symbols, and terms are listed in the *Glossary*.

Related Documentation

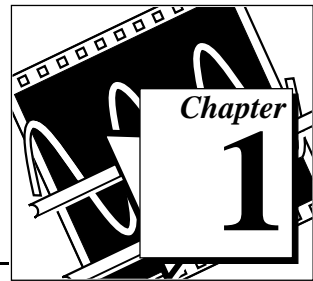
The following documents contain information that you may find helpful as you read this manual:

- *LabVIEW Digital Filter Design Toolkit Reference Manual*
- *LabVIEW Analysis VI Reference Manual*
- *LabWindows/CVI Advanced Analysis Library Reference Manual*
- *Joint Time-Frequency Analysis Manual*

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix C, *Customer Communication*, at the end of this manual.

Introduction



This chapter lists the contents of the Wavelet and Filter Bank Design (WFBD) Toolkit, describes the installation procedure, and provides an overview of the WFBD Toolkit.

Package Contents

Your WFBD Toolkit contains the following materials:

- The *Wavelet and Filter Bank Design Toolkit Reference Manual*.
- The WFBD Toolkit diskettes, containing the following four parts:
 - `WFBD.exe`—The main program for designing and testing the wavelet and filter bank. This stand-alone software does not require special application software to run it.
 - `lvsrc`—The folder containing the VI libraries for LabVIEW users. It contains the source code for `WFBD.exe`. If you design your wavelets and filter banks in the LabVIEW environment, you have more design power. For example, under LabVIEW, you can design your own real-time test program.
 - `WFBD32.FP`—An instrument driver for Windows 95/NT LabWindows/CVI users. The driver contains the essential functions for you to build your own wavelet and filter bank systems.
 - `WFBD32.DLL`—A 32-bit dynamic link library (DLL) containing the same functions as `WFBD32.FP`, which you can use for LabWindows/CVI and other Windows environments.

Installation Procedure

The following sections contain instructions for installing the WFBD Toolkit on the Macintosh, Windows, Sun SPARCstation, and HP-UX platforms.

Windows

These instructions apply to Windows 3.x, Windows NT, and Windows 95. Complete the following steps to install the toolkit:

1. Launch Windows.
2. If you are installing the WFBD Toolkit on Windows 3.x or Windows NT, select **File»Run...** from the Program Manager. If you are installing the WFBD Toolkit on Windows 95, select **Start»Run...** Insert disk 1 and type:
`x: \SETUP`
where x is your floppy drive.
3. Follow the instructions on your screen.

Once you complete the on-screen installation instructions, you can run the WFBD Toolkit.

Macintosh and Power Macintosh

Complete the following steps to install the toolkit:

1. Insert disk 1 of the WFBD Toolkit into your 3.5 inch disk drive and double-click on the WFBD Toolkit Installer icon when it appears on your desktop.
2. Follow the instructions on your screen.

Once you complete the on-screen installation instructions, you can run the WFBD Toolkit.

Sun and HP-UX

Complete the following steps to install the toolkit on your hard drive:

1. Insert disk 1 of the WFBD Toolkit into your 3.5 inch disk drive.
2. In the UNIX shell, enter the following line in a directory for which you have write permission.

```
tar xvf /dev/rfd0a INSTALL
```

This entry extracts the installation script file, `INSTALL`.

3. Enter the following command to start the installation process:
`./INSTALL`
4. Follow the instructions on your screen.

Once you complete the on-screen installation instructions, you can run the WFBD Toolkit.

WFBD Toolkit Applications

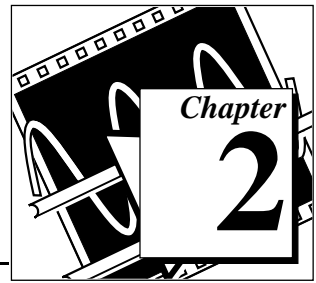
The success of wavelet analysis applications hinges on the selection of a proper wavelet function. Unlike current wavelet software, the WFBD Toolkit provides an intuitive and interactive way of designing wavelets and filter banks. With the WFBD Toolkit, you can choose from a variety of two-channel perfect reconstruction filter banks and wavelet functions. In particular, you can examine the performance of wavelet waveforms and filter banks on your data online. This unique feature greatly facilitates the design process because you can save all design results as text files for future use.

The WFBD Toolkit is a stand-alone application as well as an add-on toolkit for LabVIEW. Using LabVIEW, you can build your own real-time test experiment.

The WFBD Toolkit also provides an instrument driver if you are using LabWindows/CVI for Windows 95/NT. This instrument driver contains all the necessary functions to build your own analysis and synthesis filter banks system. The instrument driver also provides utility functions to read the filter bank coefficients designed by using the stand-alone WFBD application.

In addition, the WFBD Toolkit provides a dynamic link library (DLL) that contains functions for both LabWindows/CVI and other Windows users.

Wavelet Analysis



This chapter describes the history of wavelet analysis, compares Fourier transform and wavelet analysis, and describes some applications of wavelet analysis.

History of Wavelet Analysis

Although Alfred Haar first mentioned the term *wavelet* in a 1909 thesis, the idea of wavelet analysis did not receive much attention until the late 1970s. Since then, wavelet analysis has been studied thoroughly and applied successfully in many areas. Current wavelet analysis is thought by some to be no more than the recasting and unifying of existing theories and techniques. Nevertheless, the breadth of applications for wavelet analysis is more expansive than ever was anticipated.

Conventional Fourier Transform

The development of wavelet analysis originally was motivated by the desire to overcome the drawbacks of traditional Fourier analysis and *short-time Fourier transform* processes. Fourier transform characterizes the frequency behaviors of a signal, but not how the frequencies change over time. Short-time Fourier transform, or *windowed Fourier transform*, simultaneously characterizes a signal in time and frequency. You can encounter a problem because the signal time and frequency resolutions are fixed once you select the type of window. However, signals encountered in nature always have a long time period at low frequency and a short time period at high frequency. This suggests that the window should have high time resolution at high frequency.

To understand the fundamentals of wavelet analysis, start with an artificial example.

Figure 2-1 shows a signal $s(t)$ that consists of two truncated sine waveforms. While the first waveform spans 0 to 1 second, the second waveform spans 1 to 1.5 seconds. In other words, the frequency of $s(t)$ is 1 Hz for $0 \leq t < 1$ and 2 Hz for $1 \leq t < 1.5$.

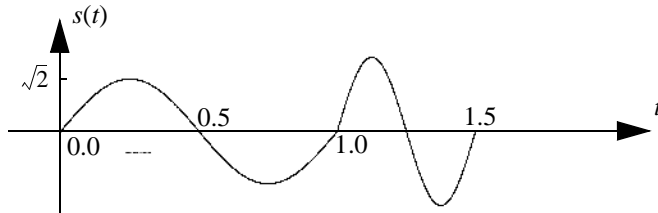


Figure 2-1. Sum of Two Truncated Sine Waveforms

When describing frequency behavior, you traditionally compare $s(t)$ with a group of harmonically related complex sinusoidal functions, such as $\exp\{j2\pi kt/T\}$. Here, the term *harmonically related complex sinusoidal functions* refers to the sets of periodic sinusoidal functions with fundamental frequencies that are all multiples of a single positive frequency $2\pi/T$.

You accomplish the comparison process with the following correlation (or inner product) operation:

$$a_k = \int_T s(t) e_k^*(t) dt = \int_T s(t) \exp\left\{-j\frac{2\pi k}{T}t\right\} dt \quad (2-1)$$

where a_k is the *Fourier coefficient*, and * denotes a complex conjugate.

The magnitude of a_k indicates the degree of similarity between the signal $s(t)$ and the elementary function $\exp\{j2\pi kt/T\}$. If this quantity is large, it indicates a high degree of correlation between $s(t)$ and $\exp\{j2\pi kt/T\}$. If this quantity is almost 0, it indicates a low degree of correlation between $s(t)$ and $\exp\{j2\pi kt/T\}$. Therefore, you can consider a_k as the measure of similarity between the signal $s(t)$ and each of the individual complex sinusoidal functions $\exp\{j2\pi kt/T\}$. Because $\exp\{j2\pi kt/T\}$ represents a distinct frequency $2\pi k/T$ (a frequency tick mark), the Fourier coefficient a_k indicates the amount of signal present at the frequency $2\pi k/T$.

In Figure 2-1, *Sum of Two Truncated Sine Waveforms*, $s(t)$ consists of two *truncated* sine waveforms. The inner product of such truncated

signal and pure sine waveforms, which extends from minus infinity to plus infinity, never vanishes, that is, a_k is not zero for all k . However, the dominant a_k , with the largest magnitude, is that which corresponds to 1 and 2 Hz elementary functions. This indicates that the primary components of $s(t)$ are 1 and 2 Hz signals. However, it is unclear, based on a_k alone, when the 1 Hz or the 2 Hz components exist in time.

There are many ways of building the frequency tick marks to measure the frequency behavior of a signal. By using complex sinusoidal functions, not only can you analyze signals, but you also can reconstruct the original signal with the Fourier coefficient a_k . For example, you can write $s(t)$ in terms of the sum of complex sinusoidal functions, according to the following formula, traditionally known as *Fourier expansion*.

$$s(t) = \sum_{k=-\infty}^{\infty} a_k e_k(t) = \sum_{k=-\infty}^{\infty} a_k \exp\left\{j \frac{2\pi t}{T} k\right\} \quad (2-2)$$

where a_k is the Fourier coefficient and $2\pi k/T$ is the frequency tick mark.

In this equation, because a_k is not zero for all k , you must use an infinite number of complex sinusoidal functions in equation (2-2) to restore $s(t)$ in Figure 2-1, *Sum of Two Truncated Sine Waveforms*.

Innovative Wavelet Analysis

Looking at $s(t)$ more closely, you find that to determine the frequency contents of $s(t)$, you only need information regarding one cycle, such as the time span of one cycle. With this information, you can compute the frequency with the following formula:

$$frequency = \frac{1}{\text{time span of one cycle}} \quad (2-3)$$

According to this equation, the higher the frequency, the shorter the time span. Therefore, instead of using infinitely long complex sinusoidal functions, you can use only one cycle of a sinusoidal waveform, or a wavelet, to measure $s(t)$. The wavelet $\psi(t)$ used to measure $s(t)$ is one cycle of sinusoidal waveform, as shown below in Figure 2-2.

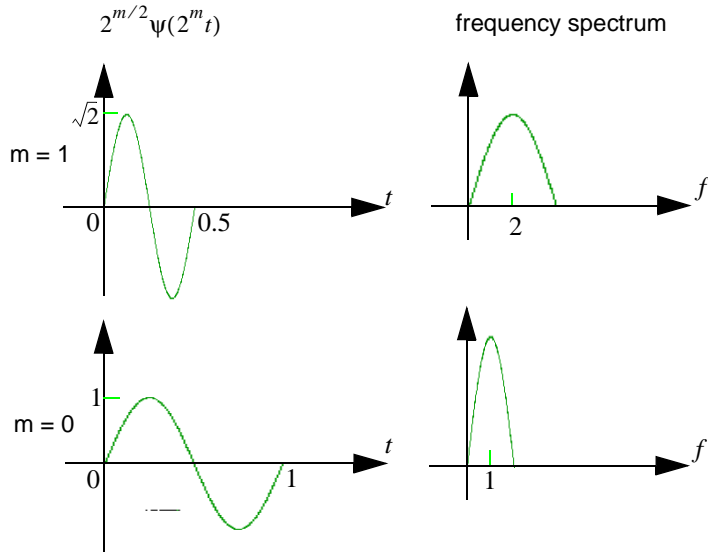


Figure 2-2. Wavelet

Because $\psi(t)$ spans 1 second, consider the frequency of $\psi(t)$ to be 1 Hz. As in the case of Fourier analysis, you can achieve the comparison process with the following correlation (or inner product) operation:

$$W_{m,n} = \int_T s(t) \psi_{m,n}(t) dt \tag{2-4}$$

where $W_{m,n}$ denotes the wavelet transform coefficients and $\psi_{m,n}(t)$ are the elementary functions of the wavelet transform.

However, the structure of the elementary functions $\psi_{m,n}(t)$ differs from the Fourier transformations, which are the *dilated* and *shifted* versions of $\psi(t)$, that is,

$$\psi_{m,n}(t) = 2^{m/2}\psi(2^m(t - n2^{-m})) \quad (2-5)$$

where m and n are integers.

By increasing n , you shift $\psi_{m,n}(t)$ forward in time. By increasing m , you compress the time duration and thereby increase the center frequency and frequency bandwidth of $\psi(t)$ (Qian and Chen 1996). You can consider the parameter m as the *scale factor* and 2^{-m} as the *sampling step*. Therefore, the shorter the time duration, the smaller the time sampling step, and vice versa.

Assuming the center frequency of $\psi(t)$ is ω_0 , then the center frequency of $\psi_{m,n}(t)$ would be $2^m\omega_0$. Consequently, you can systematically adjust the scale factor m to achieve different frequency tick marks to measure the signal frequency contents. That is, as the scale factor m increases, the center frequency and bandwidth of the wavelet increases 2^m .

Figure 2-3 depicts the wavelet transform procedure. First, let $m = n = 0$, that is, align $\psi(t)$ and $s(t)$ at $t = 0$. Then, as in equation (2-5), compare $\psi(t)$ with $s(t)$ for $0 \leq t < 1$. You obtain $W_{0,0} = 1$. Shift $\psi(t)$ to the next second, that is, let $n = 1$, and compare it with $s(t)$ for $1 \leq t < 2$. You obtain $W_{0,1} = 0$.

Next, compress $\psi(t)$ into 0 to 0.5 seconds, that is, let $m = 1$, and repeat the previous operations with the time-shift step 0.5. You obtain the following results, also displayed in the shaded table of Figure 2-3, *Wavelet Analysis*, from this process:

$$W_{1,0} = 0 \quad W_{1,1} = 0 \quad W_{1,2} = 1 \quad W_{1,3} = 0$$

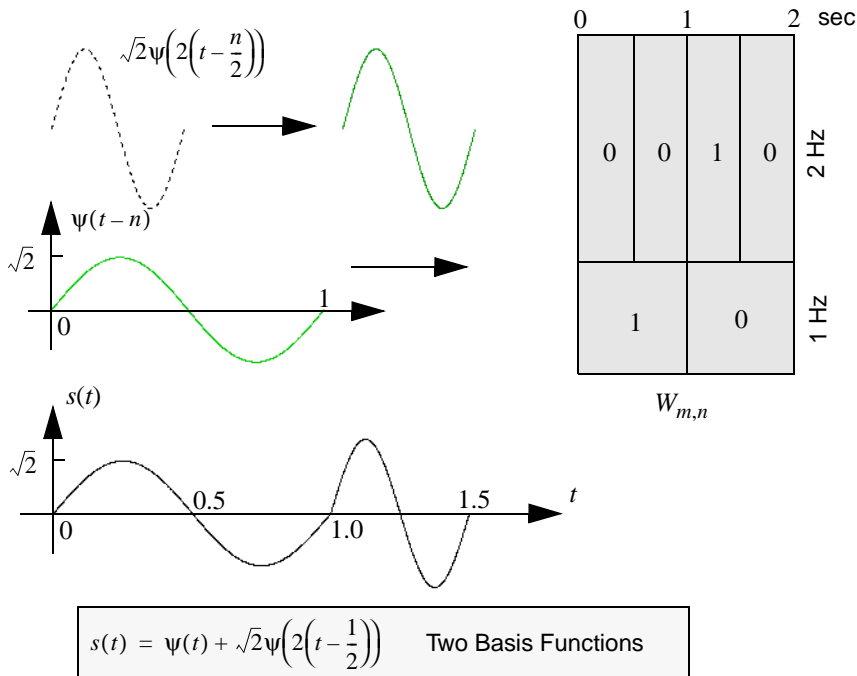


Figure 2-3. Wavelet Analysis

You can continue to compress $\psi(t)$ by increasing the scale factor m and reducing the time-shift step 2^{-m} to test $s(t)$. This procedure is called *wavelet transform*. $\psi(t)$ is called the *mother wavelet* because the different wavelets used to measure $s(t)$ are the dilated and shifted versions of this wavelet. The results of each comparison, $W_{m,n}$, are named *wavelet coefficients*. The index m and n are the scale and time indicators, respectively, which describe the signal behavior in the joint time-scale domain (As shown in Figure 2-5, *Wavelet Transform Sampling Grid*, you can easily convert the scale into frequency. Hence, $W_{m,n}$ also can be considered the signal representation in joint time and frequency domain). In this example, by checking the wavelet coefficients, you know that for $0 \leq t < 1$ the frequency of $s(t)$ is 1 Hz and for $1 \leq t < 1.5$ the frequency of $s(t)$ is 2 Hz.

Unlike Fourier analysis, wavelet transform not only indicates what frequencies the signal $s(t)$ contains, but also when these frequencies occur. Moreover, the wavelet coefficients $W_{m,n}$ of a real-valued signal $s(t)$ are always real as long as you choose real-valued $\psi(t)$. Compared

to Fourier expansion, you usually can use fewer wavelet functions to represent the signal $s(t)$. In this example, $s(t)$ can be completely represented by two terms, whereas an infinite number of complex sinusoidal functions would be needed in the case of Fourier expansion.

Wavelet Analysis vs. Fourier Analysis

You can apply short-time Fourier transform to characterize a signal in both the time and frequency domains simultaneously. However, you also can use wavelet analysis to perform the same function because of its similarity to short-time Fourier transform. You compute both by the correlation (or inner product) operation, but the main difference lies in how you build the elementary functions.

For short-time Fourier transform, the elementary functions used to test the signal are time-shifted, frequency-modulated single window functions, all with some envelope. Because this modulation does not change the time or frequency resolutions (Qian and Chen 1996), the time and frequency resolutions of the elementary functions employed in short-time Fourier transform are constant. Figure 2-4 illustrates the sampling grid for the short-time Fourier transform.

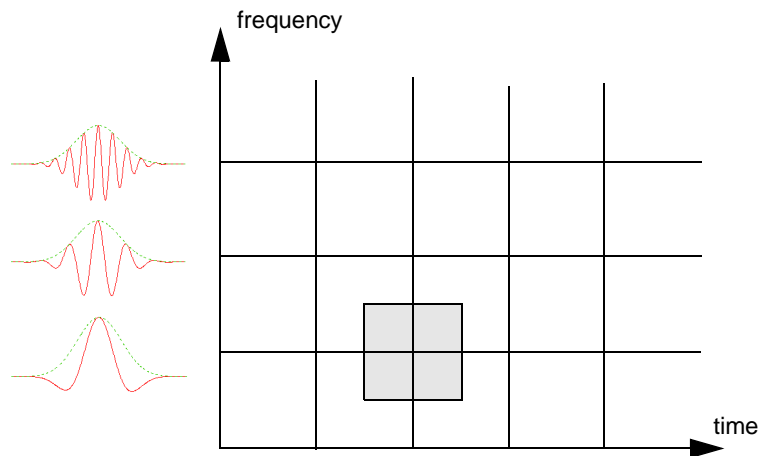


Figure 2-4. Short-Time Fourier Transform Sampling Grid

For wavelet transform, increasing the scale parameter m reduces the width of the wavelets. The time resolution of the wavelets improves and the frequency resolution becomes worse as m becomes larger. Because

of this, wavelet analysis has good time resolution at high frequencies and good frequency resolution at low frequencies.

Figure 2-5 illustrates the sampling grid for wavelet transform. Suppose that the center frequency and bandwidth of the mother wavelet $\psi(t)$ are ω_0 and $\Delta\omega$, respectively. Then, the center frequency and bandwidth of $\psi(2^m t)$ are $2^m\omega_0$ and $2^m\Delta\omega$. Although the time and frequency resolutions change at different scales m , the ratio between bandwidth and center frequency remains constant. Therefore, wavelet analysis is also called constant Q analysis, where $Q = \text{center frequency}/\text{bandwidth}$.

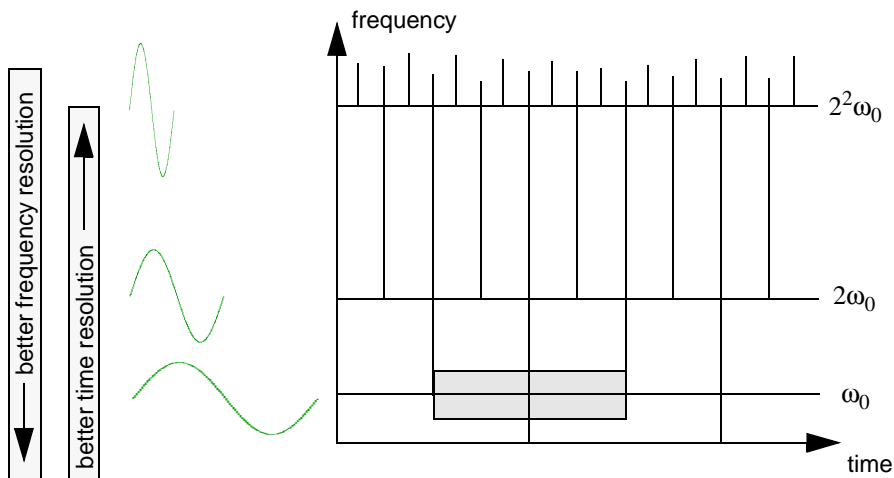


Figure 2-5. Wavelet Transform Sampling Grid

Wavelet transform is closely related to both conventional Fourier transform and short-time Fourier transform. As shown in Figure 2-6, *Comparison of Transform Processes*, all these transform processes employ the same mathematical tool, the correlation operation or inner product, to compare the signal $s(t)$ to the elementary function $b_\alpha(t)$. The difference lies in the structure of the elementary functions $\{e_\alpha(t)\}$. In some cases, wavelet analysis is more natural because the signals always have a long time cycle at low frequency and a short time cycle at high frequency.

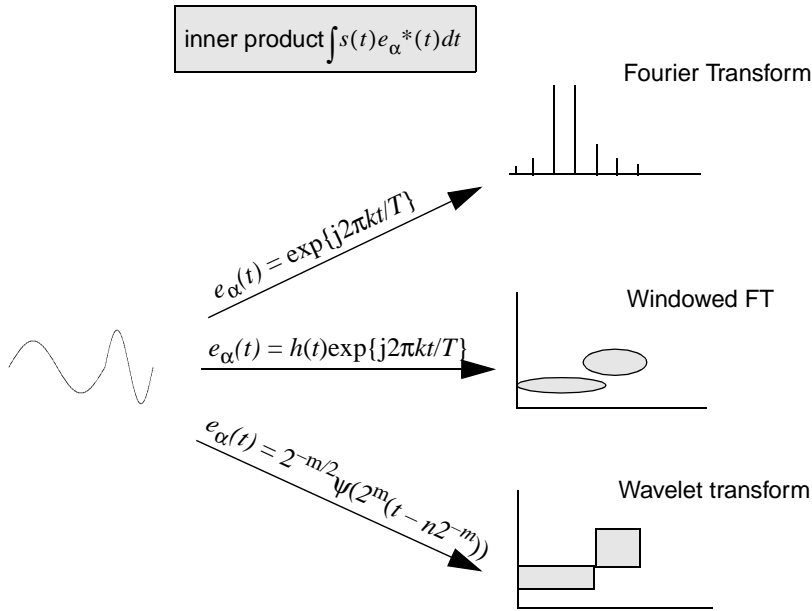


Figure 2-6. Comparison of Transform Processes

Applications of Wavelet Analysis

You can use wavelet analysis for a variety of functions, including detecting the discontinuity of a signal, removing the trend of a signal, suppressing noise, and compressing data.

Discontinuity Detection

Wavelet analysis detects signal discontinuity, such as jumps, spikes, and other non-smooth features. Ridding signals of noise is often much easier to identify in the wavelet domain than in the original domain

For example, the first plot of Figure 2-7, *Detection of Discontinuity*, illustrates a signal $s(k)$ made up of two exponential functions. The turning point or the discontinuity of the first derivative is at $k = 500$. The remaining plots are wavelet coefficients with different scale factors m .

As the scale factor increases, you can pinpoint the location of the discontinuity.

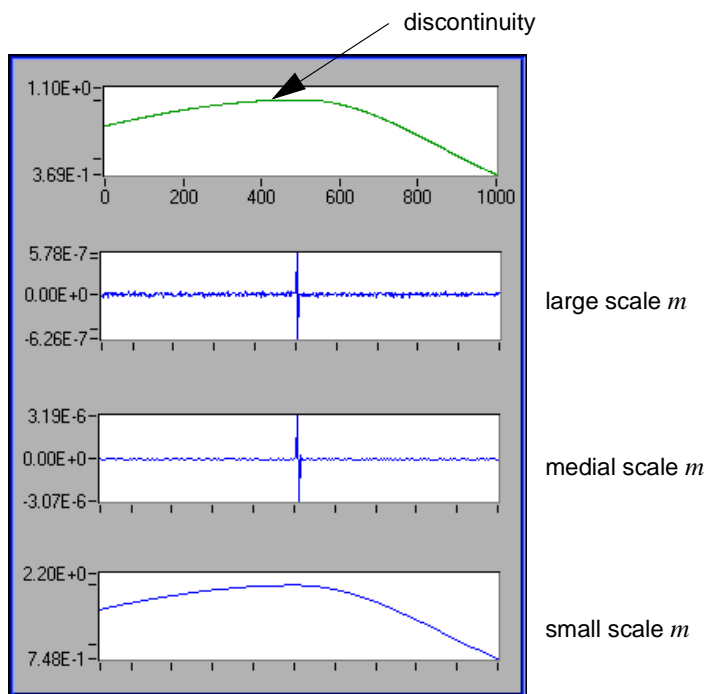


Figure 2-7. Detection of Discontinuity

Using wavelet analysis to detect the discontinuity or break point of a signal has helped to successfully repair scratches in old phonographs. The procedure works by taking the wavelet transform on the signal, smoothing unwanted spikes, and inverting the transform to reconstruct the original signal minus the noise.

In 1889, an agent of Thomas Edison used a wax cylinder to record Johannes Brahms performing his Hungarian Dance No. 1 in G minor. The recording was so poor that it was hard to discern the melody. By using wavelet transform, researchers improved the sound quality enough to distinguish the melody.

Multiscale Analysis

Using wavelet analysis, you also can look at a signal from different scales, commonly called *multiscale analysis*. Wavelet transform-based multiscale analysis helps you better understand the signal and provides a useful tool for selectively discarding undesired components, such as noise and trend, that corrupt the original signals.

Figure 2-8 illustrates a multiscale analysis of an S&P 500 stock index during the years 1947 through 1993. The first plot displays a monthly S&P 500 index while the last plot describes the long term trend of the stock movement. The remaining two plots display the short term fluctuation of the stock, at different levels, during this time. To better characterize the fluctuation that reflects the short term behavior of the stock, you must remove the trend. To do this, first adjust the wavelet decomposition level until you obtain a desired trend. Then, set the corresponding wavelet coefficients to zero and reconstruct the original samples minus the trend.

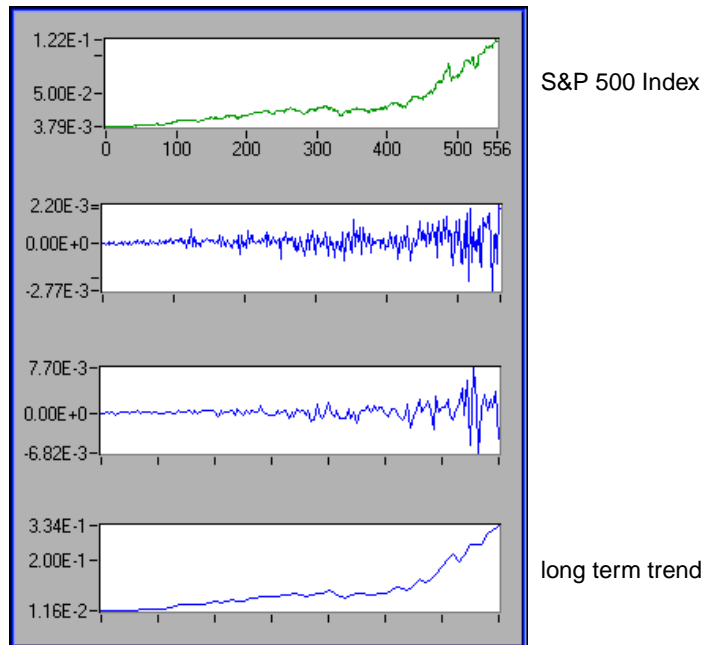


Figure 2-8. Multiscale Analysis

Detrend

One of the most important issues in the application of joint time-frequency analysis is how to remove the trend. In most applications, the trend is often less interesting. It attaches to a strong DC component in the frequency spectrum and thereby blocks many other important signal features. Traditional detrend techniques usually remove the trend by lowpass filtering, thus blurring sharp features in the underlying signal. Wavelet-based detrend is somewhat superior to this process.

Figure 2-9, *Detrend*, illustrates the same S&P 500 stock index information as Figure 2-8, *Multiscale Analysis*, but as a joint time-frequency analysis. The top plot illustrates the S&P 500 stock index as well as its corresponding long term trend (smooth curve). The center plot displays the residue between the original data and the trend, reflecting the short-term fluctuation. The bottom plot displays the joint time and frequency behavior of the residue. It shows that over the past fifty years, a four-year cycle dominates the S&P 500 index, which agrees with most economist assertions.

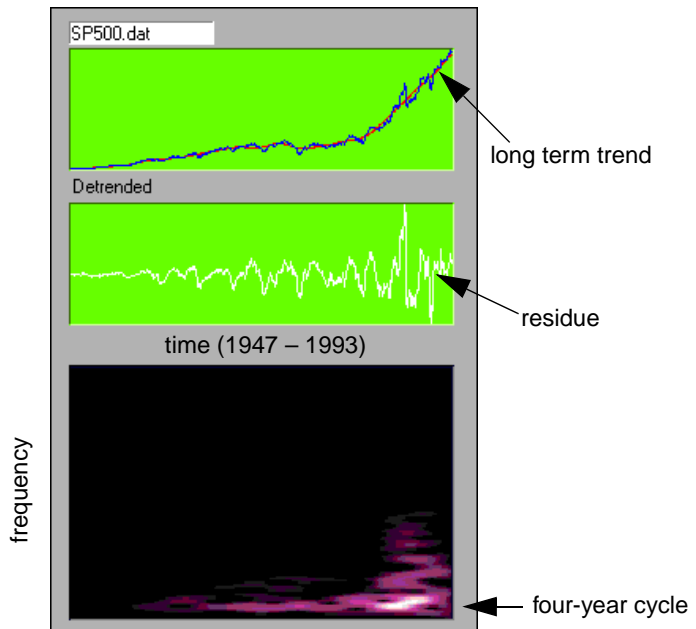


Figure 2-9. Detrend

Denoise

Unlike conventional Fourier transform, which uses only one basis function, wavelet transform provides an infinite number of mother wavelets to select. Consequently, you can select the wavelets that best match the signal. Once the wavelets match the signal, you can use a few wavelet basis to approximate the signal and achieve denoise.

Figure 2-10 illustrates one of the most successful applications of wavelet analysis—denoise. Although Figure 2-10 only uses 25% of the data, the reconstruction preserves all important features contained in the original image. The left image is transformed into the wavelet basis with 75% of the wavelet components set to zero (those of smallest magnitude). The right image is reconstructed from the remaining 25% wavelet components.

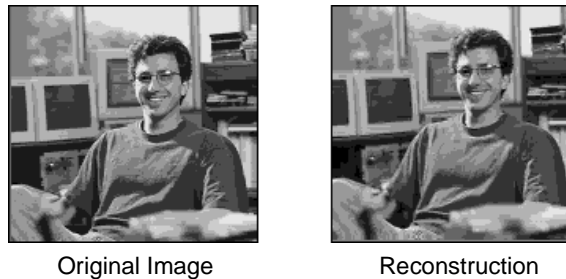


Figure 2-10. Denoise

Performance Issues

Although wavelet analysis possesses many attractive features, its numerical implementation is not as straightforward as its counterparts, such as conventional Fourier transform and short-time Fourier transform. The difficulty arises from the following two aspects.

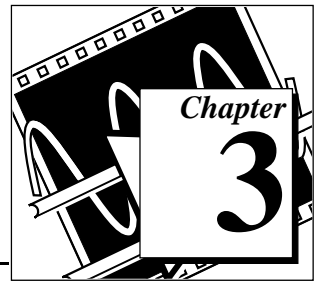
- In order to reconstruct the original signal, the selection of the mother wavelet $\psi(t)$ is not arbitrary. Although any function can be used in equation (2-4), you sometimes cannot restore the original signal based on the resulting wavelet coefficients $W_{m,n}$. $\psi(t)$ is a *valid* or *qualified* wavelet only if you can reconstruct the original signal from its corresponding wavelet coefficients. The selection of the qualified wavelet is subject to certain restrictions. On the other hand, it is not unique. Unlike the case of conventional Fourier transform, in which the basis functions must be complex sinusoidal

functions, you can select from an infinite number of mother wavelet functions. Therefore, the biggest issue of applying wavelet analysis is how to choose a desired mother wavelet $\psi(t)$. It is generally agreed that the success of the application of wavelet transform hinges on the selection of a proper wavelet function.

- Because the scale factor m could go from negative infinity to positive infinity, it is impossible to make the time index of the wavelet function, $2^m(t - n2^{-m})$, an integer number simply by digitizing t as $i\Delta_t$, where Δ_t denotes the time sampling interval. This problem prohibits us from using digital computers to evaluate wavelet transform.

Fortunately researchers discovered a relationship between wavelet transform and the perfect reconstruction filter bank, a form of digital filter banks. You can implement wavelet transform with specific types of digital filter banks known as two-channel perfect reconstruction filter banks. Chapter 3, *Digital Filter Banks*, describes the basics of two-channel perfect reconstruction filter banks and the types of digital filter banks used with wavelet analysis.

Digital Filter Banks



This chapter describes the design of two-channel perfect reconstruction filter banks and defines the types of filter banks used with wavelet analysis.

Two-Channel Perfect Reconstruction Filter Banks

Two-channel perfect reconstruction (PR) filter banks were recognized as useful in signal processing for a long time, particularly after their close relationship with wavelet transform was discovered. Since then, it has become a common technique for computing wavelet transform.

Figure 3-1 illustrates a typical two-channel filter bank system. The signal $X(z)$ is first filtered by a filter bank constituted by $G_0(z)$ and $G_1(z)$.

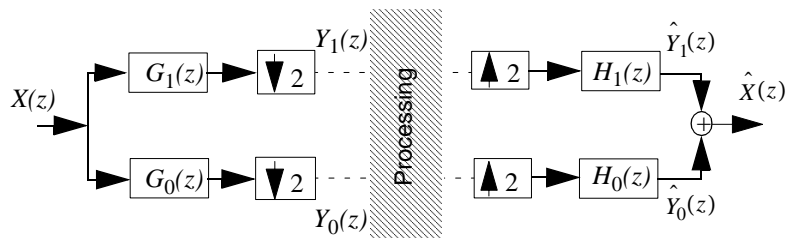


Figure 3-1. Two-Channel Filter Bank



Note: For a finite impulse response (FIR) digital filter $g[n]$, its z -transform is defined as

$$G(z) = \sum_{n=0}^N g[n]z^{-n} = G(e^{j\omega}) = G(\omega) = \sum_{n=0}^N g[n]e^{-j\omega n}$$

where N denotes the filter order. Consequently, the filter length is equal to $N + 1$. Clearly, $\omega = 0$ is equivalent to $z = 1$. $\omega = \pi$ is equivalent to $z = -1$. That is, $G(0)$ and $G(\pi)$ in the frequency domain correspond to $G(1)$ and $G(-1)$ in the z -domain.

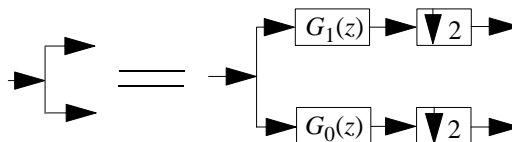
Then, outputs of $G_0(z)$ and $G_1(z)$ are downsampled by 2 to obtain $Y_0(z)$ and $Y_1(z)$. After some processing, the modified signals are upsampled and filtered by another filter bank constructed by $H_0(z)$ and $H_1(z)$. If no processing takes place between the two filter banks, that is, $Y_0(z)$ and $Y_1(z)$ are not altered, the sum of outputs of $H_0(z)$ and $H_1(z)$ is identical to the original signal $X(z)$, except for the time delay. Such a system is commonly referred to as two-channel PR filter banks. $G_0(z)$ and $G_1(z)$ form an analysis filter bank, whereas $H_0(z)$ and $H_1(z)$ form a synthesis filter bank.



Note: $G(z)$ and $H(z)$ can be interchanged. For instance, you can use $H_0(z)$ and $H_1(z)$ for analysis and $G_0(z)$ and $G_1(z)$ for synthesis. $H_0(z)$ and $H_1(z)$ are usually considered as the dual of $G_0(z)$ and $G_1(z)$, and vice versa.

Traditionally, $G_0(z)$ and $H_0(z)$ are lowpass filters, while $G_1(z)$ and $H_1(z)$ are highpass filters, where the subscripts 0 and 1 represent lowpass and highpass filters, respectively. Because the two-channel PR filter banks process $Y_0(z)$ and $Y_1(z)$ at half the sampling rate of the original signal $X(z)$, they attract many signal processing applications.

If you assume the following convention,



then the relationship between two-channel PR filter banks and wavelet transform can be illustrated by Figure 3-2.

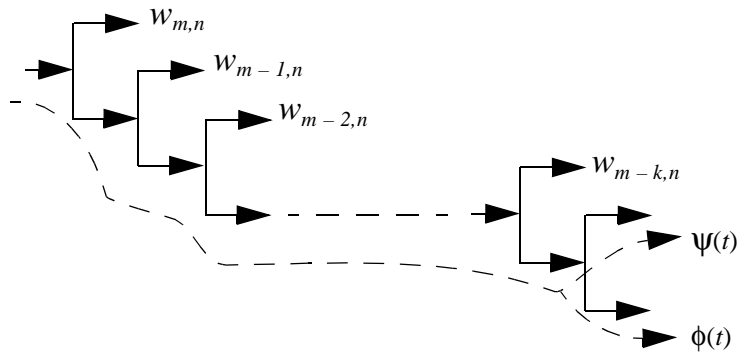


Figure 3-2. Relationship of Two-Channel PR Filter Banks to Wavelet Transform

It is proven (Qian and Chen 1996) that under certain conditions, two-channel PR filter banks are related to wavelet transform in two ways:

- The impulse response of the lowpass filters converges to the scaling function $\phi(t)$. Once you obtain $\phi(t)$, you can compute the mother wavelet function $\psi(t)$ by highpass $\phi(t)$, as shown in Figure 3-2.
- The outputs of each of the highpass filters are *approximations* of the wavelet transform. You can accomplish wavelet transform with a tree of two-channel PR filter banks. The selection of a desirable mother wavelet becomes the design of two-channel PR filter banks.

Figure 3-3 illustrates the relationship of filter banks and wavelet transform coefficients.

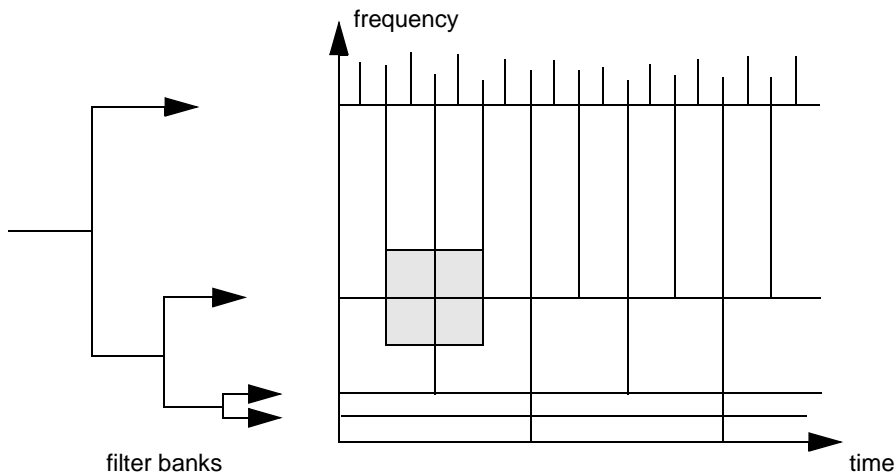


Figure 3-3. Filter Bank and Wavelet Transform Coefficients

The following sections describe the design fundamentals for two types of two-channel PR filter banks, *biorthogonal* and *orthogonal*. In most equations, only results are given without justifications. You can find mathematical treatments in the related references, listed in Appendix A, *References*.

Biorthogonal Filter Banks

Refer back to Figure 3-1, *Two-Channel Filter Bank*. You can define (Strang and Nguyen 1995; Vaidyanathan 1993) the output of the low-channel as

$$\hat{Y}_0(z) = \frac{1}{2}H_0(z)[G_0(z)X(z) + G_0(-z)X(-z)]. \quad (3-1)$$

Similarly, you can define the output of the up-channel as

$$\hat{Y}_1(z) = \frac{1}{2}H_1(z)[G_1(z)X(z) + G_1(-z)X(-z)]. \quad (3-2)$$

Add them together to obtain,

$$\frac{1}{2}[H_0(z)G_0(z) + H_1(z)G_1(z)]X(z) + \frac{1}{2}[H_0(z)G_0(-z) + H_1(z)G_1(-z)]X(-z). \quad (3-3)$$

One term involves $X(z)$ while the other involves $X(-z)$. For perfect reconstruction, the term with $X(-z)$, traditionally called the *alias term*, must be zero. To achieve this, you want,

$$H_0(z)G_0(-z) + H_1(z)G_1(-z) = 0, \quad (3-4)$$

which you accomplish by letting

$$H_0(z) = G_1(-z) \quad \text{and} \quad H_1(z) = -G_0(-z). \quad (3-5)$$

The relationship in equation (3-5) implies that you can obtain $h_0[n]$ by alternating the sign of $g_1[n]$, that is,

$$h_0[n] = (-1)^n g_1[n]. \quad (3-6)$$

Similarly,

$$h_1[n] = (-1)^{n+1} g_0[n]. \quad (3-7)$$

Therefore, $g_1[n]$ and $h_1[n]$ are the highpass filters if $g_0[n]$ and $h_0[n]$ are the lowpass filters. For perfect reconstruction, you also want the first term in equation (3-3), called the *distortion term*, to be a constant or a pure time delay. For example,

$$H_0(z)G_0(z) + H_1(z)G_1(z) = 2z^{-l}, \quad (3-8)$$

where l denotes a time delay.

If you satisfy both equations (3-4) and (3-8), the output of the two-channel filter bank in Figure 3-1, *Two-Channel Filter Bank*, is a delayed version of the input signal, that is,

$$\hat{X}(z) = z^{-l}X(z). \quad (3-9)$$

However, there remains a problem computing $G_0(z)$ and $G_1(z)$ [or $H_0(z)$ and $H_1(z)$]. Once you determine $G_0(z)$ and $G_1(z)$, you can find the rest of the filters with equation (3-5).

You also can write equation (3-5) as,

$$G_1(z) = H_0(-z) \quad \text{and} \quad H_1(z) = -G_0(-z).$$

Substituting it into equation (3-8) yields,

$$G_0(z)H_0(z) - G_0(-z)H_0(-z) = P_0(z) - P_0(-z) = 2z^{-l}, \quad (3-10)$$

where $P_0(z)$ denotes the product of two lowpass filters, $G_0(z)$ and $H_0(z)$, that is,

$$P_0(z) = G_0(z)H_0(z). \quad (3-11)$$

Equation (3-10) indicates that all odd terms of the product of two lowpass filters, $G_0(z)$ and $H_0(z)$, must be zero except for order l , where l must be odd. But, even order terms are arbitrary. You can summarize these observations by the following formula:

$$p_0[n] = \begin{cases} 0 & n \text{ odd and } n \neq l \\ 2 & n = l \\ \text{arbitrary} & n \text{ even} \end{cases}. \quad (3-12)$$

This reduces the design of two-channel PR filter banks to two steps:

1. Design a filter $P_0(z)$ satisfying equation (3-12).
2. Factorize $P_0(z)$ into $G_0(z)$ and $H_0(z)$. Then use equation (3-5) to compute $G_1(z)$ and $H_1(z)$.

Two types of filters are frequently used for $P_0(z)$:

- an equiripple *halfband* filter (Vaidyanathan and Nguyen 1987)
- a *maximum flat* filter

In the first filter, the halfband refers to a filter in which $\omega_s + \omega_p = \pi$, where ω_s and ω_p denote the passband and stopband frequencies, respectively, as in Figure 3-4, *Halfband Filter*.

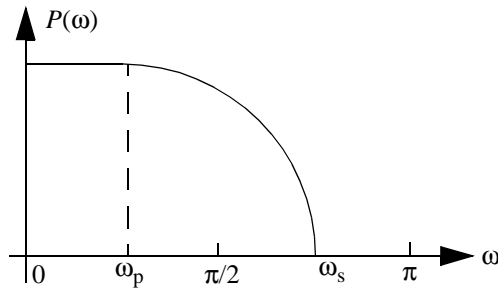


Figure 3-4. Halfband Filter

The second filter is the maximum flat filter with a form according to the following formula:

$$P_0(z) = (1 + z^{-1})^{2p} Q(z), \quad (3-13)$$

which has $2p$ zeros at $z = -1$ or $\omega = \pi$. If you limit the order of the polynomial $Q(z)$ to $2p - 2$, then $Q(z)$ is unique.



Note:

The maximum flat filter here differs from the Butterworth filter. The low-frequency asymptote of the Butterworth filter is a constant, while the maximum flat filter is not.

In all cases, the product of lowpass filter $P_0(z)$ is a type I filter, that is,

$$p_0[n] = p_0[N - n] \quad N \text{ even}, \quad (3-14)$$

where N denotes the filter order.

Consequently, the number of coefficients $p_0[n]$ is odd, $N + 1$.

Figure 3-5 plots the zeros distribution of a maximum flat filter $P_0(z)$ for $p = 3$.

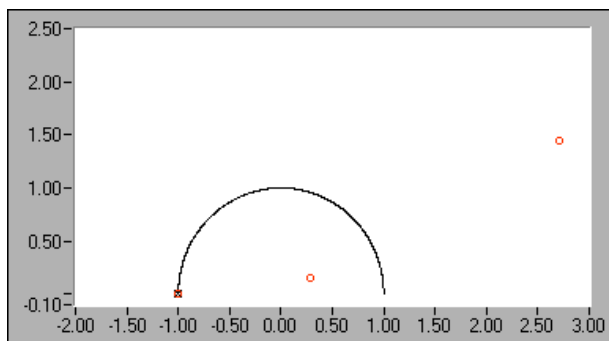


Figure 3-5. Zeros Distribution for $(1 - z^{-1})^6 Q(z)$

There are six zeros at $\omega = \pi$. In this case, the order of the unique polynomial $Q(z)$ is four, which contributes another four zeros (not on the unit circle). If you let three zeros at $\omega = \pi$ go to $G_0(z)$ according to the formula,

$$G_0(z) = (1 + z^{-1})^3, \quad (3-15)$$

and the rest of the zeros go to $H_0(z)$, you obtain *B-spline filter banks*. The coefficients of $g_0[n]$ and $g_1[n]$ and the corresponding scaling function and mother wavelet are plotted in Figure 3-6, *B-spline Filter Bank*. Both the scaling function and mother wavelet generated by $g_0[n]$ and $g_1[n]$ are smooth.

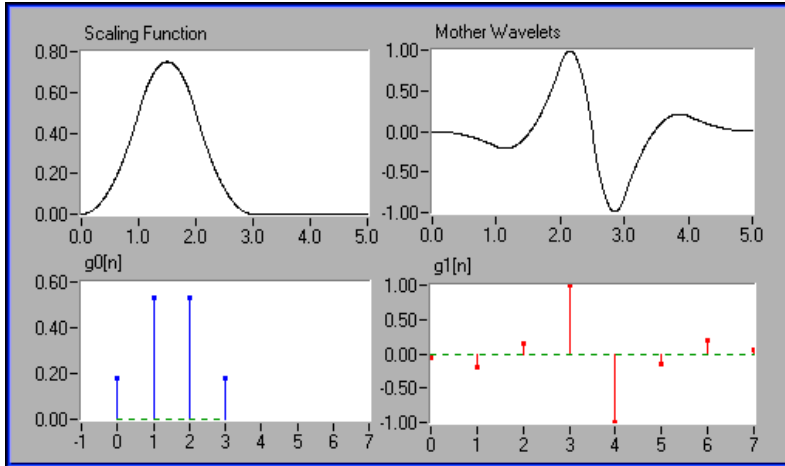


Figure 3-6. B-Spline Filter Bank

Figure 3-7 depicts the dual filter bank, $h_0[n]$ and $h_1[n]$, and corresponding scaling function and mother wavelet. You also can use $h_0[n]$ and $h_1[n]$ for analysis. In Figure 3-7, the tree filter banks constituted by $h_0[n]$ and $h_1[n]$ do not converge.

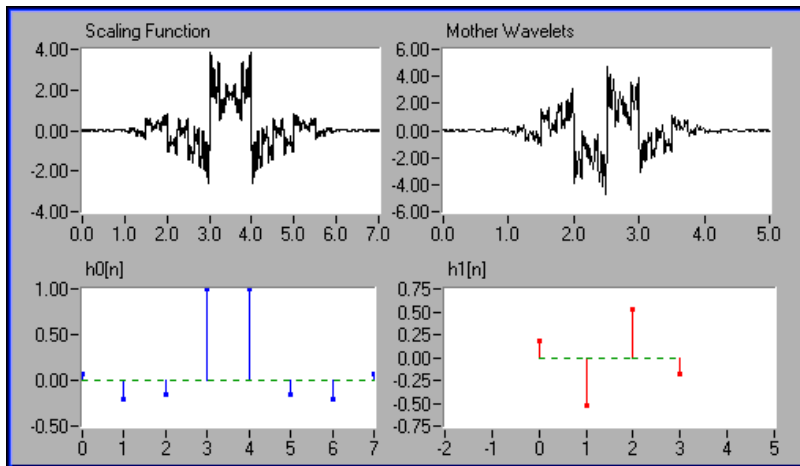


Figure 3-7. Dual B-Spline Filter Bank

You must remember that two-channel PR filter banks do not necessarily correspond to the wavelet transform. The wavelet transformations are special cases of two-channel PR filter banks. The conditions of two-channel PR filter banks are more moderate than those for the wavelet transform.

Finally, the analysis filter banks and synthesis filter banks presented in this section are orthogonal to each other. That is

$$\sum_n g_i[n-2k]h_i[n] = \delta(k) \quad (3-16)$$

and

$$\sum_n g_i[n-2k]h_l[n] = 0 \quad i \neq l, \forall k$$

The filters banks that satisfy equation (3-16) are traditionally called *biorthogonal filter banks*. In addition to equation (3-16), if the analysis filter banks also satisfy the following equations

$$\sum_n g_i[n-2k]g_i[n] = \delta(k) \quad (3-17)$$

and

$$\sum_n g_i[n-2k]g_l[n] = 0 \quad i \neq l, \forall k$$

the resulting filter banks are called *orthogonal filter banks*. Orthogonal filter banks are special cases of biorthogonal filter banks.

Orthogonal Filter Banks

As shown in the preceding section, once you determine $P_0(z)$, the product of two lowpass filters, you must factorize $P_0(z)$ into $G_0(z)$ and $H_0(z)$. Evidently, the combinations of zeros are not unique. Different combinations lead to different filter banks. Sometimes $G_0(z)$ and $G_1(z)$ work well, but $H_0(z)$ and $H_1(z)$ might not (see Figure 3-6, *B-spline Filter Bank*, and Figure 3-7, *Dual B-spline Filter Bank*). One way to make this process easier is to limit the selections into a subset. The most effective approach is to require $G_0(z)$ and $G_1(z)$, and thereby $H_0(z)$ and $H_1(z)$, to be orthogonal, as described by equation (3-17).

These constraints reduce the filter banks design to one filter design. Once you select $G_0(z)$, you can easily find all other filters. The constraints imposed by equation (3-17) not only guarantee that both filter banks have the same performance, but also provide other advantages. For example, many applications demonstrate that the lack of orthogonality complicates quantization and bit allocation between bands, eliminating the conservation of energy.

To achieve equation (3-17), let

$$G_1(z) = -z^{-N}G_0(-z^{-1}), \quad (3-18)$$

which implies that $g_1[n]$ is *alternating flip* of $g_0[n]$, that is,

$$(g_1[0], g_1[1], g_1[2], \dots) = (g_0[N], -g_0[N-1], g_0[N-2], \dots). \quad (3-19)$$

Equation (3-18) implies that for orthogonal wavelets and filter banks,

$$H_0(z) = z^{-N}G_0(z^{-1}), \quad (3-20)$$

where you use the relation in equation (3-5). Consequently, equation (3-11) can be written as,

$$P_0(z) = z^{-N}G_0(z)G_0(z^{-1}). \quad (3-21)$$

If,

$$G_0(z)G_0(z^{-1}) = P(z), \quad (3-22)$$

then,

$$P(e^{j\omega}) = \sum_{n=-N}^N p[n]e^{-j\omega n} = \left| \sum_{n=0}^N g_0[n]e^{-j\omega n} \right|^2, \quad (3-23)$$

which implies that $P(z)$ is non-negative.

Similar to biorthogonal cases, the selection of $P_0(z)$ in orthogonal cases is dominated by maximum flat and equiripple halfband filters.

However, because of constraints imposed by equation (3-23), $P_0(z)$ must be the time-shifted non-negative function $P(z)$. While the

maximum flat filter in equation (3-13) ensures this requirement, special care must be taken when $P_0(z)$ is an equiripple halfband filter.

Figure 3-8 plots the third order Daubechies filter banks and wavelets. It is derived from the same maximum flat filter as that depicted in Figure 3-5, *Zeros Distribution for $(1 - z^{-1})^6 Q(z)$* , but in this case, $G_0(z)$ contains three zeros at $\omega = \pi$ as well as all zeros inside of the unit circle, therefore possessing minimum phase. Because of the orthogonality, its dual filter bank has the same convergence property. Compared to the B-spline cases in Figure 3-6, *B-spline Filter Bank*, and Figure 3-7, *Dual B-spline Filter Bank*, the third order Daubechies wavelet and scaling function are less smooth than that of $G_0(z)$ and $G_1(z)$ (see Figure 3-6, *B-spline Filter Bank*), but much smoother than that of $H_0(z)$ and $H_1(z)$ (see Figure 3-7, *Dual B-spline Filter Bank*).

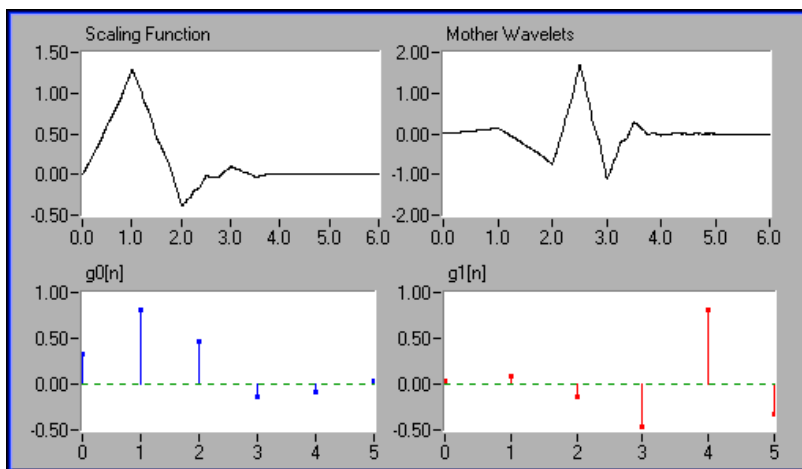


Figure 3-8. Third Order Daubechies Filter Banks and Wavelets

Two-Dimensional Signal Processing

The preceding sections introduced two-channel PR filter banks for one-dimensional (1D) signal processing. In fact, two-channel PR filter banks also can be used for two-dimensional (2D) signals as shown in Figure 3-9. In this case, you process rows first and then columns. Consequently, one 2D array splits to the following four 2D sub-arrays:

- low-low
- low-high
- high-low
- high-high

Each sub-arrays is a quarter size of the original 2D signal.

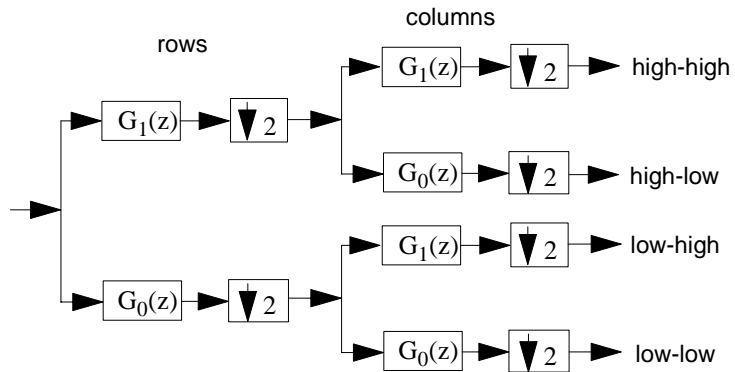


Figure 3-9. 2D Signal Processing

Figure 3-10, *2D Image Decomposition*, illustrates 2D image decomposition by two-channel PR filter banks. In this case, the original 128-by-128 2D array is decomposed into four 64-by-64 sub-arrays, but the total size of the four sub-arrays is the same as the original 2D array. For example, the total number of elements in the four sub-arrays is 16,384, which equals 128 times 128. However, if the filters are selected properly, you can make sub-arrays such that the majority elements are small enough to be neglected. Consequently, you can use a fraction of the entire wavelet transform coefficients to recover the original image and thereby achieve data compression. In this example, you use the largest 25% wavelet transform coefficients to rebuild the original image. Among them, the majority (93.22%) are from the

low-low sub-array. The remaining three sub-arrays contain limited information. If you repeat the wavelet transform to the low-low sub-array, you can reduce the compression rate further.

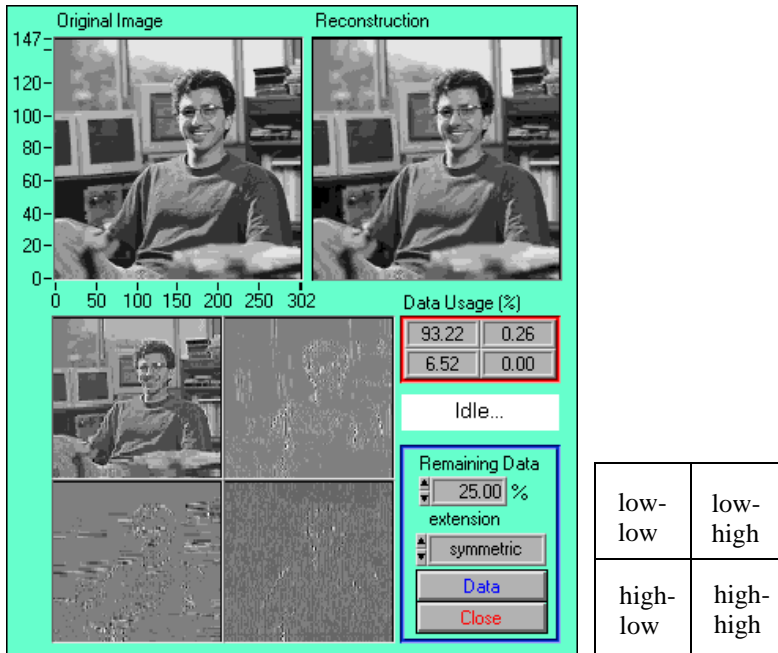
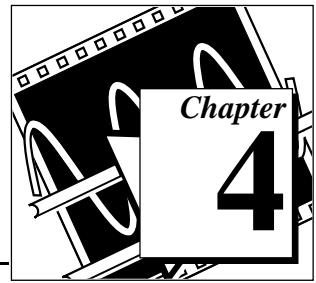


Figure 3-10. 2D Image Decomposition

Using the Wavelet and Filter Bank Design Toolkit



This chapter describes the architecture of the WFBD Toolkit, lists the design procedures, and describes some applications you can create with the WFBD Toolkit.

This chapter also describes how to use WFBD Toolkit to design a desired wavelet and filter bank. Although you can use it without understanding the fundamentals of wavelets and filter banks introduced in the previous two chapters, for the best results, it is highly recommended that you review those chapters before running the WFBD Toolkit.

Wavelet and Filter Bank Design

Figure 4-1, *Design Procedure for Wavelets and Filter Banks*, lists the choices of wavelets and filter banks available in the WFBD Toolkit. The design of wavelets and filter banks contains three steps.

1. Determine the type of wavelet and filter banks—orthogonal or biorthogonal.
2. Select the type of filters based on the product $P_0(z)$ of the two lowpass filters, $G_0(z)$ and $H_0(z)$.
3. Factorize $P_0(z)$ into $G_0(z)$ and $H_0(z)$.

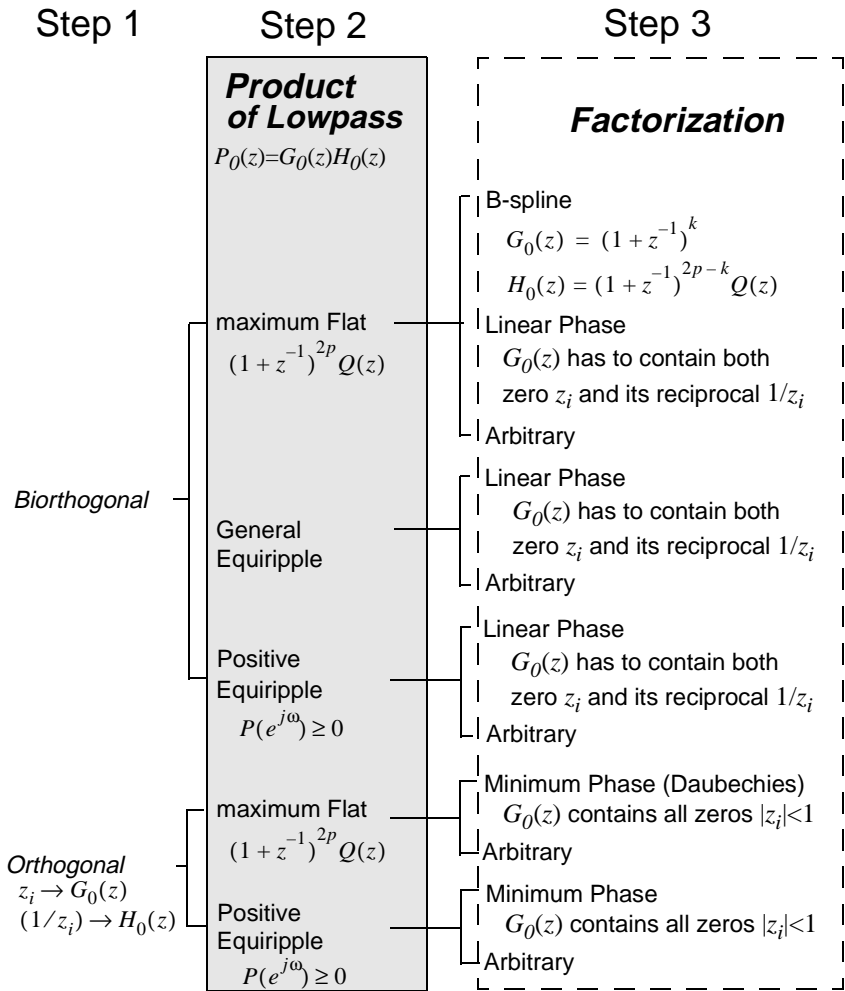


Figure 4-1. Design Procedure for Wavelets and Filter Banks

Because all filters in the WFBD Toolkit act as real-valued FIR filters, the zeros of $P_0(z)$, $G_0(z)$, and $H_0(z)$ are symmetrical in the z -plane. This implies that for any zero z_i , there always exists z_i^* , if z_i is complex (see Figure 4-2). Hence, you only need to deal with half of the z -plane. Once you select z_i , the program automatically includes its complex conjugate z_i^* .

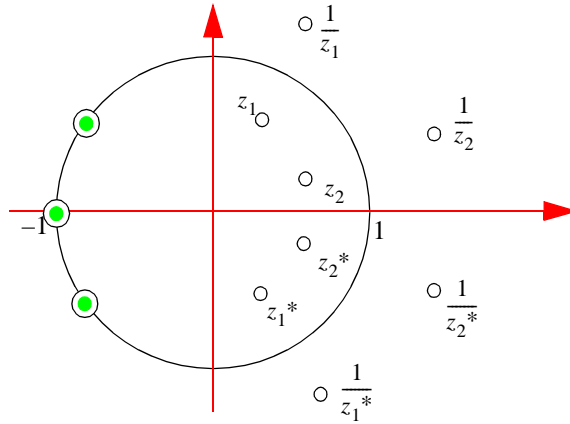


Figure 4-2. Non-Negative Equiripple Halfband Filter

For both orthogonal and biorthogonal wavelets and filter banks, you can use either maximum flat or equiripple filters for the product of lowpass filters $P_0(z)$. The maximum flat filters have good frequency attenuation, but wider transition band. Because the filter has the form

$$P_0(z) = (1 + z^{-1})^{2p} Q(z),$$

you can impose as many zeros at $\omega = \pi$ as you like. On the other hand, the halfband equiripple filters only can have a pair of zeros at $\omega = \pi$, which gives the equiripple type filters slower convergence rates. However, it is easier to balance the frequency attenuation and transition band for an equiripple filter. For a given transition band, the attenuation is proportional to the filter order of $P_0(z)$. The larger the order, the better the attenuation.

Once you determine $P_0(z)$, you must factorize it into the lowpass filters, $G_0(z)$ and $H_0(z)$. The combination of zeros is not unique. Table 4-1,

Filter Comparison, summarizes some important filter combinations, while Figure 4-3 through Figure 4-5 plot the zeros distribution.

Table 4-1. Filter Comparison

Filter	Zeros	Property	Illustration
Real	complex conjugate symmetrical	easy to implement	Figure 4-2
Minimum Phase	$ z_i \leq 1$ for all i (on or inside of unit circle) all zeros have to be on or inside of the unit circle	important for control systems	Figure 4-3
Linear Phase	must contain both z_i and its reciprocal $1/z_i$ the pair of reciprocals must be in the same filter	desirable for image processing	Figure 4-4
Orthogonal	cannot have z_i and its reciprocal $1/z_i$ simultaneously z_i and its reciprocal has to be in the separated filters, contradictory to linear phase	analysis and synthesis have the same performance convenient for bit allocation and quantization error control not linear phase even length (N odd)	Figure 4-5

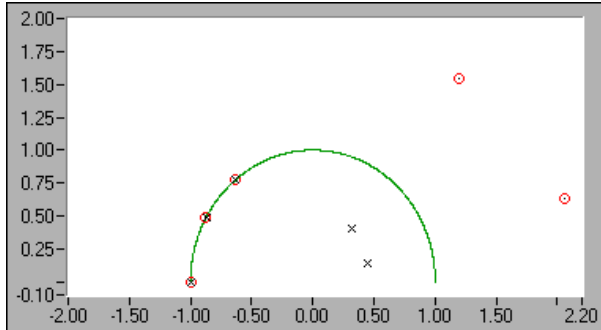


Figure 4-3. Minimum Phase Filter

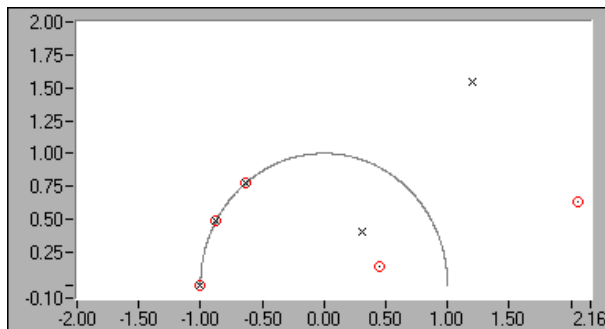


Figure 4-4. Linear Phase Filter

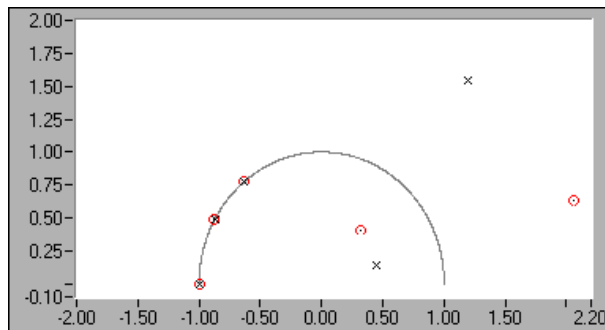


Figure 4-5. Orthogonal Filter



Note: *The conditions for linear phase and orthogonality are contradictory. In general, you cannot achieve linear phase and orthogonality simultaneously.*

Design Panel

If you use LabVIEW with the WFBD Toolkit, you design your wavelets and filter banks by using the Design Panel. To access the Design Panel, open WaveMain.llb and click on Design Panel. This VI opens the panel shown in Figure 4-6. If you do not have LabVIEW, run wfbd.exe. The first panel to appear is the Design Panel. Use the Design Panel to complete the three steps in designing wavelets and filter banks. Refer to Chapter 3, *Digital Filter Banks*, for more background information on wavelets and filter banks.

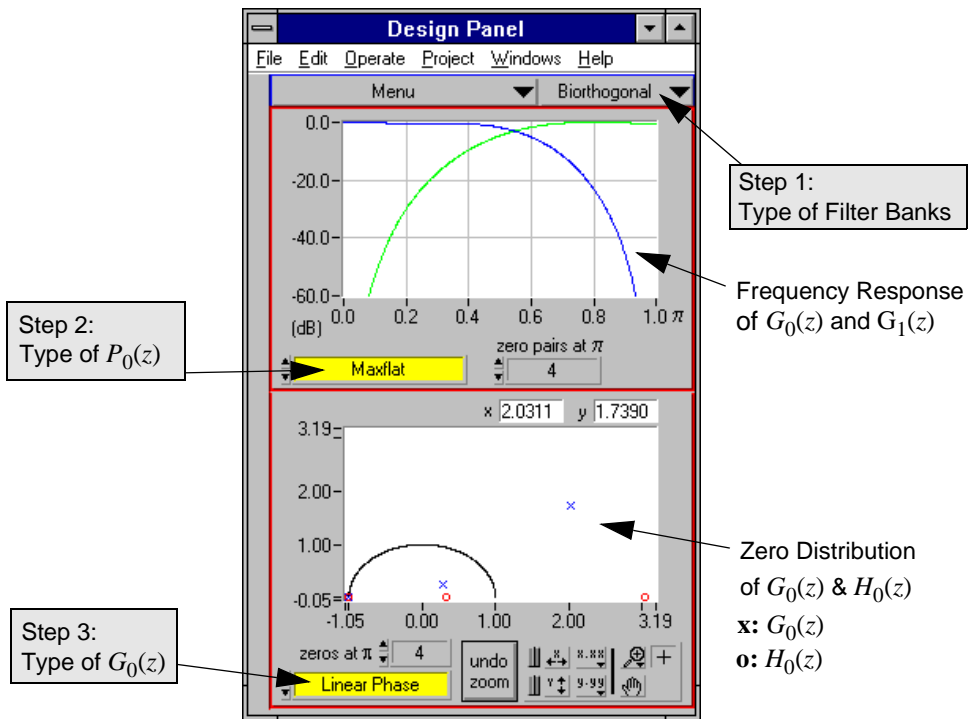


Figure 4-6. Design Panel

The three steps in designing wavelets and filter banks are as follows:

1. Select the type of filter bank.

You can select from two types of wavelets and filter banks, *orthogonal* and *biorthogonal*. You can design orthogonal filters and wavelets easily because they involve fewer parameters, but the filter banks cannot be linear phase.

2. Find the product $P_0(z)$.

$P_0(z)$ denotes the product of $G_0(z)$ and $H_0(z)$, that is,
 $P_0(z) = G_0(z)H_0(z)$.



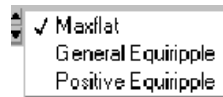
Note:

*In the WFBK Toolkit, **G** denotes an analysis filter and **H** denotes a synthesis filter. The subscript 0 denotes a lowpass filter and 1 denotes a highpass filter.*

In orthogonal filter banks, $P_0(z)$ can be either maximum flat or positive equiripple.



In biorthogonal filter banks, $P_0(z)$ can be maximum flat, general equiripple, or positive equiripple.



The maximum flat filter differs from the Butterworth filter. It has a form,

$$P_0(z) = (1 + z^{-1})^{2p} Q(z). \quad (4-1)$$

The parameter p is controlled by the button **zero pairs at π** . $Q(z)$ is a $2p - 2$ order polynomial, which you can uniquely determine if p is decided. Therefore, the total number of coefficients of $P_0(z)$ is $4p - 1$.

The equiripple is further divided into the general equiripple and positive equiripple filters; however, you only can select general equiripple filters for biorthogonal filter banks. Although both are halfband filters, that is, the sum of the normalized passband and stopband frequencies equals 0.5, the Fourier transform of the positive equiripple filter $p_0[n]$ is always real and non-negative.

There are two parameters associated with equiripple filters, the **# of taps** and normalized **passband** frequency as illustrated in Figure 4-7.

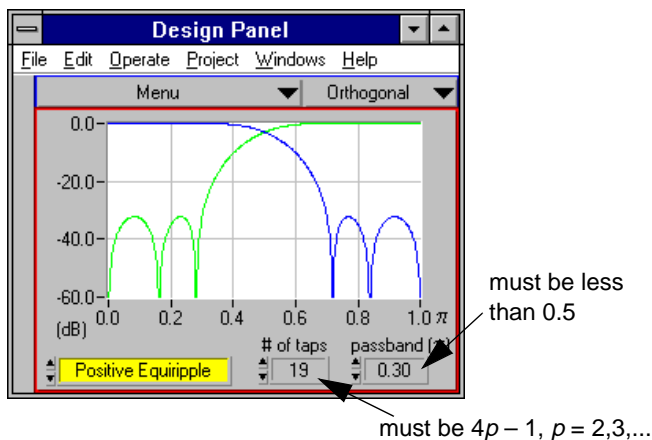


Figure 4-7. Equiripple Filter

of taps—Number of coefficients of $P_0(z)$. Because $P_0(z)$ is a type I FIR filter, the length of $P_0(z)$ must be $4p - 1$, where $p = 2, 3, \dots$

passband—Normalized cutoff frequency of $P_0(z)$, which must be less than 0.5.

Once you determine the product of lowpass filters $P_0(z)$, you must factorize $P_0(z)$ into $G_0(z)$ and $H_0(z)$.

- Factorize $P_0(z)$ into $G_0(z)$ and $H_0(z)$.

The plot in the lower half of the Design Panel in Figure 4-6 displays the zeros distribution of $P_0(z)$. Because all the zeros are symmetrical with respect to x-axis, only the upper half of the plane is displayed. The selection of $G_0(z)$ and $H_0(z)$ is to group different zeros. The blue circle represents the zeros in $G_0(z)$ and the red cross represents the zeros in $H_0(z)$.

To select a zero, place the cursor on the zero that you want to choose and click the left mouse button. This switches the zeros from $G_0(z)$ to $H_0(z)$, and vice versa. All the zeros go to either $G_0(z)$ or $H_0(z)$. The plot in the upper half of the panel displays the frequency response of filters $G_0(z)$ and $G_1(z)$. $G_1(z)$ is the sign-alternated version of $H_0(z)$. Therefore, $G_1(z)$ must be a highpass filter if $H_0(z)$ is a lowpass filter.

If two zeros are too close to choose, use the **Zoom Tool** palette, located in the lower right corner of the Design Panel to zoom in on these zeros until you can identify these zeros. For maximum flat filters, there are multiple zeros at $z = 0$. Use the **zeros at π** button to control how many zeros at $z = 0$ go to $G_0(z)$.

For the given $P_0(z)$, you have four choices for $G_0(z)$ and $H_0(z)$.

- **Linear Phase**—Any zero and its reciprocal must belong to the same filter.
- **Minimum Phase**— $G_0(z)$ contains all the zeros inside the unit circle. When $P_0(z)$ is maximum flat and $G_0(z)$ is minimum phase, the resulting wavelets are traditionally named *Daubechies* wavelets.
- **B-Spline**—Only available when the filter is biorthogonal and maximum flat. In this case,

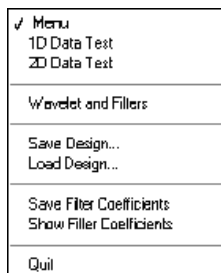
$$G_0(z) = (1 + z^{-1})^k \quad H_0(z) = (1 + z^{-1})^{2p-k} Q(z),$$

where k is decided by the button, **zeros at π** . p is decided by the button, **zero pairs at π** , as mentioned earlier.

- **Arbitrary**—No specific constraints.

Once you decide the type of $G_0(z)$ and $H_0(z)$, the program automatically computes the constraints. For example, once you select a zero, its reciprocal automatically is included if you choose $G_0(z)$ for linear phase. All possible design combinations provided by this panel are summarized in Figure 4-1, *Design Procedure for Wavelets and Filter Banks*, of this chapter.

The Design Panel also provides other utilities. You can access these utilities from the **Menu** ring.



1D Data Test—Invokes the 1D_Test panel. Use this panel to examine the performance of the wavelet and filter bank you designed with the Design Panel.

2D Data Test—Invokes the 2D_Test panel. Use this panel to test the wavelet and filter bank you designed for a two-dimensional image.

Wavelet and Filters—Invokes the Wavelets and Filters panel. Use this panel to display the mother wavelet, scaling functions, and the filter coefficients.

Save Design...—Saves the design information in a binary file.

Load Design...—Loads a saved design information file.

Save Filter Coefficients—Saves the designed analysis filter coefficients and synthesis filter coefficients in a text file.

Show Filter Coefficients—Displays a table listing the designed analysis and synthesis filter coefficients.

The **1D Data Test**, **2D Data Test**, and **Wavelet and Filters** menu options access other panels, described in the remaining sections of this chapter.

One-Dimensional Data Test

The 1D_Test panel is shown in Figure 4-8. You access this panel by selecting **1D Data Test** from the **Menu** ring of the Design Panel. Use this panel to test the designed wavelet and filter bank for 1D data.

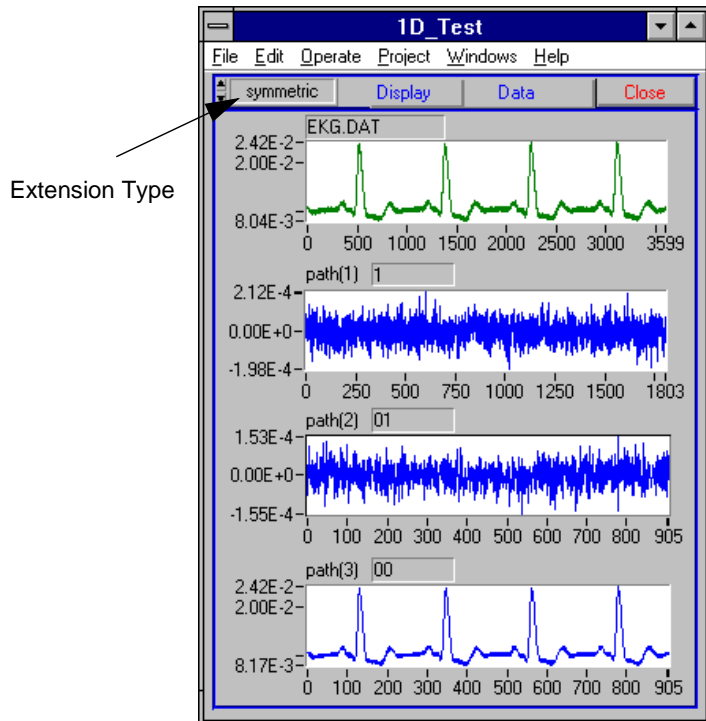


Figure 4-8. 1D_Test Panel

Extension Type—Determines the padding method for the data.

Zero Padding—Adds zeros at the beginning and end of the original data.

Symmetric Extension—Symmetrically adds the input data at the beginning and end of the original data.

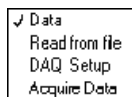
In both cases, you can add the number of points at the beginning and the end of the original data with the following formula:

$$\frac{N_p - 1}{2} = \frac{N_g + N_h}{2} - 1$$

where N_p is the number of coefficients of $P_0(z)$, N_g is the number of coefficients of filter $G_0(z)$, and N_h is the number of coefficients of filter $H_0(z)$.

Display—Displays either a time waveform or a histogram.

Data—Provides the following choices:



Read from file—Reads one-dimensional input data from a text file.

Acquire Data—Uses the data acquisition board specified in the DAQ Setup panel to acquire a block of data and then analyze it. You must run the DAQ Setup panel before you acquire any data.

DAQ Setup—Invokes the DAQ Setup panel, shown in Figure 4-9, *DAQ Setup Panel*. Use this panel to configure your data acquisition board.

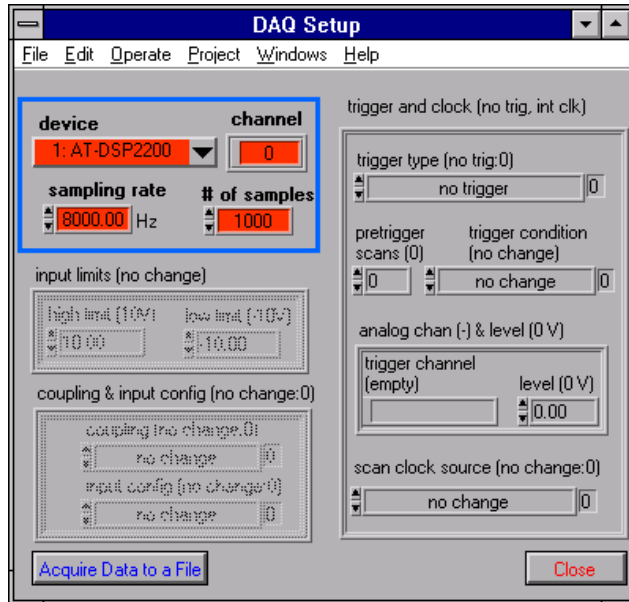


Figure 4-9. DAQ Setup Panel

Usually, you only need to configure the following parameters:

device—Indicates which DAQ board you are using to acquire data.

channel—Indicates which channel on your DAQ board you are using to acquire data. You only can specify one channel.

sampling rate—Indicates how fast you sample your data.

of samples—Indicates how many samples to acquire.

Acquire Data to a File—Acquires a block of data and saves it to a text file.



Note:

The remaining parameters on the DAQ Setup panel are for advanced data acquisition users. Refer to the corresponding DAQ manual from National Instruments Corporation for detailed information about these parameters.

path—Specifies a path for the output for that plot. You can type in any path. While 0 represents passing a lowpass filter $G_0(z)$, 1 represents passing a highpass filter $G_1(z)$. An example of this is demonstrated in Figure 4-10.

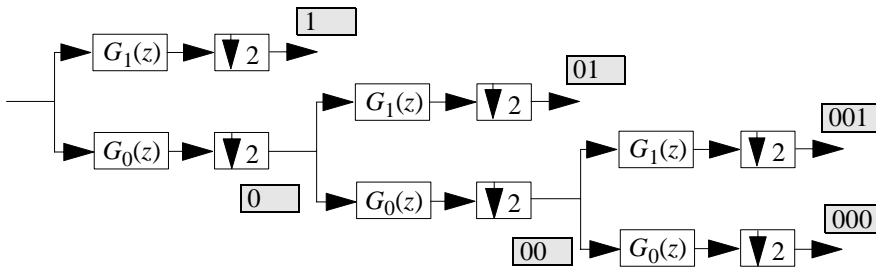


Figure 4-10. Specifying a Path

Two-Dimensional Data Test

The 2D_Test panel is shown in Figure 4-11. You access this panel by selecting **2D Data Test** from the **Menu** ring of the Design Panel. You can use this panel to test the designed wavelet and filter banks for a 2D image.

As introduced in Chapter 3, *Digital Filter Banks*, by applying wavelet transform, one image is broken into four subimages. The four subimages are arranged as follows:

low-low	low-high
high-low	high-high



Figure 4-11. 2D_Test Panel

Data Usage (%)—Displays the percentage of the wavelet coefficients from each of the subimages used to restore the image. In Figure 4-11, the original image size is 353 times 148, or 52,244 data samples. The reconstruction uses 25% of the largest wavelet transform coefficients, that is, 25% of 52,244 or 13,061 samples. Among these samples, 93.05% are from the low-low subimage (12,147 coefficients), 6.23% are from the low-high subimage (814 coefficients), and 0.85% are from the high-low subimage (111 coefficients). None are from the high-high subimage.

Remaining Data—Displays your choice for the percentage of the largest data from the four subimages, which is used for the reconstruction.

extension—Determines the padding method for the data.

Zero Padding—Adds zeros at the beginning and end of the original data.

symmetric extension—Symmetrically adds the input data at the beginning and end of the original data.

In both cases, the number of points added at the beginning and the end of the original data are found by the following formula:

$$\frac{N_p - 1}{2} = \frac{N_g + N_h}{2} - 1$$

where N_p is the number of coefficients of $P_0(z)$, N_g is the number of coefficients of filter $G_0(z)$, and N_h is the number of coefficients of filter $H_0(z)$.

Data—Reads a 2D spreadsheet text file or stand image file, such as a .TIF or .BMP file. Be sure to choose the correct data type when reading the data file.

Wavelets and Filters

As illustrated in Figure 4-12, *Wavelets and Filters*, the Wavelets and Filters panel displays the mother wavelet and scaling functions, as well as the filter coefficients, $G_0(z)$, $G_1(z)$, $H_0(z)$, and $H_1(z)$. You access this panel by selecting **Wavelet and Filters** from the **Menu** ring of the Design Panel. Filter banks do not always converge to a wavelet function. This panel helps you examine whether the filter bank you selected converges.

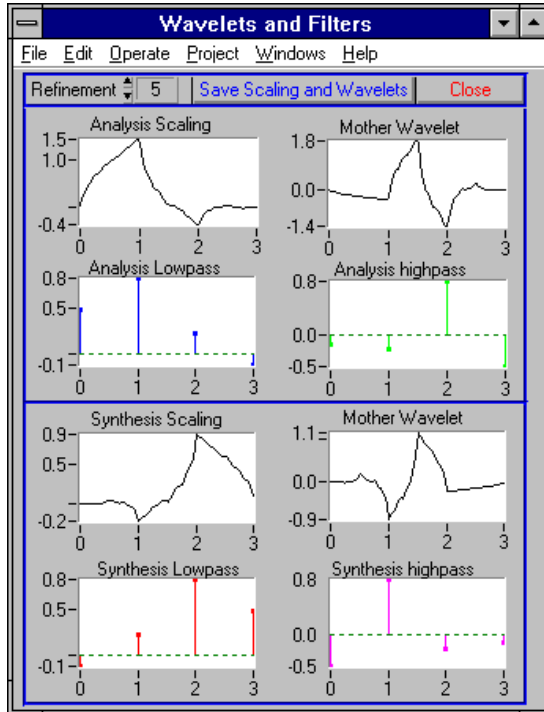


Figure 4-12. Wavelets and Filters

Refinement—Defines how many levels to go through to compute the wavelet and scaling function. A proper wavelet usually converges after 4 or 5 levels.

Save Scaling and Wavelets—Saves the scaling functions and wavelets for the analysis and synthesis filters in a text file.

Create Your Own Applications

You can save all design results as text files for use in other applications. Moreover, the WFBD Toolkit includes three LabVIEW VI libraries, `WaveMain.llb`, `Wavesubs.llb`, and `Wavemisc.llb`, for LabVIEW users. These three libraries contain the basic wavelet analysis VIs, such as 1D and 2D analysis and synthesis filters, as well as many other useful functions. For more information regarding these VIs, refer to Chapter 5, *Function Reference*. Consequently, you can test the design not only with the two built-in testing panels as described in the previous sections, but also from your own applications.

This section introduces a few applications that you can develop with the help of this toolkit. You can create all the examples described in this section with or without LabVIEW, because you always can incorporate the filter bank coefficients into your applications from previously saved text files.

Wavelet Packet Analysis

The preceding sections introduce wavelet analysis in which the signal is continuously decomposed in the lowpass path, similar to the path shown in Figure 3-2, *Relationship of Two-Channel PR Filter Banks to Wavelet Transform*. You also can apply other decomposition schemes to the signal and still maintain the perfect reconstruction. Figure 4-13, *Full Path of a Three Level PR Tree*, illustrates the full path for a three level decomposition. For example, you can decompose the signal X as 0, 100, 101, and 11, then use those coefficients to reconstruct the original signal by the synthesis filter banks as shown in Figure 4-14, *Wavelet Packet*. Although you do not follow the ordinary wavelet decomposition scheme discussed in the earlier chapters in this case, you can still fully recover the original signal X if the coefficients are not altered. This generalized wavelet decomposition is called a *wavelet packet*, which offers a wider range of possibilities for signal processing.

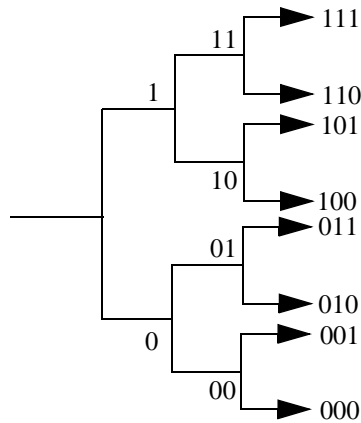


Figure 4-13. Full Path of a Three Level PR Tree

The path is completely determined by the applications on the hand. One common method is to check each node of the decomposition tree and quantify the information. Then, continue to decompose those nodes which contain more information. Such technique is traditionally called an *entropy-based criterion*.

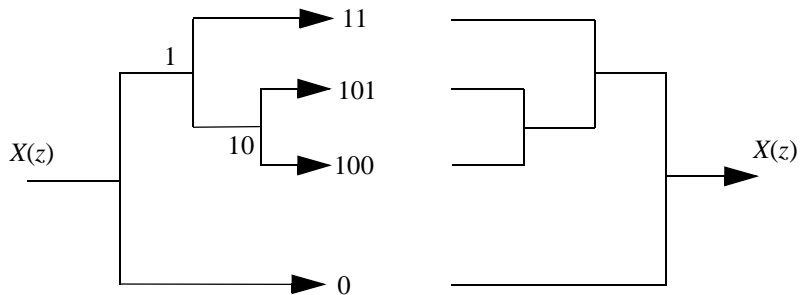


Figure 4-14. Wavelet Packet

On-Line Testing Panel

To assist you with testing your own applications, the main design panel saves the filter coefficients as the following global variables in the Wavelet Global VI:

- *Analysis Filter Coefficients*—Contains coefficients of $G_0(z)$ and $G_1(z)$.
- *Synthesis Filter Coefficients*—Contains coefficients of $H_0(z)$ and $H_1(z)$.

These variables simultaneously change as you change the design. If you incorporate those parameters into your own application, you can see the effect of the different design. Figure 4-15, shown below, illustrates how LabVIEW uses these two parameters to implement a Wavelet Packet similar to the one displayed in Figure 4-14, *Wavelet Packet*.

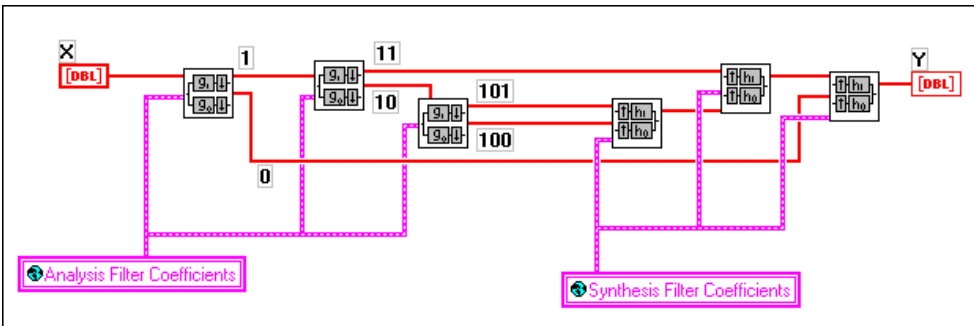
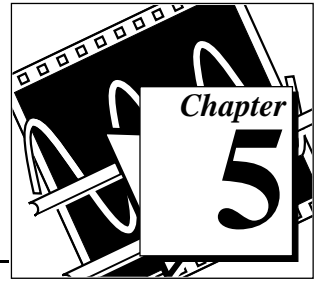


Figure 4-15. Implementation of a Wavelet Packet

Denosing

One of the most successful applications of wavelet transform is called *denoising*. This application works by first taking the wavelet transform of the signal, setting the coefficients below a certain threshold to zero, and finally inverting the transform to reconstruct the original signal. The resulting signal has less noise interference if the threshold is set properly. You can find a detailed description of such wavelet transform-based denoising in D. L. Donoho's article "De-noise by soft thresholding" in the May 1995 issue of *IEEE Transactions Information Theory*.



Function Reference

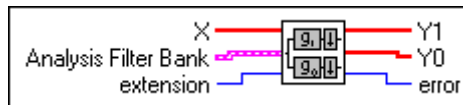
This chapter describes the VIs in the WFBD Toolkit package, the instrument driver for LabWindows/CVI, and the functions in the DLLs.

LabVIEW VI Applications

This section contains the VIs you can use when operating the WFBD Toolkit with LabVIEW.

2-Channel Analysis Filter Bank VI

This VI computes the outputs of an analysis filter bank.



[DBL]

X is the input data array.

[DBL]

Analysis Filter Bank contains the analysis filter bank coefficients.

[DBL]

Lowpass contains the lowpass analysis filter coefficients G_0 .

[DBL]

Highpass contains the highpass analysis filter coefficients G_1 .

[I32]

extension decides the initial condition X_i and final condition X_f .

extension has two options:

0: zero padding—all the initial condition and final condition are zeros.

1: symmetric extension—extends signal **X** symmetrically as the initial condition and final condition.

[DBL]

Y1 is the output of analysis highpass filter.

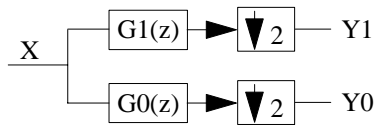
[DBL]

Y0 is the output of analysis lowpass filter.

[I32]

error. See Appendix B, *Error Codes*, for a description of the error.

The VI performs the following operation:



If we define input as \mathbf{X} , outputs as $\mathbf{Y0}$ and $\mathbf{Y1}$, then,

$$y0_n = \sum_{i=1}^{n_{g0}} x1_{2n+i} g_{n_{g0}-i}$$

$$y1_m = \sum_{i=1}^{n_{g1}} x1_{2n+i} g_{1_{n_{g1}-i}}$$

where $n = 0, 1, \dots, n_{y0} - 1$, $m = 0, 1, \dots, n_{y1} - 1$,

n_{y0} is the length of output $\mathbf{Y0}$, $n_{y0} = \text{ceil}((n_x + n_{g1} - 1)/2)$,

n_{y1} is the length of output $\mathbf{Y1}$, $n_{y1} = \text{ceil}((n_x + n_{g0} - 1)/2)$,

n_x is the size of input array \mathbf{X} ,

n_{g0} is the size of $G0$,

n_{g1} is the size of $G1$,

$G0$ is the analysis lowpass filter coefficients,

$G1$ is the analysis highpass filter coefficients, and

$X1$ is the input signal \mathbf{X} plus the initial condition X_i and final condition X_f , which is decided by the selection of **extension**.

If **extension** is zero padding, zeros are added at the beginning and the end of \mathbf{X} .

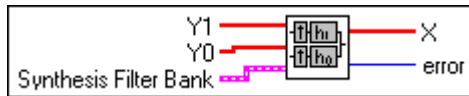
Figure 5-1 shows how the VI constructs $X1$ in this case. All the elements in X_i and X_f are zeros and the lengths of X_i and X_f are

$$n_p = \frac{n_{g0} + n_{g1} - 1}{2}$$

After constructing X1, this VI computes outputs **Y0** and **Y1** the same way as the outputs in the Decimation Filter VI. Refer to the section *Decimation Filter VI* in this chapter to learn how to compute **Y0** and **Y1**.

2-Channel Synthesis Filter Bank VI

This VI computes the outputs of a synthesis filter bank.



[DBL]

Y1 is the input data array for the synthesis highpass filter, often the output from the analysis highpass filter.

[DBL]

Y0 is the input data array for the synthesis lowpass filter, often the output from the analysis lowpass filter.

[DBL]

Synthesis Filter Bank contains the synthesis filter coefficients.

[DBL]

Lowpass contains the lowpass synthesis filter coefficients.

[DBL]

Highpass contains the highpass synthesis filter coefficients.

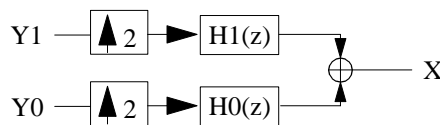
[DBL]

X is the output from the synthesis filter bank.

[I32]

error. See Appendix B, *Error Codes*, for a description of the error.

The VI performs the following operation:



The output **X** can be described by,

$$x_n = \sum_{i=0}^{n_{h0}/2-1} y0_{n+i} h0_{n_{h0}-2i} + \sum_{i=0}^{n_{h1}/2-1} y1_{n+i} h1_{n_{h1}-2i}$$

where $n = 0, 1, \dots, L - 1$,

L is the size of output \mathbf{X} ,

$$L = 2n_{y0} - n_{h0} + 1 \text{ or } L = 2n_{y1} - n_{h1} + 1$$



Note: *The lengths of $\mathbf{Y0}$ and $\mathbf{Y1}$ must satisfy $2n_{y0} - n_{h0} = 2n_{y1} - n_{h1}$. If your inputs $\mathbf{Y0}$ and $\mathbf{Y1}$ are the outputs from the same analysis filter bank, this condition is satisfied automatically.*

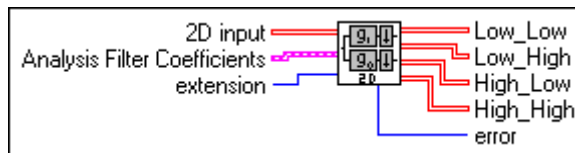
$H0$ is the synthesis lowpass filter coefficients, where n_{h0} is the size of $H0$, and

$H1$ is the synthesis highpass filter coefficients, where n_{h0} is the size of $H1$.

Refer to the section *Interpolation Filter VI* in this chapter for the more information about this operation.

2D Analysis Filter Bank VI

This VI computes the outputs of a 2D image passing through an analysis filter bank.



When a 2D image passes an analysis filter bank, it is broken into four sub-images. Refer to Chapter 3, *Digital Filter Banks*, for more information about computing the four sub-images.



2D Input contains 2D input image data.



Analysis Filter Coefficients contains the analysis filter bank coefficients.



Lowpass contains the lowpass analysis filter coefficients.



Highpass contains the highpass analysis filter coefficients.



extension decides the initial condition and final condition. **extension** has two options:

0: zero padding—all the initial condition and final condition are zeros.

1: symmetric extension—extends signal \mathbf{X} symmetrically as the initial condition and final condition.

Refer to the section *AnalysisFilterBank* in this chapter for detailed information about how to add data in these two cases.

[DBL]

Low_Low contains the output of the first subimage from the analysis filter bank.

[DBL]

Low_High contains the output of the second subimage from the analysis filter bank.

[DBL]

High_Low contains the output of the third subimage from the analysis filter bank.

[DBL]

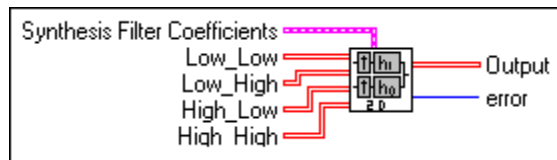
High_High contains the output of the fourth subimage from the analysis filter bank.

[I32]

error. See Appendix B, *Error Codes*, for a description of the error.

2D Synthesis Filter Bank VI

This VI computes the 2D output of a synthesis filter bank. It reconstructs the four subimages into the original image, if the four images are the outputs from the same 2D Analysis Filter Bank VI.



[DBL]

Synthesis Filter Coefficients contains the synthesis filters coefficients.

[DBL]

Lowpass contains the lowpass synthesis filter coefficients.

[DBL]

Highpass contains the highpass synthesis filter coefficients.

[DBL]

Low_Low contains the first subimage from the analysis filter bank.

[DBL]

Low_High contains the second subimage from the analysis filter bank.

[DBL]

High_Low contains the third subimage from the analysis filter bank.

[DBL]

High_High contains the fourth subimage from the analysis filter bank.

[DBL]

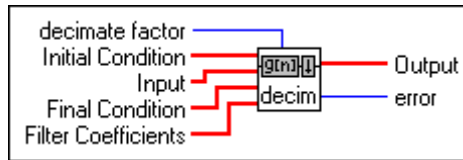
Output is the reconstructed X image of the signal.

[I32]

error. See Appendix B, *Error Codes*, for a description of the error.

Decimation Filter VI

This VI performs a decimation filter.



I32

decimate factor indicates the data reduction rate in the **Output** array. Only every M th point of the output from filter G is kept in the **Output** array.

[DBL]

Initial Condition contains the initial condition of the **Input**.

[DBL]

Input contains the input signal.

[DBL]

Final Condition contains the final condition of the **Input**.

[DBL]

Filter Coefficients contains the filter coefficients.

[DBL]

Output contains the output array.

I32

error. See Appendix B, *Error Codes*, for a description of the error.

This VI performs the following operation:



That is,

$$y_i = \sum_{k=1}^{n_g} x_{Mi+k} g_{n_g-k} \quad i = 0, 1, \dots, \text{ceil}((L - n_g + 1)/M)$$

n_x is the size of **X**,

G are the **Filter Coefficients**,

n_g is the size of G ,

Y is the **Output** array,

M is the **decimate factor**, and

$X1$ is **Initial Condition** plus **Input** plus **Final Condition**, according to the following formula:

$$x1_i = \begin{cases} xi_i & i = 0, 1 \dots n_{xi} - 1 \\ xi_{i-n_{xi}} & i = n_{xi}, n_{xi} + 1 \dots n_{xi} + n_x - 1 \\ xf_{i-n_{xi}-n_x} & i = n_{xi} + n_x \dots n_{xi} + n_x + n_{xf} - 1 \end{cases}$$

where Xi is the array of **Initial Condition**,

n_{xi} is the size of Xi ,

xf_i is the array of **Final Condition**, and

n_{xf} is the size of Xf .

The first plot in Figure 5-3, *Filtering Operation*, shows the signal of $X1$. The VI performs a regular FIR filtering or convolution followed by a decimation factor of M . The operation is illustrated in Figure 5-3, *Filtering Operation*.

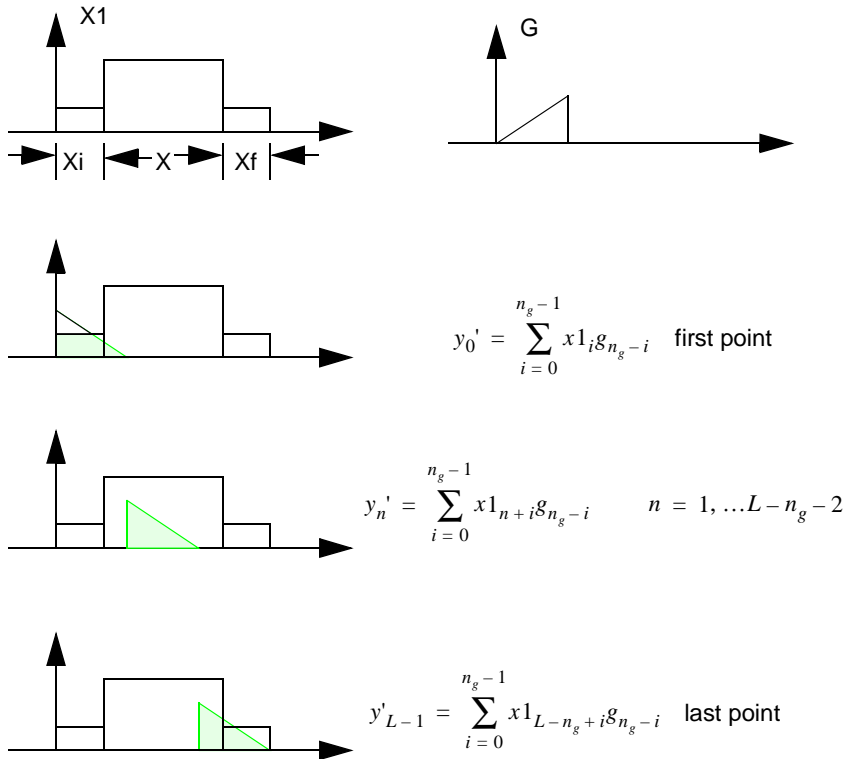


Figure 5-3. Filtering Operation

where Y is the decimation version of Y' , that is $y_n = y'_{Mn}$, where $n = 0, 1, \dots, L - 1/M$, and L is the length of $X1$, where $L = n_{xi} + n_x + n_{xf}$.

For the two-channel PR filter banks, the requirement for n_{xi} and n_{xf} is

$$n_{xi} = n_{xf} = (n_{g0} + n_{g1})/2 - 1$$

where n_{g0} is the length of the analysis lowpass filter, and

n_{g1} is the length of the analysis highpass filter.

If n_{xi} and n_{xf} meet the above condition, using the Decimation Filter and Interpolation Filter VIs produces a perfect reconstructed signal with no delay.

Figure 5-4 shows how to build a two-channel perfect reconstruction system using the Decimation Filter and Interpolation Filter VIs. The **Input** and **Output** arrays are the same. In this example, the **Initial Condition** and **Final Condition** arrays are constructed as zero padding. You can construct any values in **Initial Condition** and **Final Condition** as long as the sizes meet the requirements previously mentioned. If these requirements are met, you receive the same **Output** as **Input**.

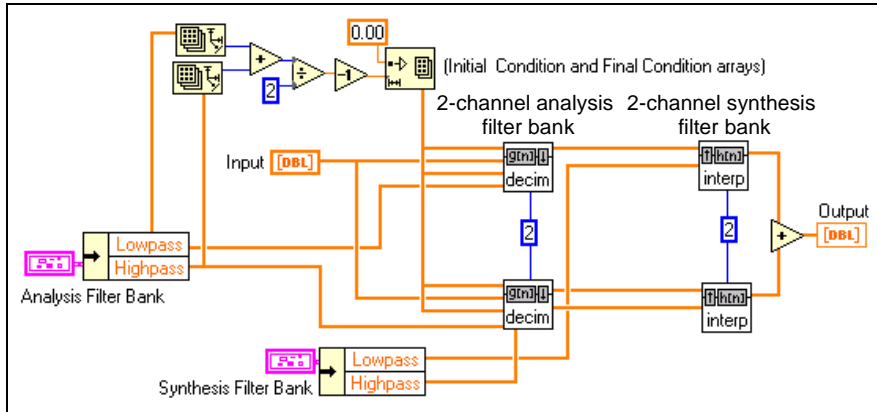
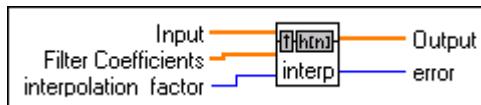


Figure 5-4. Two-Channel Perfect Reconstruction System

This VI is a subVI of the 2-Channel Analysis Filter Bank VI, which limits the **Initial Condition** and **Final Condition** to two cases, zero padding and symmetric extension.

Interpolation Filter VI

This VI performs an interpolation filter.



[DBL]

Input contains the input signal.

[DBL]

Filter Coefficients contains the filter coefficients.

[I32]

interpolation factor indicates the number of zeros to add among the **Input** data points. $L - 1$ zeros are inserted among each data point of the **Input** array before the filtering operation.

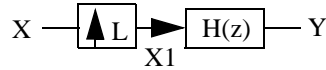
[DBL]

Output contains the output array.

[I32]

error. See Appendix B, *Error Codes*, for a description of the error.

This VI performs the following operation:



That is,

$$y_i = \sum_{k=0}^{n_h-1} x1_{i+k} g_{n_h-k} \quad i = 0, 1 \dots Ln_x - n_h + 1$$

where H is the array of **Filter Coefficients**,

n_h is the size of H ,

Y is the **Output** array,

L is the **interpolation factor**, and

$X1$ is the interpolated **Input**, that is, you insert $L - 1$ zeros among each point of **Input** data.

Figure 5-5 shows the case when $L = 2$.

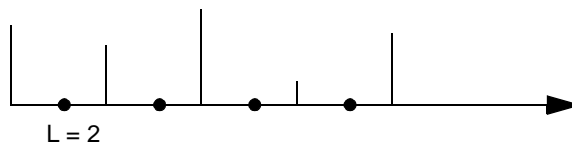


Figure 5-5. Signal Interpolated by 2

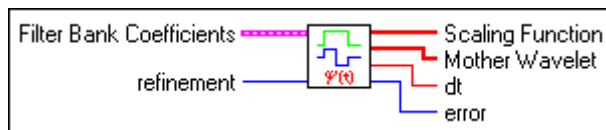
From $X1$ to Y is a regular FIR filtering or convolution. The operation is the same as shown in Figure 5-3, *Filtering Operation*.

If you set the proper length of the **Initial Condition** and **Final Condition** in the Decimation Filter VI, when building a two-channel filter bank using the Decimation Filter and Interpolation Filter VIs, you can get a perfect reconstructed signal with no delay, as shown in Figure 5-4, *Two-Channel Perfect Reconstruction System*.

This VI is a subVI of the 2-Channel Synthesis Filter Bank VI.

Mother Wavelet and Scaling Function VI

This VI computes the mother wavelet and scaling function of a filter bank. For a detailed description of the mother wavelet and scaling function, refer to Figure 3-2, *Relationship of Two-Channel PR Filter Banks to Wavelet Transform*, in Chapter 3, *Digital Filter Banks*.



Filter Bank Coefficients contains the filter bank coefficients.



Lowpass contains the lowpass filter coefficients.



Highpass contains the highpass filter coefficients.



refinement indicates how many levels of lowpass filters to go through to calculate the **Mother Wavelet** and **Scaling Function**.



Scaling Function. See Figure 3-2, *Relationship of Two-Channel PR Filter Banks to Wavelet Transform*, in Chapter 3, *Digital Filter Banks*.



Mother Wavelet. See Figure 2-3, *Wavelet Analysis*, in Chapter 2, *Wavelet Analysis*.



dt indicates the time duration between two points in the **Mother Wavelet** and **Scaling Function** outputs.



error. See Appendix B, *Error Codes*, for a description of the error.

LabWindows/CVI Applications

This section describes the LabWindows/CVI utilities you can use with the WFBD Toolkit.

Calling WFBD functions in LabWindows/CVI

Add the `wfbd32.fpp` to your project to call any functions in the instrument driver in your C code. You can find `wfdb32.fpp` under the `cvisrc\instr` subdirectory of your installation directory.

`wfbd32.fpp` needs import library `wfbd32.lib` in the same directory to run correctly. `wfbd32.lib` calls `wfbd32.dll` which is installed under your system directory. LabWindows/CVI compiler is compatible with four commonly used C extensions:

- Visual C/C++
- Symantec C/C++
- Borland C/C++
- Watcon C/C++

The installer automatically detects which extension CVI supports and installs the correct import library `wfbd32.lib`. If you change CVI to a different extension later on, you need to copy the right import library `wfbd32.lib` to the same directory as `wfdb32.fpp`. You can find four different import libraries under `windll\lib` subdirectory of your installation directory.

For examples of how to call these functions, check the directory `\cvisrc\examples` subdirectory of your installation directory.

WFBD Instrument Driver

The Wavelet and Filter Bank Design Toolkit provides an instrument driver, `wfbd.fpp`, for LabWindows/CVI developers using the Windows 95/NT platform. You can find this file in the `cvisrc\instr` subdirectory of your installation directory.

The following are the function prototypes in the instrument driver.

```
typedef struct {
double *Lowpass;      /* pointer to the lowpass filter coefficients */
long    nl;          /* number of coefficients in Lowpass */
```

```

double *Highpass;      /* pointer to the lowpass filter coefficients */
long   nh;             /* number of coefficients in Lowpass */
}FilterBankStruct, *FilterBankPtr;

long status=AnalysisFilterBank(double x[],long nx,FilterBankPtr
AnalysisFilters,long padtype,double y0[],long ny0,double y1[],long
ny1);

long status = SynthesisFilterBank(double y0[],long ny0,double
y1[],long ny1,FilterBankPtr SynthesisFilters,double x[],long nx);

long status = DecimationFilter(double x[],long nx,double coef[],long
nf,double init[],long ni,double final[],long nf,long decfact,double
y[],long ny);

long status = InterpolationFilter(double x[],long nx,double
coef[],long nf,long interfact,double y[],long ny);

long status = AnalysisFilterBank2D(void *x,long rows,long
cols,FilterBankPtr AnalysisFilters,long padtype,void* low_low,void*
low_high,void* high_low,void* high_high,long outsize[]);

long status = Analysis2DArraySize(long xrows,long xcols,long nl,long
nh,long nsize[8]);

long status = SynthesisFilterBank2D(void* low_low,void*
low_high,void* high_low,void* high_high,long insize[],FilterBankPtr
SynthesisFilters,void *x,long xrows,long xcols);

long status = Synthesis2DArraySize(long nsize[8],long nl,long nh,long
*rows,long *cols);

FilterBankPtr fptr=AllocCoeffWFBD(void);

long status=ReadCoeffWFBD(char coeffPath[],FilterBankPtr
AnalysisFilter,FilterBankPtr SynthesisFilter);

long err = FreeCoeffWFBD(FilterBankPtr fptr);

```

AllocCoeffWFBD

FilterBankPtr **fptr**=AllocCoeffWFBD(void);

Use this function to allocate the WFBD filter bank coefficients structure. You must call this function once to properly allocate the WFBD filter coefficients structure.

Return Value

fptr	FilterBankPtr	Pointer to allocated filter bank structure.
-------------	---------------	---

Analysis2DArraySize

```
long status=Analysis2DArraySize(long xrows,long xcols,long nl,long nh,long nsize[8]);
```

Computes the sizes of four output arrays for AnalysisFilterBank2D. Call this function to compute the sizes for four arrays before calling AnalysisFilterBank2D.

Parameters

Input	xrows	long integer	The row size of 2D input array x .
	xcols	long integer	The column size of 2D input array x .
	nl	long integer	The size of lowpass filter in the analysis filter bank.
	nh	long integer	The size of highpass filter in the analysis filter bank.

Output	nsize	long-integer array	<p>The array contains all the size information of four output arrays in AnalysisFilterBank2D. The array size of nsize must be 8. Assume the four arrays are low_low, low_high, high_low and high_high, then</p> <p>nsize[0]: the number of rows of array low_low.</p> <p>nsize[1]: the number of columns of array low_low.</p> <p>nsize[2]: the number of rows of array low_high.</p> <p>nsize[3]: the number of columns of array low_high.</p> <p>nsize[4]: the number of rows of array high_low.</p> <p>nsize[5]: the number of columns of array high_low.</p> <p>nsize[6]: the number of rows of array high_high.</p> <p>nsize[7]: the number of columns of array high_high.</p>
--------	--------------	--------------------	---

Return Value

status	integer	Refer to Appendix B for error codes.
---------------	---------	--------------------------------------

AnalysisFilterBank

```
long status = AnalysisFilterBank(double x[], long nx, FilterBankPtr
AnalysisFilters, long padtype, double y0[], long ny0, double y1[], long ny1);
```

Computes the outputs of a 2-Channel Analysis Filter Bank. It performs the same operation as 2-Channel Filter Bank VI. Please refer to the description about that VI for the detailed information.

Parameters

Input	x	double-precision array	The input data array.
	nx	long integer	The size of input array x .
	AnalysisFilters	FilterBankPtr	The structure holding the analysis filter bank coefficients.
	padtype	long integer	The type of padding used at the beginning and the end of the input data. 0: zero padding 1: symmetric extension
	ny0	long integer	The array size of y0 . It must be $\text{ceil}((\mathbf{nx} + \mathbf{nh} - 1) / 2)$. \mathbf{nh} is the size of the highpass filter in AnalysisFilters .
	ny1	long integer	The array size of y1 . It must be $\text{ceil}((\mathbf{nx} + \mathbf{nl} - 1) / 2)$. \mathbf{nl} is the size of the lowpass filter in AnalysisFilters .
Output	y0	double-precision array	The output from the analysis lowpass filter.
	y1	double-precision array	The output from the analysis highpass filter.

Return Value

status	integer	Refer to Appendix B for error codes.
---------------	---------	--------------------------------------

Example

```

/* Example 1: How to call function AnalysisFilterBank */
include "wfbd.h"
FilterBankPtr          anaptr, synptr;
double                 *x, *y0, *y1;
long                   err, nx, ny0, ny1;
anaptr = AllocCoeffWFBD(); /* allocate filter bank structure */
if(!anaptr) return;
synptr = AllocCoeffWFBD();
if(!synptr) {
    free(anaptr);
    return;
}

err = ReadCoeffWFBD("coef.dat", anaptr, synptr); /* Read filter bank coefficients */
if(err) goto errend;
nx = 128;
x = (double*)malloc(nx*sizeof(double));
if(!x) goto errend;
Chirp (nx, 1.0, 0.0, 0.5, x);

ny0 = ceil(0.5*(nx+anaptr->nh-1)); /* Compute the size of output array */
y0 = (double*)malloc(ny0*sizeof(double));
if(!y0) goto errend;
ny1 = ceil(0.5*(nx+anaptr->nl-1));
y1 = (double*)malloc(ny1*sizeof(double));
if(!y1) {
    free(y0);
    goto errend;
}
err=AnalysisFilterBank(x, nx, anaptr, 0, y0, ny0, y1, ny1);

errend:
    free(x);
    FreeCoeffWFBD(anaptr);
    FreeCoeffWFBD(synptr);

```

AnalysisFilterBank2D

```

long status = AnalysisFilterBank2D(void *x, long rows, long cols, FilterBankPtr
AnalysisFilters, long padtype, void* low_low, void* low_high, void*
high_low, void* high_high, long outside[]);

```

Computes the output from an analysis filter bank of a 2D signal. It performs the same operation as in 2D Analysis Filter Bank VI. Please refer to the description for that VI for detailed information.

Parameters

Input	x rows cols AnalysisFilters padtype outsizes	2D double-precision array long integer long integer FilterBankPtr long integer long integer array	The input 2D data array. The number of rows of input array x . The number of columns of input array x . The structure holding the analysis filter bank coefficients. The type of padding used at the beginning and the end of the input data. 0: zero padding 1: symmetric extension Contains the size information for four output arrays. Call the Analysis2DArraySize function to compute this array. outsizes[0] : the number of rows of array low_low ; it must be $\text{ceil}((\mathbf{rows} + nh - 1) / 2)$. outsizes[1] : the number of columns of array low_low ; it must be $\text{ceil}((\mathbf{cols} + nh - 1) / 2)$. outsizes[2] : the number of rows of array low_high ; it must be $\text{ceil}((\mathbf{rows} + nl - 1) / 2)$. outsizes[3] : the number of columns of array low_high ; it must be $\text{ceil}((\mathbf{cols} + nh - 1) / 2)$. outsizes[4] : the number of rows of array high_low ; it must be $\text{ceil}((\mathbf{rows} + nh - 1) / 2)$.
-------	---	--	---

			<p>outsized[5]: the number of columns of array high_low; it must be $\text{ceil}((\text{cols} + nl - 1) / 2)$.</p> <p>outsized[6]: the number of rows of array high_high; it must be $\text{ceil}((\text{rows} + nl - 1) / 2)$.</p> <p>outsized[7]: the number of columns of array high_high; it must be $\text{ceil}((\text{cols} + nl - 1) / 2)$.</p> <p>For all equations, nl is the size of the lowpass filters and nh is the size of the highpass filters in the Analysis Filters.</p>
Output	<p>low_low</p> <p>low_high</p> <p>high_low</p> <p>high_high</p>	<p>double-precision 2D array</p> <p>double-precision 2D array</p> <p>double-precision 2D array</p> <p>double-precision 2D array</p>	<p>The upper left subimage from the analysis filter bank.</p> <p>The upper right subimage from the analysis filter bank.</p> <p>The lower left subimage from the analysis filter bank.</p> <p>The lower right subimage from the analysis filter bank.</p>

Return Value

status	integer	Refer to Appendix B for error codes.
---------------	---------	--------------------------------------

Example

```

/* Example 2: How to call function AnalysisFilterBank2D */
#include "wfbd.h"
FilterBankPtr      anaptr, synptr;
double             *x,*ll,*lh,*hl,*hh;
long               err,rows,cols,nsz[8];

anaptr = AllocCoeffWFBD(); /* allocate filter bank structure */
if(!anaptr) return;
synptr = AllocCoeffWFBD();
if(!synptr) {
    free(anaptr);
    return;
}

err = ReadCoeffWFBD("coef.dat",anaptr,synptr); /* Read filter bank coefficients */
if(err) goto errend;
rows = 128;
cols = 256;
x = (double*)malloc(rows*cols*sizeof(double));
if(!x) goto errend;

/* compute the size of output arrays */
err = Analysis2DArraySize(rows,cols,anaptr->nl,anaptr->nh,nsz);
if(err) goto errend;

/* Allocate memory for the output arrays*/
ll = (double*)malloc(nsz[0]*nsz[1]*sizeof(double));
if(!ll) goto errend;
lh = (double*)malloc(nsz[2]*nsz[3]*sizeof(double));
if(!lh) {
    free(ll);
    goto errend;
}
hl = (double*)malloc(nsz[4]*nsz[5]*sizeof(double));
if(!hl) {
    free(ll);
    free(lh);
    goto errend;
}
hh = (double*)malloc(nsz[6]*nsz[7]*sizeof(double));
if(!hh) {
    free(ll);
    free(lh);
    free(hl);
    goto errend;
}

err = AnalysisFilterBank2D(x,rows,cols,anaptr,0,ll,lh,hl,hh,nsz);
free(ll);
free(lh);
free(hl);
free(hh);

errend:
    free(x);
    FreeCoeffWFBD(anaptr);
    FreeCoeffWFBD(synptr);

```

DecimationFilter

```
long status = DecimationFilter(double x[],long nx,double coef[],long ncoef,double
init[],long ni,double final[],long nf,long defact,double y[],long ny);
```

Performs a decimation filtering. It performs the same operation as the Decimation Filter VI. Please refer to the description for that VI for detailed information.

Parameters

Input	x	double-precision array	The input data array.
	nx	long integer	The size of input array x .
	coef	double-precision array	The array of filter coefficients.
	ncoef	long integer	The size of array coef .
	init	double-precision array	The initial condition of the input data
	ni	long integer	The size of array init .
	final	double-precision array	The final condition of the input data.
	nf	long integer	The size of array final .
	defact	long integer	The decimation factor.
	ny	long integer	The size of output array y . It must be $\text{ceil}((\mathbf{nx} + \mathbf{ni} + \mathbf{nf} - \mathbf{ncoef} + 1) / \mathbf{defact})$.
Output	y	double-precision array	The output from the decimation filter.

Return Value

status	integer	Refer to Appendix B for error codes.
---------------	---------	--------------------------------------

Example

```
/* Example 3: How to build 2-channel analysis filter bank using DecimationFilter */
#include "wfbd.h"

FilterBankPtr          anaptr, synptr;
double                 *x,*y0,*y1,*init,*final, *xtmp;
long                   err,nx,ny0,ny1,nl,nh,npad,i;

anaptr = AllocCoeffWFBD(); /* allocate filter bank structure */
if(!anaptr) return;
synptr = AllocCoeffWFBD();
```

```

if(!synptr) {
    free(anaptr);
    return;
}

err = ReadCoeffWFBD("coef.dat", anaptr, synptr); /* Read filter bank coefficients */
if(err) goto errend;

nl = anaptr->nl;
nh = anaptr->nh;

x = (double*)malloc(nx*sizeof(double));
if(!x) goto errend;

npad = (nl+nh)/2-1; /* Compute the size of initial and final condition arrays */
init = (double*) malloc(npad*sizeof(double));
if(!init) goto errend;
final = (double*) malloc(npad*sizeof(double));
if(!final) {
    free(init);
    goto errend;
}

/* Initialize the initial and final condition arrays to zeros, you can initilize these
two arrays to different values base on your requirments */

xtmp = init;
for(i=npad;i--;) *xtmp++ = 0.0;
xtmp = final;
for(i=npad;i--;) *xtmp++ = 0.0;

/* Compute the size of output arrays and allocate memory for them */
ny0 = ceil(0.5*(nx+nh-1));
y0 = (double*)malloc(ny0*sizeof(double));
if(!y0) {
    free(init);
    free(final);
    goto errend;
}
ny1 = ceil(0.5*(nx+nl-1));
y1 = (double*)malloc(ny1*sizeof(double));
if(!y1) {
    free(y0);
    free(init);
    free(final);
    goto errend;
}

/* Compute the ouput from the analysis lowpass filter */
if(nl>0)
    err = DecimationFilter(x, nx, anaptr->Lowpass, nl, init, npad, final, npad, 2, y0, ny0);

/* Compute the ouput from the analysis highpass filter */
if(!err) {
    if(nh>0)
        err = DecimationFilter(x, nx, anaptr->Highpass, nh, init, npad, final, npad, 2, y1, ny1);
}

errend:
    free(x);
    FreeCoeffWFBD(anaptr);
    FreeCoeffWFBD(synptr);

```

Parameter Discussion

To obtain the perfect reconstruction, you must meet the following condition:

$$\mathbf{ni} = \mathbf{nf} = (nl + nh)/2 - 1$$

where nl is the size of the analysis lowpass filter and nh is the size of the analysis highpass filter.

You can use this function to build a 2-Channel Analysis Filter Bank.

FreeCoeffWFBD

```
long err = FreeCoeffWFBD(FilterBankPtr fptr);
```

Use this function to free the WFBD filter bank coefficients structure and all of its coefficients arrays.

Parameters

Input	fptr	FilterBankPtr	Pointer to allocated filter bank structure.
-------	-------------	---------------	---

Return Value

err	integer	Refer to Appendix B for error codes.
------------	---------	--------------------------------------

InterpolationFilter

```
long status = InterpolationFilter(double x[], long nx, double coef[], long nf, long  
interfact, double y[], long ny);
```

Performs an interpolation filter. It performs the same operation as the Interpolation Filter VI. Please refer to the description for that VI for detailed information. You can use this function to build a 2-Channel Synthesis Filter Bank.

Parameters

Input	x	double-precision array	The input data array.
	nx	long integer	The size of input array x .
	coef	double-precision array	The array of filter coefficients.
	nf	long integer	The size of array coef .
	interfact	long integer	The interpolation factor.
	ny	long integer	The size of output array y . It must be nx*interfact – nf + 1 .
Output	y	double-precision array	The output from the interpolation filter.

Return Value

status	integer	Refer to Appendix B for error codes.
---------------	---------	--------------------------------------

Example

```

/* Example 4: How to build 2-channel synthesis filter bank using InterpolationFilter */
#include "wfbd.h"

test()
{
  FilterBankPtr      anaptr, synptr;
  double             *x, *y0, *y1, *tmp, *x0;
  long               err, nx, ny0, ny1, nl, nh, npad, i;

  anaptr = AllocCoeffWFBD(); /* allocate filter bank structure */
  if(!anaptr) return;
  synptr = AllocCoeffWFBD();
  if(!synptr) {
    free(anaptr);
    return;
  }
  err = ReadCoeffWFBD("coef.dat", anaptr, synptr); /* Read filter bank coefficients */
  if(err) goto errend;

  nl = anaptr->nl;
  nh = anaptr->nh;

  x = (double*)malloc(nx*sizeof(double));
  if(!x) goto errend;

  /* Compute the size of output arrays and allocate memory for them */
  ny0 = ceil(0.5*(nx+anaptr->nh-1));
  y0 = (double*)malloc(ny0*sizeof(double));
  if(!y0) goto errend;

  ny1 = ceil(0.5*(nx+anaptr->nl-1));
  y1 = (double*)malloc(ny1*sizeof(double));

```

```

if(!y1) {
    free(y0);
    goto errend;
}

err = AnalysisFilterBank(x,nx,anaptr,0,y0,ny0,y1,ny1);
if(err) {
    free(y0);
    free(y1);
    goto errend;
}

/* Allocate memory for the output of synthesis filter bank */
x0 = (double*)malloc(nx*sizeof(double));
if(!x0) {
    free(y0);
    free(y1);
    goto errend;
}
tmp = (double*)malloc(nx*sizeof(double));
if(!tmp) {
    free(y0);
    free(y1);
    free(x0);
    goto errend;
}

/* Compute the output from synthesis lowpass filter */
err = InterpolationFilter(y0,ny0,synptr->Lowpass,synptr->nl,2,x0,nx);
if(err) {
    free(tmp);
    return (err);
}

/* Compute the output from synthesis highpass filter */
err = InterpolationFilter(y1,ny1,synptr->Highpass,synptr->nh,2,tmp,nx);
if(err) {
    free(tmp);
    return (err);
}

/* Compute the output from the synthesis filter bank */
for(i=0;i<nx;i++) x0[i] += tmp[i];
errend:
    free(x);
    FreeCoeffWFBD(anaptr);
    FreeCoeffWFBD(synptr);
}

```

ReadCoeffWFBD

```
long status=ReadCoeffWFBD(char coeffPath[],FilterBankPtr
AnalysisFilter,FilterBankPtr SynthesisFilter);
```

Reads the analysis and synthesis filter bank coefficients from a text file. You must call **AllocCoeffWFBD** to allocate filter bank structures for both analysis filter banks and synthesis filter banks. The text file is created by using `WFBD.exe`.

Parameters

Input	coeffPath	char array	The path of the text file to read.
Output	AnalysisFilter	FilterBankPtr	The structure that holds the analysis filter bank coefficients. If this pointer is set to NULL, the function will not read the analysis filter bank coefficients.
	SynthesisFilter	FilterBankPtr	The structure that holds the synthesis filter bank coefficients. If this pointer is set to NULL, the function will not read the synthesis filter bank coefficients.

Return Value

status	integer	Refer to Appendix B for error codes.
---------------	---------	--------------------------------------

Synthesis2DArraySize

```
long status=Synthesis2DArraySize(long nsize[8], long nl,long nh,long *rows,long
*cols);
```

Computes the size of the 2D output array for `SynthesisFilterBank2D`. Call this function to compute the sizes for the output array before calling `SynthesisFilterBank2D`.

Parameters

Input	nsiz	long integer array	<p>The array contains all the size information of four input arrays for the SynthesisFilterBank2D.</p> <p>The array size of nsiz must be 8.</p> <p>Assume the four input arrays are low_low, low_high, high_low, and high_high, then</p> <p>nsiz[0]: the number of rows of array low_low.</p> <p>nsiz[1]: the number of columns of array low_low.</p> <p>nsiz[2]: the number of rows of array low_high.</p> <p>nsiz[3]: the number of columns of array low_high.</p> <p>nsiz[4]: the number of rows of array high_low.</p> <p>nsiz[5]: the number of columns of array high_low.</p> <p>nsiz[6]: the number of rows of array high_high.</p> <p>nsiz[7]: the number of columns of array high_high.</p>
	nl	long integer	The size of lowpass filter in the synthesis filter bank.
	nh	long integer	The size of highpass filter in the synthesis filter bank.
Output	rows	long integer	The row size of the 2D output array for SynthesisFilterBank2D.
	cols	long integer	The column size of the 2D output array for SynthesisFilterBank2D.

Return Value

status	integer	Refer to Appendix B for error codes.
---------------	---------	--------------------------------------

SynthesisFilterBank

```
long status = SynthesisFilterBank(double y0[], long ny0, double y1[], long
ny1, FilterBankPtr SynthesisFilters, double x[], long nx);
```

Computes the output of a 2-Channel Synthesis Filter Bank. It performs the same operation as in 2-Channel Synthesis Filter Bank VI. Please refer to the description for that VI for detailed information.

Parameters

Input	y0	double-precision array	The input data array for the synthesis lowpass filter.
	ny0	long integer	The size of input array y0 .
	y1	double-precision array	The input data array for the synthesis highpass filter.
	ny1	long integer	The size of input array y1 .
	Synthesis Filters	FilterBankPtr	The structure holding the synthesis filter bank coefficients.
	nx	long integer	The array size of x . It must be $2*\mathbf{ny0} - n_l + 1 = 2*\mathbf{ny1} - n_h + 1$. The values of ny0 , ny1 , n_l , and n_h must meet the above condition. n_l is the size of lowpass filter in SynthesisFilters . n_h is the size of highpass filter in SynthesisFilters .
Output	x	double-precision array	The output from the synthesis filter bank.

Return Value

status	integer	Refer to Appendix B for error codes.
---------------	---------	--------------------------------------

Example

```

/* Example 5: How to call function SynthesisFilterBank */

#include "wfbd.h"

FilterBankPtr      anaptr, synptr;
double             *x,*y0,*y1,*x0;
long               err,nx,ny0,ny1;

anaptr = AllocCoeffWFBD(); /* allocate filter bank structure */
if(!anaptr) return;
synptr = AllocCoeffWFBD();
if(!synptr) {
    free(anaptr);
    return;
}

err = ReadCoeffWFBD("coef.dat",anaptr,synptr); /* Read filter bank coefficients */
if(err) goto errend;
nx = 128;
x = (double*)malloc(nx*sizeof(double));
if(!x) goto errend;
Chirp (nx, 1.0, 0.0, 0.5, x);

ny0 = ceil(0.5*(nx+anaptr->nh-1));/* Compute the size of output array */
y0 = (double*)malloc(ny0*sizeof(double));
if(!y0) goto errend;
ny1 = ceil(0.5*(nx+anaptr->nl-1));
y1 = (double*)malloc(ny1*sizeof(double));
if(!y1) {
    free(y0);
    goto errend;
}
err=AnalysisFilterBank(x,nx,anaptr,0,y0,ny0,y1,ny1);

/* Allocate memory for the output of synthesis filter bank */
x0 = (double*)malloc(nx*sizeof(double));
if(!x0) {
    free(y0);
    free(y1);
    goto errend;
}
err = SynthesisFilterBank(y0,ny0,y1,ny1,synptr,x0,nx); /* x0 and x should be the same
value */free(x0);

errend:
    free(x);
    FreeCoeffWFBD(anaptr);
    FreeCoeffWFBD(synptr);

```

SynthesisFilterBank2D

```
long status = SynthesisFilterBank2D(void* low_low,void* low_high,void*
high_low,void* high_high,long insize[],FilterBankPtr SynthesisFilters,void
*x,long xrows,long xcols);
```

Computes the output from a synthesis filter bank of a 2D signal.

Parameters

Input	low_low	double-precision 2D array	The upper left subimage from the analysis filter bank.
	low_high	double-precision 2D array	The upper right subimage from the analysis filter bank.
	high_low	double-precision 2D array	The lower left subimage from the analysis filter bank.
	high_high	double-precision 2D array	The lower right subimage from the analysis filter bank.
	insize	long integer array	Contains the size information for all four input arrays. outsized[0] : the number of rows of array low_low . outsized[1] : the number of columns of array low_low . outsized[2] : the number of rows of array low_high . outsized[3] : the number of columns of array low_high . outsized[4] : the number of rows of array high_low . outsized[5] : the number of columns of array high_low . outsized[6] : the number of rows of array high_high . outsized[7] : the number of columns of array high_high .

	SynthesisFilters	FilterBankPtr	The structure holding the synthesis filter bank coefficients.
	xrows	long integer	The row size of output array x . Call the function Synthesis2DArraySize to compute xrows .
	xcols	long integer	The column size of output array x . Call the function Synthesis2DArraySize to compute xcols .
Output	x	double-precision 2D array	The output from the synthesis filter bank.

Return Value

status	integer	Refer to Appendix B for error codes.
---------------	---------	--------------------------------------

Example

```

/* Example 6: How to call function SynthesisFilterBank2D */
#include "wfbd.h"

FilterBankPtr      anaptr, synptr;
double             *x,*l1,*lh,*hl,*hh,*x0;
long               err,rows,cols,nsiz[8],x0rows,x0cols;

anaptr = AllocCoeffWFBD(); /* allocate filter bank structure */
if(!anaptr) return;
synptr = AllocCoeffWFBD();
if(!synptr) {
    free(anaptr);
    return;
}

err = ReadCoeffWFBD("coef.dat",anaptr,synptr); /* Read filter bank coefficients */
if(err) goto errend;
rows = 128;
cols = 256;
x = (double*)malloc(rows*cols*sizeof(double));
if(!x) goto errend;

/* compute the size of output arrays */
err = Analysis2DArraySize(rows,cols,anaptr->nl,anaptr->nh,nsiz);
if(err) goto errend;

/* Allocate memory for the output arrays*/
l1 = (double*)malloc(nsiz[0]*nsiz[1]*sizeof(double));
if(!l1) goto errend;
lh = (double*)malloc(nsiz[2]*nsiz[3]*sizeof(double));
if(!lh) {
    free(l1);
    goto errend;
}

```

```

hl = (double*)malloc(nsize[4]*nsize[5]*sizeof(double));
if(!hl) {
    free(ll);
    free(lh);
    goto errend;
}
hh = (double*)malloc(nsize[6]*nsize[7]*sizeof(double));
if(!hh) {
    free(ll);
    free(lh);
    free(hl);
    goto errend;
}

err = AnalysisFilterBank2D(x,rows,cols,anaptr,0,ll,lh,hl,hh,nsize);
if(err){
    free(ll);
    free(lh);
    free(hl);
    free(hh);
    goto errend;
}

/* Compute the size of 2D output from the synthesis filter bank */
Synthesis2DArraySize(nsize,synptr->nl,synptr->nh,&x0rows,&x0cols);

/* Allocate 2D output for the synthesis filter bank */
x0 = (double*)malloc(x0rows*x0cols*sizeof(double));
if(!hh) {
    free(ll);
    free(lh);
    free(hl);
    free(hh);
    goto errend;
}

err = SynthesisFilterBank2D(ll,lh,hl,hh,nsize,synptr,x0,x0rows,x0cols);

free(ll);
free(lh);
free(hl);
free(hh);
free(x0);

errend:
    free(x);
    FreeCoeffWFBD(anaptr);
    FreeCoeffWFBD(synptr);

```

Windows Applications

The WFBD Toolkit provides a 32-bit dynamic link library (DLL), `wfbd32.dll`, for all Windows platforms users. The DLL is located in the `windll` subdirectory of your installation directory. Four import libraries for different compilers also are provided:

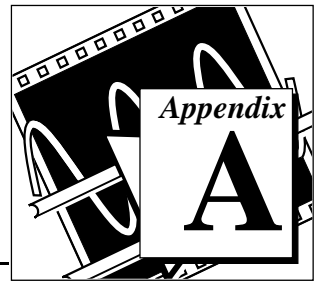
- Microsoft Visual C/C++
- Borland C/C++
- Watcom C/C++
- Symantec C/C++

You can find these four import libraries under `windll\lib` subdirectory of your installation directory.

The functions in the DLL are the same as for `LabWindows\CVI`. Please refer to the previous section *WFBD Instrument Driver* for the function description.

Call these functions the same way in your code as you call any functions in DLLs.

References



This appendix lists the reference material used to produce the Wavelet and Filter Bank Design Toolkit. For more information about the theories and algorithms implemented in the WFBD Toolkit, refer to the following documents.

Crochiere, R. E., and L.R. Rabiner. *Multirate Digital Signal Processing*. Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

Donoho, D. L. "De-noise by soft thresholding." *IEEE Transactions Information Theory*, no. 41 (May 1995): 613-627.

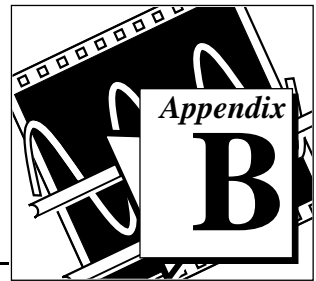
Qian, S., and D. Chen. *Joint Time-Frequency Analysis*. Upper Saddle River, New Jersey: Prentice-Hall, 1996.

Strang, G., and T. Nguyen. *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press, 1995.

Vaidyanathan, P.P. *Multirate Systems and Filter Banks*, Englewood Cliffs, New Jersey: Prentice-Hall, 1993.

Vaidyanathan, P.P., and T. Nguyen. "A 'Trick' for the Design of FIR Half-Band Filters." *IEEE Transactions on Circuits and Systems*, March 1987: 297-300.

Error Codes



This appendix lists the error codes returned by the LabVIEW VIs, including the error number and a description. Each VI returns an error code that indicates whether the function was performed successfully.

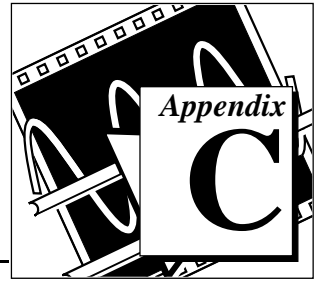
Table B-1. LabVIEW VI Error Codes

Code	Name	Description
0	NoErr	No error; the call was successful.
-20001	OutOfMemErr	There is not enough memory left to perform the specified routine.
-20002	EqSamplesErr	The input sequences must be the same size.
-20003	SamplesGTZeroErr	The number of samples must be greater than zero.
-20004	SamplesGEZeroErr	The number of samples must be greater than or equal to zero.
-20005	SamplesGEOneErr	The number of samples must be greater than or equal to one.
-20008	ArraySizeErr	The input arrays do not contain the correct number of data values for this VI.
-20009	PowerOfTwoErr	The size of the input array must be a power of two: $\text{size} = 2^m$, $0 < m < 23$.
-20012	CyclesErr	The number of cycles must be greater than zero and less than or equal to the number of samples.

Table B-1. LabVIEW VI Error Codes

Code	Name	Description
-20020	NyquistErr	The cutoff frequency, f_c , must meet the condition $0 \leq f_c \leq f_s/2$
-20021	OrderGTZeroErr	The order must be greater than zero.
-20031	EqRplDesignErr	The filter cannot be designed with the specified input values.
-20033	EvenSizeErr	The number of coefficients must be odd for this filter.
-20034	OddSizeErr	The number of coefficients must be even for this filter.
-20038	IntervalsErr	The number of intervals must be greater than zero.
-20039	MatrixMulErr	The number of columns in the first matrix is not equal to the number of rows in the second matrix or vector.
-20040	SquareMatrixErr	The input matrix must be a square matrix.
-20041	SingularMatrixErr	The system of equations cannot be solved because the input matrix is singular.
-20062	MaxIterErr	The maximum iterations have been exceeded.
-20065	ZeroVectorErr	The vector cannot be zero.

Customer Communication



For your convenience, this appendix contains forms to help you gather the information necessary to help us solve your technical problems and a form you can use to comment on the product documentation. When you contact us, we need the information on the Technical Support Form and the configuration form, if your manual contains one, about your system configuration to answer your questions as quickly as possible.

National Instruments has technical assistance through electronic, fax, and telephone systems to quickly provide the information you need. Our electronic services include a bulletin board service, an FTP site, a Fax-on-Demand system, and e-mail support. If you have a hardware or software problem, first try the electronic support systems. If the information available on these systems does not answer your questions, we offer fax and telephone support through our technical support centers, which are staffed by applications engineers.

Electronic Services



Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call (512) 795-6990. You can access these services at:

United States: (512) 794-5422

Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422

Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 01 48 65 15 59

Up to 9,600 baud, 8 data bits, 1 stop bit, no parity



FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as anonymous and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.



Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at (512) 418-1111.



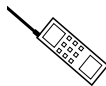
E-Mail Support (currently U.S. only)

You can submit technical support questions to the applications engineering team through e-mail at the Internet address listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

support@natinst.com

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.



Telephone



Fax

Australia	02 9874 4100	02 9874 4455
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 527 2321	09 502 2930
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 5734815	03 5734816
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	01635 523545	01635 523154

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

Fax (____) _____ Phone (____) _____

Computer brand _____ Model _____ Processor _____

Operating system (include version number) _____

Clock speed _____MHz RAM _____MB Display adapter _____

Mouse ___yes ___no Other adapters installed _____

Hard disk capacity _____MB Brand _____

Instruments used _____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is: _____

List any error messages: _____

The following steps reproduce the problem: _____

WFBD Toolkit Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

DAQ hardware _____

Interrupt level of hardware _____

DMA channels of hardware _____

Base I/O address of hardware _____

Programming choice _____

LabVIEW or LabWindows/CVI version _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Other Products

Computer make and model _____

Microprocessor _____

Clock frequency or speed _____

Type of video board installed _____

Operating system version _____

Operating system mode _____

Programming language _____

Programming language version _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: Wavelet and Filter Bank Design Toolkit Reference Manual

Edition Date: January 1997

Part Number: 321380A-01

Please comment on the completeness, clarity, and organization of the manual.

If you find errors in the manual, please record the page numbers and describe the errors.

Thank you for your help.

Name _____

Title _____

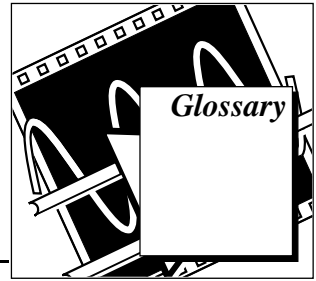
Company _____

Address _____

Phone (____) _____ Fax (____) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
(512) 794-5678



Numbers/Symbols

1D	One-dimensional
2D	Two-dimensional
Hz	Hertz

A

alias term	An image term in the frequency domain.
alternating flip	For a periodic sequence $g[n]$ with a period N , the sequence $(-1)^n g[N - n]$ is considered as alternating flip of $g[n]$.
analysis filter bank	A filter bank that converts a signal from time domain into wavelet domain.

B

basis function	An elementary function that can be used to build arbitrary signals.
biorthogonal filter bank	A filter bank in which analysis and synthesis filter banks are orthogonal to each other.
Butterworth filter	A special kind of filter in which the low-frequency asymptote is a constant.

C

constant Q analysis	Analysis where the ratio between the center frequency and frequency bandwidth is constant.
---------------------	--

D

DAQ	Data acquisition.
Daubechies wavelet and filter bank	Wavelet and filter bank that has a maximum number of zeros at π . The wavelet and filter bank was initially developed by Ingrid Daubechies.
decimation filter	The output of the filter does not preserve all points.
denoise	Remove the noise from the original signal.
distortion term	A term that causes distortion in a filter output.

E

equiripple filter	A filter with equiripples in the passband and stopband.
-------------------	---

F

filter bank	A group of filters.
finite impulse response filter	A filter without feedback and containing only zeros in the z -domain.
FIR filter	<i>See</i> Finite Impulse Response filter.

H

halfband filter	A filter with a cut-off frequency at a half of the frequency band.
-----------------	--

I

image compression	Using only part of the data to recover the original image.
inner product	A mathematical operation used to test the difference between two functions.

M

maximum flat filter	A type I filter that has a maximum number of zeros at π .
mother wavelet	An elementary wavelet.
multiscale analysis	Analyzing a signal in several different scales.

O

orthogonal filter bank	A filter bank where both the analysis and synthesis filter banks are orthogonal to themselves. It is a special case of biorthogonal filter banks.
------------------------	---

S

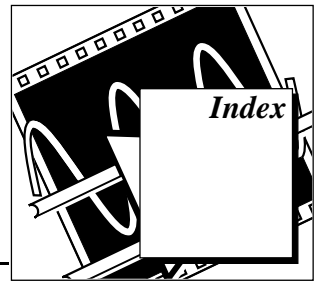
signal discontinuity	The point where the first derivative does not exist.
synthesis filter bank	A filter bank that transfers a signal from the wavelet domain into the time domain.

T

type I filter	The filter coefficients are symmetric among the middle point.
---------------	---

W

wavelet	A transform using wavelet as the elementary functions.
wavelet-based detrend	A method of detrend, which is achieved by wavelet transform.



Numbers

- 1D_Test panel. *See* One-Dimensional (1D) Data Test panel.
- 2-Channel Analysis Filter Bank VI, 5-1 to 5-4
- 2-Channel Synthesis Filter Bank VI, 5-4 to 5-5
- 2D Analysis Filter Bank VI, 5-5 to 5-6
- 2D Synthesis Filter Bank VI, 5-6
- 2D_Test panel. *See* Two-Dimensional (2D) Data Test panel.

A

- alias term, 3-5
- AllocCoeffWFBD function, 5-14
- analysis filter functions/VIs
 - 2-Channel Analysis Filter Bank VI, 5-1 to 5-4
 - 2D Analysis Filter Bank VI, 5-5 to 5-6
 - Analysis2DArraySize function, 5-15 to 5-16
 - AnalysisFilterbank function, 5-17 to 5-18
 - AnalysisFilterbank2D function, 5-18 to 5-21
- analysis filters
 - designations in WFBD Toolkit (note), 4-7
 - interchangeability of $G(z)$ and $H(z)$ (note), 3-2
- approximations of wavelet transform, 3-3

B

- bibliographic references, A-1
- biorthogonal filter banks
 - two-channel perfect reconstruction filter banks, 3-4 to 3-10
 - wavelet and filter bank design, 4-3, 4-7

- design procedure (figure), 4-2
- filter comparison (table), 4-4
- B-spline filter banks
 - choices for $G_0(z)$ and $H_0(z)$, 4-9
 - dual, 3-9
 - two-channel perfect reconstruction filter banks, 3-8 to 3-9
- bulletin board support, C-1
- Butterworth filter (note), 3-7

C

- coefficients
 - filter bank coefficient functions
 - AllocCoeffWFBD function, 5-14
 - FreeCoeffWFBD function, 5-24
 - ReadCoeffWFBD function, 5-27
 - Fourier coefficient, 2-2
 - wavelet coefficients
 - definition, 2-6
 - relationship with filter banks (figure), 3-4
- complex sinusoidal functions, 2-2 to 2-3
 - harmonically related, 2-2
- customer communication, *xi*, C-1 to C-2

D

- DAQ Setup panel, 4-13
- Daubechies filter banks and wavelets, third order, 3-12
- Decimation Filter VI, 5-7 to 5-10
- DecimationFilter function, 5-22 to 5-24
- denoising application, 2-13, 4-20
- Design Panel, 4-6 to 4-10
 - accessing, 4-6

- choices for $G_0(z)$ and $H_0(z)$, 4-9
- illustration, 4-6
- steps for designing wavelets and filter banks, 4-7 to 4-8
- utilities, 4-10
- designing wavelet and filter banks. *See* wavelet and filter bank design.
- detrend application, 2-12
- digital filter banks, 3-1 to 3-14
 - two-channel perfect reconstruction, 3-1 to 3-12
 - biorthogonal, 3-4 to 3-10
 - finite impulse response (FIR) filter (note), 3-2
 - $G(z)$ and $H(z)$ filter banks, 3-2
 - orthogonal, 3-10 to 3-12
 - relationship to wavelet transform, 3-3 to 3-4
 - typical two-channel system (figure), 3-1
 - two-dimensional signal processing, 3-13 to 3-14
- discontinuity detection, 2-9 to 2-10
- diskettes, for Wavelet and Filter Bank Design Toolkit, 1-1
- distortion term, 3-5
- documentation
 - conventions used in manual, *x-xi*
 - organization of manual, *x-x*
 - related documentation, *xi*
- dual B-spline filter banks, 3-9

E

- electronic support services, C-1 to C-2
- e-mail support, C-2
- entropy-based criterion, 4-19
- equiripple halfband filter
 - biorthogonal filter banks, 3-6 to 4-7
 - non-negative (figure), 4-3
 - $P_0(z)$ filter design, 3-6
 - wavelet and filter bank design, 4-7 to 4-8
- error codes, B-1 to B-2

Extension Type

- 1D_Test panel, 4-11 to 4-12
 - Symmetric Extension, 4-11 to 4-12
 - Zero Padding, 4-11
- 2D_Test panel, 4-16
 - Symmetric Extension, 4-16
 - Zero Padding, 4-16

F

- Fax-on-Demand support, C-2
- filter bank coefficient functions
 - AllocCoeffWFBD function, 5-14
 - FreeCoeffWFBD function, 5-24
 - ReadCoeffWFBD function, 5-27
- filter banks. *See* digital filter banks.
- finite impulse response (FIR) filters
 - behavior of filters in WFBD Toolkit, 4-3
 - definition of z -transform (note), 3-2
- FIR filters. *See* finite impulse response (FIR) filters.
- flat filter, maximum. *See* maximum flat filter.
- Fourier coefficient, 2-2
- Fourier expansion, 2-3
- Fourier transform
 - conventional, 2-1 to 2-3
 - short-time
 - definition, 2-1
 - sampling grid (figure), 2-7
 - wavelet analysis vs., 2-7 to 2-9
 - windowed, 2-1
- FreeCoeffWFBD function, 5-24
- frequency, calculating, 2-3 to 2-4
- FTP support, C-1
- functions
 - LabVIEW VI applications
 - 2-Channel Analysis Filter Bank VI, 5-1 to 5-4
 - 2-Channel Synthesis Filter Bank VI, 5-4 to 5-5
 - 2D Analysis Filter Bank VI, 5-5 to 5-6
 - 2D Synthesis Filter Bank VI, 5-6

- Decimation Filter VI, 5-7 to 5-10
- Interpolation Filter VI, 5-10 to 5-12
- Mother Wavelet and Scaling function VI, 5-12
- LabWindows/CVI applications
 - AllocCoeffWFBD, 5-14
 - Analysis2DArraySize, 5-15 to 5-16
 - AnalysisFilterbank, 5-17 to 5-18
 - AnalysisFilterbank2D, 5-18 to 5-21
 - calling WFBD functions, 5-13
 - DecimationFilter, 5-22 to 5-24
 - FreeCoeffWFBD, 5-24
 - InterpolationFilter, 5-24 to 5-26
 - ReadCoeffWFBD, 5-27
 - Synthesis2DArraySize, 5-27 to 5-29
 - SynthesisFilterBank, 5-29 to 5-30
 - SynthesisFilterBank2D, 5-31 to 5-33
 - WFBD instrument driver, 5-13 to 5-14
- Windows applications, 5-34

G

$G(z)$ filter banks, 3-1 to 3-2

H

- Haar, Alfred, 2-1
- halfband filter. *See* equiripple halfband filter.
- harmonically related complex sinusoidal functions, 2-2
- highpass filters
 - approximations of wavelet transform, 3-3
 - $G_1(z)$ and $H_1(z)$ as, 3-2
- HP-UX installation, 1-2
- $H(z)$ filter banks, 3-2

I

- images
 - image compression, 3-13 to 3-14
 - subimage arrangement (figure), 4-14 to 4-15

- installation, 1-1 to 1-2
 - Macintosh and Power Macintosh, 1-2
 - Sun and HP-UX, 1-2
 - Windows, 1-2
- Interpolation Filter VI, 5-10 to 5-12
- InterpolationFilter function, 5-24 to 5-26

L

- LabVIEW VIs
 - 2-Channel Analysis Filter Bank VI, 5-1 to 5-4
 - 2-Channel Synthesis Filter Bank VI, 5-4 to 5-5
 - 2D Analysis Filter Bank VI, 5-5 to 5-6
 - 2D Synthesis Filter Bank VI, 5-6
 - Decimation Filter VI, 5-7 to 5-10
 - Interpolation Filter VI, 5-10 to 5-12
 - Mother Wavelet and Scaling function VI, 5-12
- LabWindows/CVI functions
 - AllocCoeffWFBD, 5-14
 - Analysis2DArraySize, 5-15 to 5-16
 - AnalysisFilterbank, 5-17 to 5-18
 - AnalysisFilterbank2D, 5-18 to 5-21
 - calling WFBD functions, 5-13
 - DecimationFilter, 5-22 to 5-24
 - FreeCoeffWFBD, 5-24
 - InterpolationFilter, 5-24 to 5-26
 - ReadCoeffWFBD, 5-27
 - Synthesis2DArraySize, 5-27 to 5-29
 - SynthesisFilterBank, 5-29 to 5-30
 - SynthesisFilterBank2D, 5-31 to 5-33
 - WFBD instrument driver, 5-13 to 5-14
- linear phase filter
 - choices for $G_0(z)$ and $H_0(z)$, 4-9
 - filter comparison (table), 4-4
 - zeros distribution (figure), 4-5
- Load Design utility, 4-10
- lowpass filters
 - $G_0(z)$ and $H_0(z)$ as, 3-2
 - impulse response, 3-3
- lvsrc folder, 1-1

M

Macintosh installation, 1-2
 manual. *See* documentation.
 maximum flat filter

- biorthogonal filter banks, 3-6 to 3-7
- wavelet and filter bank design, 4-3, 4-7

 minimum phase filter

- choices for $G_0(z)$ and $H_0(z)$, 4-9
- filter comparison (table), 4-4
- zeros distribution (figure), 4-5

 mother wavelet, 2-6
 Mother Wavelet and Scaling function VI, 5-12
 multiscale analysis, 2-11

N

noise, removing. *See* denoising application.

O

One-Dimensional (1D) Data Test panel,
 4-11 to 4-14

- Data, 4-12 to 4-14
 - Acquire Data, 4-12
 - DAQ Setup, 4-12 to 4-13
 - path, 4-14
 - Read from file, 4-12
- Display, 4-12
- Extension Type, 4-11 to 4-12
 - Symmetric Extension, 4-11 to 4-12
 - Zero Padding, 4-11

 illustration, 4-11
 on-line testing panel, 4-20
 orthogonal filter banks

- two-channel perfect reconstruction filter banks, 3-10 to 3-12
- wavelet and filter bank design, 4-3, 4-7
 - design procedure (figure), 4-2
 - filter comparison (table), 4-4
 - zeros distribution (figure), 4-5

P

performance issues, 2-13 to 2-14
 Power Macintosh installation, 1-2

Q

qualified wavelets, 2-13

R

ReadCoeffWFBD function, 5-27
 references, A-1

S

Save Design utility, 4-10
 Save Filter Coefficients utility, 4-10
 short-time Fourier transform

- definition, 2-1
- sampling grid (figure), 2-7

 Show Filter Coefficients utility, 4-10
 sine waveforms, truncated, 2-2
 sinusoidal functions, complex, harmonically related, 2-2
 subimage arrangement (figure), 4-14 to 4-15
 Sun installation, 1-2
 synthesis filter functions/VIs

- 2-Channel Synthesis Filter Bank VI, 5-4 to 5-5
- 2D Synthesis Filter Bank VI, 5-6
- Synthesis2DArraySize function, 5-27 to 5-29
- SynthesisFilterBank function, 5-29 to 5-30
- SynthesisFilterBank2D function, 5-31 to 5-33

 synthesis filters

- designations in WFBD Toolkit (note), 4-7
- interchangeability of $G(z)$ and $H(z)$ (note), 4-2

T

technical support, C-1 to C-2
 telephone and fax support, C-2
 third order Daubechies filter banks and wavelets, 3-12
 trend, removing, 2-12
 truncated sine waveforms, 2-2
 two-channel perfect reconstruction filter banks, 3-1 to 3-12

- biorthogonal, 3-4 to 3-10
- finite impulse response (FIR) filter (note), 3-2
- $G(z)$ and $H(z)$ filter banks, 3-2
- orthogonal, 3-10 to 3-12
- relationship to wavelet transform, 3-3 to 3-4
- typical two-channel system (figure), 3-1

 Two-Dimensional (2D) Data Test panel, 4-14 to 4-16

- Data, 4-16
- Data Usage, 4-15
- extension, 4-16
 - symmetric extension, 4-16
 - Zero Padding, 4-16
- illustration, 4-15
- Remaining Data (%), 4-15
- subimage arrangement (figure), 4-14 to 4-15

 two-dimensional signal processing, 3-13 to 3-14

V

valid wavelets, 2-13
 VIs. *See* LabVIEW VIs.

W

wavelet analysis, 2-1 to 2-14

- applications, 2-9 to 2-13
- denoise, 2-12
- detrend, 2-12
- discontinuity detection, 2-9 to 2-10
- multiscale analysis, 2-11

 Fourier analysis vs., 2-7 to 2-9
 history, 2-1 to 2-7
 innovative analysis, 2-3 to 2-7
 performance issues, 2-13 to 2-14

wavelet and filter bank design, 4-1 to 4-5

- choices available in WFBD Toolkit (figure), 4-2
- filter comparison (table), 4-4
- non-negative equiripple halfband filter (figure), 4-24-3
- steps, 4-1
- zeros distribution
 - linear phase filter (figure), 4-5
 - minimum phase filter (figure), 4-5
 - orthogonal filter (figure), 4-5

 Wavelet and Filter Bank Design Toolkit

- applications, 1-3
- creating applications, 4-18 to 4-20
 - denoising, 4-20
 - on-line testing panel, 4-20
 - wavelet packet analysis, 4-18 to 4-19
- Design Panel, 4-6 to 4-10
 - accessing, 4-6
 - choices for $G_0(z)$ and $H_0(z)$, 4-9
 - illustration, 4-6
 - steps for designing wavelets and filter banks, 4-7 to 4-8
 - utilities, 4-10
- installation procedure, 1-1 to 1-2
- 1D_Test panel, 4-11 to 4-14
 - Data, 4-12 to 4-14
 - Display, 4-12
 - Extension Type, 4-11 to 4-12
 - illustration, 4-11
- package contents, 1-1
- 2D_Test panel, 4-14 to 4-16
 - Data, 4-16
 - Data Usage, 4-15
 - extension, 4-16

- illustration, 4-15
- Remaining Data (%), 4-15
- wavelet and filter bank design, 4-1 to 4-5
- Wavelets and Filters panel, 4-16 to 4-17
 - illustration, 4-17
 - Refinement, 4-17
 - Save Scaling and Wavelets, 4-17
- wavelet coefficients
 - definition, 2-6
 - relationship with filter banks (figure), 3-4
- wavelet packet analysis application, 4-18 to 4-19
- wavelet transform
 - approximations, 3-3
 - compared with Fourier transform, 2-8 to 2-9
 - definition, 2-6
 - relationship to two-channel perfect reconstruction filter banks, 3-3 to 3-4
 - sampling grid (figure), 2-8
- wavelets
 - definition, 2-4
 - mother wavelet, 2-6
 - qualified, 2-13
 - valid, 2-13
- Wavelets and Filters panel, 4-16 to 4-17
 - Refinement, 4-17
 - Save Scaling and Wavelets, 4-17
- WFBD instrument driver, 5-13 to 5-14
- WFBD Toolkit. *See* Wavelet and Filter Bank Design Toolkit.
- WFBD32.DLL, 1-1
- WFBD32.FP instrument driver, 1-1
- WFBD.EXE, 1-1
- windowed Fourier transform, 2-1
- Windows 3.x, Windows NT, and Windows 95
 - dynamic link libraries, 5-34
 - installation, 1-2

Z

- zeros distribution
 - linear phase filter (figure), 4-5
 - minimum phase filter (figure), 4-5
 - orthogonal filter (figure), 4-5