Wavelets 2.0

October 22, 2003

TIBCO Software Inc.

Preface

Acknowledgments

The S+Wavelets module is based upon work supported by the Air Force Office of Scientific Research under contracts #F49620-99-C-0068 and #F49620-00-C-0039, and by the National Science Foundation under contract #DMI-9628595. The technical monitor for the Air Force contract is Dr. Don Washburn at Kirtland Air Force Base, Albuquerque, New Mexico. We would like to thank Dr. Arje Nachman of the Air Force Office of Scientific Research, and Major Mike Johnson and Lieutenant Colonel Randy Lefevre of the Airborne Laser System Program Office for their guidance and support. The technical monitor for the NSF contract is G. Patrick Johnson at the division of Design, Manufacture, and Industrial Innovation.

This supplement and the S+Wavelets module are products of the efforts made by the Insightful Corporation. Many of the mathematical and statistical ideas behind this software are described more thoroughly in the book co-authored by one of us (Percival) and Andrew Walden entitled, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000. We are particularly indebted to the efforts of Prof. Peter Craigmile, Department of Statistics, Ohio State University, who was partially supported by the Air Force STTR contract during his doctoral candidacy in the Department of Statistics at the University of Washington. His thesis work helped establish the theory behind many of the principles used in wavelet-based analysis of fractionally differenced processes. The S+Wavelets 2.0 module was developed primarily by one of use (Constantine) as principal investigator on the Air Force phase I and phase II contract as well the principal investigator on the NSF contract during phase II. We would also like to thank the many co-workers who contributed to the development of the S+Wavelets module including Luca Cazzanti, Dr. Keith Davidson, Dr. Jill Goldschneider and and Dr. Sylvia Isler. Particular thanks and credit go to Dr. Alan Gibbs, who developed the code for the dual tree wavelet transform and wrote the section of this manual describing it (§3.7).

William L.B. Constantine
Donald B. Percival
September 2002

Contents

1
Introduction

1.1 Advantages of New Wavelet Transforms

1.1.1 Discrete Wavelet Transforms

In this version of the S+Wavelets module we introduce a variety of new discrete wavelet transforms based on those developed in [PW00] and in [Kin98, Kin99, Kin00]. Previous versions of the S+Wavelets module support similar transforms, but there are some distinct differences between old and new.

- **Identification of boundary coefficients.** For finite length time series, there are a number of methods used to extend the boundaries of a sequence in order to preserve the length of the time series in the transform. The number of coefficients that are influenced by these boundary treatments grows with the decomposition level. It is important to keep track of these so-called "boundary" coefficients when applying statistical measures to the wavelet transform coefficients, e.g., wavelet variance estimators. While former versions of the S+Wavelets module offer a variety of boundary treatments, there currently exists no methodology to track the corresponding boundary coefficients. In the new S+Wavelets module, we have embedded functionality to track the boundary coefficients, enabling us to calculate cross-scale unbiased statistics on the transform coefficients.

- **Assurance of energy preservation.** In orthonormal transforms, the energy of the original time series is equal to that of the collective transform coefficients. In forming certain statistics of the wavelet transform coefficients, it is vital that the energy be preserved. To guarantee energy preservation, the wavelet transforms in the new S+Wavelets module impose a periodic boundary extension rule. Other existing boundary extension treatments such as poly0, poly1, poly2, zero, and reflection do *not* guarantee energy conservation in the transform.

- **Handling non-dyadic time series.** A time series is dyadic if its length is a power of two. The definition of a discrete wavelet transform is typically based on the assumption that the time series is dyadic and, in order to overcome this restriction, special techniques must be used. In the former S+Wavelets modules, the wavelet transforms handled non-dyadic time series in a way that made the number of transform coefficients at a given level difficult for the user to predict. In the new S+Wavelets module, we use a novel treatment for non-dyadic sequences that follows a simple rule applied only to the transform coefficients corresponding to low pass filter operations, i.e., the so-called scaling coefficients: *if at any given decomposition level, the number of scaling coefficients is odd, then the last scaling coefficient is*

preserved in a special crystal named extra. For reconstruction (synthesis) operations, the extra scaling coefficients are simply put back into place at the appropriate point in the reconstruction algorithm. While this technique requires a bit more bookkeeping (on our part), it does not impose any spurious energy in the transform coefficients. Furthermore, the number of coefficients at each level is easily calculable.

- **Convolution style filtering.** Prior to this version of S+Wavelets, the default filtering style was *correlation*. In the new wavelet transforms, the filtering style is *convolution*. Both are valid means of obtaining wavelet transform coefficients, but the convolution style has the advantage of being easily related to the discrete Fourier transform. The relation we speak of states that the convolution of a time series and a filter in the time domain is equivalent to the (inverse Fourier transform of) the product of the filter and time series Fourier transforms.

We also introduce a new discrete wavelet transform to the S+Wavelets module: the dual tree wavelet transform (DTWT) developed by Dr. N.G. Kingsbury of the University of Cambridge. Like non-decimated wavelet transforms, the DTWT is an over-complete transformation of the data, producing more transform coefficients than that found in the original data. The redundancy is a factor of 2 in one dimension, and 4 in two dimensions. By comparison, conventional non-decimated DWTs impose a J-fold increase in the number of coefficients for (one-dimensional) time series, where J is the decomposition level. The advantages of the DTWT are that it retains perfect reconstruction, achieves a good (approximate) degree of shift invariance, has good directional selectivity, and has low noise amplification, all at the cost of a modest degree of redundancy. The DTWT is available for both 1-D and 2-D transformations.

A number of promising applications of the DTWT have been reported in the recent literature by Kingsbury and his co-workers. We summarize several of these here as an indication of the potential and versatility of this transform.

- In video coding, because transforms of shifted images are simply related, motion vectors between successive video frames may be estimated by analysis in the transform domain [MK98].
- In image denoising, application of soft thresholding to the complex wavelet coefficients produces results comparable to those obtained using the non-decimated transform (MODWT), but with considerably less computation [Kin98]. The denoising performance may be enhanced by use of hidden Markov tree (HMT) methods [CRK00].
- In texture analysis, the directional and shift-invariant features of the DTWT make it ideal [HMK99, dRK99, HB00].
- In image classification, techniques can be based on multiscale texture and color feature vectors obtained from the DTWT [KNKF00, RCBK00, dRK00].
- In digital image watermarking, a promising method can be based on the DTWT [LK00].

1.1.2 Gabor Transform

The new S+Wavelets contains the Gabor transform (2-D only).

1.2 Estimation of the Wavelet Variance

In this version of S+Wavelets, we introduce functionality to estimate the wavelet variance of a time series and to provide corresponding confidence intervals. The wavelet variance provides a way of analysing (decomposing) the variance of a time series and is of interest for several reasons:

- The wavelet variance breaks up the variance of certain stochastic processes into pieces that are associated with different scales. This type of decomposition has considerable appeal for studying time series that exhibit fluctuations over a range of different scales, as occurs quite commonly in financial and geophysical time series.

- The wavelet variance is closely related to the concept of the spectral density function (SDF). While the SDF decomposes the variance of a process into components that can be associated with particular frequencies, the wavelet variance decomposes the variance into components associated with particular scales. Since each scale can in turn be related to a certain range of frequencies in the data (as opposed to a single frequency), the wavelet variance often leads to a more succinct decomposition that is easier to interpret.
- The wavelet variance is a useful substitute for the variance of a process for certain processes with infinite variance or for a process whose sample variance has poorly behaved sampling properties.

1.3 Wavelet-Based Analysis of Fractionally Differenced Processes

Recently there has been wide-spread interest in the use of nonstationary multi-scale fractal processes as models for environmental, biomedical and financial time series. There are, however, a number of modeling and estimation issues that have yet to be fully resolved. In the area of modeling there is need for a flexible – yet tractable – class of models that are capable of describing the nonstationarities typically observed in recorded measurements of (fractal) processes. The challenge is to develop models that capture the salient features of fractal times series, yet do so in a relatively simple manner. One such model is the *fractionally differenced (FD) process*. For certain fractal processes, an FD process model has many advantages over conventional models (such as fractional Brownian motion and fractional Gaussian noise models) including model flexibility, simplicity and continuity over stationary-nonstationary model parameter regimes.

The S+Wavelets module utilizes novel wavelet-based techniques to estimate the (possibly time-varying) FD model parameters over finite ranges of scale. Wavelet methods have many advantages over other techniques:

- DECOMPOSITION BASED ON SCALE. Many real-world (fractal) phenomena exhibit fluctuations at various temporal or spatial scales, e.g., atmospheric turbulence measurements, financial market fluctuations, and environmental time series to name a few. A wavelet transform is a natural analyzer for such data since it transforms a time series into coefficients that correspond to (differences and averages) of such variations over a range of different scales.
- DECORRELATION OF TIME SERIES. Many fractal processes, such as long-memory processes, are highly correlated. In this case, estimating the FD model parameters via a maximum likelihood estimator (MLE) is computationally intense, but is relatively efficient for uncorrelated fractal processes. Craigmile et al show that using a DWT on correlated FD process realizations yields wavelet coefficients that are approximately uncorrelated [CPG00a]. Thus, to make MLE of FD model parameters practical, the DWT can be used as a preprocessive measure to decorrelate correlated FD processes.
- LOCALIZED TIME AND SCALE CONTENT. Each wavelet coefficient is localized in time, allowing us to track changes in the characteristics of a time series at a particular scale as a function of time.
- SEPARATION OF NONLINEAR TRENDS FROM NOISE. The wavelet coefficients are inherently 'blind' (invariant) to nonlinear polynomial trend contamination in the original time series [CPG00b].

In addition to its FD estimation routines, the S+Wavelets module contains functionality for simulating time-varying fractionally differenced (TVFD) processes. TVFD processes can serve as useful models for certain time series whose statistical properties evolve over time. The TVFD functionality developed for S+Wavelets has many advantages over other techniques:

- The FD model parameters are allowed to fluctuate as a function of time. This is beneficial for modeling some real-world processes which exhibit fluctuating stochastic fractal dynamics as time evolves.
- The TVFD routines generate mathematically "exact" realizations, thus ensuring that if we conducted a Monte Carlo experiment to determine the properties of a statistic formed from a TVFD process, our results will reflect the correct properties of the statistic rather than the inaccuracies in the simulation technique.

- The range of supported TVFD model parameters exceeds that of (related) model parameters for other techniques such as locally stationary processes [Pri65, Dah97, MPZ98, CHT98, Mal99] and locally self-similar process [GF93, FG94, WCS01]. Although these classes have served as successful models for some time series, the corresponding restricted model parameter range are not reasonable for certain real-world time series, whereas the range for our TVFD simulator is appropriate for most (if not all) real-world phenomena.

1.4 Organization of Supplement

The remainder of this S+Wavelets module is organized as follows:

In §2 we introduce the filter functions used to generate suitable wavelet and scaling filters for use in discrete wavelet transforms. In §3 we discuss both one-dimensional and two-dimensional discrete wavelet transforms and the parameters needed to fine-tune their output. In §4 we introduce the concept of wavelet variance and discuss its ramifications on scale selection and FD model validation. In §5 we introduce FD model simulation and estimation functions and show its potential in analyzing both simulated and physical stochastic fractal process realizations. In Appendix A and Appendix B we provide the data and function references, respectively, for the S+Wavelets module. In Appendix C we define the object-oriented design of the S+Wavelets module and include a description of classes and corresponding methods.

2
Filter Functions

2.1 What is a Wavelet Filter?

Qualitatively speaking, a function $\psi(\cdot)$ defined over the entire real axis is called a wavelet (or 'short wave') if $\psi(t) \to 0$ as $t \to \pm\infty$. For a contrasting example, consider a function whose value at t is given by $\cos(2\pi t/10)$. Because this function has no tendency to get smaller as t gets larger and larger, we can call it a 'long wave.' By judiciously defining a wavelet and by comparing it to various portions of another continuous function of t, we can expose or extract variations in portions of this second function over the effective width (i.e., scale or duration) of the wavelet.

In practical applications, we invariably deal with sequences of values indexed by integers rather than functions defined over the entire real axis (such sequences are often called 'time series' for convenience). In place of actual wavelets, we use short sequences of values referred to as wavelet filters. The number of values in the sequence is called the width of the wavelet filter. For technical reasons, this width, denoted by L, must be an even integer. We use the notation $\{h_l\}_{l=0}^{L-1}$ (or just $\{h_l\}$) to denote a wavelet filter, where h_l is called a filter coefficient (the set of all filter coefficients is commonly called the impulse response sequence). Quantitatively, a wavelet filter must meet the following three mathematical conditions.

1. The filter coefficients must sum to zero:
$$\sum_{l=0}^{L-1} h_l = 0.$$
2. The coefficients must have unit energy:
$$\sum_{l=0}^{L-1} h_l^2 = 1.$$
3. The coefficients must be orthogonal to their even shifts:
$$\sum_{l=0}^{L-1} h_l h_{l+2k} = 0,$$
where k is any nonzero integer, and for convenience we define $h_l = 0$ for $l < 0$ or $l \geq L$.

Together the second and third conditions are called the orthonormality condition.

The wavelet filter is used in combination with a *scaling* filter to calculate a discrete wavelet transform. The wavelet filter coefficients h_l are related to the scaling filter coefficients g_l through a quadrature mirror filter (QMF) relationship, namely,

$$g_l = (-1)^{l+1} h_{L-1-l}, \;\; l = 0, \ldots, L-1.$$

Given a scaling filter, we can obtain the corresponding wavelet filter via $h_l = (-1)^l g_{L-1-l}$. The scaling filter satisfies the same orthonormality condition as the wavelet filter, but, rather than summing to zero, we have $\sum_l g_l = \sqrt{2}$. The scaling filter and wavelet filter are sometimes referred to in wavelet literature as the 'father wavelet filter' and 'the mother wavelet filter', respectively.

2.2 Supported Wavelet Filters

There are several families of wavelet filters whose members satisfy the three mathematical conditions stated above. The following four families are supported in S+Wavelets.

- EXTREMAL PHASE (**d**): Filters in this family are designed such that, if we use an impulse as the input, then the output from the filter will have a cumulative energy that increases as fast as possible. Extremal phase filters are also know as 'daublets' or 'minimum phase' filters.
- LEAST ASYMMETRIC (**s**): Filters in this family are smoother and more symmetric looking than extremal phase filters, but they have exactly the same gain functions (i.e., the magnitudes of their discrete Fourier transforms). These filters are approximations to linear phase filters (a filter with linear phase is desireable because we can then match up patterns going into the filter with those coming out). They are also known as 'symmlets'.
- BEST LOCALIZED (**l**): These filters have the same gain functions are the extremal phase and least asymmetric filters. Like the latter, they are intended to be as symmetric as possible, but they use a slightly different definition of what constitutes good symmetry.
- COIFLET (**c**): These filters have different gain functions from the other three families and are much better approximations to linear phase filters.

The extremal phase, least asymmetric and coiflet families of wavelet filters are discussed in detail in the pioneering book on wavelets by Ingrid Daubechies [Dau92]. She coined the term 'coiflet' in honor of Ronald Coifman (another prominent 'waveletician'), to whom she credits the idea for the family. The best localized family is due to Doroslovački [Dor98].

2.3 The WaveletDaubechies Class

The wavDaubechies function retrieves wavelet and scaling filters and returns them in the form of an object of class WaveletDaubechies. The following methods can be used to view, summarize and access the data contained in a WaveletDaubechies object.

Summary Operator Methods:

print — Prints the following information about the wavelet and scaling filters:

- the filter type (e.g., 'Extremal Phase')
- the width L of the wavelet and scaling filters
- a logical flag stating the normalization status of the filters
- the coefficients (i.e., impulse response sequence) for the filters

plot — Plots the coefficients for wavelet and scaling filters or components of their frequency response functions (these functions are the discrete Fourier transforms of the filter coefficients and are also commonly called the transfer functions). The plot function takes a second argument (type), which should be set to the string "time", "gain", or "phase" to plot, respectively, the filter coefficients, the gain function (i.e., the modulus of the frequency response function) and the phase of the frequency response function.

Data Access Methods

$ — Use to access specific components of the class object. A list of accessible components can be generated using the names function.

For example, if s8 is an object of class WaveletDaubechies, then names(s8) will give a list of its components. Two of these components are wavelet and scaling, so s8$wavelet will return the wavelet filter coefficients, while s8$scaling will return the scaling filter coefficients.

2.4 An Example: Retrieving Wavelet and Scaling Filters

To retrieve a particular wavelet filter, use the wavDaubechies function along with a string denoting the desired filter. This string should start with either d (for extremal phase filters, i.e., daublets), s (for least asymmetric filters, i.e., symlets), l (for best localized filters) or c (for coiflets) and should be followed by the width of the desired filter. The wavDaubechies function also takes an optional second argument, which is a logical flag used to denote whether the filters should be unnormalized or normalized (the default). Filters with unit energy are *not* considered to be normalized – normalized filters have energies of 0.5 and are used with the 'maximal overlap' version of the discrete wavelet transform described in §3.2)

```
> s8 <- wavDaubechies("s8", norm=F)
> s8
Filter type: Least Asymmetric
Filter length: 8
Filter normalized:  FALSE

Wavelet filter:
[1]  0.03222310  0.01260397 -0.09921954 -0.29785780
[5]  0.80373875 -0.49761867 -0.02963553  0.07576571


Scaling filter:
[1] -0.07576571 -0.02963553  0.49761867  0.80373875
[5]  0.29785780 -0.09921954 -0.01260397  0.03222310
```

The wavelet and scaling filter coefficients are returned embedded in an object of class WaveletDaubechies. As shown in the example above, the print method for this class lists exactly what filters these are and their normalization status, following which it gives the coefficients (i.e., impulse response sequence) for both filters. As an example of accessing one of the components in the object s8, let us check that the wavelet filter coefficients have unit energy:

```
> sum(s8$wavelet^2)
[1] 1
```

Use the plot method to view the filter coefficients (i.e., impulse response sequence) and the components of their frequency response functions:

```
> plot(wavDaubechies("s20", norm=F), type="time")
> plot(wavDaubechies("s20", norm=F), type="gain")
> plot(wavDaubechies("s20", norm=F), type="phase")
```

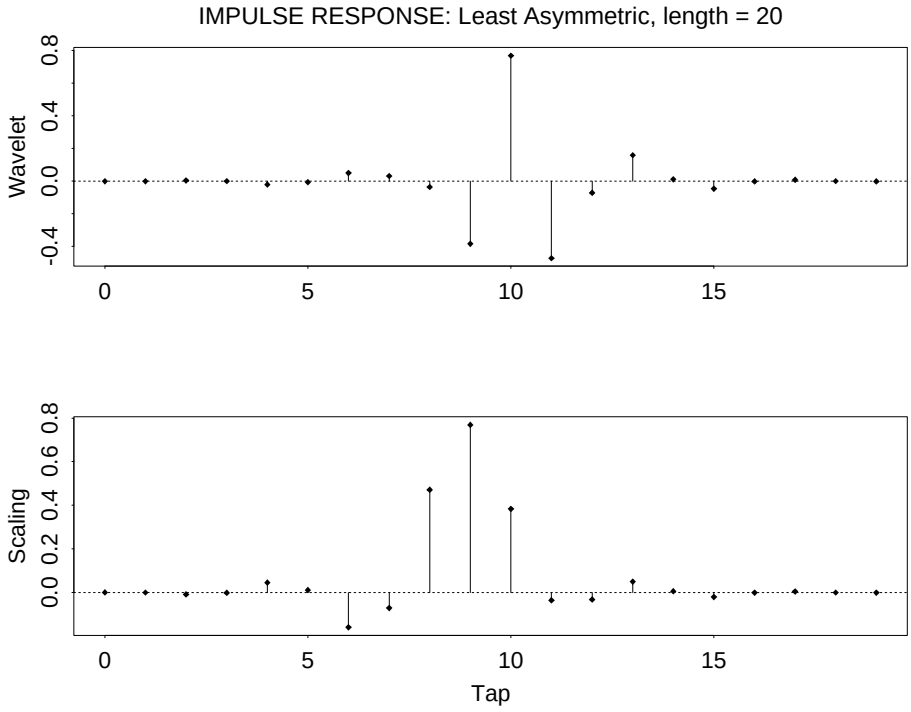

FIGURE 2.1: Filter coefficients (i.e., impulse response sequence) for Daubechies least asymmetric wavelet and scaling filters of width $L = 20$.

The gain functions reveal that the wavelet and scaling filters are high and low pass filters, respectively.

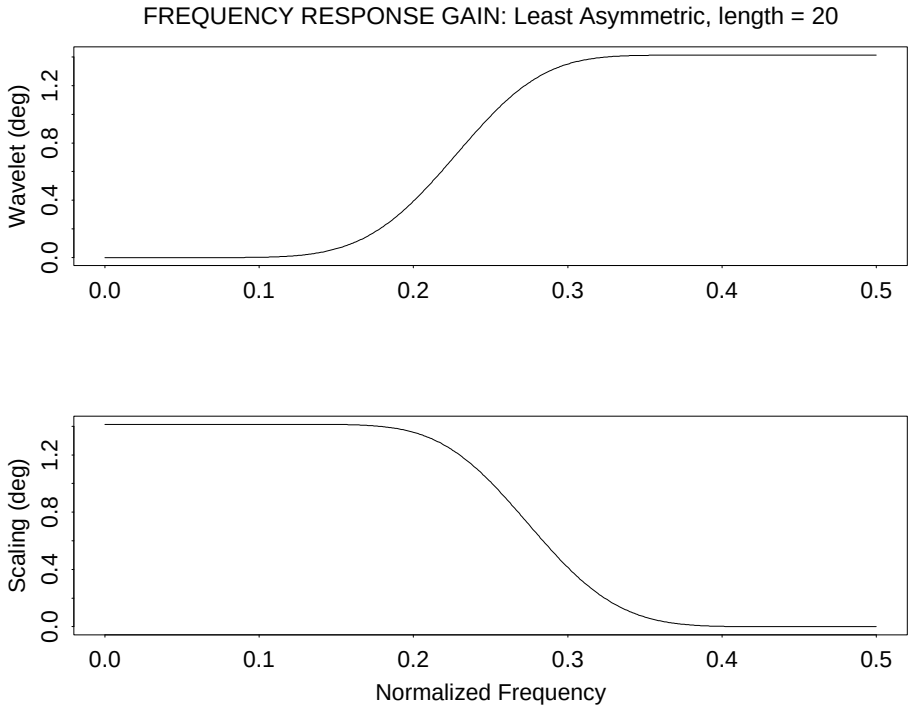

FIGURE 2.2: Gain functions (i.e., modulus of the frequency response functions) for Daubechies least asymmetric wavelet and scaling filter of width $L = 20$.

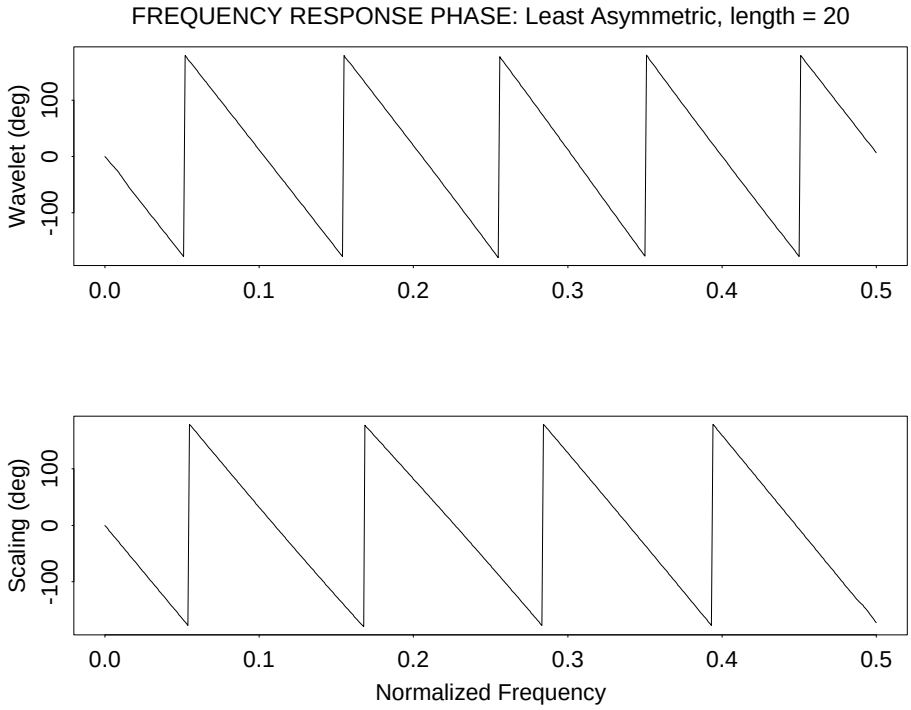

FIGURE 2.3: Phase of frequency response functions for Daubechies least asymmetric wavelet and scaling filter of width $L = 20$. The vertical axis is expressed in degrees.

3
Wavelet Transforms

3.1 The Discrete Wavelet Transform

3.1.1 Overview

The wavelet filter $\{h_l\}$ and scaling filter $\{g_l\}$ discussed in §2 are used to create a discrete wavelet transform (DWT), which maps a time series from its original representation in the *time* domain to an alternative representation in the *time-scale* domain. The DWT is implemented in the time domain by recursively applying both filters to the time series in a 'pyramid' algorithm. Each recursive use of these filters yields one level of wavelet coefficients and scaling coefficients. If we stop after reaching a particular level, say J, and if the sample size N of our time series can be expressed as an integer multiple of 2^J, the DWT consists of the wavelet coefficients for levels $j = 1, 2, \ldots, J$ along with the scaling coefficients for level J. If we let dj and sj represent the collection of wavelet and scaling coefficients, respectively, at level j, we can represent the DWT for level J as $[\mathsf{d1}|\mathsf{d2}|\ldots|\mathsf{dJ}|\mathsf{sJ}]$. We use the term 'crystal' to refer to any one of the components d1, d2, ... dJ, sJ in the DWT; i.e., a level J DWT consists of $J+1$ crystals, J of which are the wavelet coefficients at the J different levels, and one of which contains the scaling coefficients for level J. (If N is not an integer multiple of 2^J, an extra cystral is needed. This consists of at most one scaling coefficient from levels $j = 1, \ldots, J-1$. In this case, we can represent the DWT as $[\mathsf{d1}|\mathsf{d2}|\ldots|\mathsf{dJ}|\mathsf{sJ}|\mathsf{extra}]$.)

As an example, let's consider a linear chirp, i.e., a sinusoid whose frequency increases linearly with time.

```
> linchirp <- make.signal( "linchirp", n=1024 )
> plot( linchirp, type="l", ylab="linchirp" )
```

Figure 3.1 shows the resulting plot. You can use the wavDWT function to calculate a level $J = 4$ DWT using the s8 wavelet filter, i.e., the Daubechies least asymmetric filter of width $L = 8$.

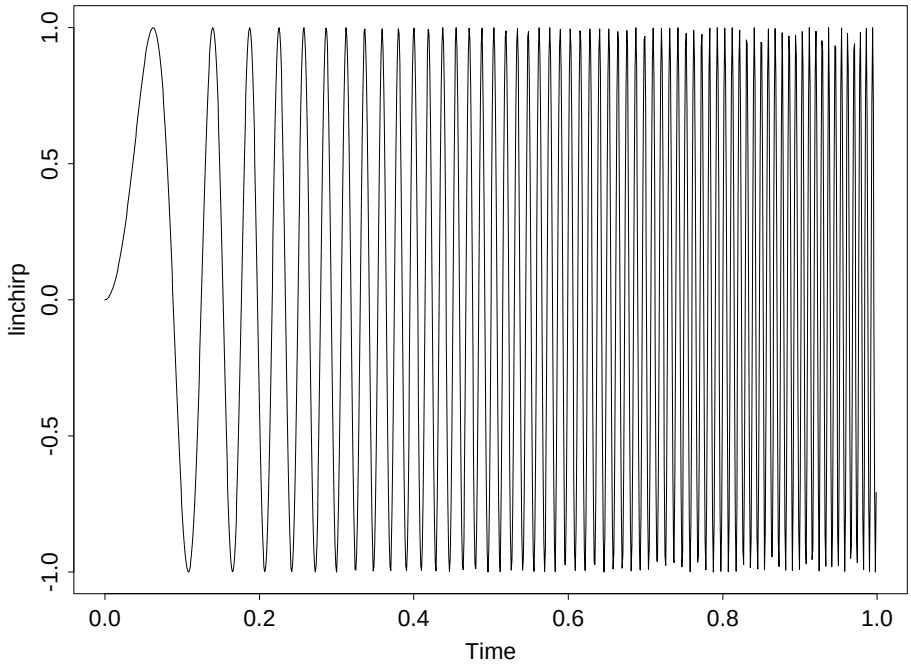

FIGURE 3.1: A 1024 point sample of a linear chirp.

```
> dwt.linchirp <- wavDWT( linchirp, wavelet="s8", n.levels=4 )
> dwt.linchirp

Discrete Wavelet Transform of linchirp
Wavelet: s8
Length of series: 1024
Number of levels: 4
Boundary correction rule: periodic
Filtering technique: convolution
Shifted for approximate zero phase: FALSE
Crystals: d1 d2 d3 d4 s4
```

To plot the DWT, simply invoke the plot method:

```
> plot( dwt.linchirp )
```

Figure 3.2 is a plot of the elements of each crystal versus a nominal position (positions 0 to 1023 correspond to times 0 to 1; §3.1.4 below discusses a refinement for associating the values in each crystal with a particular time). If we compare this figure with Fig. 3.1, we see that, as the level of the wavelet coefficients increases, the positions of its largest elements are shifted toward the beginning of the series. As we discuss in §3.1.3 below, this is in keeping with a band-pass interpretation of the DWT in which the high level wavelet coefficients are capturing information in a time series over certain bands of low frequencies. The frequencies in our linear chirp are increasing with time, which implies that the positions of large wavelet coefficients should shift to the left as the level increases – this is in agreement with Fig. 3.2. Similarly the scaling coefficients capture the very lowest frequency information about our time series, which is also consistent with what is shown in Fig. 3.2.

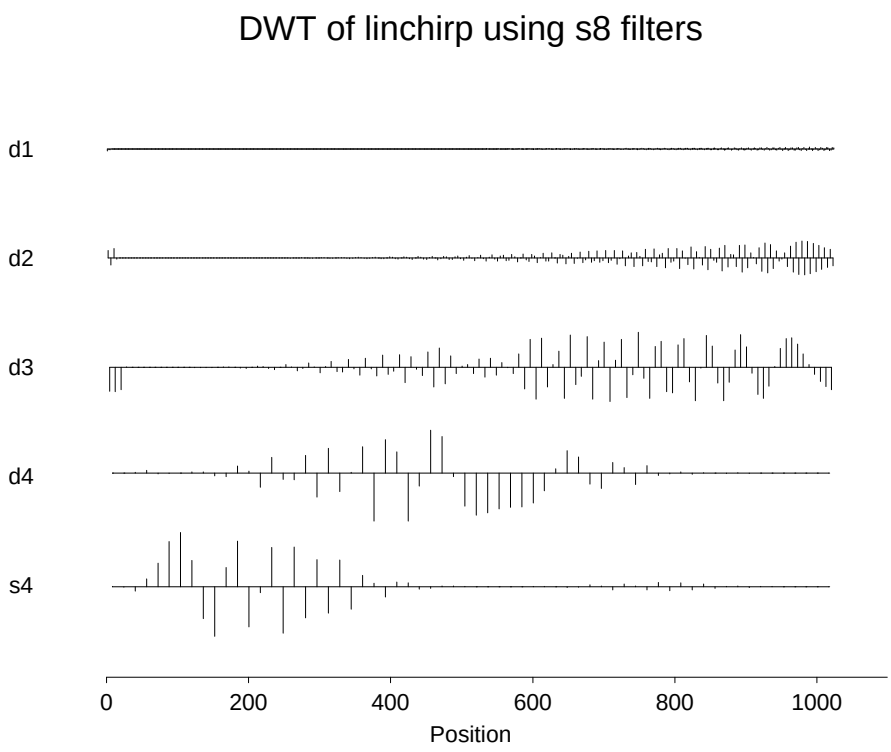

FIGURE 3.2: Discrete wavelet transform of a linear chirp.

We can easily access the components (i.e., crystals) of a DWT. For example, the wavelet coefficients for level 4 are in the crystal named d4. You can access these coefficients by using dwt.linchirp[["d4"]]. This is done in the following command, in which we calculate the 'energy' in the level 4 wavelet coefficients (i.e., the sum of squares of all the elements in d4).

```
> sum( dwt.linchirp[["d4"]]^2 )
[1] 130.638
```

The energy in particular crystals is of interest because the DWT is an 'energy preserving transform'; i.e., the energy in all the DWT coefficients is equal to the energy in the original time series. We can verify that this is true numerically here.

```
> sum( unlist(lapply(dwt.linchirp$data,
+ function(x) sum(x^2))) )
[1] 500.6865
> sum( linchirp^2 )
[1] 500.6865
```

The fact that the DWT preserves energy means that we can use it to analyze the variance in a time series (this fact is exploited in detail in §4).

In addition to the plot method, we can use the summary method to get some summary statistics about the DWT coefficients.

```
> summary(dwt.linchirp)

      Min     1Q Median    3Q   Max   Mean    SD   Var
d1 -0.144 -0.005  0.000 0.005 0.130  0.000 0.037 0.001
d2 -1.265 -0.103  0.000 0.091 1.271  0.001 0.425 0.181
d3 -2.529 -0.615 -0.001 0.603 2.629 -0.035 1.236 1.529
d4 -3.585 -0.820 -0.001 0.171 3.212 -0.224 1.422 2.023
s4 -3.711 -0.120  0.004 0.252 4.028  0.120 1.426 2.033

     MAD Energy %
d1 0.008    0.141
d2 0.146    9.194
d3 0.918   38.804
d4 0.661   26.092
s4 0.283   25.769

Energy Distribution:
           1st     1%     2%     3%     4%     5%     10%
Energy % 3.240 25.682 39.772 51.277 61.146 69.638  89.481
|coeffs| 4.028  2.966  2.494  2.315  2.110  1.865   1.042
 #coeffs 1.000 11.000 21.000 31.000 41.000 52.000 103.000

             15%     20%     25%
Energy %  96.233  98.793  99.589
|coeffs|   0.637   0.361   0.213
 #coeffs 154.000 205.000 256.000
```

The first block of the summary gives some statistics related to the empirical distribution of the values within each crystal (minimum value, first quartile, median (i.e., second quartile), third quartile, maximum value, mean, standard deviaton, sample variance and the median absolute deviation), along with the percentage of the total energy in the time series trapped within each crystal. You can also look at the energy distribution in the form of two graphs that are created by supplying a type argument to the plot function:

```
> plot( dwt.linchirp, type="energy" )
```

The result (Fig. 3.3) is a bar plot showing the energy in each crystal and a pie chart depicting the proportion of the total energy trapped by each crystal.

The summary method also gives us a look at how the energy is distributed across all the DWT coefficients, i.e., outside of their classification within particular crystals. An examination of the last part of the summary above shows that the coefficient with the largest magnitude (4.028) accounts for 3.2% of the energy in the time series. We also see that 15% of the coefficients contain 96.2% of the energy and that the magnitudes of these coefficients are bounded below by 0.637 (there are 154 coefficients with magnitudes greater than or equal to this number). A summary such as this is helpful in determining how well the DWT succinctly captures the salient features in a time series

3.1.2 The DWT in the Time Domain

The DWT can be calculated by processing the data using a *filter cascade*, where a wavelet filter $\{h_l\}$ and its associated scaling filter $\{g_l\}$ are used in a 'pyramid' algorithm to decompose the time series. To generate the first level of coefficients, the original data is filtered by convolving it separately with the wavelet and scaling filters. Both filter outputs are then decimated by a factor of two (i.e., every other point is thrown out), and the

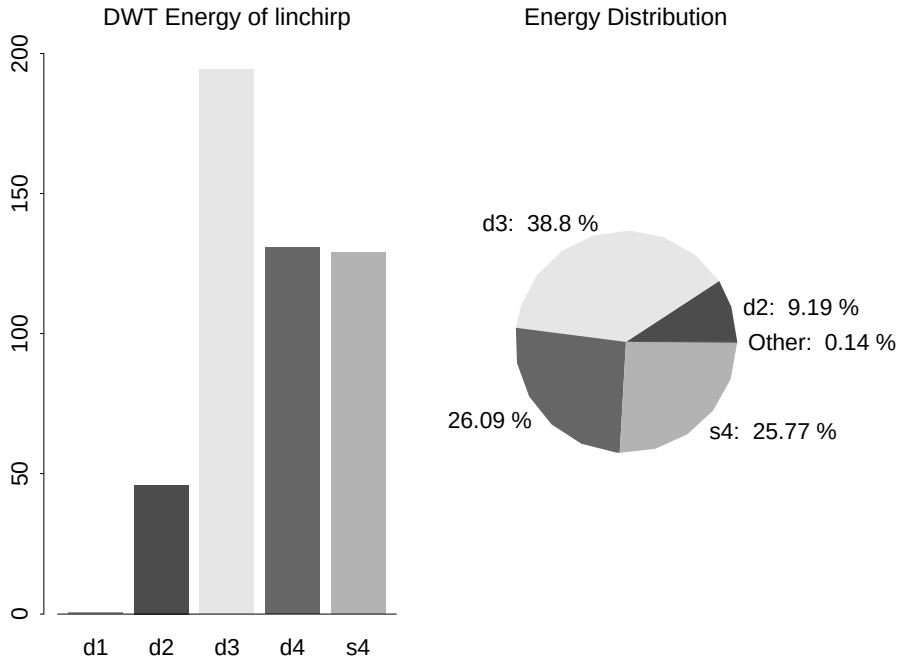

FIGURE 3.3: Energy distribution of linear chirp DWT.

remaining filter outputs are defined as the unit level ($j = 1$) wavelet and scaling coefficients. For the second level ($j = 2$), the same filtering/decimation scheme is applied, but now the unit level scaling coefficients are used as the input to the filters. This algorithm can be repeated to obtain higher level coefficients. Thus, at the jth level, the inputs to the wavelet and scaling filters are the scaling coefficients from the previous level ($j-1$), and the outputs are the jth level wavelet and scaling coefficients.

Mathematically we can describe the DWT as follows. Let $\{X_t\}_{t=0}^{N-1}$ be a uniformly sampled real-valued time series consisting of N observations, the tth of which is denoted by X_t (by uniformly sampled, we mean that the time at which we collected the tth observation can be expressed as $t_0 + t\,\Delta_t$, where t_0 is the time at which we observed X_0, and $\Delta_t > 0$ is the sampling time between observations). Define $s_{0,t} = X_t$ to be the zeroth level scaling coefficients. Starting with level $j = 1$ and recursively continuing on for levels $j = 2, \ldots, J$, we can calculate the DWT wavelet coefficients via

$$d_{j,t} \equiv \sum_{l=0}^{L-1} h_l s_{j-1,2t+1-l \bmod N_{j-1}}, \quad t = 0, \ldots, N_j - 1,$$

and the DWT scaling coefficients via

$$s_{j,t} \equiv \sum_{l=0}^{L-1} g_l s_{j-1,2t+1-l \bmod N_{j-1}}, \quad t = 0, \ldots, N_j - 1,$$

where $N_j \equiv \lfloor N/2^j \rfloor$ (here $\lfloor x \rfloor$ refers to the largest integer that is less than or equal to x). Here are four comments about the above two equations.

1. We are using 'modulo' arithmetic to specify which scaling coefficients are to be used in forming the above summations. Thus, if $0 \le 2t+1-1 \le N_{j-1}-1$, the expression '$2t+1-l \bmod N_{j-1}$' is taken to be equal to $2t+1-1$; if this condition is not true, then the expression is taken to be $2t+1-1+nN_{j-1}$, where nN_{j-1} is the unique integer multiple of N_{j-1} such that $0 \le 2t+1-1+nN_{j-1} \le N_{j-1}-1$. For

indices t such that $2t+1-l \bmod N_{j-1}$ is not the same integer as $2t+1-l$, the values of $d_{j,t}$ and $s_{j,t}$ are formed by combining together some values from the beginning of the sequence of scaling coefficients ($s_{j-1,0},\ldots$) with some values from the end ($\ldots,s_{j-1,N_{j-1}-1}$). In effect, we are treating the scaling coefficients as if they were 'circular' for the purposes of creating the filter outputs; i.e., we replace the unavailable values $\ldots,s_{j-1,-2},s_{j-1,-1}$ with $\ldots,s_{j-1,N_{j-1}-2},s_{j-1,N_{j-1}-1}$. We use the terminology 'boundary coefficient' to refer to any coefficient whose computation involves the circularity assumption in some manner. Because the physical interpretation of boundary coefficients can be problematic, there are special tools and displays in S+Wavelets that allow us to access and delineate these coefficients. Coefficients that are not boundary coefficients are deemed to be 'interior' coefficients (or 'nonboundary' coefficients).

2. In our description of the pyramid algorithm above, we noted that the jth wavelet (scaling, respectively) coefficients are formed by filtering the scaling coefficients from level $j-1$ with the wavelet (scaling) filter, after which we then decimate the filter outputs by a factor two. In the above equations, this decimation is accomplished by use of '$2t$' in the indices to the input scaling coefficients.

3. The maximum level J that we can use is given by $\lfloor \log_2(N) \rfloor$. When J is set to this level, there will be exactly one wavelet and one scaling coefficient ($d_{J,0}$ and $s_{J,0}$). If the sample size N is equal to 2^J, then $s_{J,0}$ is proportional to the sample mean of the time series (for details, see Exercise [97] of Percival and Walden, 2000).

4. To maintain backward compatibility with version 1.0 of S+Wavelets, we are using $d_{j,t}$ and $s_{j,t}$ to denote jth level wavelet and scaling coeffients. This notation differs from what is used in Percival and Walden (2000), where these coefficients are denoted by $W_{j,t}$ and $V_{j,t}$.

3.1.3 The DWT in the Frequency Domain

While the pyramid algorithm allows us to compute the DWT efficiently, we could in principle also obtain the wavelet coefficients for a given level j by using a single filter, which we call the jth level wavelet filter; likewise, we can obtain the jth level scaling coefficients by using the jth level scaling filter. Examination of the frequency domain properties of these jth level filters leads to an interpretation of the DWT as an 'octave band' filter bank.

The DWT can be calculated by processing the data using a *filter bank*, where the wavelet and scaling filters are used in a recursive (pyramid) algorithm to decompose the time series. To generate the first level transform coefficients, the original data is filtered by convolving it with both the wavelet and scaling filters (§2). The results are then decimated by a factor of two, where every other point is thrown out, and the remaining coefficients are defined as the level-one wavelet and scaling coefficients. For the second level, the same filtering-decimation scheme is applied (only) to the *scaling* coefficients. This algorithm can be repeated for higher decomposition levels until the number of wavelet and scaling coefficients is unity (and cannot be broken down any further).

We can replicate this approach by processing the original data with a set of scaled (amplitude reduced) and dilated (time stretched) versions of the wavelet and scaling filters, one filter set per level of the transform. The advantage in this approach is that the gain functions (frequency responses) of the resulting filter sets clearly show the bandpass nature of the filter bank. To illustrate this point, use the wavGain function to obtain the frequency response of the level j Daubechies 8-tap least asymmetric filters wavelet filter ($H_j(f)$) and scaling filter ($G_j(f)$) for $j = 1,\ldots,5$:

```
> gain <- wavGain( "s8", n.level = 5 )
> gain

Gain functions for s8 filters

Number of levels: 5
Number of Fourier frequencies: 1024
Filters normalized: TRUE
```

The result is an object of class WaveletGain. You can plot the *squared* gain functions by invoking the plot method:

```
> plot( gain )
```

Figure 3.4 shows the result. The vertical lines in these plots define the octaves over which the wavelet filters are associated: the wavelet filter for level j is associated with an approximate bandpass filter whose passband frequencies are $f \in [1/2^{j+1}, 1/2^j]$. Similarly, the scaling coefficients filter at level j is associated with an approximate bandpass filter whose passband frequencies are $f \in [0, 1/2^{j+1}]$.

Using these scaled and dilated filter sets, we can calculate the DWT wavelet coefficients at level j via

$$\mathbf{d_j} = \downarrow 2^j(\mathcal{F}^{-1}(H_j(f)X(f)))$$

where $\downarrow K(\cdot)$ is the downsample operator (every K^{th} point is kept), $\mathcal{F}^{-1}(\cdot)$ is the inverse discrete Fourier transform operator, $X(f)$ is the frequency response of the the original time series $\{X_t\}_{t=0}^{N-1}$, and $f \equiv k/N$ for $k = 0, \ldots, N-1$. Similarly, the DWT scaling coefficients at level j can be calculated using

$$\mathbf{s_j} = \downarrow 2^j(\mathcal{F}^{-1}(G_j(f)X(f))).$$

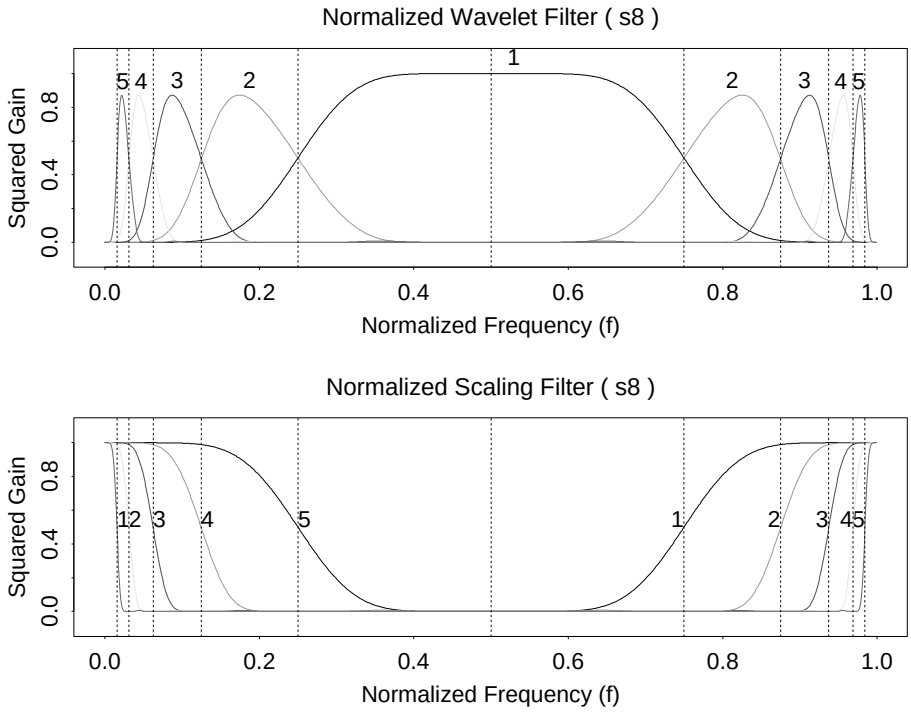

FIGURE 3.4: Squared gain functions of Daubechies least asymmetric 8-tap filters for levels $j = 1, \ldots, 5$.

3.1.4 Interpretation of DWT Coefficients

As noted in §2, the wavelet coefficients are proportional to differences between local averages of the data taken on a scale proportional to the *effective* filter width. The wavelet (and scaling) filter is scaled at each level j of

the wavelet transform such that its effective scale is

$$\tau_j \equiv 2^{j-1}\Delta_t$$

where Δ_t is the sampling interval of X_t (assumed from this point forward to be unity unless specified otherwise). Similarly, the scaling coefficients at level j are proportional to the local average of the data taken at scale τ_j. Using these dual interpretations, the wavelet transform allows us to explore the variability at different scales of the data, a sort of variably-sized mathematical microscope.

Each wavelet and scaling coefficient is typically also localized in time; i.e., except at the very largest scales, each coefficient is formed using a limited portion of the time series. By studying the phase functions that are associated with the jth level equivalent filters $\{h_{j,l}\}$ and $\{g_{j,l}\}$, it is possible to associate a time with each coefficient. This association makes most sense when filters have what is known in the engineering literature as linear phase. Within the class of filters that yield an orthonornal DWT, it is not possible to have filters with exact linear phase, but, by construction, the symmlet and coiflet filters are approximately linear phase filters. If approximate linear phase filters are used in the DWT, then the transform coefficients can be shifted (circularly permuted) so that they align with events in the original time series; i.e., they are shifted for approximate zero phase. To do so, you can use the wavShift function as in the following example:

```
> wavShift( dwt.linchirp )

Discrete Wavelet Transform of linchirp
Wavelet: s8
Length of series: 1024
Number of levels: 4
Boundary correction rule: periodic
Filtering technique: convolution
Shifted for approximate zero phase: TRUE
Crystals: d1 d2 d3 d4 s4
```

The function wavShift returns an updated version of dwt.linchirp containing cystals that have been shifted so that they can be matched up in time with events in the original time series. This fact is reflected in the next to last line above, where it is noted that the crystals have been shifted to achieve approximate zero phase. To see the shifts, you can use the plot method on the object created by wavShift:

```
> plot( wavShift(dwt.linchirp) )
```

Figure 3.5 shows the result. The shift used for each crystal appears next to each crystal name. A negative shift implies an advance (or circular shift to the left) of the data.

If approximate linear phase filters are used in the DWT, then the transform coefficients can be shifted (circularly permuted) so that they align with events in the original time series, i.e. they are shifted for approximate zero phase. The shift factors for Daubechies coiflet or least asymmetric (symmlet) families may be calculated via the wavZeroPhase function.

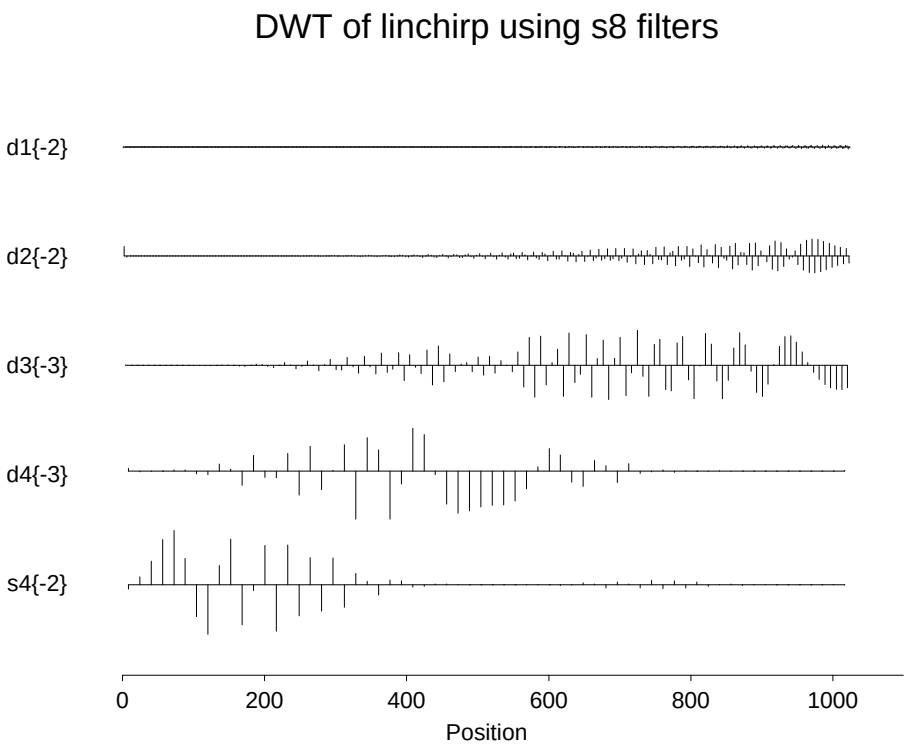

FIGURE 3.5: The DWT of the linchirp sequence circularly shifted for approximate zero phase.

```
> wavZeroPhase( wavelet = "s8", levels = 1:4 )
$dwt:
 d1 d2 d3 d4 s1 s2 s3 s4
 -2 -2 -3 -3 -1 -2 -2 -2

$modwt:
 d1  d2  d3  d4 s1 s2  s3  s4
 -4 -11 -25 -53 -3 -9 -21 -45

$dwpt:
 w1.0 w1.1 w2.0 w2.1 w2.2 w2.3 w3.0 w3.1 w3.2 w3.3 w3.4
   -1   -2   -2   -2   -3   -2   -2   -3   -3   -2   -3

 w3.5 w3.6 w3.7 w4.0 w4.1 w4.2 w4.3 w4.4 w4.5 w4.6 w4.7
   -3   -3   -2   -2   -3   -3   -3   -3   -3   -3   -2

 w4.8 w4.9 w4.10 w4.11 w4.12 w4.13 w4.14 w4.15
   -3   -3    -3    -3    -3    -3    -3    -2

$modwpt:
 w1.0 w1.1 w2.0 w2.1 w2.2 w2.3 w3.0 w3.1 w3.2 w3.3 w3.4
   -3   -4   -9  -11  -12  -10  -21  -25  -27  -23  -24

 w3.5 w3.6 w3.7 w4.0 w4.1 w4.2 w4.3 w4.4 w4.5 w4.6 w4.7
  -28  -26  -22  -45  -53  -57  -49  -51  -59  -55  -47

 w4.8 w4.9 w4.10 w4.11 w4.12 w4.13 w4.14 w4.15
  -48  -56   -60   -52   -50   -58   -54   -46
```

The wavZeroPhase function returns the shifts for the DWT as well as other transforms. You could use these shift factors to manually rotate each crystal appropriately, but a more convenient means is through the wavShift function.

```
> wavShift( dwt.linchirp )

Discrete Wavelet Transform of linchirp
Wavelet: s8
Length of series: 1024
Number of levels: 4
Boundary correction rule: periodic
Filtering technique: convolution
Shifted for approximate zero phase: TRUE
Crystals: d1 d2 d3 d4 s4
```

Notice that the display method now indicates that the crystals have been shifted for zero phase. To see the shifts, use the plot method as usual:

```
> plot( wavShift(dwt.linchirp) )
```

Figure 3.5 shows the result. The shift used for each crystal appears next to each crystal name. A negative shift implies an advance (or circular shift to the left) of the data.

3.1.5 The WaveletTransform Class

The wavDWT function calculates the DWT of a real-valued uniformly-sampled time series and returns an object of class WaveletTransform. Several methods are available to the user to view, summarize, and access the data contained in a WaveletTransform object:

Summary Operator Methods:

print	Prints useful information regarding the wavelet transform including: • Type of wavelet transform. • Information regarding the filter set. • Number of decomposition levels. • Boundary extension rule. • Filtering technique (convolution or correlation). • Whether or not the transform coefficients have been shifted for zero phase (only suitable for coiflet and symmlet filters). • The crystal names.
plot	Plots the transform coefficients. If the transform is shifted for zero phase, the shift factor appears adjacent to the crystal name. Use the string "energy" as a value for the optional type argument to display an energy distribution plot of the crystals.
eda.plot	Extendend data analysis plot displaying the transform, crystal energy distribution, and coefficient distribution
summary	Displays a statistical summary of the transform data.

Transform Operator Methods

mrd	Perform a multiresolution decomposition of the data.
mra	Perform a multiresolution analysis of the data.
reconstruct	Reconstructs the time series via an inverse transform.
wavShift	Circularly shift each crystal to achieve approximate zero phase (only suitable for coiflet and symmlet filters).
wavDetail	Calculate the detail coefficients for specified crystals.

Data Access Methods

[	Access a subset of all crystals contained in a a WaveletTransform object. For example, to obtain the first and second level wavelet crystals of a MODWT or DWT object X, use either X[1:2] or X[c("d1","d2")]. While the former is more compact, the latter is preferred because of it leaves no doubt as to which crystals are to be extracted and does not rely on any particular ordering of the crystals in the object.
[[	Access an individual crystal of a WaveletTransform object. For example, to obtain the second level wavelet crystal of a MODWT or DWT object X, use either X[[2]] or X[["d2"]]. The result is a vector of transform coefficients. If however multiple crystals are requested (ala X[[c("d1","d2")]] for example), the result is an S-PLUS list containing the requested crystals.
$	Use to access specific components of the WaveletTransform object. A list of accessible components can be generated using the names function. One such component is data which contains an S-PLUS list of all wavelet transform crystals. This gives the user yet another way to access specific transform crystal(s). For example, if X is an object of class WaveletTransform, the d2 crystal can be accessed via X$data$d2.

3.2 The Maximal Overlap Discrete Wavelet Transform

Despite its popularity, the DWT has two practical limitations. The first is the dyadic length requirement. While the DWT can be adapted to accommodate arbitrary length sequences via, e.g., polynomial extensions of the scaling coefficients, selecting an appropriate number of end points to fit or the order of fit is not a trivial task. Other techniques can be used, such as the odd-scaling coefficient storage system used for the wavDWT function, but generally involve either complicated bookkeeping or are too simple to accurately portray the dynamics of the scaling coefficients. The second limitation is a sensitivity of the DWT to where we start recording a time series; i.e., the decimation operation makes the DWT a non shift-invariant transform so that circularly shifting the time series can alter the entire DWT.

To overcome these limitations, we can use a non-decimated form of the DWT, known as the maximal overlap DWT (MODWT), that has two main advantages: (1) it handles arbitrary length sequences inherently and (2) circularly shifting the time series will result in an equivalent circular shift of the MODWT coefficients. Additionally, the number of coefficients in each scale is equal to the number of points in the original time series. This refined slicing of the data in combination with the approximate zero phase property of the least asymmetric filters allows us to calculate 'instantaneous' statistical measures of the data across scales (e.g. instantaneous fractionally differenced process model parameters, see §5 for details).

3.2.1 Definition

Let $X_t, t = 0, 1, \ldots, N-1$, represent a real-valued uniformly-sampled time series, and let h_l and $g_l, l = 0, 1, \ldots, L-1$, be the coefficients for, respectively, the wavelet and scaling filters of width L used by the DWT. To preserve

energy in the MODWT, we need to use rescaled versions of these filters, which we define as follows:

$$\tilde{h}_l \equiv h_l/\sqrt{2} \text{ and } \tilde{g}_l \equiv g_l/\sqrt{2}.$$

In order to define the MODWT for a given decomposition level j, we start by defining the MODWT scaling coefficient $\tilde{s}_{0,t}$ at level $j = 0$ and time index t to be equal to the value of the time series at index t; i.e., $\tilde{s}_{0,t} = X_t$, $t = 0, 1, \ldots, N-1$. The MODWT wavelet coefficients $\tilde{d}_{j,t}$ and scaling coefficients $\tilde{s}_{j,t}$ for level j are defined recursively in terms of the scaling coefficients for level $j-1$ as follows:

$$\begin{aligned} \tilde{d}_{j,t} &\equiv \sum_{l=0}^{L-1} \tilde{h}_l \tilde{s}_{j-1,t-2^{j-1}\, l \bmod N} \\ \tilde{s}_{j,t} &\equiv \sum_{l=0}^{L-1} \tilde{g}_l \tilde{s}_{j-1,t-2^{j-1}\, l \bmod N}, \end{aligned}$$

where $t = 0, 1, \ldots, N-1$.

While the above is an efficient way of computing the MODWT wavelet and scaling coefficients, it is also possible to obtain them directly from the time series via the equations

$$\begin{aligned} \tilde{d}_{j,t} &\equiv \sum_{l=0}^{L_j-1} \tilde{h}_{j,l} X_{t-l \bmod N} \\ \tilde{s}_{j,t} &\equiv \sum_{l=0}^{L_j-1} \tilde{g}_{j,l} X_{t-l \bmod N}, \end{aligned}$$

where $\tilde{h}_{j,l}$ and $\tilde{g}_{j,l}$, $l = 0, 1, \ldots, L_j - 1$, are called the level j equivalent MODWT wavelet and scaling filters. Both of these filters have a width given by $L_j = (2^j - 1)(L-1) + 1$. When $j = 1$, we have $L_1 = L$, $\tilde{h}_{1,l} = \tilde{h}_l$ and $\tilde{g}_{1,l} = \tilde{g}_l$. These equivalent filters are mainly of interest for theoretical developments and can be formulated using just the basic filters $\tilde{h}_l$ and $\tilde{g}_l$ (for details, see Section 5.4 of Percival and Walden, 2000).

3.2.2 An Example: The MODWT of a Linear Chirp

To calculate a MODWT, use the wavMODWT function which, like the wavDWT function, returns an object of class WaveletTransform. Here we calculate a level 4 MODWT of the linchirp sequence using Daubechies 12-tap Coiflet filters:

```
>  modwt.chirp <- wavMODWT( linchirp, wavelet = "c12",
+ n.levels = 4 )
> modwt.chirp

Maximal Overlap Discrete Wavelet Transform of linchirp
Wavelet: c12
Length of series: 1024
Number of levels: 4
Boundary correction rule: periodic
Filtering technique: convolution
Shifted for approximate zero phase: FALSE
Crystals: d1 d2 d3 d4 s4

> plot( wavShift(modwt.chirp) )
```

Figure 3.6 shows the result shifted for zero phase. A comparison of Fig. 3.5 and 3.6 shows that the (non-decimated) MODWT is visually smoother than the (decimated) DWT. Unlike the DWT, the MODWT is

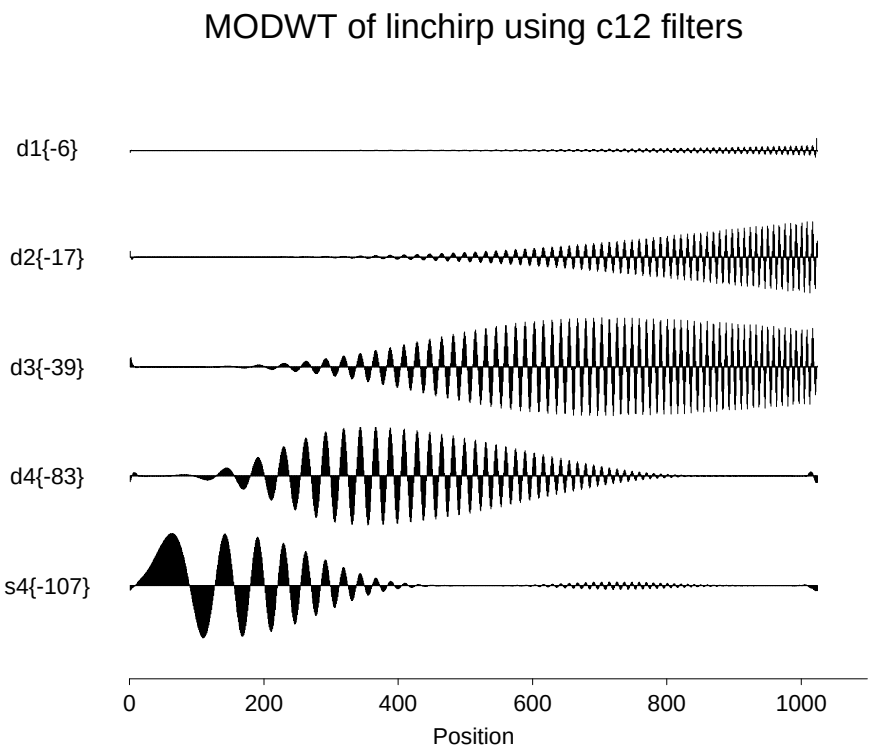

FIGURE 3.6: Maximal overlap discrete wavelet transform of a linear chirp shifted for zero phase.

non-orthogonal and highly redundant. Like the DWT, the MODWT is energy preserving. We can demonstrate this by taking the difference between the energy of the original time series and the MODWT:

```
> sum( linchirpˆ2 ) - sum( unlist(modwt.chirp$data)ˆ2 )
[1] -1.278977e-09
```

We can also test the reconstruct function in a similar manner:

```
> vecnorm( reconstruct(modwt.chirp) - linchirp )
[1] 2.439545e-10
```

The difference in reconstruction is very small and is due to roundoff error in the computation. We can use the reconstruction method for more interesting experiments such as synthesizing the MODWT after zeroing out the d2 and d4 crystals:

```
> modwt.chirp[c("d2","d4")] <- 0
> recon <- reconstruct( modwt.chirp )
> example <- list( linchirp=linchirp, synth=recon,
+ diff=linchirp-recon )
> stack.plot( example )
```

The effect is to remove the high-frequency and mid-frequency components of the chirp (Fig. 3.7).

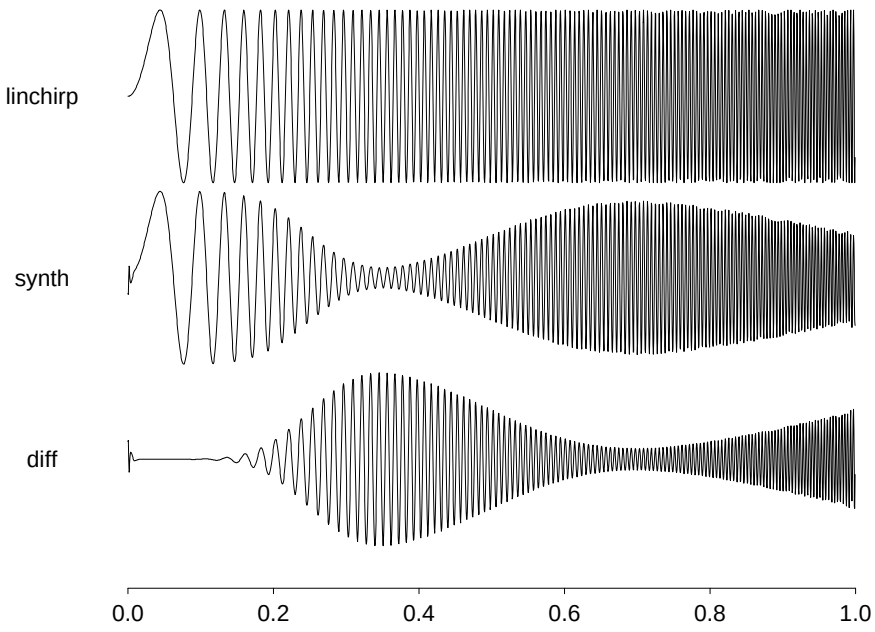

FIGURE 3.7: A synthesis example. The top plot is the original linchirp sequence. The middle plot is the reconstruction of the MODWT of the linchirp sequence with the d1 and d3 crystals zeroed out. The final plot is the difference between the synthesis and the original sequence.

3.3 The Discrete Wavelet Packet Transform

3.3.1 Overview

As we discussed in §3.1.2, the wavelet and scaling filters are used to create the DWT. This transform essentially takes a time series consisting of N values and expresses it in the time-scale domain in terms of N DWT coefficients. There are two types of DWT coefficients, namely, wavelet coefficients (encapsulating information about changes in weighted averages at different scales) and scaling coefficients (related to weighted averages). These filters are used in tandem in the pyramid algorithm. Thus, letting $j = 1, \ldots, J$ and starting with the definition that the zeroth level scaling coefficients are given by the time series itself, we take the level $j-1$ scaling coefficients and transforms them into wavelet and scaling coefficients of level j using, respectively, the wavelet and scaling filter. After J iterations of the pyramid algorithm, the DWT consists of $J+1$ crystals (components), namely, the wavelet coefficients from levels $j = 1, \ldots, J$ and the scaling coefficients from level J. In this scheme, once the wavelet coefficients are created, we leave them alone and do not subject them to any further transformations. If, however, we decide to treat the level j wavelet coefficients in a manner analogous to scaling coefficients, we can obtain what is know as a discrete wavelet packet transform (DWPT). A level j DWPT consists of 2^j crystals, each of which contains $N/2^j$ values, where N is the sample size of the time series (for the purposes of this discussion, we assume that N is a power of two, but in fact the DWPT in S+Wavelets can handle arbitrary sample sizes).

The algorithm that dictates how we go from level j to level $j+1$ says that we take each crystal and subject its contents to both a wavelet filter and a scaling filter, yielding two new crystals that form part of the level $j+1$ DWPT. We can illustrate the DWPT algorithm with the help of Fig. 3.8, which shows the creation of DWPTs of levels 1, 2 and 3. Each box of Fig. 3.8 represents a crystal in the wavelet packet tree denoted by wj.n where $0 \leq j \leq J$ is the decomposition level and $0 \leq n < 2^j$ is the oscillation index. By definition, the original

FIGURE 3.8: Wavelet packet table with 3 decomposition levels

time series is stored in crystal w0.0. The DWPT begins by filtering w0.0 with both a wavelet and scaling filter, followed by decimation process where every other coefficient is dropped (decimation by 2) yielding crystals w1.0 and w1.1. Each of these crystals contains $N/2$ coefficients and accordingly the level 1 DWPT is length preserving. The level 2 DWPT is formed in the same manner, where each level 1 crystal is filtered and decimated to form the crystals w2.0 through w2.3. Each of these crystals contains $N/4$ coefficients and again we find that the collection of level 2 DWPT coefficients is length preserving. The algorithm for higher levels continues in the same way, where each *parent* crystal is filtered and subsequently decimated to form two *children* crystals. Each level of the DWPT is length preserving and (due to the decimation by 2 process at each level) we are restricted to levels $0 \leq j \leq J$ where $J = \log_2 N$.

Finally, as a comparison, note the shaded portions of Fig. 3.8 which collectively denote the DWT. Thus, the DWT is actually a subset of the DWPT and represents one of many possible orthonormal (disjoint and dyadic) transforms of the orginal time series.

3.3.2 The DWPT in the time domain

Given that j,n,t are the decomposition level, oscillation index, and time index, respectively, the DWPT is given by

$$W_{j,n,t} = \sum_{l=0}^{L-1} u_{n,l} W_{j-1,\lfloor n/2 \rfloor, 2t+1-l \bmod N_{j-1}}, \qquad t = 0,\ldots,N_j - 1.$$

where $N_j \equiv N/2^j$ and $\lfloor \cdot \rfloor$ denotes the integer part. The variable L is the length of the filters defined by

$$u_{n,l} \equiv \begin{cases} g_l, & \text{if } n \bmod 4 = 0 \text{ or } 3; \\ h_l, & \text{if } n \bmod 4 = 1 \text{ or } 2, \end{cases}$$

where g_l and h_l are the scaling filter and wavelet filter, respectively. By definition, $W_{0,0,t} \equiv X_t$ where $\{X_t\}$ is the original time series.

3.3.3 The DWPT in the Frequency Domain

The filtering and subsequent decimation of each crystal in the DWPT evenly divides the frequency content of a parent crystal amongst its children. For example, if we assume that the original sequence denoted by w0.0 has a normalized frequency bandwidth of $1/2$, then its children w1.0 and w1.1 will inherit respectively the normalized frequency bands of $[0,1/4]$ and $[1/4,1/2]$. In general, the coefficients of a given crystal wj.n correspond to a time-varying decomposition of the (normalized) frequency interval $[n/2^{j+1},(n+1)/2^{j+1}]$. For example, crystal w3.2 corresponds to the frequency interval $[1/8,3/16]$.

To illustrate the DWPT, let us return to the linear chirp we created and stored in linchirp (Fig. 3.1).

```
> linchirp <- make.signal("linchirp", n=1024)
> dwpt.chirp <- wavDWPT( linchirp, n.levels = 3 )
> dwpt.chirp

Discrete Wavelet Packet Transform of linchirp
Wavelet: s8
Length of series: 1024
Number of levels: 3
Boundary correction rule: periodic
Filtering technique: convolution
Shifted for approximate zero phase: FALSE
Crystals:

 w0.0 w1.0 w1.1
 w2.0 w2.1 w2.2
 w2.3 w3.0 w3.1
 w3.2 w3.3 w3.4
 ... (15 bases)

> plot(dwpt.chirp)
```

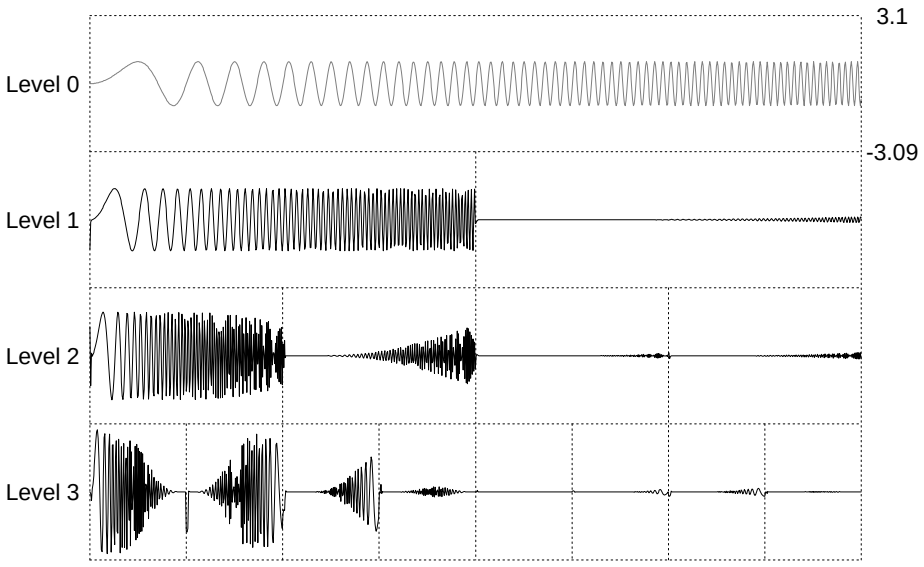

FIGURE 3.9: Discrete wavelet packet transform of a linear chirp.

The division of the frequency content into octaves is clearly visible in the DWPT of the linear chirp, where each crystal represents a band-pass filering of the original sequence.

3.4 The Maximal Overlap Wavelet Packet Transform

The MODWT discussed in §3.2 can be easily extended to the *maximal overlap discrete wavelet packet transform* (MODWPT). The MODWPT is seen as a collection of non-orthogonal transforms. Each level of the transform is associated with a level j, where the j^{th} level MODWPT decomposes the frequency interval $[0, 1/2]$ into 2^j uniform divisions. Similar to the Short-Time Fourier Transform, the MODWPT is seen as a time-frequency decomposition because it decomposes a time series into crystals such that each crystal is associated with a particular range of frequencies and each coefficient of a crystal corresponds to to a particular time.

3.4.1 The MODWPT in the Time Domain

Given j, n, t are the decomposition level, oscillation index, and time index, respectively, the MODWPT is given by

$$\widetilde{W}_{j,n,t} \equiv \sum_{l=0}^{L-1} \widetilde{u}_{n,l} \widetilde{W}_{j-1,\lfloor n/2 \rfloor, t-2^{j-1} \, l \bmod N}, \qquad t = 0, \ldots, N-1.$$

The variable L is the length of the filters defined by

$$\widetilde{u}_{n,l} \equiv \begin{cases} \widetilde{g}_l/\sqrt{2}, & \text{if } n \bmod 4 = 0 \text{ or } 3; \\ \widetilde{h}_l/\sqrt{2}, & \text{if } n \bmod 4 = 1 \text{ or } 2, \end{cases}$$

where g_l and h_l are the scaling filter and wavelet filter, respectively. By definition, $\widetilde{W}_{0,0,t} \equiv X_t$ where $\{X_t\}$ is the original time series.

3.4.2 A Comparison of the MODWPT and the DWPT

The interpretation of the MODWPT in the frequency domain is the same as that for the DWPT, namely that the frequency content of a parent crystal is evenly divided amongst the children. The main differences between the DWPT and the MODWPT are that (i) there is no decimation operation performed after filtering a given MODWPT crystal, and as a result (ii) there is no theoretical limit on the number of decomposition levels one can perform using the MODWPT (although in practice one typically limits the MODWPT to $J = \log_2 N$ since the transform coefficients for levels beyond J do not provide further insight into the variations over scales greater than the extent of the time series, see, e.g., page 199–200 in [PW00]). Although the high level of redundancy in the MODWPT is a computational detrement, there are certain advantages that the MODWPT has over the DWPT including:

1. The MODWPT can be used to form a multiresolution analysis (MRA) that is associated with zero phase filters, thus ensuring that the feautres of the original time series align with those in the MRA details and smooths.

2. The MODWPT MRA is shift-invariant so that a circular shift of the original time series will result in the same shift of the MRA components.

3. The MODWPT MRA is less affected by the choice of filters used in the original decomposition.

See §3.6 for a discussion on MRAs in S+Wavelets and see [PW00] for a more in-depth discussion of the differences between the MODWPT and the DWPT.

For purposes of comparison with the DWPT, consider the MODWPT of the linear chirp series linchirp (Fig. 3.1).

```
> modwpt.chirp <- wavMODWPT( linchirp, wavelet = "s8",
+ n.levels = 3 )
> modwpt.chirp

Maximal Overlap Discrete Wavelet Packet Transform of linchirp
Wavelet: s8
Length of series: 1024
Number of levels: 3
Boundary correction rule: periodic
Filtering technique: convolution
Shifted for approximate zero phase: FALSE
Crystals:

 w0.0 w1.0 w1.1
 w2.0 w2.1 w2.2
 w2.3 w3.0 w3.1
 w3.2 w3.3 w3.4
 ... (15  bases)

> plot( modwpt.chirp )
```

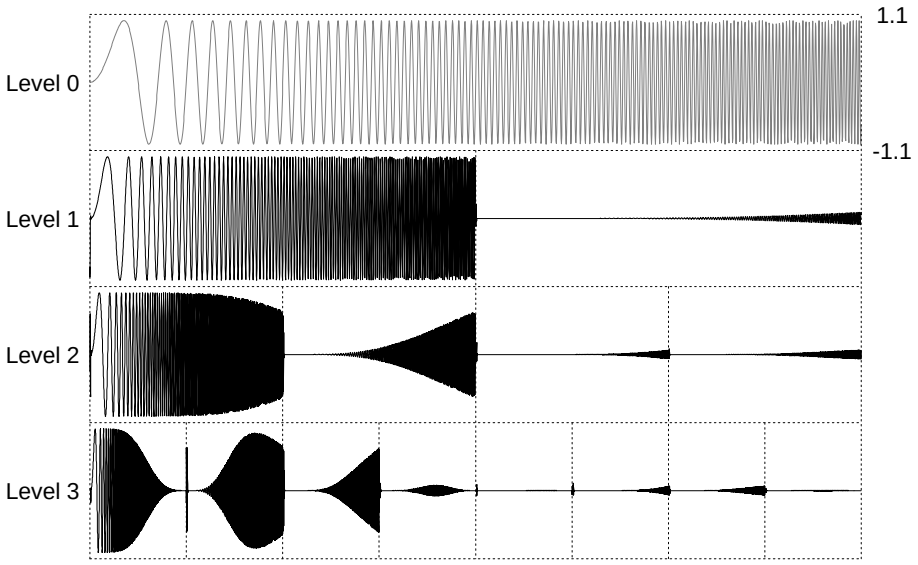

FIGURE 3.10: Maximal overlap discrete wavelet packet transform of a linear chirp.

Figure 3.10 shows the MODWPT of the linear chirp out to three levels. A comparison of the MODWPT (Fig. 3.10) and the DWPT (Fig. 3.9) shows the redundant nature of the MODWPT with each crystal containing N coefficients for an N-point time series.

3.5 The WaveletPacket Class

The wavDWPT and wavMODWPT functions calculate the DWPT and MODWPT, respectively, of a real-valued uniformly-sampled time series and returns an object of class WaveletPacket. Methods are available to the user to view and summarize the data contained in a WaveletPacket object:

Summary Operator Methods:

display Prints useful information regarding the wavelet packet transform including:

- Type of wavelet packet transform.
- Information regarding the filter set.
- Number of decomposition levels.
- Boundary extension rule.
- Filtering technique (convolution or correlation).
- The crystal names.

plot Plots the transform coefficients.

Transform Operator Methods

mrd Perform a multiresolution decomposition of the data.

mra Perform a multiresolution analysis of the data.

reconstruct Reconstructs the time series via an inverse transform.

wavShift Circularly shift each crystal to achieve approximate zero phase (only suitable for coiflet and symmlet filters).

wavDetail Calculate the detail coefficients for specified crystals.

Data Access Methods

[Access a subset of all crystals contained in a a WaveletPacket object. For example, to obtain the first and second level wavelet crystals of a MODWPT or DWPT object X, use X[level = 1:2]. Individual crystals may be accessed as we using the crystal name(s) as an input ala X[c("w1.0","w3.1")] for example. In all cases, the original WaveletPacket object is returned with a subset of the original crystals.

[[Access an individual crystal of a WaveletPacket object. For example, to obtain the node correpsonding to level 2 and oscillation index 3 of a WaveletPacket object X, use X[["w3.2"]]. The result is a vector of transform coefficients. If, however, multiple crystals are requested (ala X[[c("w3.0","w1.1")]] for example), the result is an S-PLUS list containing the requested crystals. In all cases, only the specified transform coefficients are returned, i.e., all other components of the original WaveletPacket object are excluded.

**$** Use to access specific components of the WaveletPacket object. A list of accessible components can be generated using the names function. One such component is data which contains an S-PLUS list of all wavelet transform crystals. This gives the user yet another way to access specific transform crystals. For example, if X is an object of class WaveletPacket, the w3.0 crystal can be accessed via X$data$w3.0.

3.6 Multiresolution Decomposition and Analysis

Multiresolution decomposition (MRD) involves breaking down a time series or image into its fundamental components. Each component contains information related to the variations in the original data at a certain scale. Summing all of the components together will give you the original data. The process of successively summing the data based on its MRD is known as multiresolution analysis (MRA). Summing only a few of the components removes those portions deemed undesirable in the data and can be used as a very effective 'denoising' tool. Consider, for example, the following MRD of an electrocardiogram signal using the MODWT:

```
> modwt.ecg <- wavMODWT( ecg, wavelet = "s8", n.levels = 6 )
> mrd.ecg <- mrd( modwt.ecg )
> mrd.ecg

MODWT Decomposition of ecg
Wavelet: s8
Length of series: 2048
Number of levels: 6
Boundary correction rule: periodic
Filtering technique: convolution
Signal Components: D1 D2 D3 D4 D5 D6 S6

> plot( mrd.ecg )
```

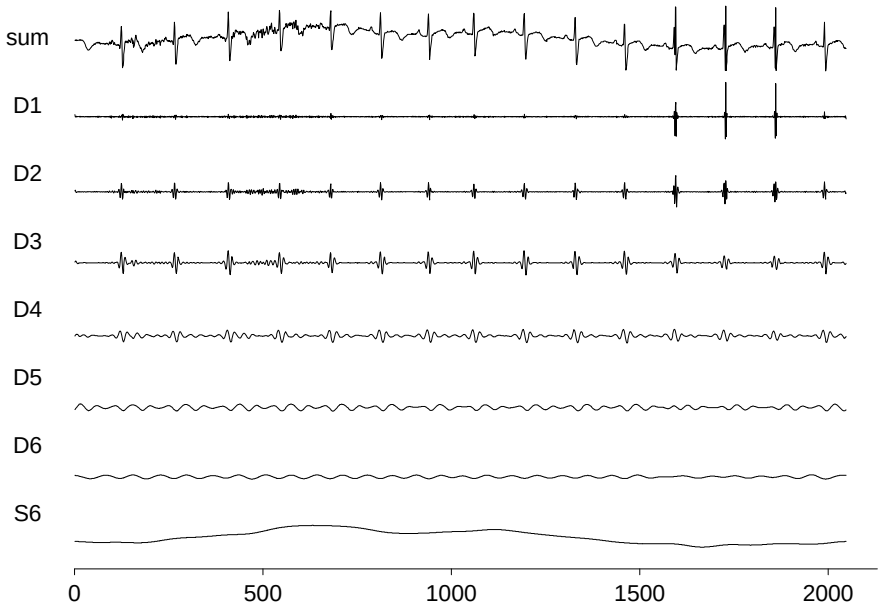

FIGURE 3.11: MODWT multiresolution decomposition of an ECG sequence.

Figure 3.11 shows the resulting MRD of the ecg sequence out to six decomposition levels. We see that the baseline drift (due to respiration) is captured mainly in the S6 crystal while a small amount of noise is seen in the first portion of the D1 crystal. Suppose we wish to get rid of the small scale (high frequency) noise that exists in the D1 component and to exclude the low frequency base-line drift due to respiration captured in component S6. To do so, we simply sum components D2-D6:

```
> nobaseline <- apply( mrd.ecg[2:6,], MARGIN=1, sum )
> comparison <- list( ecg=ecg, detrend=nobaseline )
> stack.plot( comparison )
```

Figure 3.12 shows a comparison of the "denoised" and original ecg sequence. The denoised version is much cleaner than the original. We can also prove that a summation over all crystals of the MRD will give us the

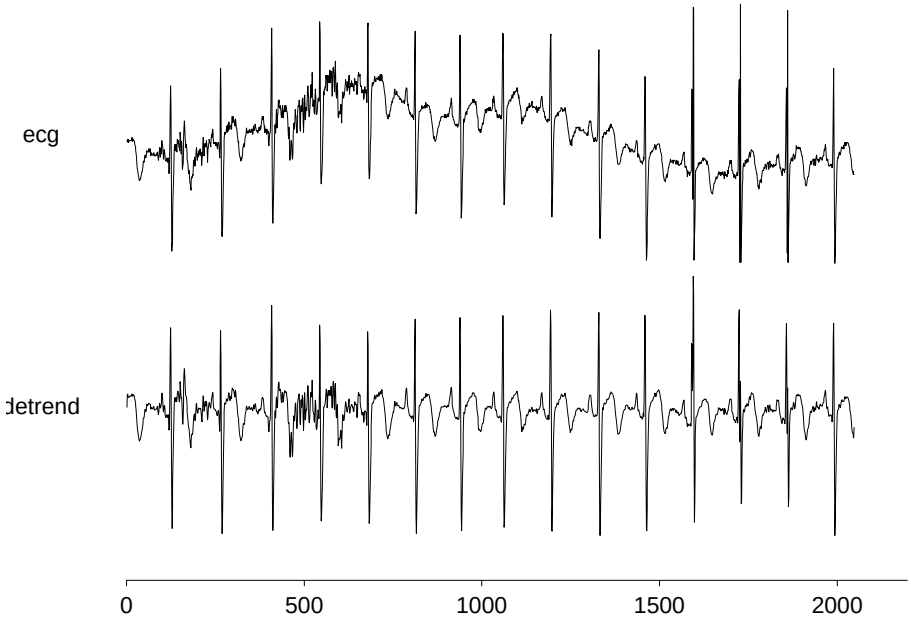

FIGURE 3.12: Denoising an ECG sequence using a MODWT MRD. The top plot shows the ecg sequence. The bottom plot shows the summation of the MRD crystals D2-D6. The effect is to remove the low-frequency baseline drift and high-frequency noise components.

original sequence:

```
> vecnorm( ecg@data - reconstruct(mrd.ecg) )
[1] 3.338254e-11
```

You can also obtain a subset of the components of an MRD through the wavDetail function. The wavDetail function is highly flexible in that it is usable for the DWT, MODWT, DWPT, and MODWPT. To illustrate, let us generate these transforms for the first difference of the atomic clock sequence atomclock:

```
> x <- diff( as.vector(atomclock) )
> x.dwt <- wavDWT( x, n.levels = 3 )
> x.dwpt <- wavDWPT( x, n.levels = 3 )
> x.modwt <- wavMODWT( x, n.levels = 3 )
> x.modwpt <- wavMODWPT( x, n.levels = 3 )
```

The wavDetail function takes two arguments, level and osc, which are used to specify the decomposition level and oscillation index, respectively. A (level, osc) doublet defines the location of a particular node or crystal in a sequency-ordered wavelet packet tree. If you call the wavDetail function without specifying any values for level or osc, then by default an MRD is returned.

```
> wavDetail( x.dwt )

DWT Decomposition of x
Wavelet: s8
Length of series: 1025
Number of levels: 3
Boundary correction rule: periodic
Filtering technique: convolution
Signal Components: D1 D2 D3 S3

> wavDetail( x.dwpt )

DWPT Decomposition of x
Wavelet: s8
Length of series: 1025
Number of levels: 3
Boundary correction rule: periodic
Filtering technique: convolution
Signal Components: W3.0 W3.1 W3.2 W3.3 W3.4 W3.5 W3.6 W3.7
```

Notice that, in the case where the transform is a wavelet packet, an MRD of the nodes found in the last level is returned. In the non-packet transform case (e.g., in the DWT or MODWT) the default action is to return the MRD of the available crystals which span all levels in the transform. The plot method can be used to display the MRD.

```
> plot( wavDetail(x.dwt) )
> plot( wavDetail(x.dwpt) )
```

The MRD for the x.dwt and x.dwpt objects are shown in Fig. 3.13 and 3.14, respectively. If we wish to decompose only those crystals associated with a particular decoposition level, we can do so by specifying the level in the call to wavDetail:

```
> plot( wavDetail(x.dwt, level = 3), show.sum = F )
```

Figure 3.15 shows the level three details for the x.dwt object. You can also form the details for a single node or group of nodes in a particualr level as well by specifying both the level and osc arguments of the wavDetail function.

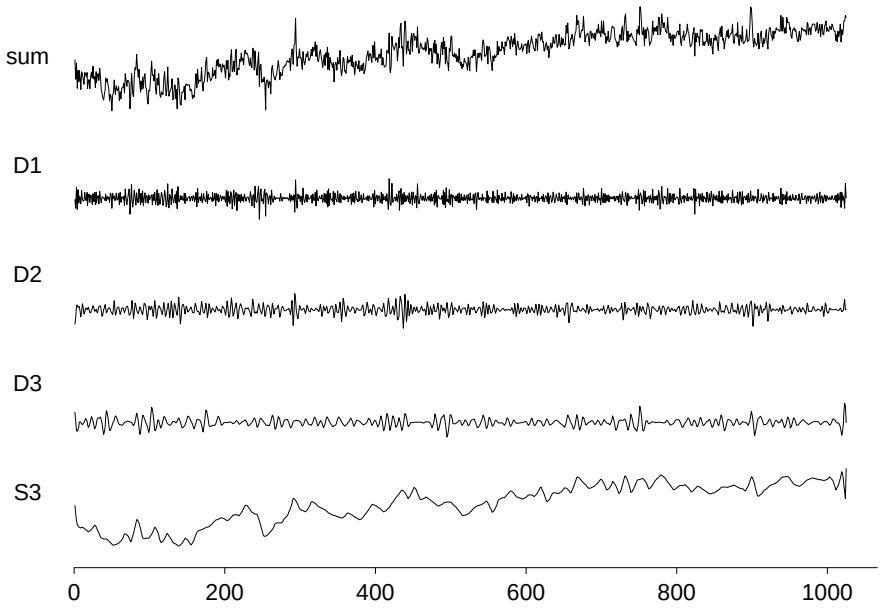

FIGURE 3.13: DWT multiresolution decomposition of the atomclock differences.

```
> wavDetail( x.modwt, level = 2, osc = 1 )

MODWT Decomposition of x
Wavelet: s8
Length of series: 1025
Number of levels: 3
Boundary correction rule: periodic
Filtering technique: convolution
Signal Components: D2

> wavDetail( x.modwpt, level = 2, osc = c(0,3) )

MODWPT Decomposition of x
Wavelet: s8
Length of series: 1025
Number of levels: 3
Boundary correction rule: periodic
Filtering technique: convolution
Signal Components: W2.0 W2.3
```

NOTE: For non-packet transforms such as the DWT or MODWT, the osc argument in the wavDetail function must be set as osc=1 for levels $j < J$ where J is the total number of decomposition levels. For level $j = J$,

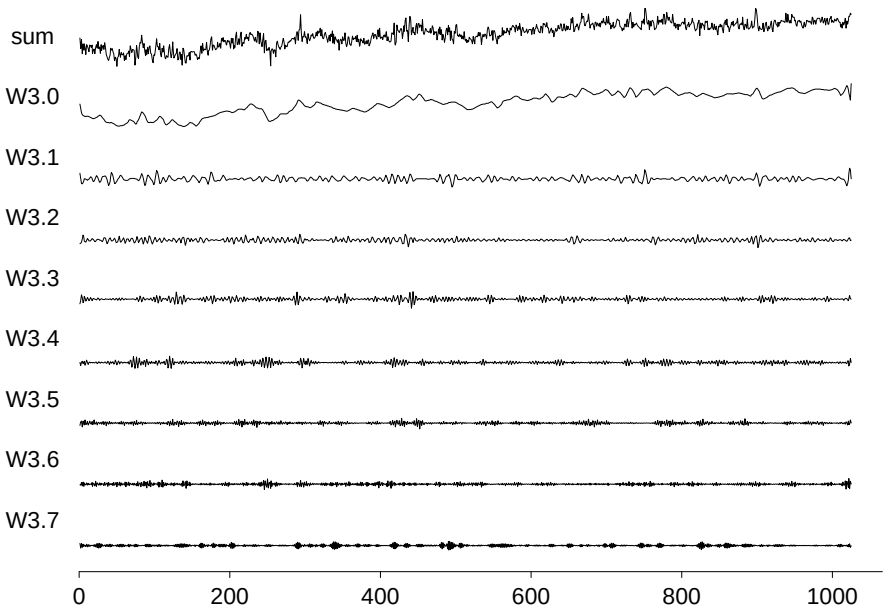

FIGURE 3.14: DWPT multiresolution decomposition of the atomclock differences.

the osc argument must be set such that $0 \leq \text{osc} \leq 1$. This format is consistent with the sequency ordering of wavelet packet trees (see Fig. 3.8 and corresponding text for details).

3.7 The Dual Tree Wavelet Transform

The Dual-Tree Complex Wavelet Transform (DTWT) is a recent development in wavelet theory due to Nick Kingsbury at Cambridge University [Kin98, Kin99, Kin00]. The research that resulted in the DTWT was motivated by the quest for a wavelet transform that would have a combination of desirable properties, some of them shared by classical wavelet transforms, and some of them new. These are:

- **Invertibility.** The ability to "perfectly" reconstruct the data after transformation, i.e., the transform of a function has an inverse and, in the absence of round-off errors, the inverse is identical to the original function. This property is easily demonstrated for the DTWT, for example by transforming a random sequence:

```
> set.seed(0)
> rand <- rnorm( 128 )
> rand.dtwt <- wavDTWT( rand )
> rand.recon <- reconstruct( rand.dtwt )
> vecnorm( rand - rand.recon ) / vecnorm( rand )
[1] 7.319817e-15
```

  While many classical wavelet transforms have this property, the requirements for achieving perfect reconstruction can conflict with those for achieving shift invariance and directional selectivity [Kin99].

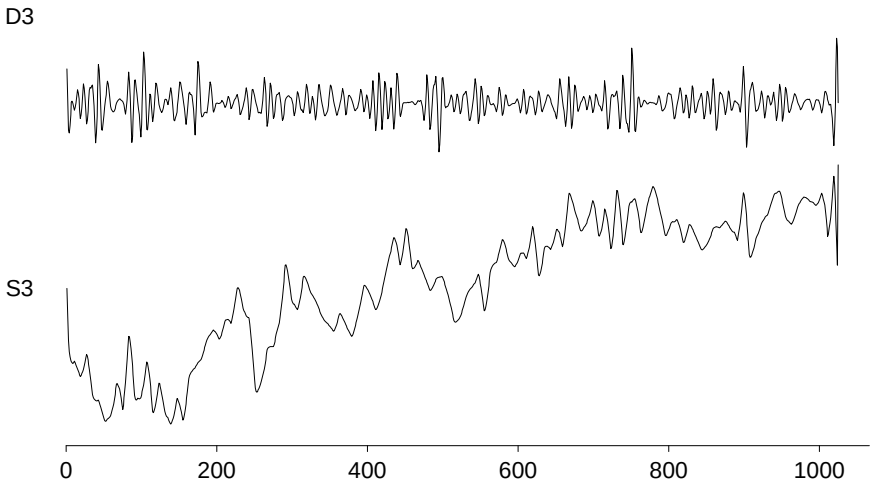

FIGURE 3.15: DWT multiresolution decomposition of the atomclock differences.

- **Shift invariance.** In the ideal case, shift invariance would mean that if the location of a feature (e.g., a line or an edge) in the original function or image were shifted then the wavelet coefficients, and the components of the multiresolution decomposition, would exhibit a similar shift, and there would be no transfer of energy between different levels of the transform. This property would be highly desirable, but it is not found in the classical wavelet transforms. There, shifting a feature often results in considerable shifting of energy between different transform levels. For an example of this behavior, see the following section. Also see §5.1 of [PW00].

  As noted in [BG96], it is possible to achieve shift invariance by use of a non-decimated wavelet transform such as the MODWT. However, in that transform, the wavelet coefficients are evaluated without any decimation. Thus, the MODWT achieves shift invariance only at the expense of a high degree of redundancy, and with a considerable increase in computation and data handling.

- **Directional selectivity.** In a 2D image, if certain features are aligned to the horizontal at angles of $\pm$ 15 deg, $\pm$ 45 deg, or $\pm$ 75 deg, then these alignments also will be evident in the magnitudes of the wavelet coefficients. Ideally, features aligned in the positive (+) directions would be distinct from those aligned at the negative (-) directions. Unfortunately, achievement of this ideal has been elusive, and classical wavelet transforms show an equal emphasis of the + and - directions.

- **Low noise amplification.** Small changes in the wavelet coefficients do not lead to large changes in the reconstructed signal. Many classical wavelet transforms do have this property, but maintaining it places constraints on the achievement of the other properties listed here. Thus, noise amplification limits the usefulness of wavelet transforms based on a single tree of complex filters [dR00].

The DTWT is a recent attempt to create a transform having the features described above. The DTWT retains perfect reconstruction, achieves a good (approximate) degree of shift invariance, has good directional selectivity, and has low noise amplification. These features do come with a modest degree of redundancy (a factor

of two in one dimension, or four in two dimensions), but this is considerably less than the redundancy of the non-decimated wavelet transform.

We note that Kingsbury has presented two versions of the DTWT that differ in the types of filters used. The first version was termed the 'Odd-Even' version [Kin98]. The second, more recent, version is termed the 'Q-shift' version [Kin00]. It is the Q-shift version that is implemented in Wavelets 2.0.

3.7.1 Some Features of the DTWT

There are two principal differences between the DTWT and classical wavelet transforms:

1. Decimation of the transform coefficients is eliminated at level 1, but is retained at all higher levels. The output of the level 1 transform is separated into two streams. The odd-indexed data is passed into one 'tree' of filters that generate the higher levels of the transform, using a different set of filters than used at level 1. The even-indexed data is passed into a second 'tree' of filters that also generate the higher levels of the transform, but with a set of filters whose coefficients are reversed versions of those in the first tree. It is this dual set of filter trees that is referred to by the words 'Dual Tree' in the name of the transform. The overall picture is equivalent to having two completely separate trees, each including the same level 1 filters, but with the second tree's level 1 filters including a unit delay. This viewpoint is illustrated, for a DTWT through level 3, in Fig. 3.16. If additional levels were present, their filters would repeat those

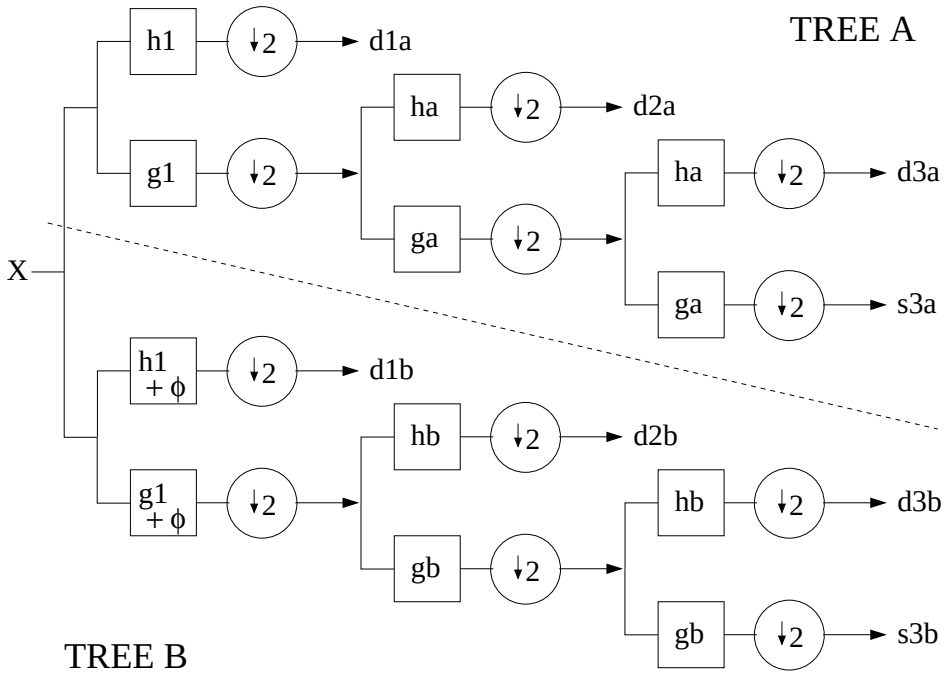

FIGURE 3.16: The three-level Dual-Tree Complex Wavelet Transform as a filter bank.

shown for levels 2 and 3.

2. The filters in the two trees are designed to have the character of 'real' and 'imaginary' parts of an overall complex wavelet transform. (But, we must emphasize, this is accomplished with filters that have purely real coefficients!) Here, the 'complex' nature of the result must be understood in the analytic signal sense of the term. Thus, the mother wavelets associated with each tree are discrete versions of Hilbert transforms of one another [Sel00]. Equivalently, the lowpass filters of one tree interpolate midway between the lowpass filters of the second tree [Kin99]. Again, we emphasize that no complex numbers are in-

volved in the evaluation of the transforms themselves. However, the results obtained from the two trees, *TreeA* and *TreeB*, are interpreted as $(TreeA) + i(TreeB)$

Kingsbury has provided four sets of filters that can be used at level 1, and four sets that can be used at the higher levels. The level 1 filters all are biorthogonal filters, and are termed *nearsyma*, *nearsymb*, *antonini*, and *legall*. The filters for use at the higher levels all are quarter-shift ('Q-shift') filters and are termed *a*, *b*, *c*, and *d*. Specific sets of filter coefficients can be selected via the function wavDTWTFilters, and the impulse responses of the selected filters can be displayed, as in Fig. 3.17.

```
> filts <- wavDTWTFilters( biorthogonal="antonini", qshift="b" )
> plot( filts, type = "time" )
```

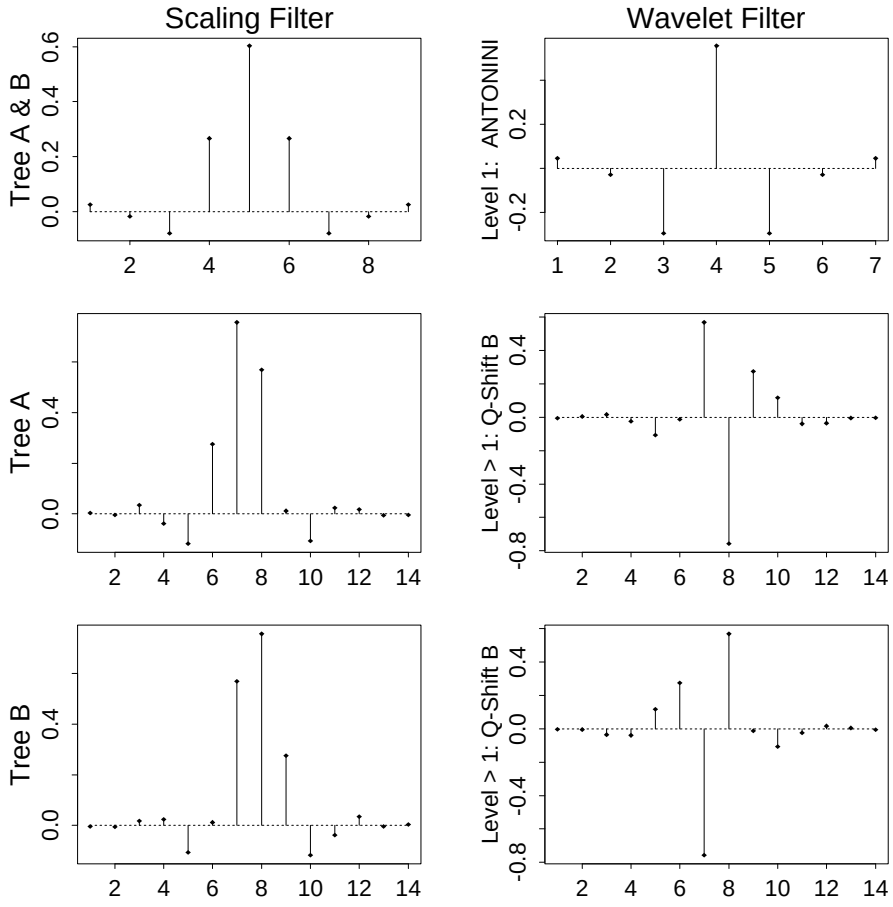

FIGURE 3.17: Impulse responses of DTWT filters. This example has selected the Antonini filters for level 1 and the Q-Shift *b* filters for the higher levels.

Alternatively, the frequency responses of the filters can be displayed, as shown in Fig. 3.18.

```
> plot( filts, type = "gain" )
```

The approximate shift-invariance of the DTWT, and the corresponding lack of shift-invariance of a classical wavelet transform, are illustrated in Fig. 3.19 and 3.20. These figures compare the multi-resolution decompo-

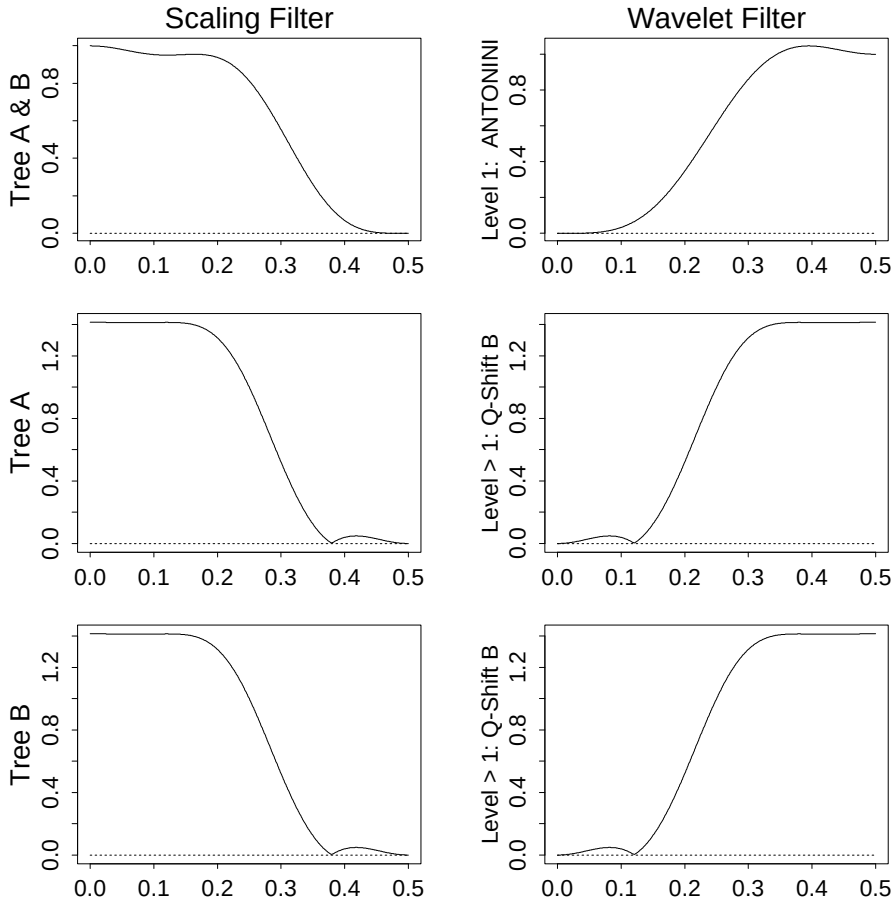

FIGURE 3.18: Frequency Responses of DTWT Filters.

sitions of a sequence of sixteen shifted unit step functions as generated by the DTWT using the nearsyma and Q-shift a wavelets, and by a DWT using the s8 wavelet. The upper plot in each figure shows the step functions. As an aid in visualization of the entire sequence, each successive function has been drawn at a slightly lower elevation. The next five plots in each figure show the components of the level 4 multiresolution decompositions of the step functions, specifically the D1, D2, D3, D4, and S4 components. These too have been drawn with each successive case at a slightly lower elevation. In every instance, the original step function is the sum of its five components. These detailed figures were generated by combining a number of similar S-PLUS commands into a source file. Generation of any single component, however, is straightfoward. For example, if u.step is a step function of length 128 then the D1 component of its multiresolution decomposition is generated by

```
> u.step <- c( rep(0,64), rep(1,64) )
> u.step.dtwt <- wavDTWT( u.step, n.levels=4 )
> u.temp.dtwt <- u.step.dtwt
> u.temp.dtwt[ c("d2", "d3", "d4", "s4") ] <- 0
> D1.step <- reconstruct( u.temp.dtwt )
```

As Fig. 3.19 clearly shows, when the multiresolution decomposition is based on the DTWT, the principal effect of a shift in step location is a corresponding shift of each component, with very little transfer of energy between different levels. Figure 3.20, however, which shows the results for the DWT, provides a sharp contrast. While

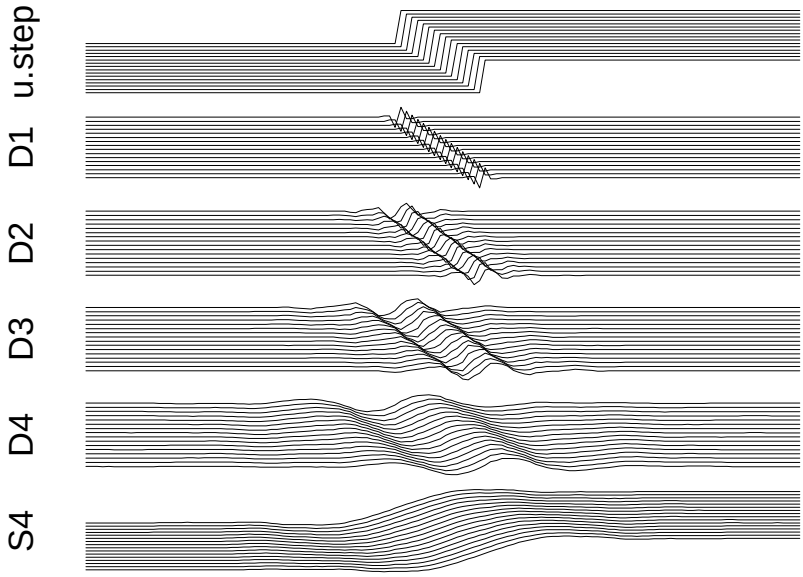

FIGURE 3.19: DTWT multiresolution decompositions of shifted step functions.

each step function still is the sum of its individual components, none of the components exhibit a simple shift behavior. The D1 component appears to oscillate between two different shifted states, while the D2, D3, and D4 components show major transfers of energy between the different levels. The S4 component comes closest to showing a simple shift, but it too has noticeable distortion compared to what is seen in Fig. 3.19. The conclusion is that DTWT achieves a high degree of shift invariance that is not present in the ordinary DWT, and it does so at the expense of only a two-fold degeneracy (in one dimension).

The directional selectivity of the DTWT can be illustrated best by application to an idealized image. To that end, the function make.image can be used to construct a binary image in the form of a twelve-petal flower, as shown in Fig. 3.21.

```
> posy <- make.image( "flower", nrow=256 )
> par( pty="s" )
> image( posy, axes=F )
```

The petal centerlines are oriented at the angles of 15 deg, 45 deg, 75 deg, ...to the horizontal. The two-dimensional DTWT of the flower can be evaluated, and the magnitudes of different combinations of the DTWT coefficients can be displayed as in Fig. 3.22.

```
> posy.dtwt <- wavDTWT.2d( posy, n.levels=3 )
> plot( posy.dtwt )
```

Features in the image that are oriented at different angles have their counterparts in the DTWT coefficients. Different orientations can be emphasized by selecting DTWT coefficients associated with different combinations of lowpass (scaling) and highpass (wavelet) filter outputs in the transforms of the rows and columns. Thus, at any particular level, the row coefficients and, independently, the column coefficients can be associated with

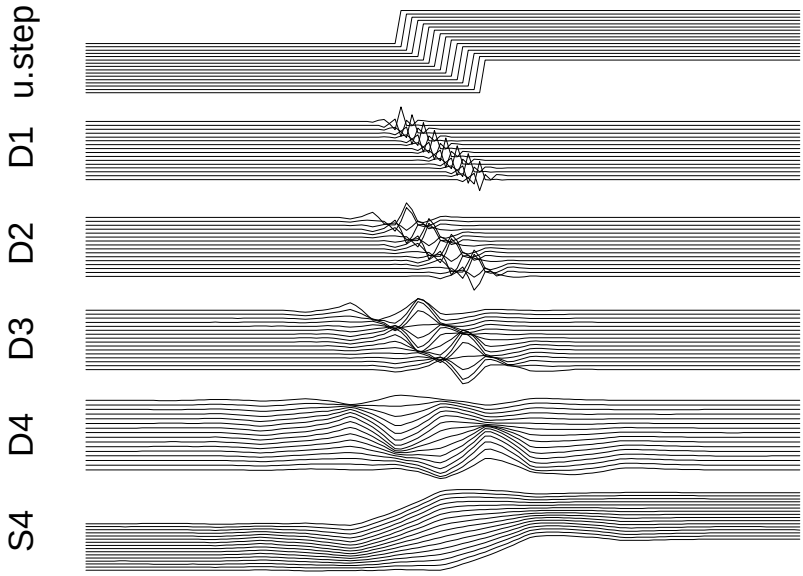

FIGURE 3.20: DWT multiresolution decompositions of shifted step functions.

either the highpass filters or the lowpass filters. Positive or negative angular orientations are emphasized in the DTWT by choosing the 'imaginary' parts of the row and column transforms to be associated with imaginary units 'i' having the same algebraic signs, or the opposite algebraic signs, respectively. In the analytic signal interpretation, this corresponds to the selection of frequency ranges in the row and column transforms that have the same algebraic signs, or the opposite signs, respectively. Of course, we display the complex magnitudes of the wavelet coefficients. This ability to emphasize positive and negative orientations separately is unique to the DTWT and would not be seen in a display based on a real DWT.

An enlarged view of any one, or more, of the six small images in Fig. 3.22 also can be obtained. For example, the $+45$ deg image is shown in Fig. 3.23.

```
> plot( posy.dtwt, level=2, angle=+45 )
```

It is possible to suppress less prominent features in the plot of a DTWT by use of the threshold argument in plot. This argument ranges from 0 to 1, and has the default value 1. For example:

```
> plot( posy.dtwt, level=2, angle=+45, threshold=0.8 )
```

The result is shown in Fig. 3.24, which displays the same DTWT image as in Fig. 3.23 but plotted with threshold = 0.8. The effect of this setting is to plot a zero for any DTWT coefficient whose complex magnitude is less than 0.8 times the maximum amplitude of the DTWT.

While the directional selectivity of the DTWT is most striking for an idealized image, such as the flower, it also is apparent with images from the real world. As an illustration, we consider the classical Lena image shown in Fig. 3.25 as a 512×512 pixel gray-scale image.

```
> image( lena, axes=F )
```

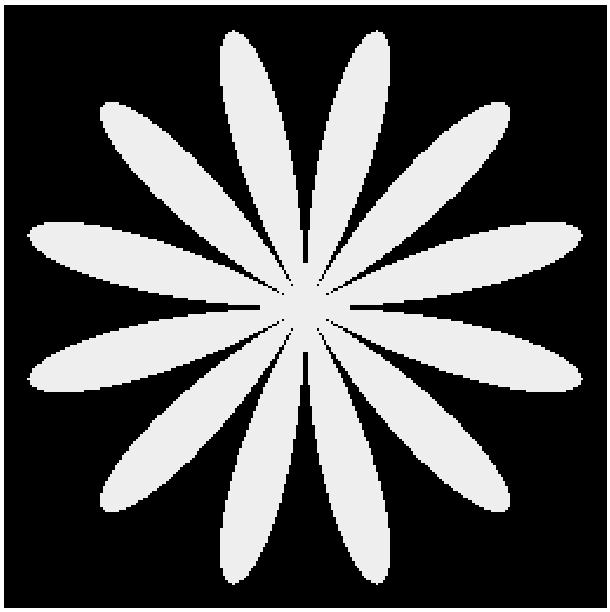

FIGURE 3.21: Binary image of an idealized twelve-petal flower as a 256×256 matrix.

The magnitudes of the DTWT coefficients of the Lena image are shown in Fig. 3.25.

```
> lena.dtwt <- wavDTWT.2d( lena, n.levels=2 )
> plot( lena.dtwt )
```

An expanded view of any one, or more, of the small images also can be displayed. For example, a view of the images for the angles ± 45 deg is obtained via the commands

```
> plot( lena.dtwt, angle=c(+45,-45) )
```

and is shown in Fig. 3.27. The ability of the DTWT to emphasize features oriented at specific angles in the image is apparent.

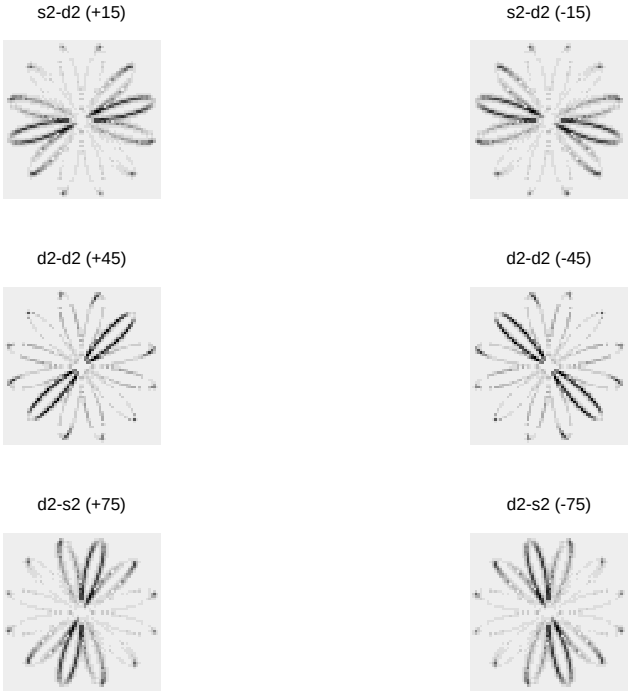

FIGURE 3.22: Complex magnitudes of the DTWT of the flower image.

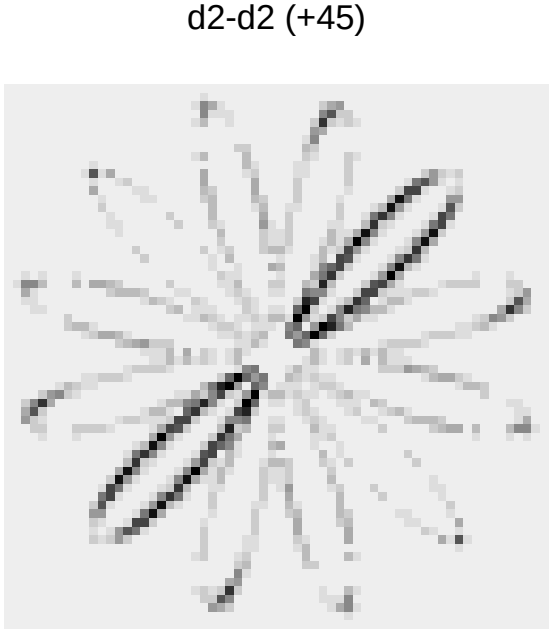

FIGURE 3.23: Complex magnitude of the DTWT of the flower image to emphasize features at +45 deg.

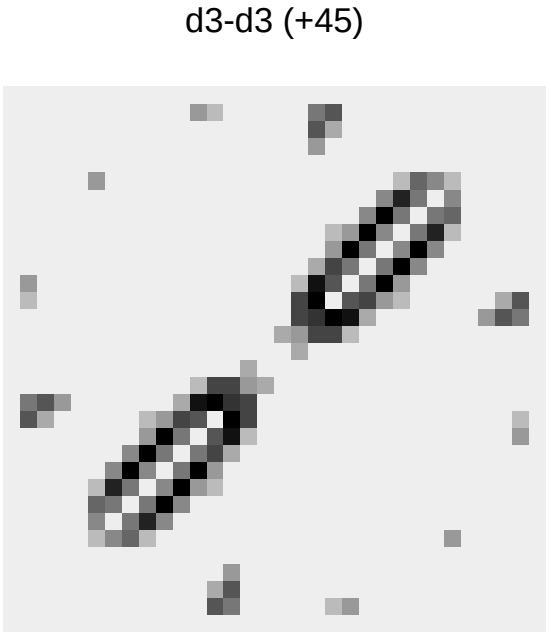

FIGURE 3.24: Complex magnitudes of the DTWT of the 45 deg flower image plotted with threshold = 0.8.

FIGURE 3.25: The Lena Image

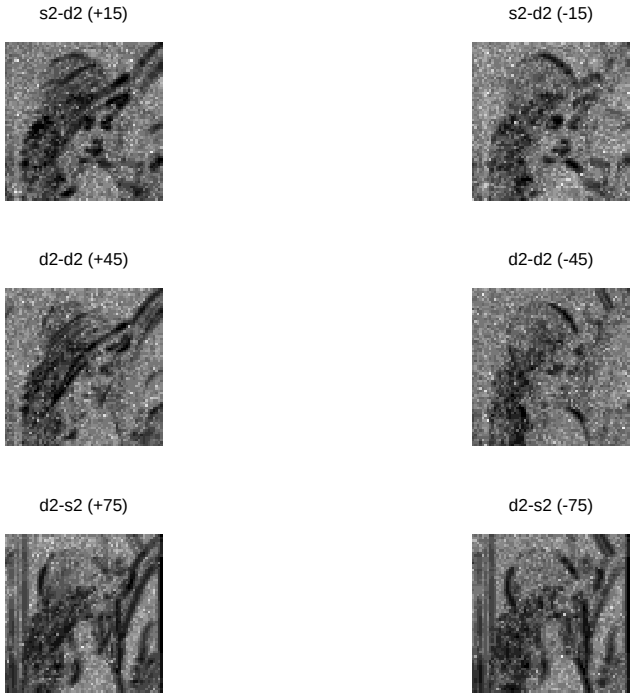

FIGURE 3.26: Complex magnitudes of the level 2 DTWT of the Lena image.

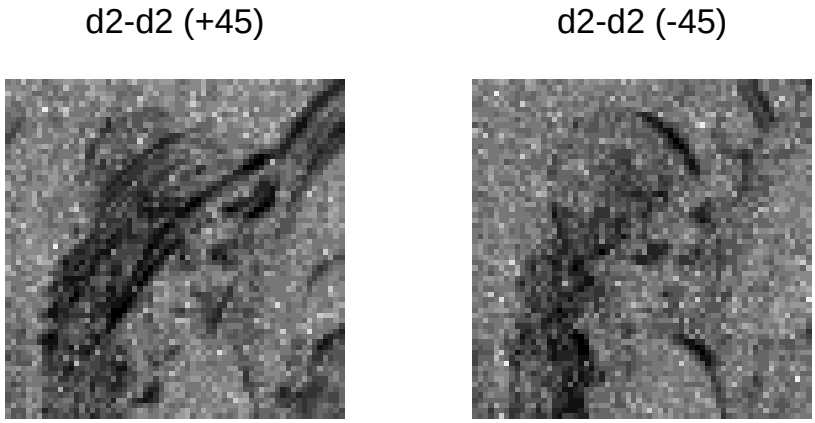

FIGURE 3.27: Complex magnitudes of DTWT of the Lena image selected to emphasize features oriented at ± 45 deg.

4
Wavelet Variance Analysis

Wavelet variance analysis is a method of partitioning the variance of a time series into pieces that are associated with different scales. This type of analysis tells us what scales are important contributors to the overall variability of a time series. Wavelet variance analysis has been found to be useful in the analysis of, e.g., financial time series, genome sequences, frequency fluctuations in atomic clocks, changes in the variance of soil properties, canopy gaps in forests, the accumulation of snow fields in the polar regions, turbulence in the atomosphere and ocean, and regular and semi-regular variable stars.

Wavelet variance analysis involves the estimation of a sequence of theoretical quantities called the wavelet variance (see §4.1 below for a precise definition). Each term in this sequence is associated with a particular scale $\tau_j \equiv 2^{j-1}$, $j = 1, 2, 3, \ldots$. If we regard the time series $X_0, X_1, \ldots, X_{N-1}$ under analysis as a realization of a portion of a stationary process with variance σ_X^2 and if we let $\nu_X^2(\tau_j)$ denote the wavelet variance for scale τ_j, then we have the fundamental relationship

$$\sigma_X^2 = \sum_{j=1}^{\infty} \nu_X^2(\tau_j).$$

The above relationship is analogous to the fundamental relationship between the variance of a stationary process and its spectral density function (SDF). If we let $S_X(f)$ denote this SDF at the frequency $f \in [-1/2, 1/2]$, then we have

$$\sigma_X^2 = \int_{-1/2}^{1/2} S_X(f)\, df.$$

Thus, just as the SDF for a stationary process decomposes the process variance across different frequencies, so does the wavelet variance decompose it across different scales.

The wavelet variance is of interest for the following reasons.

- Because the wavelet variance offers a scale-by-scale decomposition of the variability in certain stochastic processes, it has considerable appeal for researchers studying processes that exhibit fluctuations over a range of different scales. Such processes commonly occur in, e.g., the atmospheric sciences and oceanography.

- The wavelet variance is closely related to the concept of the SDF. Both provide an analysis of the variance of a stationary process across a physically interpretable independent variable (frequency in the case of the SDF, and scale in the case of the wavelet variance). Since the scale τ_j can be related to a range of

frequencies in the interval $[1/2^{j-1}, 1/2^j]$, the wavelet variance often leads to a more succinct decomposition. In addition, the square root of the wavelet variance has the same units as the original data, which can make it more easily interpretable than the SDF (if the units for the time series are in, say, meters, and if the time between observations is in, say, seconds, then the units for the SDF are in meters squared per Hertz, where a Hertz is defined to be one cycle per second).

- For certain stationary processes, the sample variance of a time series, namely,

$$\hat{\sigma}_X^2 \equiv \frac{1}{N}\sum_{t=0}^{N-1}(X_t - \overline{X})^2, \text{ where } \overline{X} \equiv \frac{1}{N}\sum_{t=0}^{N-1} X_t,$$

can grossly underestimate the process variance σ_X^2 even when the sample size N is quite large. For these processes, the wavelet variance is a useful substitute because it replaces the problematic notion of a 'global' variance with a sequence of variances over particular scales, for which we can readily formulate unbiased estimators. In addition, the wavelet variance is well-defined and can be easily estimated for certain nonstationary processes for which the variance is either infinite or an ever increasing function of the sample size N.

4.1 Definition of the Wavelet Variance

Let $\tilde{h}_{j,l}$ be the level j equivalent MODWT wavelet filter based upon the MODWT wavelet and scaling filters $\tilde{h}_l$ and $\tilde{g}_l$ of width L (see Section 3.2.1 for details). Let $\{X_t, t = \ldots, -1, 0, 1, \ldots\}$ represent a discrete parameter stochastic process, i.e., a collection of random variables (RVs) indexed by the set of all integers. Define the level j MODWT wavelet coefficients for this process as

$$\overline{d}_{j,t} = \sum_{l=0}^{L_j-1} \tilde{h}_{j,l} X_{t-l},$$

where t ranges over all the integers. By definition, the wavelet variance $\nu_X^2(\tau_j)$ for scale $\tau_j \equiv 2^{j-1}$ is defined to be the variance of $\overline{d}_{j,t}$:

$$\nu_X^2(\tau_j) \equiv \text{var}\{\overline{d}_{j,t}\}.$$

If we make certain assumptions about the process $\{X_t\}$, then $\nu_X^2(\tau_j)$ will be finite and independent of the time index t, and moreover we will have $E\{\overline{d}_{j,t}\} = 0$ so that $E\{\overline{d}_{j,t}^2\} = \text{var}\{\overline{d}_{j,t}\}$ (for these conditions to hold, it suffices that the backward differences of $\{X_t\}$ of a certain order constitute a zero mean stationary process; see Chapter 8 of Percival and Walden, 2000, for details). In what follows, we assume that $\nu_X^2(\tau_j)$ and $\overline{d}_{j,t}$ obey these conditions.

4.2 Estimation of the Wavelet Variance

Suppose now that we have a time series of N values that we regard as a portion $X_0, X_1, \ldots, X_{N-1}$ of a stochastic process $\{X_t\}$. Let $\{\tilde{h}_l\}$ be a MODWT wavelet filter of width L, and assume that $\{X_t\}$ satisfies conditions such that the wavelet variance $\nu_X^2(\tau_j)$ for scale $\tau_j \equiv 2^{j-1}$ based upon this filter is finite and independent of time. Let $L_j \equiv (2^j - 1)(L-1) + 1$ be the width of the equivalent MODWT filter $\{\tilde{h}_{j,l}\}$ for level j (when $j = 1$, we have $\tilde{h}_{1,l} = \tilde{h}_l$ and $L_1 = L$). Then the unbiased MODWT estimator of the wavelet variance is defined as

$$\hat{\nu}_X^2(\tau_j) \equiv \frac{1}{M_j}\sum_{t=L_j-1}^{N-1} \tilde{d}_{j,t}^2,$$

where $\tilde{d}_{j,t}$ is the MODWT wavelet coefficient at level j and time index t, and $M_j \equiv N - L_j + 1$ (we assume that $M_j \geq 1$). Note that, while there are N MODWT wavelet coefficients at each level j, we only use the last M_j of

these. The unbiased wavelet variance estimator avoids the first $L_j - 1$ coefficients on each level because these are the so-called boundary coefficients; i.e., these coefficients make explicit use of the circularity assumption built into the MODWT and hence are formed by combining together values both at the beginning and the end of the time series, with the result that $E\{\tilde{d}_{j,t}^2\}$ need not be equal to $\nu_X^2(\tau_j)$. If we retain these boundary coefficients, we obtain a biased MODWT estimator of the wavelet variance, namely,

$$\widetilde{\nu}_X^2(\tau_j) \equiv \frac{1}{N} \sum_{t=0}^{N-1} \tilde{d}_{j,t}^2.$$

The DWT can also be used to formulate estimators of the wavelet variance, but the sampling properties of these estimators are generally inferior to those of the MODWT estimators. Let $N_j' \equiv \lfloor N/2^j \rfloor$ be the number of DWT wavelet coefficients at level j, and $L_j' \equiv \lceil (L-2)(1-2^{-j}) \rceil$ be the number of DWT boundary coefficients at level j (assuming $N_j' > L'j$). Then the DWT version of the unbiased estimator of the wavelet variance is defined as

$$\hat{\nu}_X^2(\tau_j) \equiv \frac{1}{(N_j' - L_j')2^j} \sum_{t=L_j'-1}^{N_j-1} d_{j,t}^2,$$

where $d_{j,t}$ is the DWT coefficient at level j and time index t. Similarly, the DWT version of the biased wavelet variance is defined as

$$\widetilde{\nu}_X^2(\tau_j) \equiv \frac{1}{N_j' 2^j} \sum_{t=0}^{N_j'-1} d_{j,t}^2.$$

4.3 Distribution of Wavelet Variance Estimators

An approximation to the distribution of the unbiased MODWT wavelet variance estimator $\hat{\nu}_X^2(\tau_j)$ has been worked out and can be used to assess its sampling variability and to obtain confidence intervals for the true wavelet variance $\nu_X^2(\tau_j)$ (see Sections 8.3 and 8.4 of Percival and Walden, 2000, for details). This approximation is based on the assumption that the statistic $\hat{\nu}_X^2(\tau_j)$ has a distribution that is equal to an RV given by the product of a chi-square RV χ_η^2 with η degrees of freedom and the constant $\nu_X^2(\tau_j)/\eta$. The starting point for this approximation is to note that, if we have M independent and identically distribution Gaussian RVs with mean zero, then the sum of their squares forms an RV whose distribution is given by the product of a chi-square RV χ_M^2 with M degrees of freedom and a constant. By assumption, the MODWT wavelet coefficients $\tilde{d}_{j,t}$ that we use to form $\hat{\nu}_X^2(\tau_j)$ are Gaussian RVs with mean zero and variance $\nu_X^2(\tau_j)$; however, because these coefficients are in general correlated with each other, their sum of squares is not a chi-square RV with M_j degrees of freedom. We can adjust for this correlation by setting η equal to a value such that the RV $\nu_X^2(\tau_j)\chi_\eta^2/\eta$ has the same theoretical variance as $\hat{\nu}_X^2(\tau_j)$ (we can obtain a good approximation to this theoretical variance by appealong to a large sample approximation). In this approach, η is known as the 'equivalent degrees of freedom' (EDOF) and in effect becomes a parameter that we need to determine somehow.

The S+Wavelets module supports three different modes for setting the EDOF η.

1. **EDOF** $\hat{\eta}_1$ (based upon large sample theory):

$$\hat{\eta}_1 = \frac{M_j \hat{\nu}_X^4(\tau_j)}{\hat{A}_j},$$

where

$$\hat{A}_j \equiv \frac{\hat{\nu}_X^4(\tau_j)}{2} + \sum_{\tau=1}^{M_j-1} \hat{s}_{j,\tau}^2,$$

and $\hat{s}_{j,\tau}$ is a sample lag τ autocovariance defined by

$$\hat{s}_{j,\tau} \equiv \frac{1}{M_j} \sum_{t=L_j-1}^{N-1-|\tau|} \tilde{d}_{j,t} \tilde{d}_{j,t+|\tau|} \quad 0 \le |\tau| \le M_j - 1.$$

2. **EDOF** $\hat{\eta}_2$ (based upon the assumption that the shape of the SDF for $\{X_t\}$ is known *a priori*):

$$\hat{\eta}_2 = \frac{2\left(\sum_{k=1}^{\lfloor (M_j-1)/2 \rfloor} C_j(f_k)\right)^2}{\sum_{k=1}^{\lfloor (M_j-1)/2 \rfloor} C_j^2(f_k)},$$

where $f_k \equiv k/M_j$ and $C_j(f) \propto \widetilde{\mathcal{H}}_j^{(D)}(f) S_X(f)$. i.e., the product of the squared gain function for the Daubechies MODWT equivalent wavelet filter $\{\tilde{h}_{j,l}\}$ for level j and the SDF for $\{X_t\}$ (assumed to be known up to a constant of proportionality).

3. **EDOF** $\hat{\eta}_3$ (large sample approximation based upon a band-pass approximation):

$$\hat{\eta}_3 = \max\{M_j/2^j, 1\}.$$

Once η has been set to either $\hat{\eta}_1$, $\hat{\eta}_2$ or $\hat{\eta}_3$, we can calculate an approximate $100(1-2p)\%$ confidence interval for $\nu_X^2(\tau_j)$ via

$$\left[\frac{\eta \hat{\nu}_X^2(\tau_j)}{Q_\eta(1-p)}, \frac{\eta \hat{\nu}_X^2(\tau_j)}{Q_\eta(p)}\right],$$

where $Q_\eta(p)$ is the $p \times 100\%$ percentage point for the chi-square distribution with η degrees of freedom (setting $p = 0.025$ yields an approximate 95% confidence interval).

4.4 The WaveletVariance Class

The wavVariance function calculates the wavelet variance of a real-valued uniformly-sampled time series and returns an object of class WaveletVariance. Several methods are available to the user to view, summarize and access the data contained in a WaveletVariance object:

Summary Operator Methods:

print	Prints useful information regarding the wavelet variance including: • Type of wavelet transform. • Information regarding the filter set. • Number of decomposition levels. • Boundary extension rule. • Filtering technique (convolution or correlation). • The crystal names. • The scale of each crystal.
plot	Plots the wavelet variance estimates. The plot function has two optional arguments type and edof. Set type to "unbiased" or "biased" to plot unbiased or biased estimates, respectively. To select the EDOF mode of confidence intervals to be plotted, set edof to a vector of integers representing the desired EDOF mode as a third argument (e.g., c(1, 3) for both EDOF mode 1 and 3).
summary	Displays a statistical summary of the transform data.

Data Access Methods

$	Use to access specific components of the class object. A list of accessible components can be generated using the names function.

4.5 Example: Wavelet Variance of Atomic Clock Data

```
> var.clock <- wavVariance( atomclock, n.levels = 5 )
> summary( var.clock )

MODWT wavelet variance of atomclock
               d1       d2       d3       d4       d5
  Biased 3.55e+00 4.95e+00 9.17e+00 1.80e+01 3.58e+01
Unbiased 2.39e-06 5.61e-06 1.47e-05 4.13e-05 1.73e-04

Confidence Interval Data for Unbiased
Time Independent Wavelet Variance:

           d1   d2  d3  d4  d5
   Level    1    2   3   4   5
   Scale    1    2   4   8  16
 [EDOF1]  805  353 172  99  50
 [EDOF2]   NA   NA  NA  NA  NA
 [EDOF3]  510  251 122  58  25
# coeffs 1019 1005 977 921 809
```

The summary method for the WaveletVariance object gives a complete picture of the wavelet variance estimates as well as the EDOF and corresponding confidence intervals. To plot the data, simply invoke the plot method using an optional character string to denote biased or unbiased estimates as a second argument and a vector denoting the EDOF mode to use in displaying the confidence intervals as an optional third argument:

```
> plot( var.clock, type = "unbiased", edof = c(1,3) )
```

Figure 4.1 shows the wavelet variance versus scale (measured in days) for an atomic clock sequence formed by measuring the difference in time kept by a clock 571 and by an ensemble of cesium beam atomic clocks used in the 1970s by the US Naval Observatory to form the official time scale for the US. The fact that the values of the wavelet variance for the four smallest scales (1 to 8 days) lie roughly on a line in a log-log plot indicates that the SDF varies approximately as a power law over the corresponding range of frequencies (i.e., 1/32 to 1/2 cycles per day). The fact that the wavelet variances increase monotonically with scale indicates that the ability of clock 571 to keep time decreases as the time span (scale) increases.

The DWT estimation of the wavelet variance can also be calculated by setting the transform argument appropriately.

```
>  wavVariance( atomclock, transform = "dwt" )

DWT wavelet variance of atomclock
Wavelet: s8
Length of series: 1026
Number of levels: 7
Boundary correction rule: periodic
Filtering technique: convolution
Sampling interval:  1
       d1 d2 d3 d4 d5 d6 d7
Scales  1  2  4  8 16 32 64
```

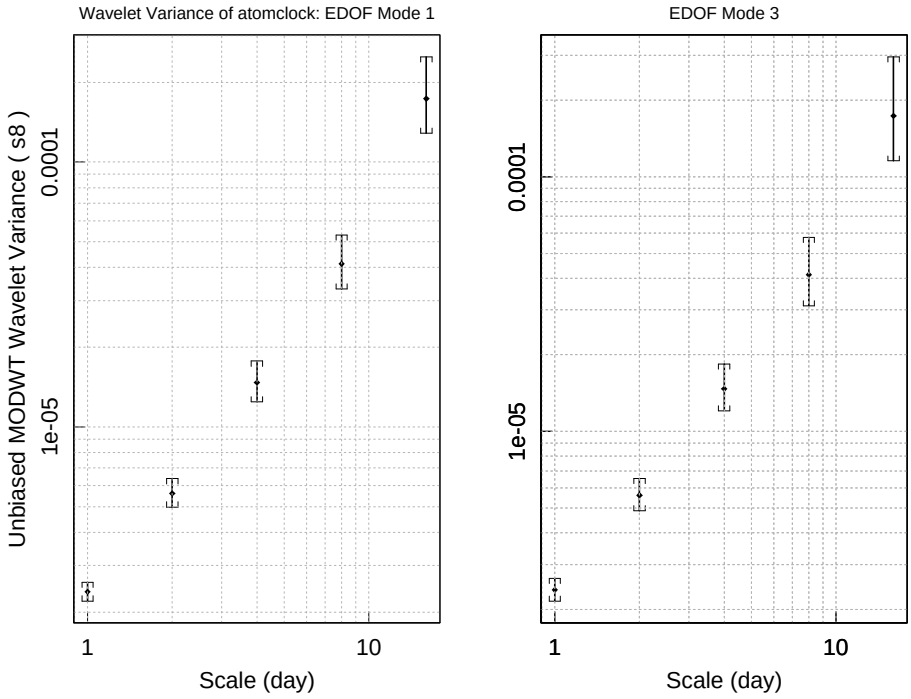

FIGURE 4.1: Unbiased MODWT wavelet variance estimates of the atomclock time series for levels $j = 1, \ldots, 5$ using Daubechies s8 filters. The left and right plots display confidence intervals using EDOF mode 1 and 3, respectively.

NOTE: No confidence limits are available for the DWT wavelet wavelet variance estimator. The default transform (MODWT) also produces more accurate estimates than does the DWT version and can be used to produce instantaneous variance estimates.

4.6 Other Wavelet Variance Functions

You can also use the wavEDOF and wavConfidenceLimits functions individually to obtain EDOF and wavelet variance confidence interval estimates, respectively. Here, we estimate the EDOF and wavelet variance confidence intervals for the ocean vertical shear time series:

```
> edof <- wavEDOF(ocean)
> edof
$EDOF1:
       d1       d2       d3       d4      d5       d6
 1808.758 940.1328 522.7423 274.3481 78.1384 81.87458

       d7      d8       d9
 27.16933 18.4152 6.777816

$EDOF2:
[1] NA NA NA NA NA NA NA NA NA

$EDOF3:
     d1      d2      d3       d4       d5       d6
 2044.5 1018.75 505.875 249.4375 121.2188 57.10938

       d7       d8       d9
 25.05469 9.027344 1.013672

$variance.unbiased:
           d1           d2           d3           d4
 0.0002309741 0.0004410549 0.0005715682 0.001836403

          d5         d6        d7        d8       d9
 0.009440761 0.04758687 0.4106984 0.9862548 0.469676

$n.coeff:
   d1   d2   d3   d4   d5   d6   d7   d8  d9
 4089 4075 4047 3991 3879 3655 3207 2311 519
```

The wavEDOF function returns a list with members variance.unbiased and EDOF1, EDOF2, EDOF3 which can be used as inputs to the wavConfidenceLimits function. Note that (in general) only unbiased wavelet variance estimates are suitable for confidence interval estimation so the unbiased version is accordingly not returned by wavEDOF. To calculate the 95% confidence intervals corresponding to EDOF mode 1 we call the wavConfidenceLimits function as follows:

```
> wavConfidenceLimits( edof$variance.unbiased, edof$EDOF1 )
$low:
           d1           d2           d3           d4
 0.0002188667 0.0004094998 0.0005177851 0.001604601

          d5         d6        d7        d8        d9
 0.007393772 0.03746485 0.2767846 0.6179611 0.2262995

$high:
           d1           d2          d3          d4
 0.0002441744 0.0004766303 0.000634739 0.002125904

         d5         d6        d7       d8       d9
 0.01255137 0.06281859 0.6854098 1.874926 1.468822
```

The result shows the low and high 95% confidence interval limits based on the unbiased wavelet variance estimates and the chi-square EDOF estimates.

4.7 Testing for Homogeneity of Variance

An implicit assumption we have made in constructing an estimator of the wavelet variance is that var$\{\tilde{d}_{j,t}\}$ does not depend upon the time index t; i.e., we are assuming that the variances of the MODWT wavelet coefficients iare homogeneous across time. If this assumption appears to be questionable for a particular time series, we can perform a statistical test based upon the corresponding DWT (recall that we can extract the DWT coefficients from the MODWT coefficients by an appropriately subsampling and rescaling the latter).

The wavVarianceHomogeneity function tests for homogeneity of variance for each level of a DWT decomposition. Based on the assumption that the DWT decorrelates the data, the nonboundary (interior) wavelet coefficients in a given level (d_j) can be regarded as a zero mean Gaussian white noise process. For a homogeneous distribution of $d_{j,t}$, there is an expected linear increase in the cumulative energy as a function of time. The so called D-statistic denotes the maximum deviation of the $d_{j,t}$ from a hypothetical linear cumulative energy trend. This D-statistic is then compared to percentage points dictated by the distribution of D under the null hypothesis that the variance is in fact homogeneous. Comparing the D-statistic for the $d_{j,t}$ values o the corresponding percentage points provides a means of quantitatively rejecting or failing to reject the null hypothesis.

The wavVarianceHomogeneity function can be used to perform a homogeneity test on a level by level basis. This function accpets a number of input arguments that are used to specify how the test is to be performed. The main arguments are an input time series as well as the optional arguments: wavelet, n.levels, significance, lookup, n.realization, n.repetition. The arguments wavelet and n.levels are the same as those used for the wavelet transforms. The remaining optional arguments are used as used to determine the percentage points under the null hypothesis. An Inclan–Tiao approximation to the percentage points is used for sample sizes $N \geq 128$ while a Monte Carlo technique is used for $N < 128$. For the Monte Carlo technique, the D-statistic for a Gaussian white noise sequence of length N is calculated. This process is repeated n.realization times, forming a distribution of the D-statistic under the null hypothesis. The upper 10%, 5% and 1% percentage points based upon n.realization realizations are formed a total of n.repetition times, and averaged to form an estimate of the true percentage points for the D-statistic. Because the Monte Carlo study can be both computationally and memory intensive, it is highly recommended that lookup be set to TRUE, its default value.

As an example, let us apply the test for homogeneity of variance to the fdp045 series (a simulated stochastic fractal time series). To begin the test, we simply call the wavVarianceHomogeneity function.

```
> wavVarianceHomogeneity( fdp045 )
Homogeneity Test for Discrete Wavelet Transform of fdp045

Pass:
   10% 5% 1%
d6   F  T  T
d5   T  T  T
d4   F  T  T
d3   T  T  T
d2   T  T  T
d1   T  T  T

D-statistic critical values comparison:
                  N          D       10%        5%
d6: Monte Carlo   2 0.99561682 0.9931909 0.9984854
d5: Monte Carlo  10 0.26449419 0.5027442 0.5577445
d4: Monte Carlo  26 0.34851101 0.3185281 0.3549792
d3: Monte Carlo  58 0.09888661 0.2183442 0.2425481
d2: Monte Carlo 123 0.10464938 0.1515953 0.1679855
d1: Inclan-Tiao 253 0.03866207 0.1088155 0.1207195

                       1%
d6: Monte Carlo 0.9999397
d5: Monte Carlo 0.6590028
d4: Monte Carlo 0.4280126
d3: Monte Carlo 0.2945787
d2: Monte Carlo 0.2028810
d1: Inclan-Tiao 0.1447117

Lookup: TRUE
Inclan-Tiao tolerance:  1e-06
Repetitions: 3 (lookup), 3 (non-lookup)
Realizations/repetition: 10000 (lookup), 10000 (non-lookup)

Wavelet: s8
Length of series: 512
Number of levels: 6
Boundary correction rule: periodic
Filtering technique: convolution
```

4.8 Estimation of the Wavelet Covariance

In a manner analogous to how we defined the wavelet variance for a time series, we can define a wavelet covariance between two time series. The wavCovariance function is used to estimate the wavelet covariance of two sequences. As in the case of the wavelet variance, we can formulate biased and unbiased estimators of the wavelet covariance based upon either the MODWT or the DWT. Given the time series $X_0, X_1, \ldots, X_{N-1}$ and $Y_0, Y_1, \ldots, Y_{N-1}$, the MODWT biased and unbiased estimators of the wavelet covariance are defined as

$$\text{Biased: } \widetilde{\nu}^2_{XY} \equiv \frac{1}{N} \sum_{t=0}^{N-1} \widetilde{W}^{(X)}_{j,t} \widetilde{W}^{(Y)}_{j,t} \tag{4.1}$$

$$\text{Unbiased: } \widehat{\nu}^2_{XY} \equiv \frac{1}{M_j} \sum_{t=L_j}^{N-1} \widetilde{W}^{(X)}_{j,t} \widetilde{W}^{(Y)}_{j,t} \tag{4.2}$$

where, as before, $M_j \equiv N - L_j + 1$ is the number of nonboundary (interior) MODWT wavelet coefficients, and $L_j = (2^j - 1)(L-1) + 1$ is the length of the level j MODWT equivalent filter $\{h_{j,l}\}$. An integer lag ϕ can be used to study the covariance between to processes whose events are assumed to be correlated at different times. For example, the lagged biased wavelet covariance is defined as

$$\text{Biased, Lagged: } \widetilde{\nu}^2_{XY,\phi} \equiv \frac{1}{N} \sum_{t=0}^{N-1} \widetilde{W}^{(X)}_{j,t} \widetilde{W}^{(Y)}_{j,t+\phi \bmod N}. \tag{4.3}$$

The wavelet covariance can also be estimated using DWT wavelet coefficients. Given the same time series as before, the DWT biased estimator of the wavelet covariance is defined as

$$\text{Biased: } \tilde{\nu}^2_{XY} \equiv \frac{1}{N} \sum_{t=0}^{N_j-1} d^{(X)}_{j,t} d^{(Y)}_{j,t}, \tag{4.4}$$

where $N_j \equiv N/2^j$ for $j = 1, \ldots, J$. As before, only the nonboundary (interior) wavelet coefficients are used to form the corresponding unbiased DWT-based wavelet covariance estimator. Assuming that $N_j > L'_j$, this estimator is given by

$$\text{Unbiased: } \hat{\nu}^2_{XY} \equiv \frac{1}{M'_j} \sum_{t=L'_j}^{N_j-1} d^{(X)}_{j,t} d^{(Y)}_{j,t}, \tag{4.5}$$

where $M'_j \equiv N - 2^j L'_j$ with $L'_j \equiv \lceil (L-2)(1-2^{-j}) \rceil$ (for large j, $L'_j = L-2$). The lagged biased wavelet covariance estimator is defined as

$$\text{Biased, Lagged: } \tilde{\nu}^2_{XY,\phi} \equiv \frac{1}{N} \sum_{t=0}^{N_j-1} d^{(X)}_{j,t} d^{(Y)}_{j,t+\phi \bmod N_j}. \tag{4.6}$$

As a simple (and somewhat artificial) example, let us perform the wavelet covariance calculations for consecutive 1024 point segments of the sunspots series.

```
> sun1 <- sunspots[1:1024]
> sun2 <- sunspots[(1:1024)+1024]
> xcov <- wavCovariance( sun1, sun2 )
> xcov

MODWT wavelet covariance of sun1 and sun2
Wavelet: s8
Length of series: 1024
Number of levels: 7
Boundary correction rule: periodic
Filtering technique: convolution
Shifted for approximate zero phase: FALSE
Scales:
 d1 d2 d3 d4 d5 d6 d7
  1  2  4  8 16 32 64

Lags:
 d1 d2 d3 d4 d5 d6 d7
  0  0  0  0  0  0  0

> plot( xcov )
```

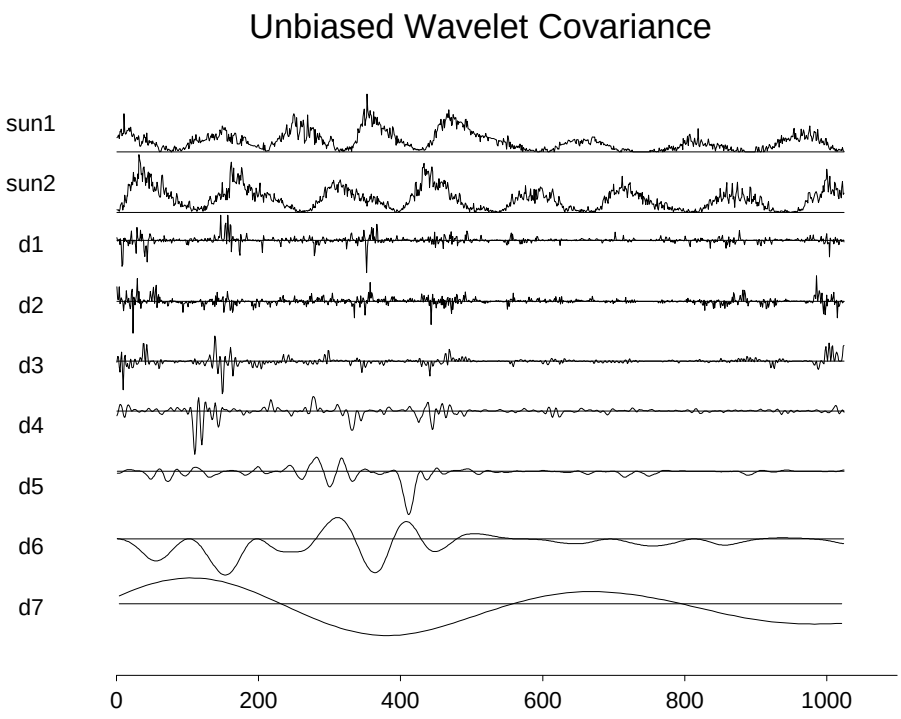

FIGURE 4.2: The discrete wavelet covariance of the successive sun-spots series.

The results are shown in Fig. 4.2. There is an apparent lag between the two sequences of approximately 75 points. We can impose a corresponding lag in the wavelet covariance results.

```
> xcov.lag <- wavCovariance( sun1, sun2, lag = 75 )
> xcov.lag

MODWT wavelet covariance of sun1 and sun2
Wavelet: s8
Length of series: 1024
Number of levels: 7
Boundary correction rule: periodic
Filtering technique: convolution
Shifted for approximate zero phase: FALSE
Scales:
 d1 d2 d3 d4 d5 d6 d7
  1  2  4  8 16 32 64

Lags:
 d1 d2 d3 d4 d5 d6 d7
 75 75 75 75 75 75 75

> plot( xcov )
```

The results are shown in Fig. 4.3. Relative to Fig. 4.2, the convariance results shown in Fig. 4.3 exhibit a shift in covariance over various scales in the data. The lag argument may also be used to specify different lags for

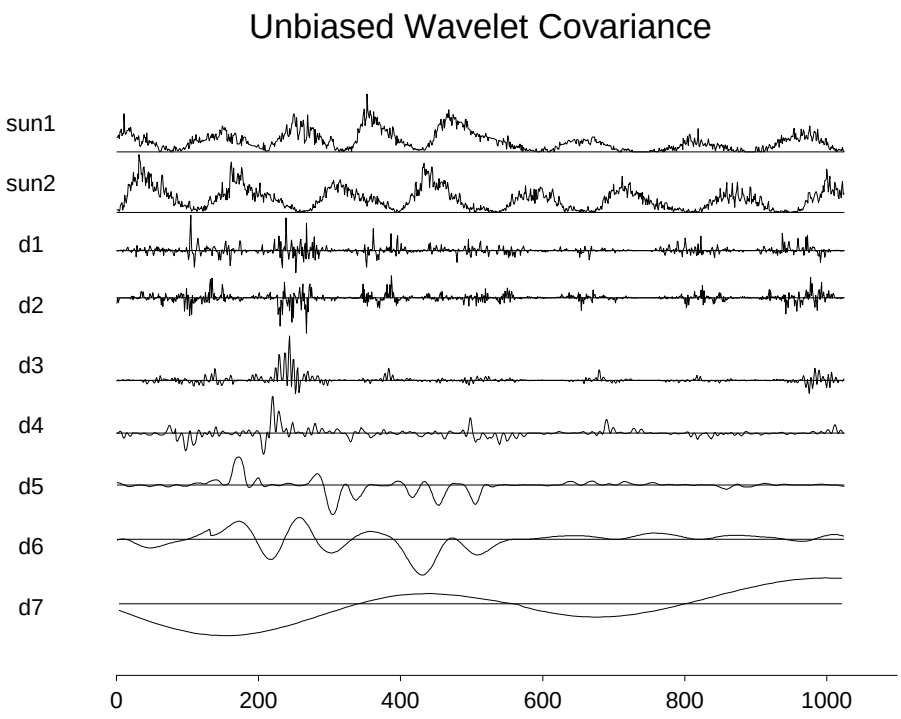

FIGURE 4.3: The discrete wavelet covariance of the sun-spots series with an imposed lag $\phi = 75$ for each decomposition level.

different decomposition levels.

```
> wavCovariance( sun1, sun2, lag = c(20, 30, 40) )

MODWT wavelet covariance of sun1 and sun2
Wavelet: s8
Length of series: 1024
Number of levels: 7
Boundary correction rule: periodic
Filtering technique: convolution
Shifted for approximate zero phase: FALSE
Scales:
 d1 d2 d3 d4 d5 d6 d7
  1  2  4  8 16 32 64

Lags:
 d1 d2 d3 d4 d5 d6 d7
 20 30 40 40 40 40 40
```

Notice that if the length of the lag vector is less than the number of decomposition levels, then the the last lag is replicated accordingly.

5

Wavelet-Based Analysis of Fractionally Differenced Processes

In recent years there has been considerable interest in using stochastic processes with so-called 'power-law' or 'fractal' properties as models for various financial, biomedical and geophysical time series. Specific applications for these processes include

- forward premiums, interest rate differentials and inflation rates [Bai96];
- voltage fluctuations across cell membranes;
- density fluctuations in sand passing through an hour glass;
- traffic fluctuations on Japanese expressways;
- impedance fluctuations in geophysical borehole;
- fluctuations in the rotation of the earth; and
- X-ray time variability of galaxies.

The basic idea behind such processes is that their spectral density functions (SDFs) vary as a power law over certain ranges of frequencies. Early attempts (dating from the 1960s) at defining stochastic processes that capture these ideas include fractional Brownian motion (FBM) and fractional Gaussian noise (FGN), both of which have seen wide usage. In the early 1980s, Granger and Joyeux [GJ80] and Hosking [Hos81] introduced a flexible class of models known as fractionally differenced (FD) processes. An FD process can be regarded as an extension to an autoregressive, integrated, moving average model in which the order of integration is allowed to assume non-integer values, and the orders for the autoregressive and moving average processes are both set to zero. FD processes capture the power-law and fractal ideas in a flexible and tractable manner and have the following advantages over FBM and FGN.

- UNLIMITED POWER LAW EXPONENT RANGE. Both FBM and FGN are stochastic power law processes because their SDFs are approximately proportional to $|f|^{\alpha}$ at low frequencies, where α is the exponent of the power law. For FBM, this exponent is limited to the open interval $(-3,-1)$; for FGN, it is limited to $(-1,1)$. Neither process actually includes the case $\alpha = -1$. Interestingly enough, this exceptional case seems to occur quite often in practical applications [BLW94] and is refered to in the literature as (pure) $1/f$ noise, pink noise and flicker noise. An FD process is also a stochastic power law process, but its exponent can assume any real-value, including the important case $\alpha = -1$.

- MODEL CONTINUITY. While practitioners have been tempted to group FBM and FGN processes together in an attempt to obtain coverage of power laws ranging from -3 up to 1, the FGN and FBM processes do not smoothly transition into one another at the $\alpha = -1$ boundary due to a discontinity in their SDFs at high frequencies. This discontinuity can lead to subtle problems in model selection. FD processes have no such discontinuity. In addition, an FD process is closed under differencing operations; i.e., if $\{X_t\}$ is an FD process and if we define $Y_t = X_t - X_{t-1}$, then the process $\{Y_t\}$ is also an FD process. By contrast, differencing an FGN or FBM process will not yield the same type of process. Because differencing plays such a prominent role in time series analysis, FD processes are thus more flexible and tractable as models.

- TRACTABLE TIME AND FREQUENCY DOMAIN CHARACTERIZATIONS. In contrast to FBM and FGN processes, an FD process has simple expressions for both its SDF and (when stationary) its autocovariance sequence (ACVS). This means that both the SDF and ACVS of the FD processes can be manipulated analytically and also readily computed without having to approximate any infinite summations (as occurs in the expression for the SDF for FGN).

- MODEL FLEXIBILITY. Both autoregressive and moving average components can be added to an FD process to provide additional modeling flexibility, leading to the well-known class of autoregressive, fractionally integrated, moving average (ARFIMA) models [Ber94]. This flexibility is sometimes needed to model the high frequency content of measured data, which is often contaminated by exogenous noise sources. Similar extensions could be made to FBM and FGN processes, but such additions would further complicate their SDFs and ACVSs.

In the remainder of this chapter, we first give a definition for FD processes (§5.1) and then describe wavelet-based schemes for estimating the parameters of this process in a flexible manner (for further details, see Chapters 7 and 9 of Percival and Walden [PW00], and Constantine, et al, [CPR01], upon which the material that follows is based).

5.1 Definition of a Fractionally Differenced Process

The easiest way to define an FD process is via its SDF. Let δ be any real-valued number, let $\sigma_\varepsilon^2 > 0$, and let $\mathbb{Z}$ denote the set of all integers. The stochastic process $\{X_t, t \in \mathbb{Z}\}$ is called an FD$(\delta, \sigma_\varepsilon^2)$ process if it possesses an SDF given by

$$S_X(f) = \frac{\sigma_\varepsilon^2}{|2\sin(\pi f)|^{2\delta}} \qquad |f| \le 1/2. \tag{5.1}$$

An FD process thus depends on two parameters. The first, δ, is called the fractionally differenced parameter and determines the shape of the SDF; the second, σ_ε^2, is called the innovations variance and sets the level (height) of the SDF. When $\delta < 1/2$, an FD process is stationary and has an ACVS $\{s_{X,\tau}, \tau \in \mathbb{Z}\}$ that can be conveniently obtained by first computing

$$s_{X,0} = \mathrm{var}\{X_t\} = \frac{\sigma_\varepsilon^2 \Gamma(1-2\delta)}{\Gamma^2(1-\delta)},$$

where $\Gamma(\cdot)$ is Euler's gamma function [AS64], and then recursively using the formula

$$s_{X,\tau} = s_{X,\tau-1}\frac{\tau+\delta-1}{\tau-\delta}$$

for $\tau = 1, 2, \ldots$ (this also gives us the values of $s_{X,\tau}$ when $\tau < 0$ since $s_{X,\tau} = s_{X,-\tau}$). When $\delta \ge 1/2$, we obtain a class of nonstationary processes that are stationary if $\{X_t\}$ is differenced $d = \lfloor \delta + 1/2 \rfloor$ times, where $\lfloor x \rfloor$ is the greatest integer less than or equal to x; i.e., while $\{X_t\}$ is nonstationary, the process

$$Y_t \equiv \sum_{k=0}^{d} \binom{d}{k} (-1)^k X_{t-k}$$

is stationary (thus $Y_t = X_t - X_{t-1}$ when $d = 1$, $Y_t = X_t - 2X_{t-1} + X_{t-2}$ when $d = 2$, and so forth). If we use the small angle approximation $\sin(x) \approx x$, we see from Eq. 5.1 that the SDF for an FD$(\delta, \sigma_\varepsilon^2)$ process approximately obeys a power law process, i.e., $S_X(f) \propto |f|^\alpha$, at low frequencies with $\alpha = -2\delta$ (the error in this approximation is quite small for $|f| \leq 1/8$).

5.2 Wavelet-Based Estimation of FD Parameters

Suppose that we have a time series that can be regarded as a realization of a portion $\mathbf{X} = [X_0, X_1, \ldots, X_{N-1}]^T$ of an FD$(\delta, \sigma_\varepsilon^2)$ process, where δ is presumed to be unknown and is to be estimated from the time series. If the FD process is Gaussian, then arguably the best estimator for δ is the maximum likelihood estimator (MLE); however, this estimator can be very difficult to compute even for moderate sample sizes [Ber94]. Here we consider three computationally efficient schemes for estimating the parameter δ via a wavelet transform of the time series. The first two schemes make use of the fact that the relationship between the variance of the wavelet coefficients across scales is dictated by δ in such a manner that we can construct a least squares estimator (LSE) of δ (Abry *et al*. [AGF93, AG95], Abry and Veitch [AV98] and Jensen [Jen99b] consider similar estimators). The third scheme is a wavelet-based approximation to the MLE of δ (Wornell and Oppenheim [WO92], Wornell [Wor93, Wor96], Kaplan and Kuo [Kap93], McCoy and Walden [MW96] and Jensen [Jen99a, Jen00] discuss related wavelet-based MLEs). The first LSE and the MLE make use of the entire time series and hence are called 'block-dependent' estimators; by contrast, the second LSE utilizes only certain coefficients that are colocated in time, and we refer to it as an 'instantaneous' estimator.

5.2.1 Block-Dependent FD Model Parameter Estimators

Block-Dependent Weighted Least Squares Estimator

Let $\tilde{\mathbf{d}}_j$ be a vector of length N containing the MODWT wavelet coefficients for scale τ_j. Here we develop a weighted LSE (WLSE) of δ based upon an estimator of the variance of the nonboundary (interior) coefficients in $\tilde{\mathbf{d}}_j$ over a range of scales τ_j given by $J_0 \leq j \leq J_1$ (the selection of J_0 and J_1 is application dependent). Under the assumption that the length L of the wavelet filter is chosen such that $L/2 \geq \lfloor \delta + \frac{1}{2} \rfloor$, these nonboundary coefficients are a portion of a stationary process obtained by filtering the time series with the equivalent MODWT wavelet filter $h_{j,l}$. Since the squared gain function for $h_{j,l}$ is given by $\widetilde{\mathcal{H}}_{j,L}(f)$, the SDF for the nonboundary coefficients is given by $\widetilde{\mathcal{H}}_{j,L}(f)\, S_X(f)$, and hence their variance can be expressed as

$$\nu_X^2(\tau_j) \equiv \mathrm{var}\{\widetilde{W}_{j,t}\} = \int_{-1/2}^{1/2} \widetilde{\mathcal{H}}_{j,L}(f)\, S_X(f)\, df. \tag{5.2}$$

Using the approximation that $\widetilde{\mathcal{H}}_{j,L}(f)$ is an ideal bandpass filter over $|f| \in [1/2^{j+1}, 1/2^j]$ and taking into consideration the even symmetry of SDFs, an approximation to the wavelet variance is given by

$$\nu_X^2(\tau_j) \approx 2 \int_{1/2^{j+1}}^{1/2^j} S_X(f)\, df. \tag{5.3}$$

For FD processes, we have

$$\nu_X^2(\tau_j) \approx 2 \int_{1/2^{j+1}}^{1/2^j} \frac{\sigma_\varepsilon^2}{|2\sin(\pi f)|^{2\delta}}\, df. \tag{5.4}$$

When $j \geq 3$, so that $\sin \pi f \approx \pi f$, Equation **??** can be approximated by

$$\nu_X^2(\tau_j) \approx \sigma_\varepsilon^2\, \widetilde{c}(\delta)\, \tau_j^{2\delta-1}, \tag{5.5}$$

where $\widetilde{c}(\delta) \equiv \pi^{-2\delta}(1 - 2^{2\delta-1})/(1-2\delta)$. Equation **??** suggests that a direct means of estimating δ is to fit a least squares line to the logarithm of an estimate of the wavelet variance, say $\hat{\nu}_X^2(\tau_j)$. The slope of the line, say β,

that best fits $\ln(\hat{\nu}_X^2(\tau_j))$ versus $\ln(\tau_j)$ in a least squares sense is related to the FD parameter by $\delta = (\beta+1)/2$ and the power law exponent by $\alpha = -(\beta+1)$.

Given a time series of length N, we can obtain an unbiased MODWT-based estimate of the wavelet variance by defining

$$\widehat{\nu}_X^2(\tau_j) \equiv \frac{1}{\widetilde{M}_j} \sum_{t=\widetilde{L}_j-1}^{N-1} \tilde{d}_{j,t}^2, \tag{5.6}$$

where $\widetilde{M}_j \equiv N - \widetilde{L}_j + 1$ is the number of MODWT nonboundary wavelet coefficients. As a caveat, it should be noted that the wavelet variance estimates are somewhat sensitive to the order L of the wavelet filter used in the analysis. In particular, there can be a significant bias in estimating δ (and hence α) if we use the Haar wavelet filter (for which $L = 2$) [PW00]. This bias can be attributed to a spectral leakage phenomenon and can be attenuated by increasing L. In practice the choice $L = 8$ often works well.

As discussed in §4.3, the distribution for $\hat{\nu}_X^2(\tau_j)$ is approximately that of a random variable (RV) given by $\chi_{\eta_j}^2 \nu_X^2(\tau_j)/\eta_j$, where $\chi_{\eta_j}^2$ is a chi-square RV with η_j degrees of freedom. Define

$$Y(\tau_j) \equiv \ln(\hat{\nu}_X^2(\tau_j)) - \psi\Big(\frac{\eta_j}{2}\Big) + \ln\Big(\frac{\eta_j}{2}\Big), \tag{5.7}$$

where $\psi(\cdot)$ is the digamma function. The properties of the chi-square distribution dictate that

$$E\{Y(\tau_j)\} = \ln(\nu_X^2(\tau_j)) \text{ and } \operatorname{var}\{Y(\tau_j)\} = \psi'(\eta_j/2), \tag{5.8}$$

where $\psi'(\cdot)$ is the trigamma function. By assuming the approximation afforded by Eq. 5.5, we can now formulate a linear regression model $Y(\tau_j) = \gamma + \beta \ln(\tau_j) + e_j$, where $e_j \equiv \ln(\hat{\nu}_X^2(\tau_j)/\nu_X^2(\tau_j)) - \psi(\eta_j/2) + \ln(\eta_j/2)$ defines a sequence of errors, each with zero mean and variance $\psi'(\eta_j/2)$. If we take into account the inhomogeneity in the variance in these errors, we arrive at the WLSE of the slope term β given by

$$\hat{\beta}_{wlse} = \frac{\sum \omega_j \sum \omega_j \ln(\tau_j) Y(\tau_j) - \sum \omega_j \ln(\tau_j) \sum \omega_j Y(\tau_j)}{\sum \omega_j \sum \omega_j \ln^2(\tau_j) - (\sum \omega_j \ln(\tau_j))^2}, \tag{5.9}$$

where $\omega_j \equiv [\psi'(\eta_j/2)]^{-1}$, and all sums are over $j = J_0, \ldots, J_1$. The weighted least squares estimate of the FD parameter is then

$$\hat{\delta}_{wlse} = \frac{1}{2}(\hat{\beta}_{wlse} + 1). \tag{5.10}$$

If we ignore the possible correlation between the error terms (which we can decrease by increasing L), the variance of $\hat{\beta}_{wlse}$ is given by

$$\operatorname{var}\{\hat{\beta}_{wlse}\} = \frac{\sum \omega_j}{\sum \omega_j \sum \omega_j \ln^2(\tau_j) - (\sum \omega_j \ln(\tau_j))^2}, \tag{5.11}$$

and thus the variance of the $\hat{\delta}_{wlse}$ is given by

$$\operatorname{var}\{\hat{\delta}_{wlse}\} = \frac{1}{4}\operatorname{var}\{\hat{\beta}_{wlse}\}. \tag{5.12}$$

Monte Carlo studies indicate that Eq. 5.11 tends to overestimate the variability in $\hat{\beta}_{wlse}$ somewhat and thus can be regarded as a conservative upper bound [PW00].

Block-Dependent Maximum Likelihood Estimator

Wavelet-based maximum likelihood techniques can be used in harmony with an FD model as another means of obtaining estimates for FD parameters. Using the DWT is advantageous in that it is known to decorrelate long memory FD and related processes, forming a near independent Gaussian sequence, and thus simplifying the

statistics significantly [CPG00a]. The basic idea is to formulate the likelihood function for the FD parameters δ and σ_ε^2 directly in terms of the interior DWT wavelet coefficients. Let $\mathbf{d}_I$ be an $M = \sum_j M_j$ point vector containing all of the interior DWT wavelet coefficients over a specified range of scales $j = J_0, \ldots, J_1$. We can write the exact likelihood function for δ and σ_ε^2 as

$$\mathcal{L}(\delta, \sigma_\varepsilon^2 | \mathbf{d}_I) \equiv \frac{e^{-\mathbf{d}_I^T \Sigma_{\mathbf{d}_I}^{-1} \mathbf{d}_I / 2}}{(2\pi)^{M/2} |\Sigma_{\mathbf{d}_I}|^{1/2}}, \tag{5.13}$$

where $\Sigma_{\mathbf{d}_I}$ is the covariance matrix of $\mathbf{d}_I$, and $|\Sigma_{\mathbf{d}_I}|$ is the determinant of $\Sigma_{\mathbf{d}_I}$. Note that the dependence of the likelihood function on δ and σ_ε^2 is through $\Sigma_{\mathbf{d}_I}$ alone. Under the assumption that the wavelet coefficients in $\mathbf{d}_I$ are approximately uncorrelated, Eq. 5.13 can be approximated by

$$\tilde{\mathcal{L}}(\delta, \sigma_\varepsilon^2 | \mathbf{d}_I) \equiv \prod_{j=J_0}^{J_1} \prod_{t=0}^{M_j - 1} \frac{e^{-d_{j,t+L'_j}^2 / (2C_j(\delta, \sigma_\varepsilon^2))}}{(2\pi C_j(\delta, \sigma_\varepsilon^2))^{1/2}}, \tag{5.14}$$

where $C_j(\delta, \sigma_\varepsilon^2)$ is an approximation to the variance of $d_{j,t}$ given by the average value of the SDF in Eq. 5.1 over the nominal pass-band $[1/4\tau_j, 1/2\tau_j]$ for the equivalent wavelet filter $h_{j,l}$. The estimate $\hat{\delta}_{mle}$ of δ is obtained by maximizing $\tilde{\mathcal{L}}(\delta, \sigma_\varepsilon^2 | \mathbf{d}_I)$ with respect to δ. Equivalently we can consider the *reduced (natural) log likelihood function*

$$\tilde{l}(\delta | \mathbf{d}_I) \equiv M \ln(\tilde{\sigma}_\varepsilon^2(\delta)) + \sum_{j=J_0}^{J_1} M_j \ln(C'_j(\delta)), \tag{5.15}$$

where $C'_j(\delta) \equiv C_j(\delta, \sigma_\varepsilon^2)/\sigma^2\varepsilon$, and

$$\tilde{\sigma}_\varepsilon^2(\delta) \equiv \frac{1}{M} \sum_{j=J_0}^{J_1} \frac{1}{C'_j(\delta)} \sum_{t=0}^{M_j - 1} d_{j,t+L'_j}^2 \tag{5.16}$$

(see [PW00] for explicit details on the development of the reduced (natural) log likelihood function using the DWT coefficients). Minimizing Eq. **??**, which is a function of δ alone, yields the maximum likelihood estimate $\hat{\delta}_{mle}$, after which we can compute the corresponding estimate for σ_ε^2 by plugging $\hat{\delta}_{mle}$ into Eq. 5.16.

Under the assumption that $\delta \in [-1/2, L/2]$, the estimator $\hat{\delta}_{mle}$ for large M is approximately Gaussian distributed with mean δ and variance

$$\sigma_{\hat{\delta}_{mle}}^2 \equiv 2 \left[\sum_{j=J_0}^{J_1} M_j \phi_j^2 - \frac{1}{M} \left(\sum_{j=J_0}^{J_1} M_j \phi_j \right)^2 \right]^{-1}, \tag{5.17}$$

where

$$\begin{aligned} \phi_j &\equiv -\frac{4\sigma_\varepsilon^2}{\mathrm{var}\{d_{j,t}\}} \int_0^{1/2} \mathcal{H}_{j,L}(f) \frac{\ln(2\sin(\pi f))}{[2\sin(\pi f)]^{2\delta}} \, df \\ &\approx -\frac{2^{j+2}}{C'_j(\hat{\delta}_{mle})} \int_{1/4\tau_j}^{1/2\tau_j} \frac{\ln(2\sin(\pi f))}{[2\sin(\pi f)]^{2\hat{\delta}_{mle}}} \, df \end{aligned} \tag{5.18}$$

(see [CPG00b] for details). In practice, the right-hand integral can be approximated through either (i) numerical integration or (ii) a Taylor series expansion about the mid-band frequencies for levels $j = 1, 2$ along with direct integration using a small angle assumption for $j > 2$. The approximation above is based upon the view that the wavelet transform forms an octave band decomposition. There is generally a large increase in computational speed when using this bandpass approximation with relatively small loss of accuracy.

5.2.2 Instantaneous FD Model Parameter Estimators

The block-dependent estimators we formulated in 5.2.1 depend upon the entire time series $X_0, \ldots, X_{N-1}$. For time series whose statistical properties are evidently evolving over time, the assumptions behind this estimator are violated, and it is problematic to use this estimator on the entire times series. If, however, we can divide the time series up into blocks within which we can assume that the data are the realization of an FD process (with parameters that are now allowed to vary from one block to the next), we can apply the WLSE estimator on a block by block basis. In practice, each of the blocks will contain the same number of points, so we can now consider N to be the size of each block rather than the length of the entire time series. The choice of N is usually subjective and thus open to question, so it is useful to have some means of verifying that a particular choice is appropriate. We can do so by formulating an 'instantaneous' estimator that is independent of N and that can be used to check for departures from statistical consistency within a proposed block size.

Instantaneous Least Squares Estimator

The idea behind an instantaneous least squares estimate of δ is to use only a single wavelet coefficient from each scale; i.e., we only use $\tilde{d}^2_{j,t_j}$ to estimate $\nu^2_X(\tau_j)$, where t_j is the time index of the j^{th} level MODWT coefficient associated with time t in $\{X_t\}_{t=0}^{N-1}$. The time index t_j can be meaningfully determined *only* if (approximate) linear phase wavelet filters are used. With this substitution, the time dependent form of Eq. 5.10 becomes

$$\hat{\delta}_{lse,t} \quad = \quad \frac{\Delta_J \sum \ln(\tau_j) Y_t(\tau_j) - \sum \ln(\tau_j) \sum Y_t(\tau_j)}{2(\Delta_J \sum \ln^2(\tau_j) - (\sum \ln(\tau_j))^2)} + 1/2, \tag{5.19}$$

where $\Delta_J = J_1 - J_0 + 1$ and all sums are over $j = J_0, \ldots, J_1$ and

$$Y_t(\tau_j) \equiv \ln(\tilde{d}^2_{j,t_j}) - \psi(1/2) - \ln(2). \tag{5.20}$$

To decrease the variability of the estimates Δ_J should ideally be set to be as large as is feasible.

Instantaneous Maximum Likelihood Estimator

To form an instantaneous maximum likelihood estimator, we alter slightly the block-dependent reduced log likelihood function (Eq. **??**) so that (only) a single interior wavelet coefficient is used at each scale:

$$\tilde{l}(\delta|\mathbf{d}_I) \equiv \Delta_{\tilde{J}} \ln(\tilde{\sigma}^2_\varepsilon(\delta)) + \sum_{j=J_0}^{J_1} \ln(C'_j(\delta)), \tag{5.21}$$

where $C'_j(\delta) \equiv C_j(\delta, \sigma^2_\varepsilon)/\sigma^2\varepsilon$, and

$$\tilde{\sigma}^2_\varepsilon(\delta) \equiv \frac{1}{\Delta_{\tilde{J}}} \sum_{j=J_0}^{J_1} \frac{1}{C'_j(\delta)} \sum_{j=J_0}^{J_1} \tilde{d}^2_{j,t_j} \tag{5.22}$$

where t_j is the j^{th} level (zero phase shifted) MODWT coefficient that is associated with time t and $\Delta_{\tilde{J}} \equiv J_1 - J_0 + 1$ is the number of interior MODWT coefficients at time t_j (assuming $J_0 < J_1$ and that level J_0 contains at least one interior wavelet coefficient at time t_j).

5.2.3 The WaveletFDP Class

The wavFDPBlock and wavFDPTime functions estimate the block-dependent and instantaneous FD process model parameters, respectively, of a real-valued uniformly-sampled time series and each returns an object of class WaveletFDP. Several methods are available to the user to view, summarize, and access the data contained in a WaveletFDP object:

Summary Operator Methods:

print — Prints useful information regarding the FD model parameter estimation including:

- A summary of the estimated FD parameters: δ, var$\{\delta\}$, and σ_ε^2.
- The method used to estimate the FD process model parameters.
- Information regarding the handling of boundary coefficients.
- Decomposition levels over which the estimates were calculated.
- The search range of the FD parameter (δ).
- Information regarding the wavelet transform filters.

plot — Plots the estimates FD parameter δ_t. For instantaneous LSEs, an optional numeric constant can be used as a second argument to the plot function in order to display the confidence intervals about a known value of the FD parameter δ. This method is only applicable for instantaneous estimates produced by the wavFDPTime function.

Data Access Methods

$ — Use to access specific fields of the class object. A list of accessible fields can be generated using the names function.

5.2.4 An Example: FD Parameter Estimation of a Simulated FD Realization

Consider the 512 point FD$(0.45, 1.0)$ realization:

```
> plot( fdp045, type = "l" )
```

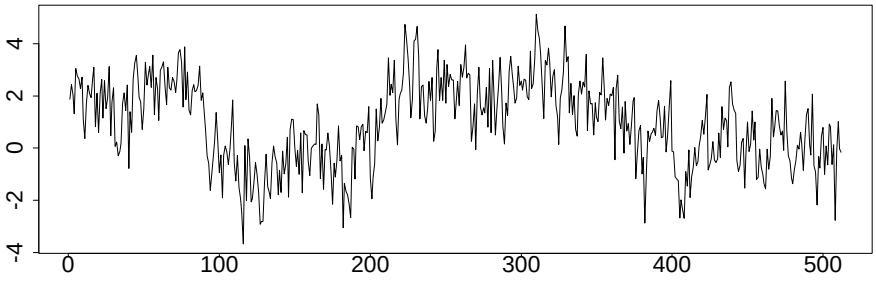

FIGURE 5.1: A 512 point realization of a FD process whose model parameters are constant: $\delta = 0.45$, $\sigma_\varepsilon^2 = 1.0$.

Figure 5.1 shows the time history of the FD realization. Since we know the true values of the FD model parameters, we can scrutinize the accuracy of block-dependent and instantaneous estimators. Let us first examine the block-dependent estimators. To perform a weighted least squares estimation we use the wavFDPBlock command as follows:

```
> wavFDPBlock( fdp045, lev=1:6, est="wlse", boundary="unbiased", edof=1 )

Block-dependent FD parameter estimation of fdp045

FD parameter estimate (delta): 0.37451
vardelta estimate: 0.00121
Innovations variance estimate: NA
Estimator: WLSE
Levels: 1 2 3 4 5 6
Boundary mode: unbiased
EDOF mode: 1
Delta range: -10 10
Wavelet: s8
Length of series: 512
Number of levels: 6
Boundary correction rule: periodic
Filtering technique: convolution
```

We see that the unbiased WLSE of delta is close to the known value of 0.45. But perhaps we could do better with the biased WLSE, this time over levels $2-6$:

```
> wavFDPBlock( fdp045, lev=2:6, est="wlse",
+ boundary="biased" )

Block-dependent FD parameter estimation of fdp045

FD parameter estimate (delta): 0.4432
vardelta estimate: 0.00206
Innovations variance estimate: NA
Estimator: WLSE
Levels: 2 3 4 5 6
Boundary mode: biased
EDOF mode: 1
Delta range: -10 10
Wavelet: s8
Length of series: 512
Number of levels: 5
Boundary correction rule: periodic
Filtering technique: convolution
```

Here, the biased WLSE does much better job than the unbiased case. Now we examine the blocked MLE technique using a stationary FD model. Under a stationary model, the FD parameter is assumed to be in the stationary regime, i.e. $\delta < 0.5$ which (in this case) we know to be true:

```
> wavFDPBlock( fdp045, lev=1:6, est="mle",
+ boundary="stationary" )

Block-dependent FD parameter estimation of fdp045

FD parameter estimate (delta): 0.46502
vardelta estimate: NA
Innovations variance estimate: 1.07116
Estimator: MLE
Levels: 1 2 3 4 5 6
Boundary mode: stationary FD process model
Delta range: -10 10
Wavelet: s8
Length of series: 512
Number of levels: 6
Boundary correction rule: periodic
Filtering technique: convolution
```

The MLE using a stationary model does an excellent job in estimating model parameters. For the general case, however, we will not know the value of δ a priori. In that case, let us use the stationary-nonstationary FD model which carries no restriction on the value of δ:

```
> wavFDPBlock( fdp045, lev=1:6, est="mle",
+ boundary="nonstationary" )

Block-dependent FD parameter estimation of fdp045

FD parameter estimate (delta): 0.4459
vardelta estimate: NA
Innovations variance estimate: 1.0846
Estimator: MLE
Levels: 1 2 3 4 5 6
Boundary mode: stationary-nonstationary FD process model
Delta range: -10 10
Wavelet: s8
Length of series: 512
Number of levels: 6
Boundary correction rule: periodic
Filtering technique: convolution
```

The result indicates that the stationary-nonstationary model also does an excellent job in estimating the FD model parameters.

Now we turn our attention to the instantaneous methods. We perform biased and unbiased instantaneous least squares estimates as follows:

```
> wavFDPTime( fdp045, lev=1:6, est="lse", biased=T )

Instantaneous FD parameter estimation of fdp045

Mean of FD parameter estimates (deltas): 0.45679
Mean of vardelta estimates: 0.14673
Mean of innovations variance estimates: NA
Estimator: LSE
Levels: 1 2 3 4 5 6
Boundary mode: biased
Chi-squared DOF: 1
Delta range: -10 10
Wavelet: s8
Length of series: 512
Number of levels: 6
Boundary correction rule: periodic
Filtering technique: convolution


> lse.unbiased <- wavFDPTime( fdp045, lev=1:6, est="lse",
+ biased=F )
> lse.unbiased

Instantaneous FD parameter estimation of fdp045

Mean of FD parameter estimates (deltas): 0.36536
Mean of vardelta estimates: 0.6948
Mean of innovations variance estimates: NA
Estimator: LSE
Levels: 1 2 3 4 5 6
Boundary mode: unbiased
Chi-squared DOF: 1
Delta range: -10 10
Wavelet: s8
Length of series: 512
Number of levels: 6
Boundary correction rule: periodic
Filtering technique: convolution
```

To plot the results we invoke the plot method:

```
> plot( lse.unbiased )
```

Figure 5.2 shows the result. The shaded areas about the estimates are the 95% confidence intervals based on chi-square distribution assumption on the interior wavelet coefficients. Since we know the true value of δ we can force the confidence intervals to be centered about $\delta = 0.45$:

```
> plot( lse.unbiased, mean.delta = 0.45 )
```

Figure 5.3 shows that the instantaneous LSE do a good job in estimating the true value of the time series with the exception of the endpoints which are more variable due to a lesser number of degrees of freedom. The

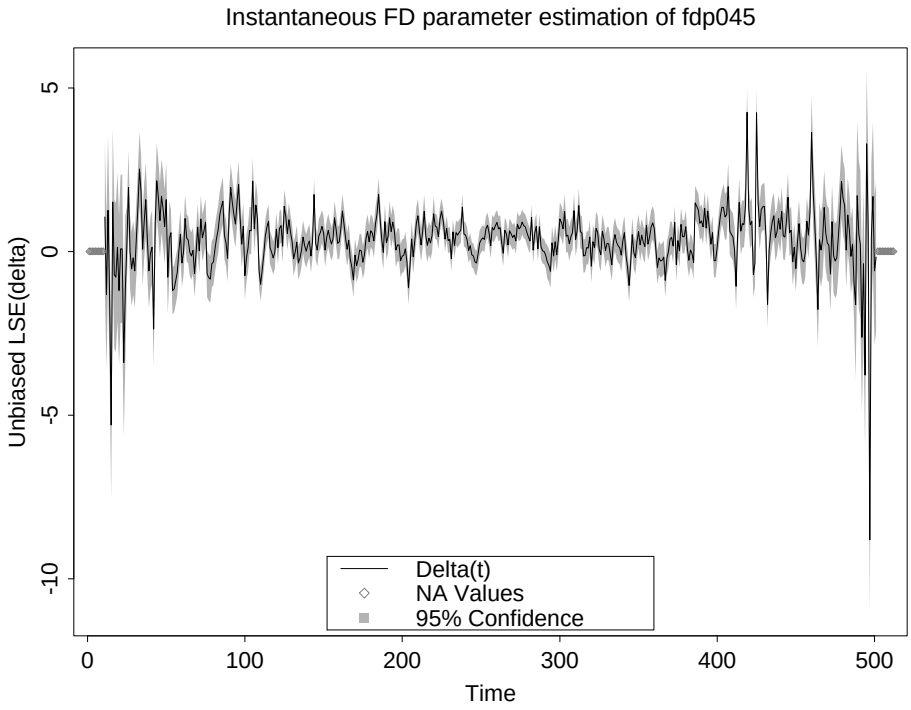

FIGURE 5.2: Unbiased instantaneous FD parameter estimation of a FD(0.45, 1.0) process. The 95% confidence intervals are displayed as the shaded covering the estimates.

endpoints of the MODWT has fewer interior wavelet coefficients at a particular time than do the points located more towards the middle of the sequence. This is and unavoidable and so we expect more variability near the endpoints.

The instantaneous estimators are inherently more variable than the blocked estimators. However, we can smooth out the instantaneous estimates by using more degrees of freedom in the estimates. What we are doing here is to average the energy of the MODWT coefficients around a given point in time, using multiple points instead on one. The more points that we use, the smoother the estimates at the price of being less localized in time. These estimates are known as *localized* estimates and lie somewhere between instantaneous and block-dependent estimators. To illustrate, let us develop localized MLEs of the fdp045 sequence with increasing DOFs:

```
> dofs <- 0:3
> deltas <- lapply(dofs,function(i)
+ wavFDPTime(fdp045, lev=1:6, est="lse", dof=i )$delta )
> names( deltas ) <- paste(2*(dofs)+1,"DOF")
> stack.plot( deltas, same.scale = T, zeroline=T )
```

Figure 5.4 shows the effects of the local smoothing on the instantaneous FD parameter estimates. The DOFs are increased through the dof argument of the wavFDPTime function, where dof represents the *DOF order, K*. The number of degrees of freedom is equal to $2 \times K + 1$) so the DOFs used in Fig. 5.4 (from top to bottom) are $1, 3, 5$, and 7, respectively. The sample mean and variance of localized LSE of δ are calculated via:

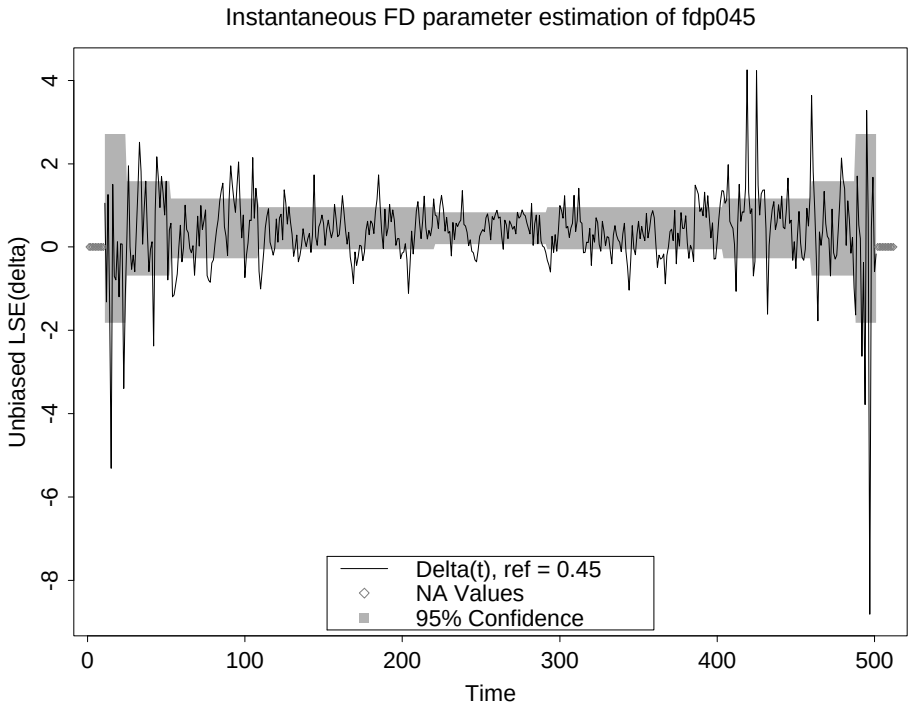

FIGURE 5.3: Unbiased instantaneous FD parameter estimation of a FD(0.45, 1.0) process. The 95% confidence intervals are displayed as the shaded covering the estimates based on the known value of $\delta = 0.45$.

```
> lapply( deltas, function(x) mean(x,na.rm=T) )
$"1 DOF":
[1] 0.3653601

$"3 DOF":
[1] 0.4133021

$"5 DOF":
[1] 0.3936822

$"7 DOF":
[1] 0.3770256

> lapply( deltas, function(x) var(x,na.method="omit") )
$"1 DOF":
[1] 0.8556084

$"3 DOF":
[1] 0.1669712

$"5 DOF":
[1] 0.09127696

$"7 DOF":
[1] 0.06862063
```

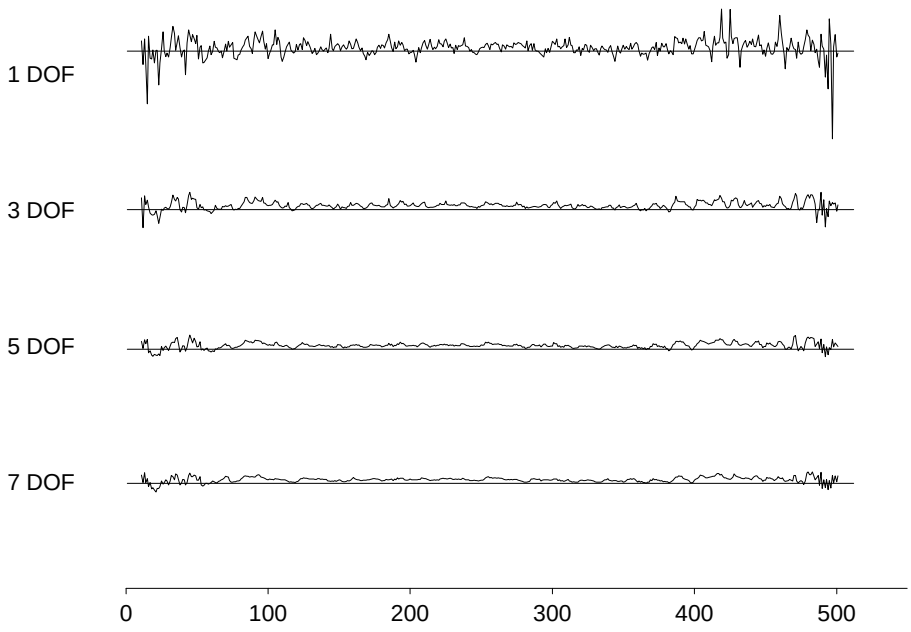

FIGURE 5.4: Localized instantaneous least squares estimates of the fdp045 sequence using (from top to bottom) $1, 3, 5$, and 7 chi-square degrees of freedom.

We see that the variance of the estimates goes down with an increase in DOF order, but at the cost of introducing more bias in the mean of the estimates. A similar effort using a maximum likelihood estimator shows the sample mean and variance trends, but the bias between the true and estimated values of δ is less than that of the LSE:

```
> deltas <- lapply(dofs,function(i)
+ wavFDPTime(fdp045, lev=1:6, est="mle", dof=i )$delta )
> names( deltas ) <- paste(2*(dofs)+1,"DOF")
> lapply( deltas, function(x) mean(x,na.rm=T) )
$"1 DOF":
[1] 0.3729772

$"3 DOF":
[1] 0.4416434

$"5 DOF":
[1] 0.4325607

$"7 DOF":
[1] 0.4151354

> lapply( deltas, function(x) var(x,na.method="omit") )
$"1 DOF":
[1] 1.022432

$"3 DOF":
[1] 0.2421365

$"5 DOF":
[1] 0.124924

$"7 DOF":
[1] 0.09654555
```

5.2.5 Example: Analysis of Aerothermal Turbulence Data

To demonstrate the potential of the S+Wavelets module, we examine block-dependent and instantaneous estimates of a 7.5 million point aerothermal turbulence sequence known as the aero series (see [CPR01] for details). Due to the large amount of data, a MA(q,r) filters (moving average using windows of length q with an overlap of r points) was used for purposes of display and comparison of results. Figure 5.5 shows the aero series smoothed with a MA$(10000, 0)$ filter. Typical of turbulence data, the aero series exhibits seemingly random fluctuations at various scales and times.

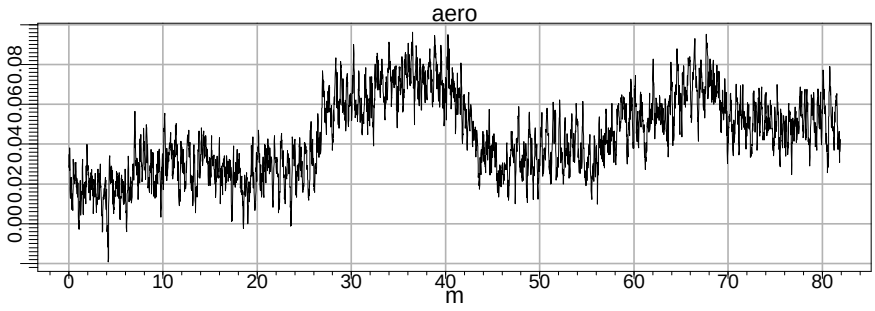

FIGURE 5.5: Aerothermal turbulence series.

The goal of the research was to quantify these random fluctuations as a function of space and scale using block-dependent FD parameter estimators. A secondary goal was to quantify the departure of the FD parameter (δ) estimates from the theoretical Kolmogorov turbulence value of $\delta = 5/6$. Figure 5.6 shows such a departure via (smoothed) WLSE of δ over scales $\tau_6 - \tau_8$ and $\tau_9 - \tau_{11}$ (here the data is plotted as the spectral density function (SDF) power law exponent α which is related to the FD parameter by $\alpha = -2\delta$). Note the apparent wide

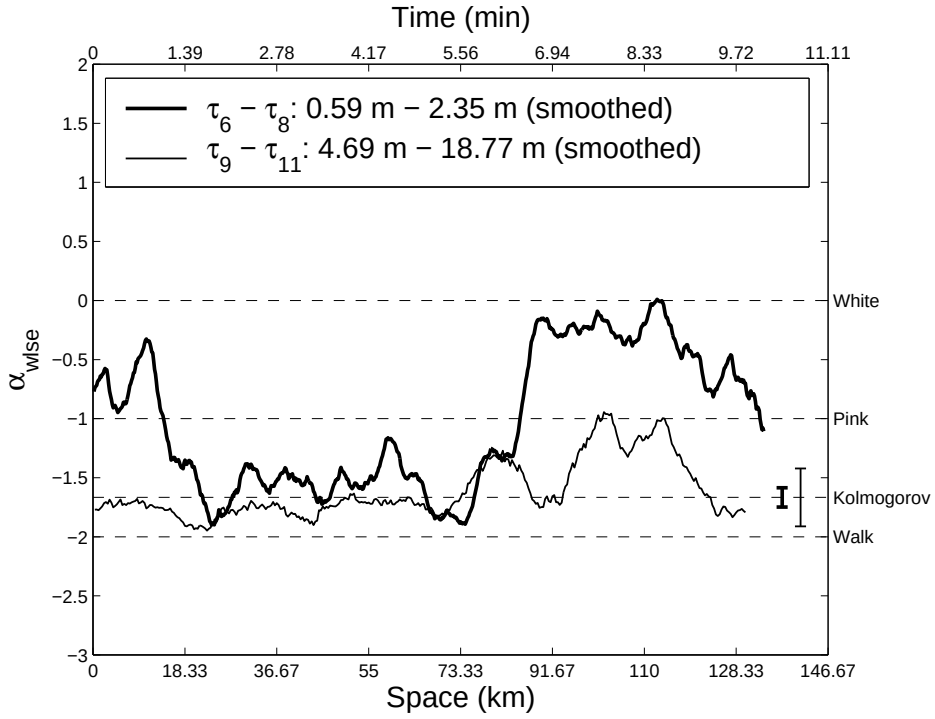

FIGURE 5.6: Block-dependent weighted least squares estimates of the FD parameter (δ) for the aero series. The δ estimates are smooth with a moving average filter and mapped to the SDF exponent α using the relation $\alpha = -2\delta$. The estimates are performed over scales $\tau_6 - \tau_8$ and $\tau_9 - \tau_{11}$.

range of $\hat{\alpha}^{wlse}_{6,8}$ and $\hat{\alpha}^{wlse}_{9,11}$, which roughly span values appropriate for stationary white noise up to nonstationary random walk noise.

The results clearly suggests that a single (Kolmogorov) exponent is not an adequate description of this aerothermal turbulence data as might be incorrectly construed from conventional Fourier-based methods. Conventionally, for example, the SDF power law exponents are estimated directly from an estimate of the SDF for the data. For example, the slope of the SDF on a log-log scale provides a direct estimate of α. Figure 5.7 shows the SDF of the entire aero series, computed by partitioning the aero series into 2^{16} point blocks, forming a spectral estimate for each block and then averaging the spectral estimates together. The average SDF portrays a strong Kolmogorov turbulence slope of $\alpha \approx -5/3$ over many octaves. This global approach masks the fact that there are significant deviations from the $-5/3$ law locally in time and hence does not accurately portray the dynamics of α. We could, of course, track the power law estimate of each block as time unfolds, but we would then need some scheme for partitioning the frequencies into regions over which a single power law is applicable. If we use a partitioning scheme that is essentially the same as what our wavelet methodology yields, the work of McCoy *et al.* [MW96] shows that wavelet-based estimates of α have better mean square error properties than do those based upon the SDF.

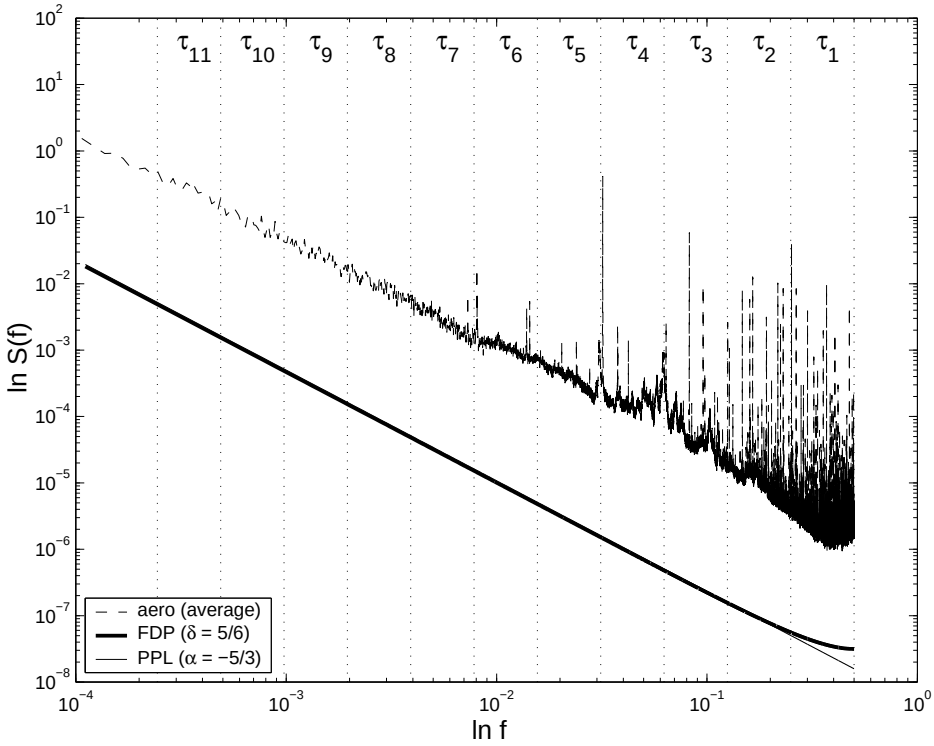

FIGURE 5.7: Averaged estimated SDF for aero series and the theoretical SDFs for an FD process and pure power law (PPL) model of fully developed Kolmogorov turbulence with an infinite inertial range. The FDP and PPL curves are purposefully offset from the average SDF of aero series so that their ln–ln SDF slopes may be easily compared over a broad range of scale. The vertical divisions represent the octaves over which the wavelet coefficients at scale τ_j are nominally associated.

5.3 Time-Varying FD Process Simulation

Time-varying fractionally differenced (TVFD) processes can serve as useful models for certain time series whose statistical properties evolve over time. The spectral density function for a TVFD process obeys a power law whose exponent can be time dependent. In contrast to locally stationary or locally self-similar processes, the power law exponent for a TVFD process is not restricted to certain intervals, which is of practical importance for modeling time series of, e.g., atmospheric turbulence.

It is possible to produce exact realizations of Gaussian TVFD processes (whose parameters fluctuate in an arbitrary manner) using either a modified Cholesky decomposition or a circulant embedding scheme [PC02]. Use of these exact methods ensures that Monte Carlo studies of the statistical properties of estimators for TVFD processes are not adversely influenced by imperfections arising from the use of approximate simulation methods.

Using the wavFDPSimulate function, you can generate finite length realizations of a TVFD process. For example, suppose we wish to emulate the dynamics of the aerothermal turbulence sequence shown in Fig. 5.5 using a multi-linear approximation to the WLSE of δ (Fig. 5.6) for scales $\tau_6 - \tau_8$:

```
> delta <- c(rep(1/4,100),seq(1/4,5/6,len=40),rep(5/6,410),
+ seq(5/6,0,len=100),seq(0,1/2,len=374))
> innov <- rep(1,length(delta))
> tvfd <- wavFDPSimulate( delta=delta, innov = innov )
> par( mfrow=c(2,1) )
> plot( delta, type="l" )
> plot( tvfd, type="l" )
```

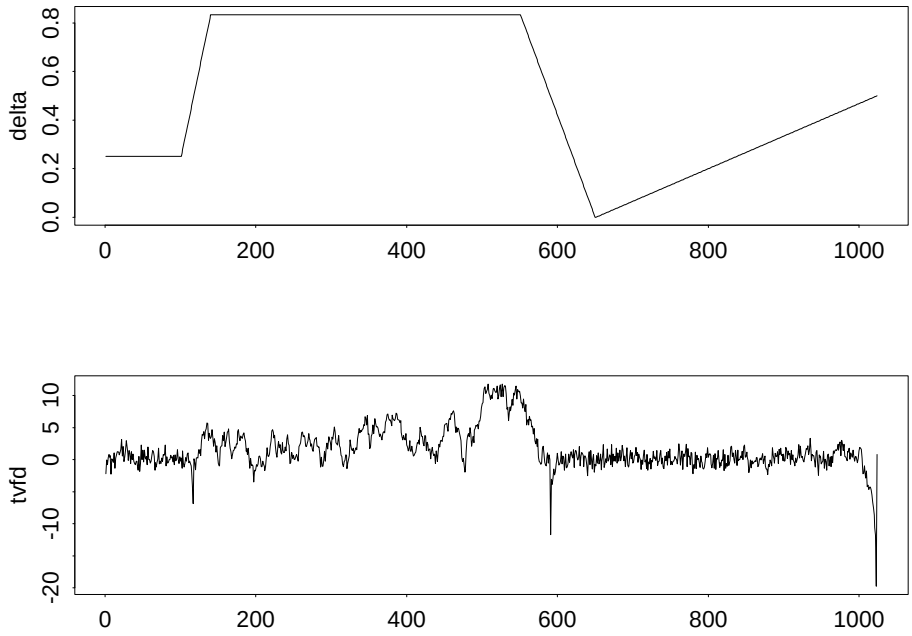

FIGURE 5.8: Simulating aerothermal turbulence dynamics. In the top plot: a crude piecewise linear approximation to the variations observed in the WLSE of aerothermal measurements (Fig. 5.6). In the bottom plot, the corresponding time-varying FD process realization generated by the wavFDPSimulate function. The innovations variance was set to unity.

The piecewise linear approximation of the aerothermal WLSE of δ_t starts out in the stationary region ($\delta < 0.5$), then evolves into a nonstationary region which is intended to mimic Kolmogorov turbulence ($\delta = 5/6$), followed by a return to a stationary region. Although δ_t fluctuates greatly, the resulting TVFD is smooth in the transitional regions of δ. This smooth transitional behavior coupled with the knowledge that the underlying methodology is statistically exact, helps to establish the efficacy of the wavFDPSimulate as a TVFD simulator.

Appendix A
Dataset Reference

Description of S+Wavelets data.

D.table.critical — *Table of critical D-Statistics.* The D-statistic denotes the maximum deviation of sequence from a hypothetical linear cumulative energy trend. The critical D-statistics define the distribution of D for a zero mean Gaussian white noise process. Comparing a D-statistic to the corresponding critical values provides a means of quantitatively rejecting or accepting a linear cumulative energy hypothesis. The D.table.critical table contains critical D-statistics for a variety of sample sizes $1 \leq N \leq 127$ and significances $10\%, 5\%$, and 1%. The table was developed using the average of 3 repititions, where for each repetition $10,000$ Monte Carlo simulations of a zero mean Gaussian white noise process were performed.

aero — *Atmospheric aerothermal turbulence data.* This data is collected by a wire probe extended from an aircraft flying at a constant altitude and speed. The current in the wire is proportional to temperature, so the series of measurements is related to temperature. The analog voltage data was sampled at 12 kHz and fed through a highpass fourth order Butterworth filter with a cutoff frequency around 0.001 Hz to eliminate the DC bias. The aero dataset is thus the AC voltage (proportional to temperature) with a spatial resolution of approximately 2 cm.

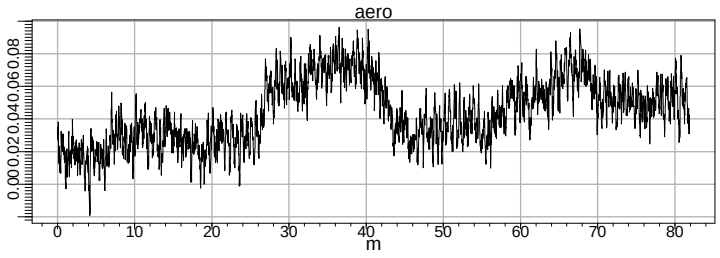

FIGURE A.1: The aero series.

atomclock

Cesium beam atomic clock data. This series represents the difference in time between a cesium beam atomic clock and an official time scale known as UTC(USNO) maintained by the US Naval Observatory, Washington DC. The UTC portion of the USNO series refers to coordinate universal time which is used as an international time standard. Negative values in the resulting (difference) series represents a lag in time relative to the UTC(USNO) standard. The differences in time were recorded at the same time for consecutive days in the 1970s resulting in a sampling interval of 1 day. The amplitude units are expressed in microseconds.

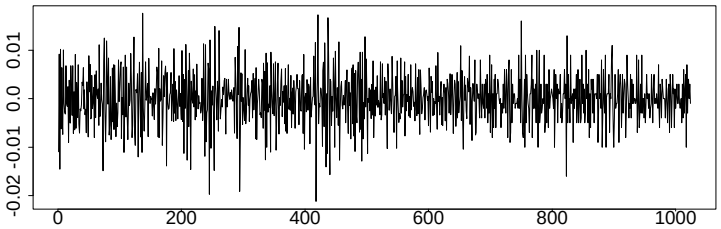

FIGURE A.2: The first difference of the atomclock series.

ecg

Electrocardiogram Data from the Human Heart. This 2048 point ECG data set represents approximately 15 beats of a normal cardiac rhythm for humans. This sequence is sampled at 180 Hz and is in units of millivolts. This series was collected and supplied by Dr. G. Bardy M.D. and Dr. P. Reinhall of the University of Washington.

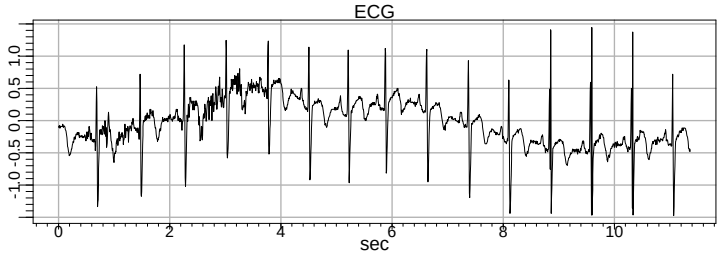

FIGURE A.3: The ecg series.

fdp045

Fractionally Differenced Process Data. These data represent a 512 point realization of a fractionally differenced (FD) process with an FD parameter of 0.45 and FD innovations variance.

nile

Yearly Nile river level minima. These data represent the measurements of the minimum yearly water level of the Nile over the years 622 to 1284 as recorded at the Roda gauge near Cairo.

ocean

Vertical shear ocean data. This data is collected by a instrument dropped over the side of a ship in the ocean. As the instruments descends vertically in the water it collects information as a function of depth. Specifically, it is a record of the horizontal (shear) velocity of the water measured in 0.1 increments starting from a depth of 489.5 meters and ending at 899.0 meters. This series was collected and supplied by Mike Gregg, Applied Physics Laboratory, University of Washington.

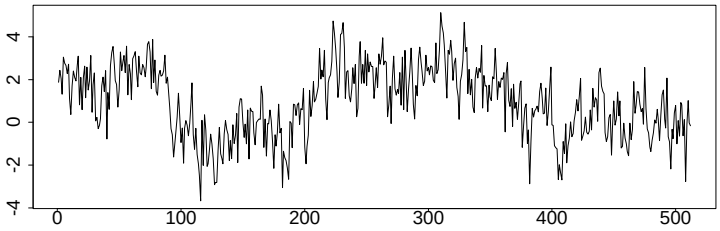

FIGURE A.4: The fdp045 series.

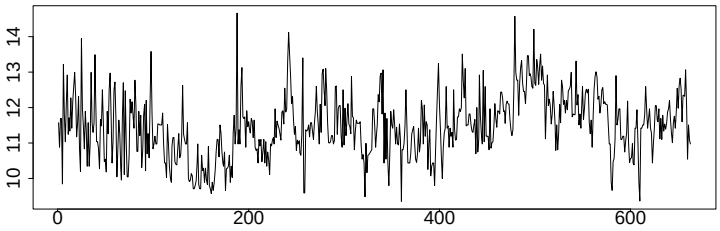

FIGURE A.5: The nile series.

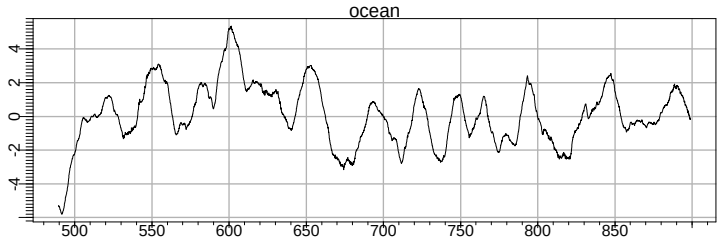

FIGURE A.6: The ocean series.

smallts1 *A test time series.* Used for various transform tests in Percival and Walden's, "Wavelet Methods for Time Series Analysis" [PW00].

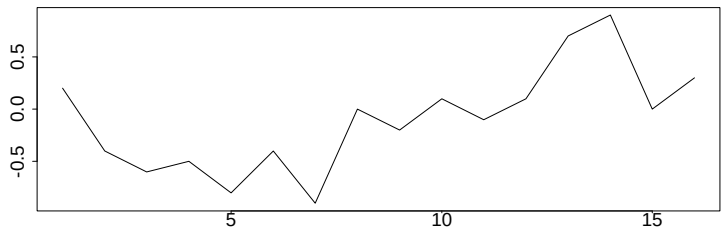

FIGURE A.7: The smallts1 series.

smallts2 *A test time series.* Used for various transform tests in Percival and Walden's, "Wavelet Methods for Time Series Analysis" [PW00].

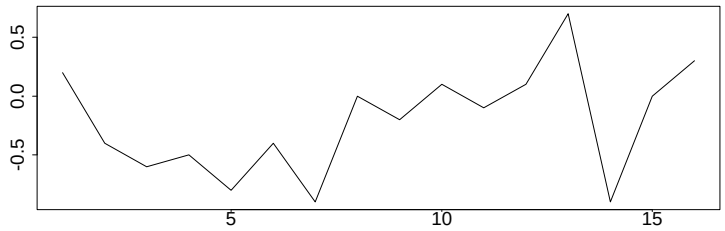

FIGURE A.8: The smallts2 series.

solarmag *Solar magnetic field data.* This data was measured by the Ulysses spacecraft as it traversed heliographic latitudes 43 deg S (day 338, 1993) to 62 deg S (day 144, 1994) with a corresponding heliocentric range of approximately 4 to 3 astronomical units, respectively (an astronomical unit is defined as the mean distance between the Earth and the Sun). The data are daily averages of the magnetic field strength measured in nanoteslas (nT).

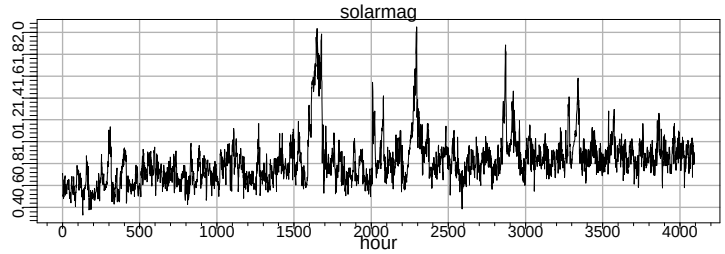

FIGURE A.9: The solarmag series.

subtidal *Subtidal sea levels in the Pacific Ocean.* The time series is a low-passed version of hourly water level observations (subsampled from 6-min data) made by the NOAA tide gauge at

Crescent City, California (41 deg 44.7 min North, 124 deg 11.0 min West) from 1/6/1980 - 12/26/1991. The water level observations were made inside the stilling well of the tide gauge to eliminate high-frequency wind waves and swell. After the data were demeaned, a Kaiser low-pass filter was used to remove the tides from the hourly observations. The series was then subsampled every 0.5 days. While the original water levels were recorded in feet, they were converted to centimeters during processing. Since these observations were collected and processed by US Federal employees supported by Federal funds, the series may be used freely without regard to copyright restrictions. Questions concerning the series should be send to Hal Mofjeld (mofjeld@pmel.noaa.gov).

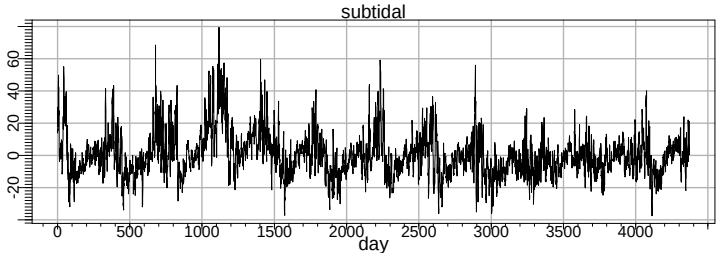

FIGURE A.10: The subtidal series.

posy *Computer generated sihlouette image of a twelve peta1 flower.* Used as a test image for the DTWT.

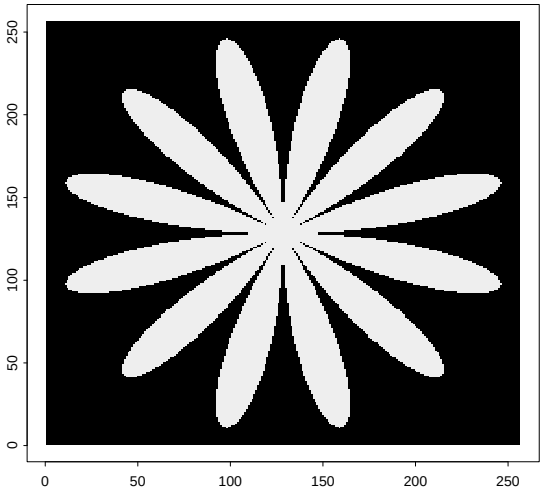

FIGURE A.11: The posy image.

Appendix B
Function Reference

The following pages describe the functions in alphabetical order.

D.table Critical D-statistic table generation.

Usage

```
D.table(n.sample = c(127, 130), significance = c(0.1, 0.05, 0.01),
lookup = T, n.realization = 10000, n.repetition = 3,
tolerance = 1e-6)
```

Description

The D-statistic denotes the maximum deviation of sequence from a hypothetical linear cumulative energy trend. The critical D-statistics define the distribution of D for a zero mean Gaussian white noise process. Comparing the sequence D-statistic to the corresponding critical values provides a means of quantitatively rejecting or accepting the linear cumulative energy hypothesis. The table is generated for an ensemble of distribution probabilities and sample sizes.

Optional Arguments

n.sample — A vector of integers denoting the sample sizes for which critical D-statistics are created. Default: c(127,130).

significance — A numeric vector of real values in the interval (0,1). The significance is the fraction of times that the linear cumulative energy hypothesis is incorrectly rejected. It is equal to the difference of the distribution probability (p) and unity. Default: c(0.1, 0.05, 0.01).

lookup — A logical flag for accessing precalculated critical D-statistics. The critical D-statistics are calculated for a variety of sample sizes and significances. If lookup is TRUE (recommended), this table is accessed. The table is stored as the matrix object D.table.critical on the S+Wavelets workspace. Missing table values are calculated using the input arguments: n.realization, n.repetition, and tolerance. Default: TRUE.

n.realization — An integer specifying the number of realizations to generate in a Monte Carlo simulation for calculating the D-statistic(s). This parameter is used either when lookup is FALSE,

or when lookup is TRUE and the table is missing values corresponding to the specified significances. Default: 10000.

n.repetition — An integer specifying the number of Monte Carlo simulations to perform. This parameter coordinates with the n.realization parameter. Default: 3.

tolerance — A numeric real scalar that specifies the amplitude threshold to use in estimating critical D-statistic(s) via the Inclan-Tiao approximation. Setting this parameter to a higher value results in a lesser number of summation terms at the expense of obtaining a less accurate approximation. Default: 1e-6.

Value

A matrix containing the critical D-statistics corresponding to the supplied sample sizes and significances.

Details

A precalculated critical D-statistics object (D.table.critical) exists on the S+Wavelets workspace and was built for a variety of sample sizes and significances using 3 repetitions and 10000 realizations/repetition. This D.table function should be used in cases where specific D-statistics are missing from D.table.critical. Note: the results of the D.table value should not be returned to a variable named D.table.critical as it will override the precalculated table that exists on the S+Wavelets workspace.

An Inclan-Tiao approximation of critical D-statistics is used for sample sizes n.sample ≥ 128 while a Monte Carlo technique is used for n.sample < 128. For the Monte Carlo technique, the D-statistic for a Gaussian white noise sequence of length n.sample is calculated. This process is repeated n.realization times, forming a distribution of the D-statistic. The critical values corresponding to the significances are calculated a total of n.repetition times, and averaged to form an approximation to the D-statistic(s).

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

wavVarianceHomogeneity, D.table.critical.

Examples

```
D.lookup <- D.table(significance = c(10,5,1)/100,
n.realization = 100, n.sample = 125:130, lookup = F)
```

---

as.signalSeries

Converts various time series to an object of class signalSeries.

Usage

```
as.signalSeries(data, position = list( from = 1, by = 1, units = ""),
from = NULL, by = NULL, to = NULL, length.out = NULL,
units = "", title.data = "", documentation = "")
```

Description

Converts numeric data to an object of class signalSeries containing one dimensional data. The input data is assumed to be uniformly sampled.

Required Arguments

data — A numeric vector, matrix or an object of class ts (uniform sampling assumed). If a matrix, the longest first row or column is extracted.

Optional Arguments

position — A list containing the arguments from, by and to which describe the position(s) of the input data. All position arguments need not be specified as missing members will be filled in by their default values. Default: list(from = 1, by = 1, units = NULL).

from — A list containing the arguments from, by and to which describe the position(s) of the input data. All position arguments need not be specified as missing members will be filled in by their default values. Default: NULL.

to — A numeric containing the end point in data from which the values should be extracted. This parameter must coordinate with the position arguments and can be used in combination with the by and to arguments. The length.out argument should not be specified if both the from and to arguments are specified. Default: NULL.

by — a numeric containing the sampling rate at which the values in data should be extracted. This parameter must coordinate with the position arguments and can be used in combination with the by, to, or length.out arguments. This argument is not the same as the position$by argument which denotes the sampling rate of the original data. Default: NULL.

length.out — an integer containing the maximum number of values to extract from data. Because data is a finite length sequence, the actual number of values returned may be less than that specified by this argument depending upon the conditions imposed by the from and by arguments. The length.out argument should not be specified if both the from and to arguments are specified. Default: NULL.

title.data — a string representing the name of the input data. Default: NULL.

documentation — a string used to describe the input data. Default: NULL.

units — A string denoting the units of the time series. Default: empty string.

na.rm — A logical flag used to indicate if NaN values should be removed from the input. Default: TRUE.

Value

an object of class signalSeries.

See Also

signalSeries.

Examples

```
## convert an explicitly developed numeric vector
x <- 1:10
as.signalSeries( x )

## now impose hypothetical positions on the data
as.signalSeries( x, pos=list(from = 0.3, by = 0.1) )

## extract the values from position 0.5 onward
as.signalSeries( x, pos=list(from = 0.3, by = 0.1), from = 0.5 )

## extract the values from position 0.5 onward, but
## keep only the first 3 values of the extraction
as.signalSeries( x, pos=list(from = 0.3, by = 0.1), from = 0.5, length = 3 )

## extract the values from position 0.5 onward
## and skip every other point (sample the data
## at 0.2 position intervals)
as.signalSeries( x, pos=list(from = 0.3, by = 0.1), from = 0.5, by = 0.2 )
```

```
## simply return the first 4 values, and supply a title and
## some documentation comments to the data
as.signalSeries( x, length = 4, title = "Faux Data", doc = "An example" )
```

---

wavBoundary — Wavelet transform boundary coefficent identification.

Usage

```
wavBoundary(x)
```

Description

A wavelet transform boundary coefficient is one subject to circular filter operations (or other boundary treatments). Conversely, the interior transform coefficients are those that are not affected by the imposed boundary treatment. The wavBoundary function separates the boundary coefficients from the interior wavelet transform coefficients.

Required Arguments

x	A DWT or MODWT transform object with class WaveletTransform.

Value

An object of class WaveletBoundary.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.
2. I. Daubechies, *Orthonormal Bases of Compactly Supported Wavelets*, Communications on Pure and, Applied Mathematics, 41, 909–96.

See Also

wavIndex, wavDWT, wavMODWT, wavShift.

Examples

```
## calculate the MODWT of the sunspots series
y <- wavMODWT(sunspots)

## identify the boundary coefficients
x <- wavBoundary(y)

## plot the results
plot(x)

## obtain a summary
summary(x)
```

---

wavConfidenceLimits — Confidence intervals for the unbiased and blocked averaged discrete wavelet variance estimates.

Usage

```
wavConfidenceLimits( variance, edof )
```

Description

Given $\hat{\nu}_X^2(\tau_j)$ are the time independent unbiased wavelet variance estimates at scales $\tau_j \equiv 2^{j-1}$ where j are the decomposition levels, the approximate $100(1-2p)\%$ confidence interval is given by

$$\left[\frac{n\hat{\nu}_X^2(\tau_j)}{Q_n(1-p)}, \frac{n\hat{\nu}_X^2(\tau_j)}{Q_n(p)} \right]$$

where $Q_n(p)$ is the $p \times 100$ percentage point for a χ_n^2 distribution.

Required Arguments

variance — A vector containing the block-averaged unbiased wavelet variance estimates.

edof — A vector containing the equivalent degrees of freedom estimates. See wavEDOF for details.

Optional Arguments

probability — The probability desired for the confidence intervals. Supported probabilities are 0.005, .025, .05, .95, .975, and .995. Default: 0.95.

Value

A matrix of size $2 \times J$. The rows of the matrix contain (in order) the low and high confidence interval limits for levels $1, \ldots, J$.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

wavVariance, wavEDOF

Examples

```
## first calculate the EDOF for the ocean series
edof <- wavEDOF( ocean )

## calculate the 95% confidence
## intervals for EDOF mode 1
wavConfidenceLimits( edof$variance.unbiased, edof$EDOF1 )
```

---

wavCovariance — Discrete wavelet covariance estimation.

Usage

```
wavCovariance( x, y, wavelet = "s8", n.levels = 4, lag = c(10,20,30,40) )
```

Description

Calculates the time (in)dependent, (un)biased, (un)lagged discrete wavelet covariance of two time series using either DWT or MODWT wavelet transform coefficients.

Required Arguments

x — A vector containing a uniformly-sampled real-valued time series.

y — A vector containing a uniformly-sampled real-valued time series.

Optional Arguments

transform — A character string denoting the type of wavelet transform: “modwt” or “dwt”. Default: “modwt”.

wavelet — A character string denoting the filter type. See wavDaubechies for details. Default: “s8”.

n.levels — The number of decomposition levels. Default: the maximum level at which there exists at least one interior wavelet coefficient.

lag — an integer denoting the time lag to be used on the series (represented by) x or y when calculating the covariance estimates at a given scale. The lag is used to study the covariance between wavelet subband processes whose events are assumed to be correlated at different times. The wavelet coefficients for y are circularly shifted by lag and the product of these shifted coefficients with the wavelet coefficients of x at the corresponding scale are used to form the wavelet covariance estimates. If lag is a vector, the j'th level y wavelet coefficients are lagged by the j'th element of lag. If the number of elements in lag is less than the number of wavelet decomposition scales, the highest scale lag is repeated for the remaining scales. Finally, lag may contain positive or negative integers, with negative lags representing an advance (or circular permutation to the left) of the wavelet coefficients. The lag used. Default: 0.

sampling.interval — The sampling interval of the time series. Default: 1.

Value

An object of class WaveletCovariance.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

wavVariance.

Examples

```
## create second data set, with the first
## being the sunspots series
revsun <- rev(sunspots)

## calculate the wavelet covariance of the sunspots
## series with a reversed sunspots series
xcov <- wavCovariance(sunspots, revsun)

## plot the results
plot(xcov)

## plot the wavelet covariance coefficients
## shifted for approximate zero phase to align
## them with events in the original time series
plot(xcov, shift = T)

## perform a lagged wavelet covariance estimation
## using the same lag for all scales
xcov <- wavCovariance(sunspots, revsun, lag = 20)

## perform a lagged wavelet covariance estimation
## using different lags for for different scales
xcov <- wavCovariance(sunspots, revsun,
+ lag = c(20, 30, 40))
print(xcov$lag)
```

---

wavDTWT.2d — The two-dimensional dual tree discrete wavelet transform (DTWT2d).

Usage

```
wavDTWT.2d( x, n.levels = 3, biorthogonal = "nearsyma", qshift = "a" )
```

Description

The two-dimensional version of the dual-tree complex wavelet transform (DTWT) applies the DTWT to each column and to each row of the input matrix (e.g., an image).

Required Arguments

x	A numeric matrix having an even number of rows and columns.

Optional Arguments

n.levels	The number of decomposition levels. Default: min(logb(dim(x), base = 2)).
biorthogonal	A character string denoting the level one biorthogonal filter type. Supported types are “antonini”, “legall”, “nearsyma”, and “nearsymb”. Default: “nearsyma”.
qshift	A character string denoting the Q-Shift filter type used for levels greater than one. Supported types are “a”, “b”, “c”, and “d”. Default: “a”.

Value

An object of class WaveletDualTree2d.

References

1. N. G. Kingsbury, *The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement*, Proc. EUSIPCO 98, Rhodes, Sept. 1998.
2. N. G. Kingsbury, *Image processing with complex wavelets*, Phil. Trans. Royal Society London A, Sept. 1999, pp. 2543–2560.
3. N. G. Kingsbury, *A dual-tree complex wavelet transform with improved orthogonality and symmetry properties*, Proc. IEEE Conf. on Image Processing, Vancouver, Sept. 11–13, 2000, Paper 1429.
4. I. Daubechies, Orthonormal Bases of Compactly Supported Wavelets, Communications on Pure and, Applied Mathematics, 41, 909–96.

See Also

wavDTWTFilters, reconstruct, wavDTWT.

Examples

```
## create an image
img <- make.image("flower", 128)

## calculate DTWT coefficients of an image x.img
## through level 3
y <- wavDTWT.2d(img, n.levels = 3, bior = "nearsymb",
+ qshift = "b", title.data = "Flower" )

## display results
print(y)

## plot complex magnitudes (decibels) of wavelet
## coefficients for all six angular orientations
## at level 1
plot(y, level=1)

## same plot, but showing actual magnitudes, not decibels
plot(y, decibels=F, level = 1)
```

```
## same plot, but with all magnitudes less than 60
## percent of maximum values set to zero
plot(y, decibels=F, level = 1, threshold=0.60)

## same type of plot, but showing an enlarged view of
## only the +15 deg plot
plot(y, decibels=F, level = 1, angle=15, threshold=0.60)

## same as above, but showing both the +15 deg and
## -15 deg plots
plot(y, decibels=F, level = 1, angle=c(15,-15),
+ threshold=0.60)

## reconstruct image from its 2D DTWT
img.recon <- reconstruct(y)

## verify reconstruction
vecnorm(img.recon - img)/vecnorm(img)
```

---

wavDTWTFilters — Dual tree discrete wavelet transform filter generation.

Usage

```
wavDTWTFilters( bio = "nearsyma", qshift = "a" )
```

Description

Calculates the wavelet and scaling filter coefficients for a dual tree wavelet transform (DTWT). There are two sets that are specified: (1) a biorthogonal set used only for level one of the DTWT and (2) an even-length, quarter-sample shift (Q-Shift) filter set used for decomposition levels greater than one.

Optional Arguments

biorthogonal — A character string denoting the level one biorthogonal filter type. Supported types are “antonini”, “legall”, “nearsyma”, and “nearsymb”. Default: “nearsyma”.

qshift — A character string denoting the Q-Shift filter type used for levels greater than one. Supported types are “a”, “b”, “c”, and “d”. Default: “a”.

Value

An object of class WaveletKingsbury.

References

1. N. G. Kingsbury, *The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement*, Proc. EUSIPCO 98, Rhodes, Sept. 1998.
2. N. G. Kingsbury, *Image processing with complex wavelets*, Phil. Trans. Royal Society London A, Sept. 1999, pp. 2543–2560.
3. N. G. Kingsbury, *A dual-tree complex wavelet transform with improved orthogonality and symmetry properties*, Proc. IEEE Conf. on Image Processing, Vancouver, Sept. 11–13, 2000, Paper 1429.
4. I. Daubechies, *Orthonormal Bases of Compactly Supported Wavelets*, Communications on Pure and, Applied Mathematics, 41, 909–96.

See Also

wavDTWT, wavDTWT.2d.

Examples

```
## calculate DTWT filters
temp <- wavDTWTFilters(bior = "nearsyma", qshift = "b")

## plot impulse responses of filters used in the
## forward DTWT
plot(temp)

## plot impulse responses of filters used in the
## inverse DTWT
plot(temp, forward = F)

## plot the squared gain response of the forward filters
plot(temp, freq = T)

## summarize the filter set
summary(temp)
```

---

wavDTWT The dual tree discrete wavelet transform (DTWT).

Usage

```
wavDTWT( x, n.levels = 3, biorthogonal = "nearsyma", qshift = "a" )
```

Description

The Dual-Tree Complex Wavelet Transform (DTWT) is a recent development in wavelet theory due to Professor Nick Kingsbury at Cambridge University. The research that resulted in the DTWT was motivated by the quest for a transform that would have several desirable properties:

- Invertibility ("perfect" reconstruction)
- Shift invariance
- Directional selectivity in the two-dimensional transform
- Numerical stability

Kingsbury's transform retains perfect reconstruction, achieves a good (but approximate) degree of shift invariance, has good directional selectivity, and is numerically stable. These features do come with a modest degree of redundancy (a factor of two in one dimension, or four in two dimensions), but this is considerably less than the redundancy of the non-decimated wavelet transform.

The transform is evaluated via two trees of filters that are designed to have the character of "real" and "imaginary" parts of an overall complex wavelet transform. We emphasize, however, that this is accomplished using purely real filters. Here, the "complex" nature of the result must be understood in the analytic signal sense. Thus, the mother wavelets associated with each tree are discrete versions of Hilbert transforms of one another. Equivalently, the lowpass filters of one tree interpolate midway between the lowpass filters of the second tree. While no complex numbers are involved in the evaluation of the transforms themselves, the results obtained from the two trees, "Tree A" and "Tree B," are interpreted as (Tree A) + j (Tree B).

The wavDTWT function calculates the wavelet coefficients and scaling coefficients for a given filter type. Two sets of filters must be specified: (1) a biorthogonal set used only

for level one of the DTWT and (2) an even-length, quarter-sample shift (Q-Shift) filter set used for decomposition levels greater than one.

Required Arguments

x — A vector containing a uniformly-sampled real-valued time series.

Optional Arguments

n.levels — The number of decomposition levels. Default: the maximum level at which there exists at least one interior wavelet coefficient.

biorthogonal — A character string denoting the level one biorthogonal filter type. Supported types are “antonini”, “legall”, “nearsyma”, and “nearsymb”. Default: “nearsyma”.

qshift — A character string denoting the Q-Shift filter type used for levels greater than one. Supported types are “a”, “b”, “c”, and “d”. Default: “a”.

Value

An object of class WaveletDualTree.

References

1. N. G. Kingsbury, *The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement*, Proc. EUSIPCO 98, Rhodes, Sept. 1998.
2. N. G. Kingsbury, *Image processing with complex wavelets*, Phil. Trans. Royal Society London A, Sept. 1999, pp. 2543–2560.
3. N. G. Kingsbury, *A dual-tree complex wavelet transform with improved orthogonality and symmetry properties*, Proc. IEEE Conf. on Image Processing, Vancouver, Sept. 11–13, 2000, Paper 1429.
4. I. Daubechies, Orthonormal Bases of Compactly Supported Wavelets, Communications on Pure and, Applied Mathematics, 41, 909–96.

See Also

wavDTWTFilters, reconstruct, wavDTWT.2d.

Examples

```
## create data vector
x <- make.signal("linchirp", n = 128)

## calculate DTWT coefficients of data vector
## x through level 3
xwav <- wavDTWT(x, n.levels = 3, bior = "nearsymb", qshift = "b" )

## display results
print(xwav)

## plot wavelet coefficients for all levels for both
## tree A and tree B
plot(xwav)

## plot complex magnitudes of wavelet coefficients
## |treeA + j*treeB|
plot(xwav, mod = T)

## display wavelet detail coefficients, but not the
## scaling coefficients
plot(xwav, plot.scaling = F)
```

```
## extract and plot wavelet crystals from Levels 1 and
## 3 only
plot(xwav, levels = c(1,3))

## reconstruct x from its DTWT
x.recon <- reconstruct(xwav)

## evaluate reconstruction
vecnorm(x.recon - x)/vecnorm(x)
```

---

wavDWPT The discrete wavelet packet transform (DWPT).

Usage

```
wavDWPT( x, wavelet = "s8", n.levels = 3 )
```

Description

Given j,n,t are the decomposition level, oscillation index, and time index, respectively, the DWPT is given by

$$W_{j,n,t} = \sum_{l=0}^{L-1} u_{n,l} W_{j-1,\lfloor n/2 \rfloor, 2t+1-l \bmod N_{j-1}}, \qquad t = 0,\ldots,N_j - 1.$$

where $N_j \equiv N/2^j$ and $\lfloor \cdot \rfloor$ denotes the integer part. The variable L is the length of the filters defined by

$$u_{n,l} \equiv \begin{cases} g_l, & \text{if } n \bmod 4 = 0 \text{ or } 3; \\ h_l, & \text{if } n \bmod 4 = 1 \text{ or } 2, \end{cases}$$

and g and h are the scaling filter and wavelet filter, respectively. Each filter is of length L. By definition, $W_{0,0,t} \equiv X_t$ where $\{X_t\}$ is the original time series.

Required Arguments

x	A vector containing a uniformly-sampled real-valued time series.

Optional Arguments

wavelet	A character string denoting the filter type. See wavDaubechies for details. Default: “s8”.
n.levels	The number of decomposition levels. Default: the maximum level at which there exists at least one interior wavelet coefficient.

Value

An object of class WaveletPacket.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

reconstruct, wavDetail, wavMODWPT, wavDWT, wavDaubechies, wavMaxLevel, wavShift, wavZeroPhase.

Examples

```
## calculate the DWPT of an electrocardiogram
## sequence out to 3 levels using Daubechies least
## asymmetric 8-tap filter set
result <- wavDWPT( ecg, wavelet = "s8", n.levels = 3 )

## plot the transform
plot( result )

## summarize the transform
summary( result )
```

---

wavDWT The discrete wavelet transform (DWT).

Usage

```
wavDWT( x, wavelet = "s8", n.levels = 3 )
```

Description

The discrete wavelet transform using convolution style filtering and periodic extension.

Let j,t be the decomposition level, and time index, respectively, and $s_{0,t} = X_{t=0}^{N-1}$ where X_t is a real-valued uniformly-sampled time series. The j^{th} level DWT wavelet coefficients ($d_{j,t}$) and scaling coefficients ($s_{j,t}$) are defined as

$$\begin{aligned} d_{j,t} &\equiv \sum_{l=0}^{L-1} h_l s_{j-1,2t+1-l \bmod N_{j-1}}, \quad t = 0,\ldots,N_j-1 \\ s_{j,t} &\equiv \sum_{l=0}^{L-1} g_l s_{j-1,2t+1-l \bmod N_{j-1}}, \quad t = 0,\ldots,N_j-1 \end{aligned}$$

for $j = 1,\ldots,J$ where $\{h_l\}$ and $\{g_l\}$ are the j^{th} level wavelet and scaling filter, respectively, and $N_j \equiv N/2^j$. The DWT is a collection of all wavelet coefficients and the scaling coefficients at the last level: $\mathbf{d_1}, \mathbf{d_2}, \ldots, \mathbf{d_J}, \mathbf{s_J}$ where $\mathbf{d_j}$ and $\mathbf{s_j}$ denote a collection of wavelet and scaling coefficients, respectively, at level j.

Required Arguments

x — A vector containing a uniformly-sampled real-valued time series.

Optional Arguments

wavelet — A character string denoting the filter type. See wavDaubechies for details. Default: “s8”.

n.levels — The number of decomposition levels. Default: the maximum level at which there exists at least one interior wavelet coefficient.

Value

An object of class WaveletTransform.

Details

This DWT imposes an ad hoc storage sytem for odd length scaling coefficient crystals: if the length of a scaling coefficient crystal is odd, the last coefficient is “stored” in the extra crystal. During reconstruction, any extra scaling coefficients are returned to their proper location. Such as system imposes no spurious energy in the transform coefficients at the cost of a little bookkeeping.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

reconstruct, wavDaubechies, wavDetail, wavDWPT, wavMODWT, wavMODWPT, wavBoundary, wavMaxLevel, wavIndex, wavShift, wavZeroPhase.

Examples

```
## calculate the DWT of an electrocardiogram
## sequence out to 4 levels
result <- wavDWT( ecg, wavelet = "s8", n.levels = 4 )

## plot the transform
plot( result )

## plot summary
eda.plot( result )

## summarize the transform
summary( result )
```

---

wavDaubechies — Daubechies wavelet and scaling filters.

Usage

```
wavDaubechies( wavelet = "d6", normalize = F )
```

Description

Ingrid Daubechies, a noted pioneer in wavelet theory, has established a number of wavelet filter types, each with different mathematical properties. This function calculates the wavelet and scaling coefficients for a given filter type. The wavelet coefficients, h_k for $k = 0, \ldots, L-1$ where L is the filter length, are related to the scaling coefficients through the quadrature mirror filter (QMF) relation

$$h_k = (-1)^{k-L} g_{L-1-k}.$$

Optional Arguments

wavelet — A character string denoting the filter type. Supported types include:

- EXTREMAL PHASE (daublet): "haar", "d2", "d4", "d6", "d8", "d10", "d12", "d14", "d16", "d18", "d20"
- LEAST ASYMMETRIC (symmlet): "s2","s4", "s6", "s8", "s10", "s12", "s14", "s16", "s18", "s20"
- BEST LOCALIZED: "l2","l4", "l6", "l14", "l18", "l20"
- COIFLET: "c6", "c12", "c18", "c24", "c30"

Default: "s8".

normalize — A logical value. If TRUE, the filters are normalized by dividing each filter coefficient by the $\sqrt{2}$ (useful for maximum overlap wavelet transforms). If FALSE, no normalization is used. Default: TRUE.

Value

An object of class WaveletDaubechies.

Details

Only relevant for Daubechies filter types. Inconsistent ordering of the coefficients in Daubechies' book was recognized and corrected by Percival (see references). The "correct" order is given here.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.
2. I. Daubechies, *Orthonormal Bases of Compactly Supported Wavelets*, Communications on Pure and, Applied Mathematics, 41, 909–96.

See Also

wavZeroPhase, wavDWT, wavMODWT, wavDWPT, wavMODWPT, wavGain.

Examples

```
## obtain Daubechies least asymmetric 8-tap filter set
filters <- wavDaubechies( "s8", normalize = T )

## display filter information
print( filters )

## plot the impulse responses
plot( filters, type = "time" )

## plot the gain function
plot( filters, type = "gain" )

## access the filter data
wavelet <- filters$wavelet
scaling <- filters$scaling
```

---

wavDetail — Calculate the detail sequences for wavelet transform crystals.

Usage

```
wavDetail( x, level = 3, osc = 2 )
```

Description

Let $\mathbf{W}_{j,n}$ be a discrete wavelet packet crystal where j is the decomposition level and n is the oscillation index. The detail sequence $\mathcal{D}_{j,n}$ is formed (essentially) by reconstructing the transform after zeroing out all other crystals except $\mathbf{W}_{j,n}$. Since the DWT and MODWT are subsets of the DWPT and MODWPT, respectively, their crystals can also be converted to detail sequences. The wavDetail function calculates the details for a DWT, MODWT, DWPT, or MODWPT in an optimized way.

Required Arguments

x — An object of class WaveletTransform or WaveletPacket.

Optional Arguments

level — An integer (vector) containing the decomposition level(s) corresponding to the crystal(s) to be decomposed. Default: If the input is of class WaveletTransform, then the default is to return the details at all levels of the transform, i.e. a full multiresolution decomposition. If the input is of class WaveletPacket, then the default is to return the details of the last (highest) decomposition level.

osc — An integer (vector) containing the oscillation indices corresponding to the crystal(s) to be decomposed. Default: the default values are coordinated with that of the level argument.

Value

An object of class decompose.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

reconstruct, mrd, mra, wavDWT, wavDWPT, wavMODWT, wavMODWPT.

Examples

```
## calculate various wavelet transforms of the
## first difference of the atomic clock sequence
x <- diff( as.vector( atomclock ) )
x.dwt <- wavDWT(x, n.levels = 3)
x.dwpt <- wavDWPT(x, n.levels = 3)
x.modwt <- wavMODWT(x, n.levels = 3)
x.modwpt <- wavMODWPT(x, n.levels = 3)

## calculate the wavelet details for all crystals
## of the DWT
## and MODWT
wavDetail( x.dwt )
wavDetail( x.modwt )

## calculate the wavelet details for the last level
## of the DWPT and MODWPT
wavDetail( x.dwpt )
wavDetail( x.modwpt )

## calculate the wavelet details for all crystals in
## the second level of the DWPT
wavDetail( x.dwpt, level = 2 )

## calculate the detail for crystal W(3,2)
## of the MODWPT
wavDetail( x.modwpt, level = 3, osc = 2 )

## calculate the detail for crystal W(3,2)
## of the DWPT
wavDetail( x.dwpt, level = 3, osc = 2 )

## plot the wavelet details for levels
## 1 and 3 of the MODWT
plot( wavDetail( x.modwt, level = c(1,3) ) )
```

---

wavEDOF Equivalent degrees of freedom (EDOF) estimates for a chi-squared distribution.

Usage

```
wavEDOF(x, wavelet = "d6", levels = 3:5 )
```

Description

Let X be a collection of M uncorrelated zero mean Gaussian random variables (RVs). The sum of the squares of the RVs in X will obey a scaled chi-square distribution with M degrees of freedom (DOF). If, however, the original Gaussian RVs are (partially) correlated, we can approximate the distribution of the sum of the squares of (correlated Gaussian) RVs using a scaled chi-square distribution with the DOF adjusted for the correlation in the RVs. These adjusted DOF estimates are known as the *equivalent degrees of freedom* (EDOF). In the context of unbiased wavelet variance analysis, the EDOF can be used to estimate confidence intervals that are guaranteed to have non-negative bounds.

This program calculates three estimates of the EDOF for each level of a discrete wavelet transform. The three modes are described as follows for the MODWT of an an input sequence $\{X_t\}_{t=0}^{N-1}$:

1. **EDOF** $\hat{\eta}_1$ (large sample approximation that requires an SDF estimation via wavelet coefficients):
$$\hat{\eta}_1 = \frac{M_j(\hat{s}_{j,0})^2}{\hat{A}_j},$$
where $\hat{s}_{j,\tau}$ is the autocovariance sequence defined by
$$\hat{s}_{j,\tau} \equiv \frac{1}{M_j}\sum_{t=0}^{M_j-1} \tilde{d}_{j,t}^{(int)}\tilde{d}_{j,t+|\tau|}^{(int)} \;\; 0 \leq |\tau| \leq M_j - 1,$$
$\tilde{d}_{j,t}^{(int)}$ are the M_j j^{th} level interior MODWT wavelet coefficients, and $\hat{A}_j$ is defined as
$$\hat{A}_j \equiv \frac{(\hat{s}_{j,0})}{2} + \sum_{\tau=1}^{M_j-1}(\hat{s}_{j,\tau})^2.$$
2. **EDOF** $\hat{\eta}_2$ (large sample approximation where the SDF is known a priori):
$$\hat{\eta}_2 = \frac{2\left(\sum_{k=1}^{\lfloor (M_j-1)/2 \rfloor} C_j(f_k)\right)^2}{\sum_{k=1}^{\lfloor (M_j-1)/2 \rfloor} C_j^2(f_k)}$$
where $f_k \equiv k/M_j$ and
$$C_j \equiv \widetilde{\mathcal{H}}_j^{(D)}(f)S_X(f)$$
is the product of Daubechies wavelet filter squared gain function and the spectral density function of X_t.
3. **EDOF** $\hat{\eta}_3$ (large sample approximation using a band-pass approximation for the SDF):
$$\hat{\eta}_3 = \max\{M_j/2^j, 1\}.$$

Required Arguments

x	An object of class WaveletTransform or a vector containing a uniformly-sampled real-valued time series.

Optional Arguments

wavelet	A character string denoting the filter type. See wavDaubechies for details. Only used if input x is a time series. Default: "s8".
levels	A vector containing the decomposition levels. Default: when x is of class WaveletTransform then levels = 1:x.n_level, otherwise levels = 1:J where J is the maximum wavelet transform level in which there exists at least one interior wavelet coefficient.
sdf	A vector containing a discretized approximation of the process spectral density function (SDF). The coefficients of this argument should correspond exactly with the normalized Fourier frequencies

$$f = [0, 1/P, 2/P, 3/P, ..., (M-1)/P]$$

where $P = 2 * (M - 1)$ and M is the number of points in the SDF vector. For example, if the sdf vector contains five elements, the corresponding frequencies will be

$f = [0, 1/8, 1/4, 3/8, 1/2]$. This argument is used only for calculating mode 2 EDOF. If the EDOF mode 2 estimates are not desired, send in an empty vector for this argument and the EDOF mode 2 and corresponding confidence intervals will not be calculated. Default: empty vector.

Value

A list containing the EDOF estimates for modes 1, 2 and 3 as well as the block-dependent unbiased wavelet variance estimates.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

wavVariance.

Examples

```
## calculate the EDOF estimates for the ocean series
wavEDOF( ocean )
```

---

wavFDPBand Mid-octave spectral density function (SDF) estimation.

Usage

```
wavFDPBand( levels = 1:5 , delta = 0.25, method = "bandpass" )
```

Description

The wavelet and scaling filters used for wavelet decompositions are nominally associated with approximate bandpass filters. Specifically, at decomposition level j, the wavelet transform coefficients correspond approximately to the normalized frequency range of $[1/2^{j+1}, 1/2^j]$. The square of the wavelet coefficients are used to form the so-called wavelet variance (or wavelet spectrum) which is seen as a regularization of the SDF. Under an assumed FD process, this function estimates the mid-octave SDF values. The estimates are calculated assuming that the wavelet transform filters form perfect (rectangular) passbands. Decomposition levels 1 and 2 are calculated using a second order Taylor series expansion about the mid-octave frequencies while, for levels greater than 2, a small angle approximation ($sin(\pi f) \approx \pi f$) is used to develop a closed form solution which is a function of FD model parameters as well as the mid-octave frequencies.

Optional Arguments

levels A vector containing the decomposition levels. If n.sample ≤ 0, then the levels may be given in any order and levels may be skipped. If, however, n.sample > 0, then levels must contain the values $1, 2, 3, \ldots, J$ where J is the maximum wavelet transform decomposition level. Default: 1:5.

delta The fractional difference parameter. If the scaling band estimates are desired (prompted by setting n.sample > 0), then delta must be less than 0.5 since the formulae for calculating the scaling band estimates implicitly assume stationarity. Default: 0.4.

n.sample The number of samples in the time series. Although no time series is actually passed to the wavFDPBand function, the n.sample argument is used in estimating the mid-octave SDF value over the band of frequencies which are nominally associated with the scaling filter in a wavelet transform. If n.sample > 0, this function will append the estimate of the average SDF value over the scaling band to the wavelet octave estimates. If n.sample ≤ 0, only the wavelet octave estimates are returned. Default: 1024.

scaling — A logical flag. If TRUE, the mid-octave value of the FDP SDF octave corresponding to the scaling coefficients is also returned. Default: TRUE.

method — A character string denoting the method to be used for estimating the average spectral density values at the center frequency (on a log scale) of each DWT octave. The choices are

- **"integration"** Numerical integration of the standard FDP spectral density function.
- **"bandpass"** A small angle approximation to the standard FDP spectral desnity functions for decomposition levels $j \geq 3$ in combination with a Taylor series approximation for levels $j = 1, 2$.

Default: "bandpass".

Value

A vector containing the mid-octave SDF estimates for an FD process.

Details

Estimates are made for the scaling filter band based upon an implicit assumption that the FD process is stationary ($\delta < 1/2$).

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000, 343–54.

See Also

wavFDPBlock, wavFDPTime, wavVariance, wavFDPSDF.

Examples

```
## calculate the mid-octave SDF values for
## an FD process over various wavelet bands
wavFDPBand( levels = c(1, 3, 5:7), delta = 0.45,
+ scaling = F )
```

---

wavFDPBlock — Block-dependent estimation of fractionally differenced (FD) model parameters.

Usage

```
wavFDPBlock( x, levels = 2:6, wavelet = "s8", estimator = "mle",
  boundary = "stationary" )
```

Description

A discrete wavelet transform is used to estimate the FD parameter, the variance of the FD parameter and the innovations variance for a given time series. Both a maximum likelihood estimation (MLE) and weighted least squares estimation (WLSE) scheme are available. If an MLE scheme is chosen, then the DWT is used for its ability to decorrelate long-memory processes. If a WLSE scheme is chosen, then the MODWT is used for its known statistical wavelet variance properties.

Required Arguments

x — A vector containing a uniformly-sampled real-valued time series.

Optional Arguments

levels — A vector containing the decomposition levels. The levels may be given in any order but must be positive. Default: 1:J where J is the maximum wavelet decomposition level at which there exists at least one interior wavelet coefficient.

filter — A character string denoting the filter type. See wavDaubechies for details. Default: "s8".

estimator — A character string denoting the estimation method. Use "wlse" for a weighted least squares estimate and "mle" for a maximum likelihood estimate. Default: "wlse".

boundary — A character string representing the different methods by which boundary wavelet coefficients and scaling coefficients are handled in calculating the FD model parameters. The options for this argument are dependent upon the estimator argument.

For the **MLE** case, the boundary options are:

stationary Under a stationary FD process model, boundary wavelet and scaling coefficients are used in estimating the FD model parameters.

nonstationary A stationary-nonstationary FD model assumes that the governing process may fall into the nonstationary regime and, accordingly, the boundary wavelet coefficients and scaling coefficients are excluded in estimating the FD model parameters.

For the **WLSE** case, the boundary options are:

biased Boundary wavelet coefficients are included in the estimate.

unbiased Boundary wavelet coefficients are excluded in the estimate.

The scaling coefficients are (always) excluded in weighted least squares estimates of FD model parameters. Default: "unbiased".

edof — The mode by which the equivalent degrees of freedom are calculated. This argument is limited to 1,2, or 3 and is used only for the WLSE scheme. See wavEDOF for details. Default: 1.

sdf — A vector containing a discretized approximation of the process spectral density function (SDF). The coefficients of this argument should correspond exactly with the normalized Fourier frequencies

$$f = [0, 1/P, 2/P, 3/P, ..., (M-1)/P]$$

where $P = 2 * (M-1)$ and M is the number of points in the SDF vector. For example, if the sdf vector contains five elements, the corresponding frequencies will be $f = [0, 1/8, 1/4, 3/8, 1/2]$. This argument is used only for the WLSE scheme when calculating EDOF mode 2 estimates. Default: empty vector.

delta_range — A two-element vector containing the search range for the FD parameter. Typically, the range $[-10, 10]$ is suitable for all physical systems. Default: c(-10 10).

Value

An object of class WaveletFDP.

Details

- When estimator = "mle" and boundary = "stationary", the levels vector is forced to take on values $1, 2, ..., J$ where J is the maximum number of levels in a full DWT. This is done because (in this case) the scaling coefficient and all wavelet coefficients are used to form the FD model parameter estimates.
- In using the WLSE scheme it is recommended that only the unbiased estimator be used since the confidence intervals for the biased estimator have not been sufficiently studied.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000, 340–92.

2. W. Constantine, D. B. Percival and P. G. Reinhall, *Inertial Range Determination for Aerothermal Turbulence Using Fractionally Differenced Processes and Wavelets*, Physical Review E, 2001, 64(036301), 12 pages.

See Also

wavFDPTime, wavFDPSimulate, wavFDPSimulateWeights, wavFDPBand, wavFDPSDF.

Examples

```
## perform a block-averaged MLE of FD
## parameters for an FD( 0.45, 1 ) realization
## over levels 1 through 6 using a
## stationary-nonstationary FD model and
## Daubechies least asymmetric 8-tap filters
wavFDPBlock( fdp045, levels = 1:6, wavelet = "s8",
est = "mle", boundary = "nonstationary" )
```

---

wavFDPSDF Spectral desnity function for a fractionally differenced process.

Usage

```
wavFDPSDF( f, delta = 0.45, variance = 1 )
```

Description

Returns the spectral density function (SDF) for a fractionally differenced (FD) process. Given a unit sampling rate, the SDF for an FD proces is

$$\frac{\sigma_\varepsilon^2}{|2\sin(\pi f)|^{2\delta}},$$

where σ_ε^2 is the innovations variance, δ is the FD parameter, and f is the normalized frequency for $|f| < 1/2$.

Required Arguments

f — A numeric value representing normalized frequency where the sampling interval is unity.

Optional Arguments

delta — The FD parameter. Default: 0.45.

response — A list containing the objects frequency and sqrgain which represent, respectively, a numeric normalized frequency vector corresponding to a wavelet squared gain response at a particular wavelet decomposition level. This argument typically will not be set by the user. Rather, it is used internally by FD process maximum likelihood estimators. Default: NULL.

variance — The FD innovations variance. Default: 1.

Value

The SDF values corresponding to the FD model parameters.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000, 340–92.

See Also

wavFDPBand, wavFDPBlock, wavFDPTime.

Examples

```
## create a normalized frequency vector
f <- seq(from = 1e-2, to = 1/2, length = 100)

## calculate the FDP SDF for delta = 0.45
## and unit innovations variance
S <- wavFDPSDF(f, delta = 0.45, variance = 1)

## plot the results
plot(f, S,log = "xy", xlab = "Frequency",
+ ylab = "SDF of FDP(0.45, 1)")
```

---

wavFDPSimulateWeights — Generate the weights for a time-varying FD process simulation.

Usage

```
wavFDPSimulateWeights( delta = c(0.2, 0.4), innovation = rep(1,2) )
```

Description

Time varying fractionally differenced (FD) process realizations are generated by cumulatively summing over the inner product of a Gaussian pseudo-random noise sequence (with zero mean and unit variance) and a series of weights that are dependent upon both the FD parameter. and innovations variance at a particular time. This function generates these weights and returns them in a matrix.

Required Arguments

delta — A vector containing time-varying FD parameters.

Optional Arguments

innovations.variance — A vector containing time-varying FD innovations variances. Default: a vector the same length as delta and filled with ones.

Value

A lower triangular matrix containing the weights needed to simulate a time-varying FD process realization corresponding to the input FD model parameters. The weights needed to simulate the t^{th} point of a time-varying FD process realization are located in result[t,1:t].

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.
2. D. B. Percival and W.L.B. Constantine, *Exact Simulations of Time-Varying Fractionally Differenced Processes*, submitted to Journal of Computational and Graphical Statistics, 2002.

See Also

wavFDPSimulate, wavFDPBlock, wavFDPTime.

Examples

```
## create a time-varying FD parameter,
## linearly varying from white to pink noise
delta <- seq( 0, 0.5, by = 0.02 )

## set the innovations variance to unity
innovation <- rep(1, length( delta ) )
```

```
## creates the weights needed to simulate a
## time-varying FD process
result <- wavFDPSimulateWeights( delta = delta,
+ innovation = innovation )
```

---

wavFDPSimulate Simulation of an FD process with time varying model parameters.

Usage

```
wavFDPSimulate( delta = c(0.2, 0.4), innovation = rep(1,2) )
```

Description

Creates a realization of a time-varying fractionally differenced (FD) process with a given vector of FD parameters and corresponding vector of innovations variances.

Required Arguments

delta — A vector containing time-varying FD parameters.

Optional Arguments

innovations.variance — A vector containing (time-varying) FD innovations variances. Default: a vector the same length as delta and filled with ones.

Value

A vector containing a (time-varying) FD process realization corresponding to the input FD model parameters.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.
2. D. B. Percival and W.L.B. Constantine, *Exact Simulations of Time-Varying Fractionally Differenced Processes*, submitted to Journal of Computational and Graphical Statistics, 2002.

See Also

wavFDPSimulateWeights, wavFDPBlock, wavFDPTime.

Examples

```
## create a time-varying FD parameter,
## linearly varying from white to pink noise
delta <- seq( 0, 0.5, by = 0.02 )

## set the innovations variance to unity
innovation <- rep(1, length( delta ) )

## simulate a time-varying FD process
result <- wavFDPSimulate( delta = delta,
+ innovation = innovation )
```

---

wavFDPTime Block-independent (instantaneous) estimation of fractionally differenced (FD) model parameters.

Usage

```
wavFDPTime( x, levels = 2:6, wavelet = "s8",
estimator = "lse", biased = F, dof = 0 )
```

Description

The MODWT is used to calculate instantaneous estimates of the FD parameter, the variance of the FD parameter and the innovations variance. The user can select between maximum likelihood and least squares estimators. Localized estimates may also be formed by using multiple chi-squared degrees of freedom in estimating the FD model parameters.

Required Arguments

x — A vector containing a uniformly-sampled real-valued time series.

Optional Arguments

levels — A vector containing the decomposition levels. The levels may be given in any order but must be positive. Default: 1:J where J is the maximum wavelet decomposition level at which there exists at least one interior wavelet coefficient.

wavelet — A character string denoting the filter type. See wavDaubechies for details. Default: “s8”.

estimator — A character string denoting the estimation method. Use “lse” for least squares estimates and “mle” for maximum likelihood estimates. Default: “lse”.

biased — A logical flag used to choose between denoting biased or unbiased estimates. Biased estimates are those which use all available levels in calculating the FD model parameters. Unbiased estimates are calculated with only those wavelet coefficients not subject to circular filter operations, i.e. only the interior wavelet coefficients are used in calculating unbiased estimates. Default: TRUE.

dof.order — The degree of freedom (DOF) order. The number of chi-square DOFs used in estimating the FD parameters is equal to $2 \times \text{dof.order} + 1$ where necessarily $\text{dof.order} > 0$. As the order increases, the estimates will become smoother but less localized in time. Default: 0.

delta.range — A two-element vector containing the search range for the FD parameter. Typically, the range $[-10, 10]$ is suitable for all physical systems. Default: c(-10, 10).

Value

An object of class WaveletFDP.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000, 340–92.
2. W. Constantine, D. B. Percival and P. G. Reinhall, *Inertial Range Determination for Aerothermal Turbulence Using Fractionally Differenced Processes and Wavelets*, Physical Review E, 2001, 64(036301), 12 pages.

See Also

wavFDPBlock, wavFDPSimulate, wavFDPSimulateWeights, wavFDPBand, wavFDPSDF.

Examples

```
## perform a unbiased instantaneous LSE of FD parameters
## for an FD(0.45, 1) realization over levels 1 through 6
## using Daubechies least asymmetric 8-tap filters.
## Use a zeroth order DOF (equivalent to 1 chi-square DOF)
result <- wavFDPTime( fdp045, levels = 1:6,
+ wavelet = "s8", est = "lse", biased = F )

## display the results
print( result )

## plot the results
plot( result )

## plot the results with the confidence intervals
## centered about the mean (known) value of the
## the FD parameter
plot( result, mean.delta = 0.45 )
```

---

wavGain The gain functions for Daubechies wavelet and scaling filters.

Usage

```
wavGain( wavelet = "s20", n.levels = 5, normalize = T )
```

Description

Given $\{g\}$ and $\{h\}$ are the impulse responses for the scaling and wavelet filters, respectively, and $G_1(f)$ and $H_1(f)$ are their corresponding gain functions, then the gain functions for decomposition level $j > 1$ are calculated using the recursive algorithm:

$$H_j(f) = H_1(2^{j-1}f)G_{j-1}(f),$$
$$G_j(f) = G_1(2^{j-1}f)G_{j-1}(f).$$

Optional Arguments

wavelet	A character string denoting the filter type. See wavDaubechies for details. Default: "s8".
n.levels	The number of decomposition levels. Default: 5.
n.fft	The number of Fourier coefficients to use in approximating the gain functions. Default: 1024.
normalize	A boolean value. If TRUE, the filters are normalized by dividing each filter coefficient by the $\sqrt{2}$ (used for maximal overlap wavelet transforms). If FALSE, no normalization is used. Default: TRUE.

Value

An object of class WaveletGain.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.
2. I. Daubechies, *Orthonormal Bases of Compactly Supported Wavelets*, Communications on Pure and, Applied Mathematics, 41, 909–96.

See Also

wavDaubechies.

Examples

```
## approximate the gain functions for the
## normalized Daubechies least
## asymmetric 20-tap filters for levels 1,...,5
## using a 1024 Fourier frequencies
result <- wavGain( wavelet = "s20", n.levels = 5,
+ norm = T )

## plot the results
plot( result )
```

---

wavIndex Boundary and interior wavelet coefficient identification.

Usage

```
wavIndex( x )
```

Description

The boundary wavelet and scaling coefficients are those subject to circular filtering operations. This function returns the range of indices which span the interior (or nonboundary) wavelet and scaling coefficients. If approximate zero phase filters are used in the wavelet trsnsform input then the shift factors needed to bring the coefficients to (approximate) zero phase are also returned.

Required Arguments

x — An object of class WaveletTransform or WaveletBoundary.

Value

A list the indices locating the interior and boundary coefficients as well as the the zero phase shift factors need for each level of the transform.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

wavDWT, wavMODWT, wavBoundary.

Examples

```
## calculate the coefficient indices for a MODWT
## of a simple time series
wavIndex( wavMODWT( 1:8, wavelet = "s8" ) )
```

---

wavMODWPT The maximal overlap discrete wavelet packet transform (MODWPT).

Usage

```
wavMODWPT( x, wavelet = "s8", n.levels = 3 )
```

Description

Given j, n, t are the decomposition level, oscillation index, and time index, respectively, the MODWPT is given by

$$\widetilde{W}_{j,n,t} \equiv \sum_{l=0}^{L-1} \widetilde{u}_{n,l} \widetilde{W}_{j-1,\lfloor n/2 \rfloor, t-2^{j-1} l \bmod N}$$

The variable L is the length of the filters defined by

$$\widetilde{u}_{n,l} \equiv \begin{cases} \widetilde{g}_l/\sqrt{2}, & \text{if } n \bmod 4 = 0 \text{ or } 3; \\ \widetilde{h}_l/\sqrt{2}, & \text{if } n \bmod 4 = 1 \text{ or } 2, \end{cases}$$

where g and h are the scaling filter and wavelet filter, respectively. By definition, $\widetilde{W}_{0,0,t} \equiv X_t$ where $\{X_t\}$ is the original time series.

Required Arguments

x	A vector containing a uniformly-sampled real-valued time series.

Optional Arguments

wavelet	A character string denoting the filter type. See wavDaubechies for details. Default: "s8".
n.levels	The number of decomposition levels. Default: the maximum level at which there exists at least one interior wavelet coefficient.

Value

An object of class WaveletPacket.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

reconstruct, wavDetail, wavMODWT, wavDWT, wavDWPT, wavDaubechies, wavShift, wavZeroPhase.

Examples

```
## calculate the MODWPT of an electrocardiogram
## sequence out to 3 levels using Daubechies least
## asymmetric 8-tap filter set
result <- wavMODWPT( ecg, wavelet = "s8", n.levels = 3 )

## plot the transform
plot( result )

## summarize the transform
summary( result )
```

---

wavMODWT — The maximal overlap discrete wavelet transform (MODWT).

Usage

```
wavMODWT( x, wavelet = "s8", n.levels = 3 )
```

Description

Let j,t be the decomposition level, and time index, respectively, and $s_{0,t} = X_{t=0}^{N-1}$ where X_t is a real-valued uniformly-sampled time series. The j^{th} level MODWT wavelet coefficients $\tilde{d}_{j,t}$ and scaling coefficients $\tilde{s}_{j,t}$ are defined as

$$\begin{aligned} \tilde{d}_{j,t} &\equiv \sum_{l=0}^{L-1} \widetilde{h}_l \tilde{s}_{j-1,t-2^{j-1}l \bmod N}, \\ \tilde{s}_{j,t} &\equiv \sum_{l=0}^{L-1} \widetilde{g}_l \tilde{s}_{j-1,t-2^{j-1}l \bmod N}. \end{aligned}$$

The variable L is the length of both the scaling filter (g) and wavelet filter (h). The $\tilde{d}_{j,t}$ and $\tilde{s}_{j,t}$ are the wavelet and scaling coefficients, respectively, at decomposition level j and time index t. The MODWT is a collection of all wavelet coefficients and the scaling coefficients at the last level: $\tilde{\mathbf{d}}_1, \tilde{\mathbf{d}}_2, \ldots, \tilde{\mathbf{d}}_J, \tilde{\mathbf{s}}_J$ where $\tilde{\mathbf{d}}_j$ and $\tilde{\mathbf{s}}_j$ denote a collection of wavelet and scaling coefficients, respectively, at level j.

Required Arguments

x — A vector containing a uniformly-sampled real-valued time series.

Optional Arguments

wavelet — A character string denoting the filter type. See wavDaubechies for details. Default: "s8".

n.levels — The number of decomposition levels. Default: the maximum level at which there exists at least one interior wavelet coefficient.

Value

An object of class WaveletTransform.

Details

The MODWT is a non-decimated form of the discrete wavelet transform (DWT) having many advantages over the DWT including the ability to handle arbitrary length sequences and shift invariance (while the wavDWT function can handle arbitrary length sequences, it does so by means of an ad hoc storage sytem for odd length scaling coefficient crystals. The MODWT needs no such scheme and is more robust in this respect). The cost of the MODWT is in its redundancy. For an N point input sequence, there are N wavelet coefficients per scale. However, the number of multiplication operations is $O(N \log_2(N))$ which is the same as the fast Fourier transform, and is acceptably fast for most situations.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

reconstruct, wavDetail, wavMODWPT, wavDaubechies, wavDWT, wavBoundary, wavIndex, wavShift, wavZeroPhase.

Examples

```
## calculate the MODWT of an electrocardiogram
## sequence out to 4 levels
result <- wavMODWT( ecg, wavelet = "s8", n.levels = 4 )

## plot the transform
plot( result )

## summarize the transform
summary( result )
```

---

wavMaxLevel

Calculates the maximum level for a wavelet transform for which there exists at least one interior wavelet coefficient.

Usage

```
wavMaxLevel( n.taps = 8, n.sample = 1024, transform = "modwt" )
```

Description — Interior wavelet coefficients are those not subject to circular filter operations. This function calculates the maximum level for a wavelet transform for which there exists at least one interior wavelet coefficient.

Optional Arguments

n.taps — The length of the wavelet filter. Default: 8.

n.sample — The number of points in the original time series. Default: 1024.

transform — A chcarcter string denoting the transform type. Supported types are “dwt”, “dwpt”, “modwt”, and “modwpt”. Default: “modwt”.

Value — An integer denoting the maximum decomposition level which contains at least one interior wavelet coefficient.

See Also — wavDWT, wavMODWT, wavDWPT, wavMODWPT.

Examples

```
wavMaxLevel( n.taps = 8, n.sample = 1024, transform = "modwt" )
```

---

wavRotateVector — Circular rotation of a numeric vector.

Usage

```
wavRotateVector(1:5,-2)
```

Description — Rotates (circularly shifts) a numeric vector by the specified shift.

Required Arguments

x — A numeric vector.

shift — An integer specifying the amount to shift the numeric vector. A negative shift implies an advance or circular permutation to the left.

Value — A vector containing the result.

See Also — vector

Examples

```
## rotate a vector forwards by 3 places
wavRotateVector( 1:10, 3 )
```

---

wavShift — Shifts wavelet transform coefficients for approximate zero phase alignment.

Usage

```
wavShift( x )
```

Description — If Daubechies symmlets or coiflets are used in a DWT, MODWT, DWPT, or MODWPT, then the transform coefficients can be circularly rotated so that they are approximately aligned (in time) with events of the original time series. An appropriate shift of the coefficients (generated by approximate linear phase filter operations) is approximately equivalent to using zero phase filters in the wavelet transform.

Required Arguments

x	An object of class WaveletTransform, WaveletBoundary or WaveletPacket.

Value An object of the same class as the input with the transform coefficients adjusted to approximate zero phase filtering operations.

Details Only relevant for transforms calculated using Daubechies coiflet and symmlet filters. A second application of wavShift to the same input object will result in the original input object, i.e. without any imposed shift in the transform coefficients.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.
2. I. Daubechies, *Orthonormal Bases of Compactly Supported Wavelets,* Communications on Pure and, Applied Mathematics, 41, 909–96.

See Also wavZeroPhase, wavDWT, wavMODWT, wavDWPT, wavMODWPT, wavBoundary.

Examples

```
## plot the zero phase shifted MODWT of a
## linear chirp sequence
linchirp <- make.signal( "linchirp", n = 1024 )
plot( wavShift( wavMODWT( linchirp, wavelet = "s8", n.levels = 4 ) ) )
```

---

wavTitle Extract the name of the data used to generate objects of various wavelet classes.

Usage

```
wavTitle(x)
```

Description Wavelet functions store the original name of the data used to create the output in various locations within the output object. This function provides a means by which the user can directly access data name.

Required Arguments

x	An object of class dwt, bpt, ptable, mra, mrd, decompose, wpt, WaveletBoundary, WaveletHomogeneity, WaveletTransform, WaveletPacket, WaveletDualTree, signalSeries, WaveletFDP, or WaveletDualTree2d.

Optional Arguments

default	A default chacter string to use if no valid time series name is found. Default: “x”.

Value A character string vector containing the result.

See Also wavDWT, wavMODWT, wavDWPT, wavMODWPT.

Examples

```
wavTitle(wavDWT(1:8))
wavTitle(wavShift(wavDWT(1:8)))
```

---

wavVarianceHomogeneity Homogeneity test for discrete wavelet transform crystals.

Usage

```
wavVarianceHomogeneity( x, wavelet = "s8", n.levels = 1:4 )
```

Description

Tests for homogeneity of variance for each scale of a discrete wavelet transform (DWT) decomposition. Based on the assumption that the DWT decorrelates colored noise processes, the interior wavelet coefficients in a given scale ($\mathbf{d}_j^{int}$) can be regarded as a zero mean Gaussian white noise process. For a homogeneous distribution of $\mathbf{d}_j^{int}$, there is an expected linear increase in the cumulative energy as a function of time. The so called D-statistic denotes the maximum deviation of the $\mathbf{d}_j^{int}$ from a hypothetical linear cumulative energy trend. This D-statistic is then compared to a table of critical D-statistics that defines the distribution of D for various sample sizes. Comparing the D-statistic of $\mathbf{d}_j^{int}$ to the corresponding critical values provides a means of quantitatively rejecting or accepting the linear cumulative energy hypothesis. This function performs this test for an ensemble of distribution probabilities.

Required Arguments

x — An object of class dwt with convolution style filtering, a corresponding wavebound object, or a numeric vector. In the latter case, DWT parameters can be passed to specify the type of wavelet to use and the number of decomposition levels to perform.

Optional Arguments

wavelet — A character string denoting the filter type. Valid only for input not of class dwt or wavebound. Default: “s8”.

n.levels — The number of decomposition levels. Valid only for input not of class dwt or wavebound. Default: the maximum decomposition level that contains at least one interior wavelet coefficient.

significance — A numeric vector of real values on the interval (0,1). Qualitatively the significance is the fraction of times that the linear cumulative energy hypothesis is incorrectly rejected. It is equal to the difference of the distribution probability (p) and unity. Default: c(0.1, 0.05, 0.01).

lookup — A logical flag for accessing precalculated critical D-statistics. The critical D-statistics are calculated for a variety of sample sizes and significances. If lookup is TRUE, this table is accessed. The table is stored as the matrix object D.table.critical and is loaded with S+Wavelets. Missing table values are calculated using the input arguments: n.realization, n.repetition and tolerance. Default: TRUE.

n.realization — An integer specifying the number of realizations to generate in a Monte Carlo simulation for calculating the D-statistic(s). This parameter is used either when lookup is FALSE, or when lookup is TRUE and the table is missing values corresponding to the specified significances. Default: 10000.

n.repetition — an integer specifying the number of Monte Carlo simulations to perform. This parameter is coordinated with the n.realization parameter. Default: 3.

tolerance — A numeric real scalar that specifies the amplitude threshold to use in estimating critical D-statistic(s) via the Inclan-Tiao approximation. Setting this parameter to a higher value results in a lesser number of summation terms at the expense of obtaining a less accurate approximation. Default: 1e-6.

Value

An object of class WaveletHomogeneity.

Details

An Inclan-Tiao approximation of critical D-statistics is used for sample sizes $N \geq 128$ while a Monte Carlo technique is used for $N < 128$. For the Monte Carlo technique, the D-statistic for a Gaussian white noise sequence of length N is calculated. This process is repeated n.realization times, forming a distribution of the D-statistic. The critical values corresponding to the significances are calculated a total of n.repetition times, and averaged to form an approximation to the D-statistic(s). Because the Monte Carlo study can be both computationally and memory intensive, it is highly recommended that lookup be set to TRUE, its default value.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

wavVariance, wavBoundary, wavDWT, D.table.

Examples

```
## perform a homogeneity of variance test for a DWT
## decomposition of a long memory process realization
homogeneity <- wavVarianceHomogeneity( fdp045 )
```

---

wavVariance Discrete wavelet variance estimation.

Usage

```
wavVariance( x, wavelet = "s8", n.levels = 4 )
```

Description

The discrete wavelet variance is a useful alternative to the spectral density function (SDF) and is seen as an octave-band regularization of the SDF. The wavelet variance decomposes the variance of certain stochastic processes on a scale-by-scale basis, and thus, is very appealing to the analyst studying physical phenomena which fluctuate both within and across a wide range of scale.

The MODWT Wavelet Variance

Let N be the the number of samples in a time series $\{X_t\}$, L be the length of the wavelet filter, $L_j \equiv (2^j - 1)(L-1)+1$ be the equivalent filter width at level j in a MODWT, and $\tau_j \equiv 2^{j-1}$ be the scale of the data at level j for $j = 1, \ldots, J$. Then the unbiased wavelet variance is defined as

$$\hat{\nu}_X^2(\tau_j) \equiv \frac{1}{M_j} \sum_{t=L_j-1}^{N-1} \tilde{d}_{j,t}^2$$

where $\tilde{d}_{j,t}$ are the MODWT coefficients at level j and time t, and $M_j \equiv N - L_j + 1$. The unbiased wavelet variance estimator avoids so-called boundary coefficients which are those coefficients subject to circular filter operations in a discrete wavelet transform. The biased estimator is defined as

$$\widetilde{\nu}_X^2(\tau_j) \equiv \frac{1}{N} \sum_{t=0}^{N-1} \tilde{d}_{j,t}^2,$$

and includes the effects of the boundary coefficients.

The DWT Wavelet Variance

The DWT can also be used to calculate wavelet variance estimates, but is not preferred over the MODWT due to its poor statistical properties. Let $N_j \equiv \lfloor N/2^j \rfloor$ be the number of DWT wavelet coefficients at level j, and $L'_j \equiv \lceil (L-2)(1-2^{-j}) \rceil$ be the number of

DWT boundary coefficients at level j (assuming $N_j > L'j$). Then the DWT version of the unbiased wavelet variance is defined as

$$\hat{\hat{\nu}}_X^2(\tau_j) \equiv \frac{1}{N - 2^j L'_j} \sum_{t=L'_j-1}^{N_j-1} d_{j,t}^2$$

where $d_{j,t}$ are the DWT coefficients at level j and time t. Similarly, the DWT version of the biased wavelet variance is defined as

$$\tilde{\tilde{\nu}}_X^2(\tau_j) \equiv \frac{1}{N} \sum_{t=0}^{N_j-1} d_{j,t}^2.$$

Required Arguments

x	A vector containing a uniformly-sampled real-valued time series.

Optional Arguments

transform	A character string denoting the type of wavelet transform: "modwt" or "dwt". Default: "modwt".
wavelet	A character string denoting the filter type. See wavDaubechies for details. Default: "s8".
n.levels	The number of decomposition levels. Default: the maximum level at which there exists at least one interior wavelet coefficient.
sdf	A vector containing a discretized approximation of the process spectral density function (SDF). The coefficients of this argument should correspond exactly with the normalized Fourier frequencies $$f = [0, 1/P, 2/P, 3/P, ..., (M-1)/P]$$ where $P = 2 * (M-1)$ and M is the number of points in the SDF vector. For example, if the sdf vector contains five elements, the corresponding frequencies will be $f = [0, 1/8, 1/4, 3/8, 1/2]$. This argument is used only for calculating mode 2 EDOF. If the EDOF mode 2 estimates are not desired, send in an empty vector for this argument and the EDOF mode 2 and corresponding confidence intervals will not be calculated. Default: empty vector.
sampling.interval	The sampling interval of the time series. Default: 1.

Value

An object of class WaveletVariance.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

wavCovariance, wavVarianceHomogeneity, wavEDOF.

Examples

```
## create sequence
x <- make.signal( "doppler" )

## perform a time independent wavelet variance analysis
vmod <- wavVariance( x )
```

```
## plot the results
plot( vmod, pch = 15, title = "Wavelet Variance of Doppler" )

## calculate wavelet variance estimaates for the ocean series
## and calculate EDOF mode 2 estimates and corresponding
## 95 percent confidence intervals
vocean <- wavVariance( ocean, sdf = oceansdf, wavelet = "d6" )

## summarize the results
plot( vocean, edof = 1:3 )
summary( vocean )
```

---

wavZeroPhase — Zero phase shift factors for Daubechies symmlet and coiflet filters.

Usage

```
wavZeroPhase( wavelet = "s8", levels = 1:3 )
```

Description

Daubechies coiflet and symmlet filters are approximate linear phase filters. Consequently, the wavelet and scaling coefficients of the DWT (using convolution style filtering), MODWT, DWPT, and MODWPT can be circularly shifted for approximate zero phase alignment with the original time series. This function calculates the circular shift factors needed to bring the wavelet and scaling coefficients to approximate zero phase.

Optional Arguments

wavelet — A character string denoting the filter type. See wavDaubechies for details. Default: “s8”.

levels — An integer vector containing the decomposition levels. Default: 1:3.

Value

A list containing the shifts for each crystal of a DWT, DWPT, MODWT, and MODWPT for the specified decomposition levels. A negative shift factor implies an advance (circular shift to the left) of the wavelet transform crystals.

Details

Only relevant for DWT, MODWT, DWPT, and MODWPT definitions as given in the above reference and is valid only for Daubechies symmlet and coiflet filters.

References

1. D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge University Press, 2000.

See Also

wavDaubechies, wavDWT, wavMODWT, wavDWPT, wavMODWPT, wavShift.

Examples

```
## calculate the zero phase
## shift factors for Daubechies coiflet
## 12-tap filters for levels 2 and 4.
wavZeroPhase( wavelet = "c12", levels = c(2,4) )
```

Appendix C
Class Reference

The S+Wavelets module is based on an object-oreiented design, where outputs from various functions produce which objects belong to a specific class. Specially designed methods are developed to facilitate the display, summary, and data access of such output. This chapter describes the S+Wavelets classes and their associated methods.

WaveletDaubechies — Daubechies filters class.

Summary Operator Methods

print — Prints the following information about the wavelet filters:

- the filter type (e.g. 'Extremal Phase')
- the width L of the wavelet and scaling filters
- a logical flag stating the normalization status of the filters
- the coefficients (i.e., impulse response sequence) for the filters

plot — Plots the coefficients for wavelet and scaling filters or components of their frequency response functions (these functions are the discrete Fourier transforms of the filter coefficients and are also commonly called the transfer functions). The plot function takes a second argument (type), which should be set to the string "time", "gain", or "phase" to plot, respectively, the filter coefficients, the gain function (i.e., the modulus of the frequency response function) and the phase of the frequency response function.

Data Access Methods

**$** — Use to access specific components of the class object. A list of accessible components can be generated using the names function. For example, if s8 is an object of class WaveletDaubechies, then names(s8) will give a list of its components. Two of these components are wavelet and scaling, so s8$wavelet will return the wavelet filter coefficients, while s8$scaling will return the scaling filter coefficients.

WaveletDualTree Dual tree wavelet transform class.

Summary Operator Methods

print — Prints useful information regarding the wavelet transform including:

- Information regarding the filter set.
- Number of decomposition levels.
- Boundary extension rule.
- Filtering technique (convolution or correlation).
- The crystal names.

plot — Plots the transform coefficients. The plot function takes two optional arguments: levels and modulus, which are, respectively, a vector specifying the decomposition levels and a boolean flag used denote the modulus (absolute value) of the transform coefficients.

Transform Operator Methods

reconstruct — Reconstructs the time series via an inverse transform.

Data Access Methods

[— Access a subset of all crystals contained in a a WaveletDualTree object. For example, to obtain the first and second level wavelet crystals of a DTWT object X, use either X[1:2] or X[c("d1","d2")]. While the former is more compact, the latter is preferred because of it leaves no doubt as to which crystals are to be extracted and does not rely on any particular ordering of the crystals in the object.

[[— Access an individual crystal of a WaveletDualTree object. For example, to obtain the second level wavelet crystal of a DTWT object X, use either X[[2]] or X[["d2"]]. The result is a vector of transform coefficients. If however multiple crystals are requested (ala X[[c("d1","d2")]] for example), the result is an S-PLUS list containing the requested crystals.

$ — Use to access specific components of the WaveletDualTree object. A list of accessible components can be generated using the names function. One such component is data which contains an S-PLUS list of all wavelet transform crystals. This gives the user yet another way to access specific transform crystal(s). For example, if X is an object of class WaveletDualTree, the d2 crystal can be accessed via X$data$d2.

WaveletDualTree2d Two-dimensional dual tree wavelet transform class.

Summary Operator Methods

print — Prints useful information regarding the wavelet transform including:

- Information regarding the filter set.
- Number of decomposition levels.
- Boundary extension rule.
- Filtering technique (convolution or correlation).

- The crystal names.

plot — Plots the transform coefficients. The plot function takes two optional arguments: levels and modulus, which are, respectively, a vector specifying the decomposition levels and a boolean flag used denote the modulus (absolute value) of the transform coefficients.

Transform Operator Methods

reconstruct — Reconstructs the time series via an inverse transform.

Data Access Methods

[— Access a subset of all crystals contained in a a WaveletDualTree2d object. For example, to obtain the "d3-s3" crystal of a 2-D DTWT object X, use X["d3-s3"].

[[— Access an individual crystal of a WaveletDualTree2d object. For example, to obtain the "d3-s3" crystal of a 2-D DTWT object X, use X[["d3-s3"]].

$ — Use to access specific components of the WaveletDualTree2d object. A list of accessible components can be generated using the names function.

WaveletFDP — Fractionally differenced process parameter estimation class

---

Summary Operator Methods

print — Prints useful information regarding the FD model parameter estimation including:

- A summary of the estimated FD parameters: δ, var$\{\delta\}$, and σ_ε^2.
- The method used to estimate the FD process model parameters.
- Information regarding the handling of boundary coefficients.
- Decomposition levels over which the estimates were calculated.
- The search range of the FD parameter (δ).
- Information regarding the wavelet transform filters.

plot — Plots the estimates FD parameter δ_t. For instantaneous LSEs, an optional numeric constant can be used as a second argument to the plot function in order to display the confidence intervals about a known value of the FD parameter δ. This method is only applicable for instantaneous estimates produced by the wavFDPTime function.

Data Access Methods

$ — Use to access specific fields of the class object. A list of accessible fields can be generated using the names function.

WaveletGain — Wavelet filter frequency reponse class.

Summary Operator Methods

print — Prints basic filter gain information including:

- Type of wavelet.

- Number of decomposition levels for which the gain functions are calculated.
- Number of (Fourier) frequencies used to approximate each gain function.
- The filter normalization state.

plot — Plots the quared gain functions.

Data Access Methods

$ — Use to access specific components of the class object. A list of accessible components can be generated using the names function.

WaveletKingsbury — Kingsbury filters class.

Summary Operator Methods

print — Prints the following information about the wavelet filters:

- the filter types for all decomposition levele (e.g. 'Antonini' or 'Q-Shift A')
- the width of each filter

plot — Plots the coefficients for wavelet and scaling filters or components of their frequency response functions (these functions are the discrete Fourier transforms of the filter coefficients and are also commonly called the transfer functions). The plot function takes a second argument (type), which should be set to the string "time" or "gain" to plot, respectively, the filter coefficients, the gain function (i.e., the modulus of the frequency response function).

Data Access Methods

$ — Use to access specific components of the class object. Class WaveletKingsbury objects contain a list of filters described below:

- **biorthogonal:** string representing the name of the biorthogonal filter set
- **qshift:** string representing the name of the Q-Shift filter set
- **analysis** A list of analysis filters, mapped as follows
  - h1: wavelet filter, level 1
  - ha: wavelet filter, levels > 1, Tree A
  - hb: wavelet filter, levels > 1, Tree B
  - g1: scaling filter, level 1
  - ga: scaling filter, levels > 1, Tree A
  - gb: scaling filter, levels > 1, Tree B
- **synthesis** A list of synthesis filters, mapped as follows
  - hh1: wavelet filter, level 1
  - hha: wavelet filter, level > 1, Tree A
  - hhb: wavelet filter, level > 1, Tree B
  - gg1: scaling filter, level 1
  - gga: scaling filter, level > 1, Tree A
  - ggb: scaling filter, level > 1, Tree B

WaveletPacket Wavelet packet transform class.

Summary Operator Methods

print	Prints useful information regarding the wavelet packet transform including: • Type of wavelet packet transform. • Information regarding the filter set. • Number of decomposition levels. • Boundary extension rule. • Filtering technique (convolution or correlation). • The crystal names.
plot	Plots the transform coefficients.

Transform Operator Methods

mrd	Perform a multiresolution decomposition of the data.
mra	Perform a multiresolution analysis of the data.
reconstruct	Reconstructs the time series via an inverse transform.
wavShift	Circularly shift each crystal to achieve approximate zero phase. (only suitable for coiflet and symmlet filters).
wavDetail	Calculate the detail coefficients for specified crystals.

Data Access Methods

[	Access a subset of all crystals contained in a a WaveletPacket object. For example, to obtain the first and second level wavelet crystals of a MODWPT or DWPT object X, use X[level = 1:2]. Individual crystals may be accessed as we using the crystal name(s) as an input ala X[c("w1.0","w3.1")] for example. In all cases, the original WaveletPacket object is returned with a subset of the original crystals.
[[	Access an individual crystal of a WaveletPacket object. For example, to obtain the node correpsonding to level 2 and oscillation index 3 of a WaveletPacket object X, use X[["w3.2"]]. The result is a vector of transform coefficients. If however multiple crystals are requested (ala X[[c("w3.0","w1.1")]] for example), the result is an S-PLUS list containing the requested crystals. In all cases, only the specified transform coefficients are returned, i.e. all other components of the original WaveletPacket object are excluded.
$	Use to access specific components of the WaveletPacket object. A list of accessible components can be generated using the names function. One such component is data which contains an S-PLUS list of all wavelet transform crystals. This gives the user yet another way to access specific transform crystal(s). For example, if X is an object of class WaveletPacket, the w3.0 crystal can be accessed via X$data$w3.0.

WaveletTransform Wavelet transform class.

Summary Operator Methods

print	Prints useful information regarding the wavelet transform including: • Type of wavelet transform. • Information regarding the filter set. • Number of decomposition levels. • Boundary extension rule. • Filtering technique (convolution or correlation). • Whether or not the transform coefficients have been shifted for zero phase (only suitable for coiflet and symmlet filters). • The crystal names.
plot	Plots the transform coefficients. If the transform is shifted for zero phase, the shift factor appears adjacent to the crystal name. Use the string "energy" as a value for the optional type argument to display an energy distribution plot of the crystals.
eda.plot	Extendend data analysis plot displaying the transform, crystal energy distribution, and coefficient distribution
summary	Displays a statistical summary of the transform data.

Transform Operator Methods

mrd	Perform a multiresolution decomposition of the data.
mra	Perform a multiresolution analysis of the data.
reconstruct	Reconstructs the time series via an inverse transform.
wavShift	Circularly shift each crystal to achieve approximate zero phase. (only suitable for coiflet and symmlet filters).

Data Access Methods

[	Access a subset of all crystals contained in a a WaveletTransform object. For example, to obtain the first and second level wavelet crystals of a MODWT or DWT object X, use either X[1:2] or X[c("d1","d2")]. While the former is more compact, the latter is preferred because of it leaves no doubt as to which crystals are to be extracted and does not rely on any particular ordering of the crystals in the object.
[[	Access an individual crystal of a WaveletTransform object. For example, to obtain the second level wavelet crystal of a MODWT or DWT object X, use either X[[2]] or X[["d2"]]. The result is a vector of transform coefficients. If however multiple crystals are requested (ala X[[c("d1","d2")]] for example), the result is an S-PLUS list containing the requested crystals.
$	Use to access specific components of the WaveletTransform object. A list of accessible components can be generated using the names function. One such component is data which contains an S-PLUS list of all wavelet transform crystals. This gives the user yet another way to access specific transform crystal(s). For example, if X is an object of class WaveletTransform, the d2 crystal can be accessed via X$data$d2.

WaveletVariance Wavelet variance class.

Summary Operator Methods

print Prints useful information regarding the wavelet variance estimates including:

- Type of wavelet transform.
- Information regarding the filter set.
- Number of decomposition levels.
- Boundary extension rule.
- Filtering technique (convolution or correlation).
- The crystal names.
- The scale of each crystal.

plot Plots the wavelet variance estimates. The plot function has two optional arguments type and edof. Set type to "unbiased" or "biased" to plot unbiased or biased estimates, respectively. To select the EDOF mode of confidence intervals to be plotted, set edof to a vector of integers representing the desired EDOF mode as a third argument (e.g. [1 3] for both EDOF mode 1 and 3).

summary Displays a statistical summary of the transform data.

Data Access Methods

$ Use to access specific components of the class object. A list of accessible components can be generated using the names function.

References

[AG95] P. Abry and P. Gonçalvès. Wavelets, spectrum analysis and $1/f$ processes. In A. Antoniadis and G. Oppenheim, editors, *Wavelets and Statistics* (Lecture Notes in Statistics, Volume 103), pages 15–29, New York, 1995. Springer–Verlag.

[AGF93] P. Abry, P. Gonçalvès, and P. Flandrin. Wavelet-based spectral analysis of $1/f$ processes. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 237–40, Minneapolis, 1993. IEEE Press.

[AS64] M. Abramowitz and I.A. Stegun, editors. *Handbook of Mathematical Functions*. US Government Printing Office, Washington, DC, 1964.

[AV98] P. Abry and D. Veitch. Wavelet analysis of long-range-dependent traffic. *IEEE Transactions on Information Theory*, 44:2–15, 1998.

[Bai96] R. T. Baillie. Long memory processes and fractional integration in econometrics. *Journal of Econometrics*, 73:5–59, 1996.

[Ber94] J. Beran. *Statistics for Long-Memory Processes*. Chapman & Hall, New York, 1994.

[BG96] A. Bruce and H.-Y. Gao. *Applied Wavelet Analysis with S-Plus*. Springer, 1996.

[BLW94] J.B. Bassingthwaighte, L.S. Liebovitch, and B.J. West. *Fractal Physiology*. Oxford University Press, New York, 1994.

[CHT98] R. Carmona, W.-L. Hwang, and B. Torrésani. *Practical Time-Frequency Analysis*. Academic Press, San Diego, 1998.

[CPG00a] P. Craigmile, D.B. Percival, and P. Guttorp. The impact of wavelet coefficient correlations on fractionally differenced processes. In *Proceedings of the Third European Conference of Mathematics*, Barcelona, Spain, 2000. Birkhauser Verlag.

[CPG00b] P. Craigmile, D.B. Percival, and P. Guttorp. Wavelet-based parameter estimation for trend contaminated fractionally differenced processes. *Journal of Time Series Analysis*, 2000. submitted for review.

[CPR01] W. Constantine, D.B. Percival, and P.G. Reinhall. Inertial range determination for aerothermal turbulence using fractionally differenced processes and wavelets. *Physical Review E*, 64(036301), 2001. 12 pages.

[CRK00] H. Choi, J. Romberg, and N. G. Kingsbury. Hidden markov tree modeling of complex wavelet transforms. In *Proceedings ICASSP 2000, Istanbul*, June 2000.

[Dah97] R. Dahlhaus. Fitting time series models to nonstationary processes. *The Annals of Statistics*, 25:1–37, 1997.

[Dau92] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, Philadelphia, 1992. Number 61 in CBMS-NSF Series in Applied Mathematics.

[Dor98] M. I. Doroslovački. On the least asymmetric wavelets. *IEEE Transactions on Signal Processing*, 46:1125–30, 1998.

[dR00] P. F. C. de Rivas. *Complex wavelet based image analysis and synthesis*. PhD thesis, University of Cambridge, 2000.

[dRK99] P. F. C. de Rivas and N. G. Kingsbury. Complex wavelet features for fast texture image retrieval. In *Proceedings of the IEEE Conference On Image Processing, Kobe*, October 1999.

[dRK00] P. F. C. de Rivas and N. G. Kingsbury. Fast segmentation using level set curves of complex wavelet surfaces. In *Proceedings ICIP 2000, Vancouver*, September 2000.

[FG94] P. Flandrin and P. Gonçalvès. From wavelets to time-energy distributions. In L. L. Schumaker and G. Webb, editors, *Recent Advances in Wavelet Analysis*, pages 309–34. Academic Press, Boston, 1994.

[GF93] P. Gonçalvès and P. Flandrin. Bilinear time-scale analysis applied to local scaling exponent estimation. In Y. Meyer and S. Roques, editors, *Progress in Wavelet Analysis and Applications*, pages 271–6. Frontieres, Paris, 1993.

[GJ80] C. W. J. Granger and R. Joyeux. An introduction to long-memory time series models and fractional differencing. *Journal of Time Series Analysis*, 1:15–29, 1980.

[HB00] P. Hill and D. Bull. Rotationally invariant texture features using the dual-tree complex wavelet transform. In *Proceedings ICIP 2000, Vancouver*, September 2000.

[HMK99] S. Hatipoglu, S. K. Mitra, and N. G. Kingsbury. Texture classification using dual-tree complex wavelet transform. In *Proceedings 7th International IEE Conference On Image Processing and Its Applications, Manchester*, July 1999.

[Hos81] J. R. M. Hosking. Fractional differencing. *Biometrika*, 68:165–176, 1981.

[Jen99a] M. J. Jensen. An approximate wavelet mle of short- and long-memory parameters. *Studies in Nonlinear Dynamics and Econometrics*, 3:239–53, 1999.

[Jen99b] Mark J. Jensen. Using wavelets to obtain a consistent ordinary least squares estimator of the long-memory parameter. *Journal of Forecasting*, 18:17–32, 1999.

[Jen00] M. J. Jensen. An alternative maximum likelihood estimator of long-memory processes using compactly supported wavelets. *Journal of Economic Dynamics and Control*, 24:361–87, 2000.

[Kap93] D. Kaplan. Evaluating deterministic structure in maps deduced from discrete-time measurements. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, 3(3):617–23, 1993.

[Kin98] N. G. Kingsbury. The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement. In *Proceedings EUSIPCO 98, Rhodes*, September 1998.

[Kin99] N. G. Kingsbury. Image processing with complex wavelets. *Philosophical Transactions of the Royal Society of London A*, pages 2543–2560, September 1999.

[Kin00] N. G. Kingsbury. A dual-tree complex wavelet transform with improved orthogonality and symmetry properties. In *Proc. IEEE Conf. on Image Processing, Vancouver, B.C.*, September 2000. paper 1429.

[KNKF00] A.H. Kam, T. T. Ng, N. G. Kingsbury, and W. J. Fitzgerald. Content based image retrieval through object extraction and querying. In *Proceedings IEEE Workshop on Content-based Access of Image and Video Libraries, Hilton Head*, June 2000.

[LK00] P. Loo and N. G. Kingsbury. Digital watermarking using complex wavelets. In *Proceedings ICIP 2000, Vancouver*, September 2000.

[Mal99] S. G. Mallat. *A Wavelet Tour of Signal Processing (Second Edition)*. Academic Press, San Diego, 1999.

[MK98] J. F. A. Magarey and N. G. Kingsbury. Motion estimation using a complex valued wavelet transform. *IEEE Transactions on Signal Processing*, 46(4), April 1998.

[MPZ98] S. G. Mallat, G. Papanicolaou, and Z. Zhang. Adaptive covariance estimation of locally stationary processes. *The Annals of Statistics*, 26:1–47, 1998.

[MW96] E.J. McCoy and A.T. Walden. Wavelet analysis and synthesis of stationary long-memory processes. *Journal of Computational and Graphical Statistics*, 5:26–56, 1996.

[PC02] D. B. Percival and W.L.B. Constantine. Exact simulation of time-varying fractionally differenced processes. submitted to the Journal of Computational and Graphical Statistics, 2002.

[Pri65] M. B. Priestley. Evolutionary spectra and non-stationary processes. *Journal of the Royal Statistical Society, Series B*, 27:204–37, 1965.

[PW00] D.B. Percival and A. Walden. *Wavelet Methods for Time Series Analysis*. Cambridge University Press, Cambridge, UK, 2000.

[RCBK00] J. Romberg, H. Choi, R. Baraniuk, and N. G. Kingsbury. Multiscale classification using complex wavelets. In *Proceedings ICIP 2000, Vancouver*, September 2000.

[Sel00] I. W. Selesnick. Image processing with complex wavelets. *Wavelet Digest*, December 2000.

[WCS01] Y. Wang, J. E. Cavanaugh, and C. Song. Self-similarity index estimation via wavelets for locally self-similar processes. *Journal of Statistical Planning and Inference*, 99:91–110, 2001.

[WO92] G. W. Wornell and A. V. Oppenheim. Estimation of fractal signals from noisy measurements using wavelets. *IEEE Transactions on Signal Processing*, 40:611–23, 1992.

[Wor93] G. W. Wornell. Wavelet-based representations for the $1/f$ family of fractal processes. *Proceedings of the IEEE*, 81:1428–50, 1993.

[Wor96] G. W. Wornell. *Signal Processing with Fractals: A Wavelet-Based Approach*. Prentice Hall, Upper Saddle River, New Jersey, 1996.

Index