Efficient computation of the discrete autocorrelation wavelet inner product matrix

Idris A. Eckley*and Guy P. Nason†

30th August 2004

Abstract

Discrete autocorrelation (a.c.) wavelets have recently been applied in the statistical analysis of locally stationary time series for local spectral modelling and estimation. This article proposes a fast recursive construction of the inner product matrix of discrete a.c. wavelets which is required by the statistical analysis. The recursion connects neighbouring elements on diagonals of the inner product matrix using a two-scale property of the a.c. wavelets. The recursive method is an $\mathcal{O}(\log(N)^3)$ operation which compares favourably with the $\mathcal{O}(N \log N)$ operations required by the brute force approach. We conclude by describing an efficient construction of the inner product matrix in the (separable) two-dimensional case.

KEYWORDS: RECURSIVE WAVELET RELATION, LOCALLY STATIONARY TIME SERIES, AUTOCORRELATION WAVELETS

1 Introduction

Locally stationary wavelet processes have recently been introduced by Nason, von Sachs and Kroisandt (2000) as models for nonstationary time series. Nonstationary

*Shell Research Limited and Department of Mathematics, University of Bristol.
†Department of Mathematics, University of Bristol, University Walk, Bristol, BS8 1TW, UK

series have a wide and growing importance in a wide range of areas including finance, medicine and genomics to name but three (see e.g. Clements and Hendry, 2001, Akay, 1998 and Lio and Vannucci, 2000). A process $\{X_{t,N}\}_{t=1}^{N}$ is *locally stationary wavelet* (LSW) if it admits the mean-square representation:

$$X_{t,N} = \sum_{j=1}^{J} \sum_{k=-\infty}^{\infty} w_{jk} \psi_{jk}(t) \xi_{jk}, \tag{1}$$

where ξ_{jk} are uncorrelated mean-zero random increments, w_{jk} are amplitudes, and $\{\psi_{jk}(t)\}$ is a set of discrete non-decimated wavelets as defined below ($j = 1$ is the finest scale, $j = J$ the coarsest) and also $N = 2^J$ for some $J \in \mathbb{N}$ for computational convenience.

The LSW model in (1) should be compared to the classical model for a stationary stochastic process, Y_t:

$$Y_t = \int_{-\pi}^{\pi} A(\omega) \, \exp(i\omega t) \, d\zeta(\omega), \tag{2}$$

where $d\zeta(\omega)$ is an orthonormal increment process (see Priestley, 1981). The stationary model in (2) is constructed from oscillatory sine and cosine functions of infinite extent whereas the LSW model in (1) uses compactly supported discrete non-decimated wavelets. In the stationary model the amplitude function $A(\omega)$ is constant over all time t whereas the the LSW amplitudes w_{jk} depend on t through the compactly supported wavelets $\psi_{jk}(t)$. The speed of evolution of the locally stationary series X_t is controlled by formally tying the amplitudes w_{jk} to a Lipschitz continuous function $W_j(z)$, $z \in (0, 1)$, so that $w_{jk} \approx W_j(k/N)$ using the rescaled time device of Dahlhaus (1997). For more information on wavelets in time series analysis see Nason and von Sachs (1999) or Percival and Walden (2000).

Analogous to the classical case the LSW model has an associated *evolutionary*

wavelet spectrum (EWS), $S_j(z)$, that quantifies the power (contribution to variance) in the process at scale j and location $z \in (0,1)$. Nason *et al.* (2000) construct an asymptotically unbiased estimator of S using the formula:

$$\hat{S} = A^{-1}P \tag{3}$$

where $S(z)$ is vector of stacked $S_j(z)$ for $j = 1, \ldots, J$ and $P(z)$ is a vector of raw wavelet periodograms: the squares of the empirical discrete non-decimated wavelet coefficients of an observed time series. The $J \times J$ matrix A is the inner product matrix of the autocorrelation functions of the non-decimated discrete wavelets. Formal definitions appear below but if $\Psi_j(\tau) = \sum_k \psi_{jk}(0)\psi_{jk}(\tau)$ for $j > 0$, $\tau \in \mathbb{Z}$ are the autocorrelation wavelets then the matrix A is defined by $A_{j,\ell} = \sum_\tau \Psi_j(\tau)\Psi_\ell(\tau)$ for $j = 1, \ldots, J$.

In (3) correction of the raw periodogram P by A^{-1} to obtain an unbiased estimate of the EWS is crucial. If the correction is not applied then spectral power is smeared across the time-scale domain giving a blurred appreciation of S especially for non-stationary time series. See Nason *et al.* (2000) for examples of this blurring and an application that demonstrates the association between EWS estimates of infant electrocardiograms and sleep state time series.

There is no known closed form formula for A except in the special cases of the Haar and Shannon wavelets (in the latter case A is diagonal). Nason *et al.* (2000) compute A using computationally intensive brute force methods. This article introduces a recursive algorithm for computing A which is fundamentally based upon the scale-recursive formulae for discrete wavelets. Our recursive algorithm efficiently computes $A_{j,k}$ from $A_{j-1,k-1}$: the flow of the computations is shown in Figure 1. Our article quantifies and compares the computational effort required to compute A using brute force and recursive techniques and this is summarized in

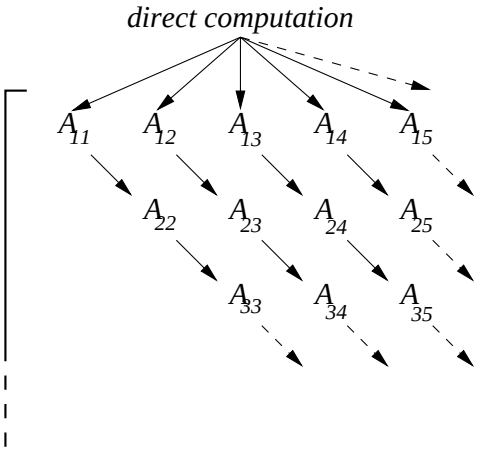

Figure 1: Recursive scheme for (symmetric) inner product matrix A calculation. All diagonal elements are obtained recursively: $A_{j+1,k+1}$ from $A_{j,k}$. The top row is populated through direct computation.

Table 1: Order of computations required for brute force and new recursive algorithms for inner product matrix and autocorrelation wavelet computation. Recall the length of the time series $N = 2^J$.

	Brute force	Recursive
Autocorrelation wavelets	2^{2J} (N^2)	2^J (N)
Inner Product matrix	$J2^J$ $(N \log N)$	J^3 $(\log N)^3$

Table 1. We also show that there appears to be a compelling recursive formula for populating the top row (column) of A but unfortunately the recursion turns out to be less efficient that direct computation.

Finally, motivated by recent work on the modelling of two-dimensional random fields in Eckley (2001), we propose a recursive construction for the inner product matrix of (separable) two-dimensional discrete autocorrelation wavelets. Proofs are presented in the appendix.

2 Discrete wavelets and autocorrelation wavelets

This section formally defines discrete wavelets, the discrete autocorrelation wavelets and their inner product matrix.

2.1 Discrete wavelets and their autocorrelation

In the Haar case discrete wavelets are simply sampled versions of their continuous cousins. For smoother Daubechies' wavelets they are the vectors obtained in Daubechies (1992) cascade algorithm used to produce successively finer approximations to the continuous wavelet.

Definition 1 *Let $\{h_k\}_{k\in\mathbb{Z}}$ and $\{g_k\}_{k\in\mathbb{Z}}$ be the low and high-pass quadrature mirror filters used in the construction of a particular Daubechies (1988, 1992) compactly supported continuous-time wavelet. For $j \in \mathbb{N}$, define the length of the discrete wavelet by*

$$L_j = (2^j - 1)(N_h - 1) + 1$$

*where N_h is simply the number of non-zero elements in $\{h_k\}$. Note that trivially, $L_1 = N_h$. The **discrete wavelets**, $\{\psi_j\}$ of scale j, length L_j, are defined by*

$$\psi_j = \left(\psi_{j,0}, \ldots, \psi_{j,(L_j-1)}\right), \tag{4}$$

where the elements are defined recursively by

$$\psi_{1,n} = \sum_k g_{2-N_h+n-2k}\delta_{0,k} = g_{2-N_h+n}, \qquad \text{for } n = 0, \ldots, L_1 - 1 \tag{5}$$

$$\text{and} \quad \psi_{j,n} = \sum_k h_{n-2k}\psi_{j-1,k}, \qquad \text{for } n = 0, \ldots, L_j - 1 \text{ when } j > 1 \tag{6}$$

and $\delta_{0,k}$ is the usual Kronecker delta.

A related set of discrete father wavelets, ϕ_j, can be similarly constructed by replacing both gs in equation (5) by h. As mentioned above the LSW processes were constructed using *non-decimated* discrete wavelets — that is $\psi_{jk}(t) = \psi_{j,k-t}$, i.e. the position of a wavelet does not depend on its scale. The consequences of a non-decimated scheme are that there are equal numbers of wavelets at every scale. The non-decimated scheme is overdetermined and provides another interpretation for the need to apply the inverse of A to obtain unbiased spectral estimates. See Shensa (1992); Nason and Silverman (1995); Mallat (1998); Vidakovic (1999) or Percival and Walden (2000) for more details of the non-decimated wavelet transform.

The autocovariance of LSW processes is a key statistical quantity of interest. Indeed, it can easily be seen that the autocovariance of X_t in (1) involves the autocorrelation functions of the discrete wavelets which we define next.

Definition 2 *Let $j \in \mathbb{N}$ and $\tau \in \mathbb{Z}$. Then the **discrete autocorrelation (a.c.) wavelet**, $\Psi_j(\tau)$, is defined by*

$$\Psi_j(\tau) = \sum_{k=\max\{0,\tau\}}^{L_j-1+\min\{0,\tau\}} \psi_{j,k}\psi_{j,k-\tau}. \tag{7}$$

*The **discrete a.c. father wavelet**, $\Phi_j(\tau)$, is defined by replacing ψ by ϕ in (7).*

The a.c. wavelets defined above have several interesting and well-known properties: they are compactly supported on the interval $[1 - L_j, \ldots, L_j - 1]$, are symmetric about $\tau = 0$ and are also positive semi-definite functions. A.c. wavelets are related to the autocorrelation shell of Saito and Beylkin (1993) and the continuous father wavelet a.c. function is also the fundamental function of the Deslauriers and Dubuc (1989) interpolation scheme.

Other useful properties of a.c. wavelets include: $\Psi_1(\tau) = (-1)^\tau \Phi_1(\tau)$ which uses equation 5.1.34 from Daubechies (1992). Furthermore, the a.c. wavelet at any

scale can be recursively obtained from the a.c. wavelet at the previous, finer, scale using knowledge only of Φ_1:

Lemma 1 *Let $\tau \in \mathbb{Z}$. The discrete a.c. wavelet at scale $j+1$ is related to that at scale $j \in \mathbb{N}$ by:*

$$\Psi_{j+1}(2\tau) = \Psi_j(\tau) \tag{8}$$

and

$$\Psi_{j+1}(2\tau+1) = \sum_{p=\max\{-L_1/2,1-L_j+\tau\}}^{\min\{L_1/2-1,L_j+\tau-1\}} \Phi_1(2p+1)\Psi_j(\tau-p). \tag{9}$$

We have paid particular attention to the summation limits here and later as this is important for determining the order of computational effort and implementation. A similar two-scale scheme is also valid for a.c. *father* wavelets. It can be shown that the brute force computation of the complete set $\{\Psi_j(\tau)\}_{j=1,\ldots,J}$ requires $\mathcal{O}(2^{2J}) = \mathcal{O}(N^2)$ operations, whereas recursive computation using Lemma 1 requires $\mathcal{O}(2^J) = \mathcal{O}(N)$ operations (see Eckley, 2001 for details).

2.2 Discrete A.c. Wavelets' Inner Product Matrix

We now define the a.c. wavelet inner product matrix.

Definition 3 *Let $J \in \mathbb{N}$. The J-dimensional **discrete a.c. wavelet inner product matrix**, A_J, is defined by $A_J = (A_{j,k})_{j,k\in\{1,\ldots,J\}}$, where,*

$$A_{j,k} = \langle \Psi_j, \Psi_k \rangle = \sum_{\tau=1-\min\{L_j,L_k\}}^{\min\{L_j,L_k\}-1} \Psi_j(\tau)\Psi_k(\tau) \tag{10}$$

$$= 1 + 2 \sum_{\tau=1}^{\min\{L_j,L_k\}-1} \Psi_j(\tau)\Psi_k(\tau), \tag{11}$$

using $\Psi_j(0) = 1$ *for all* $j \in \mathbb{N}$ *and the symmetry of* Ψ_j.

If the a.c. wavelets have already been evaluated, computation of the inner product, $A_{j,k}$, for $k \geq j$ requires the order of $\min\{L_j, L_k\} + 1 = L_j - 1$ operations. Thus, brute force construction of symmetric A_J requires $\mathcal{O}(\sum_{j=1}^{J}\sum_{l=j}^{J} L_j - 1) = \mathcal{O}(J2^J)$ operations.

3 Recursive calculation of the inner product matrix

The main goal of this article is to develop an efficient technique for computing the inner product matrix of discrete wavelet a.c. functions. Our recursive algorithm to compute A is simple and directly depends on Lemma 1 to obtain a recursive relationship between elements of A on a given diagonal. The next section considers the leading diagonal and then §3.3 considers the other diagonals. Sections 3.2 and 3.4 quantify the computational effort required for our recursive algorithm which was summarised in Table 1.

3.1 Algorithm for computing the leading diagonal

This section derives a relationship which connects neighbouring elements of the leading diagonal of the inner product matrix. The relationship enables efficient computation of $A_{k,k}$ from $A_{1,1}$ for any $k \in \mathbb{N}$. We start by defining a few key quantities Q_r and $P_{j,n}$ as follows:

Definition 4 *For* $r \in \mathbb{Z}$, *let* $l_r = -L_1/2 - \min\{0, r\}$ *and* $u_r = L_1/2 - 1 - \max\{0, r\}$. *Then define*

$$Q_r = \sum_{p=l_r}^{u_r} \Phi_1(2p+1)\,\Phi_1\left(2(p+r)+1\right). \tag{12}$$

Clearly Q_r is symmetric about $r = 0$ and has support $[1 - L_1, L_1 - 1]$. Consequently only $\{Q_r\}$ for $r \in \{0, \ldots, L_1 - 1\}$ needs to be evaluated which takes $\mathsf{Ops}(Q) = L_1(L_1 + 1)/2$ operations. The second definition is based upon a.c. wavelets.

Definition 5 *Let $j \in \mathbb{N}$, $l_n = 1 - L_j + \max\{0, n\}$ and $u_n = L_j - 1 + \min\{0, n\}$. Define*

$$P_{j,n} = \sum_{k=l_n}^{u_n} \Psi_j(k)\Psi_j(k-n). \tag{13}$$

Clearly $P_{j,n}$ is symmetric about $n = 0$, has support $[2(1 - L_j), \ldots, 2(L_j - 1)]$ and $A_{j,j} = P_{j,0}$.

Using the two-scale relationship of the discrete a.c. wavelets, the $P_{j,n}$ themselves can be recursively constructed as follows.

Proposition 1 *Let $p \in \mathbb{Z}$ and $j \in \mathbb{N}$ and let*

$$\begin{aligned}
u_e &= \min\{L_1 - 1, 2(L_{j-1} - 1) - p\}, && \textit{and} \quad l_e = \max\{1 - L_1, 2(1 - L_{j-1}) - p\};\\
u_{o1} &= \min\{L_1/2 - 1, 2(L_{j-1} - 1) - p\} && \textit{and} \quad l_{o1} = \max\{-L_1/2, 2(1 - L_{j-1}) - p\};\\
u_{o2} &= \min\{L_1/2 - 1, p - 2(1 - L_{j-1})\} && \textit{and} \quad l_{o2} = \max\{-L_1/2, p - 2(L_{j-1} - 1)\}.
\end{aligned}$$

Then,

$$P_{j,2p} = P_{j-1,p} + \sum_{q=l_e}^{u_e} P_{j-1,p+q}Q_q, \tag{14}$$

and

$$P_{j,2p+1} = \sum_{r=l_{o1}}^{u_{o1}} \Phi_1(2r+1)P_{j-1,p+r} + \sum_{r=l_{o2}}^{u_{o2}} \Phi_1(2r+1)P_{j-1,p-r}. \tag{15}$$

Thus, the $P_{j,n}$ may be calculated using knowledge of only Φ_1 and $P_{k,n}$ at finer scales k.

Once more we pay close attention to the limits of summation. The results of Lemma 1, together with the above identities, permit the derivation of a recursive relationship between neighbouring elements which lie along the leading diagonal of the inner product matrix.

Proposition 2 *Let $j \in \mathbb{N}$. Then the $(j+1, j+1)^{th}$ element of the inner product matrix is related to the $(j,j)^{th}$ element by the following recursive relation:*

$$\begin{aligned} A_{j+1,j+1} &= A_{j,j} + \sum_{r=1-L_1}^{L_1-1} P_{j,r} Q_r; \\ &= A_{j,j}(1+Q_0) + 2\sum_{r=1}^{L_1-1} P_{j,r} Q_r. \end{aligned} \tag{16}$$

In other words, the elements which lie on the leading diagonal of the inner product matrix, A_J, can be recursively obtained using only knowledge of $A_{1,1}$ and Φ_1.

3.2 Effort for leading diagonal recursion

3.2.1 Initialising values

First consider the effort required in calculating the initialising values $\{P_{1,n}\}$. Definition 5 implies

$$P_{1,n} = \sum_{k=l_1}^{u_1} \Psi_1(k)\Psi_1(k-n). \tag{17}$$

As $P_{1,n}$ is symmetric in $n=0$ it suffices to calculate it for $n \in [0, \ldots, 2(L_1-1)]$. Thus, from (17), it follows that direct evaluation of the $\{P_{1,n}\}_{n\in[0,\ldots,2(L_1-1)]}$, using pre-computed values of $\{\Psi_j(\tau)\}$, takes

$$\mathsf{Ops}(P_{\text{init}}) = \sum_{n=0}^{2(L_1-1)} 2(L_1-1) - n = (L_1-1)(2L_1-1) \quad \text{operations.}$$

In other words, for any given wavelet family, it is an $\mathcal{O}(1)$ operation

3.2.2 Calculation of the $P_{j,2p}$

Since $P_{j,2p}$ is symmetric about $2p = 0$ we only need consider $p > 0$. Calculation of $P_{j,2p}$ from (14) for any given $j \in \mathbb{N}, p \in \mathbb{Z}$, requires the following number of operations:

$$\begin{aligned} \mathsf{Ops}(P_{j,2p}) &= 1 + u_e - l_e, \\ &= 1 + \min\{L_1 - 1, 2(L_{j-1} - 1) - p\} \\ &\qquad - \max\{1 - L_1, 2(1 - L_{j-1}) - p\} \\ &= 1 + \min\{L_1 - 1, 2(L_{j-1} - 1) - p\} - (1 - L_1) \end{aligned} \tag{18}$$

since $p > 0$ and that $1 - L_1$ is always greater than $2(1 - L_{j-1}) - p$, for $j \in \mathbb{N} \setminus \{1\}$. However, it is important to note that the minimum term cannot be simplified, as there exist $p \in \mathbb{N}$ such that $2(L_{j-1}-1)-p < L_1-1$. Thus the number of operations required to calculate $P_{j,2p}$ from $P_{j-1,}$, for $j, p \in \mathbb{N}$, is given by $\mathsf{Ops}(P_{j,2p}) = L_1 + \min\{L_1 - 1, 2(L_{j-1} - 1) - p\}$.

We now consider the values of p for which we wish to evaluate $P_{j,2p}$, for any given $j \in \mathbb{N}$. The recursive identity (16) requires only those values of $P_{j,r}$ such that $r \in [1, \ldots, L_1 - 1]$. I.e. the length of the filter associated with the wavelet determines the number of $P_{j,r}$ which need to be evaluated at any given level. Thus, at first glance, it appears reasonable simply to calculate $P_{j,2p}$ for all $2p \in [1, L_1 - 1]$ for all levels $j' < j$. However, as we demonstrate in the following example, the recursive form of (14) used to generate the $\{P_{j,2p}\}$, ensures that this is not the case.

Example 1 *Assume that $J \in \mathbb{N}$ is fixed and that all $\{P_{j,r}\}_{j=1,\ldots,J-1}$ which are required for the construction of $\{P_{J,2p}\}$ have already been evaluated. As J is*

fixed, we know from (16) *that it suffices to calculate* $P_{J,2p}$ *for* $2p \in [1, \ldots, L_1 - 1]$. *However, for Daubechies' compactly supported wavelets,* L_1 *is even and so one can calculate* $P_{J,2p}$ *for* $p \in [1, \ldots, L_1/2 - 1]$. *By Proposition 1,*

$$P_{J,2p} = P_{J-1,p} + \sum_{q=l_e+p}^{u_e+p} P_{J-1,q} Q_{q-p}. \tag{19}$$

In other words, we need to know $P_{J-1,q}$ *for*

$$1 - L_1 + p \leq q \leq \min\{L_1 - 1, 2(L_{J-1} - 1) - p\} + p. \tag{20}$$

However,

$$\begin{aligned} \min_{p \in \{1,\ldots,L_1/2-1\}} \{L_1 - 1, 2(L_{J-1} - 1) - p\} &= \min\{L_1 - 1, 2L_{J-1} - L_1/2 - 1\}; \\ &= L_1 - 1. \end{aligned}$$

Hence (20) *reduces to:*

$$1 - L_1 + p \leq q \leq L_1 - 1 + p, \tag{21}$$

for $p \in \{1, \ldots, L_1/2 - 1\}$. *Thus, to calculate* $\{P_{J,2p}\}_{1,\ldots,L_1/2-1}$, *we require* $P_{J-1,q}$ *for* $q \in [0, \ldots, L_1 + L_1/2 - 2]$.

Clearly, wider and wider "intervals" of $P_{j,r}$ will be required as we progress down through the scales. However, using the above scheme, it is not at all easy to construct an identical algorithm for the evaluation of $\{P_{j,2p}\}_{j=1,\ldots,J}$ for *all* Daubechies' compactly supported wavelets. Consider, for example, the situation at the end of the above example: to construct $P_{J,2p}$ for $p \in \{1, \ldots, L_1/2 - 1\}$, we need to be able to evaluate $P_{J-1,r}$ for $r \in \{0, \ldots, L_1 + L_1/2 - 2\}$. However, as

L_1 is even, $L_1 + L_1/2 - 2$ can be either odd *or* even valued, depending on the form of N_h.

However, the situation is actually more complicated: we also need to consider the above for each scale (see Eckley, 2001 for further details). Thus, if the algorithm is to be efficient (i.e. such that only *required* $\{P_{j,r}\}$ are evaluated) not only do we need routines for each individual wavelet family but also for each scale: this is most unappealing from an implementational perspective.

As an alternative we propose the algorithm below that evaluates a slightly larger number of $P_{j,r}$ than absolutely necessary. The algorithm is easy to implement and, more importantly, still efficient and universal for each Daubechies' wavelet. The universal algorithm is:

1. Fix $J \in \mathbb{N}$.
2. Calculate $P_{1,r}$ for all $r \in [0, \ldots, 2(L_1 - 1)]$.
3. Then, for $j \in 2, \ldots, J$, calculate $P_{j,r}$ for $r \in I_j$, where

$$I_j = \{0, \ldots, \min\{2(L_j - 1), (J - j + 1)L_1\}\}, \tag{22}$$

   setting $P_{j,r} = 0$ for other r.

The universal algorithm permits the evaluation of a tractable upper bound for the number of operations required no matter what the choice of wavelet. Moreover, as $P_{j,r}$ is evaluated for values of $r \in I_j$, we know that we need only evaluate $\{P_{j,2p}\}$ for

$$p \in I_{j,\text{even}} = \{0, \ldots, u_j = \min\{L_j - 1, (J - j + 1)L_1/2\}\}.$$

Note that in general, u_j cannot be simplified to $(J - j + 1)L_1/2$, for if J is large

whilst j is small, then $u_j = L_j - 1$. However, using the definition of L_j, it is easy to show that the following holds:

$$u_j = (J - j + 1)L_1/2 \quad \text{iff} \quad \frac{N_h}{N_h - 1} \leq 2\frac{2^j - 1}{(J - j + 1)}. \tag{23}$$

Eckley (2001) shows that

$$\mathsf{Ops}\{P_{\text{even}}\} = \sum_{j=2}^{J} \sum_{p=0}^{u_j} \mathsf{Ops}(P_{j,2p})$$

is, at the very most, an $\mathcal{O}(J^2)$ operation and develop a similar algorithm for evaluating $P_{j,2p+1}$.

3.2.3 Effort in calculating the leading diagonal

Suppose that $A_{1,1}$, $\{P_{j,r}\}$ and the $\{Q_r\}$ have already been calculated. Then from (16), it follows that the calculation of $A_{j+1,j+1}$, for any level $j \in 1, \ldots, J-1$, takes L_1 operations. Thus construction of the leading diagonal of the inner product matrix via the schemes proposed in §3.1 takes

$$\begin{aligned} \mathsf{Ops}(\text{Leading Diagonal}) &= \mathsf{Ops}(A_{1,1}) + \mathsf{Ops}(A_{j+1,j+1})_{j=1,\ldots,J-1} + \mathsf{Ops}(P_{\text{odd}}) \\ &\quad + \mathsf{Ops}(P_{\text{even}}) + \mathsf{Ops}(P_{\text{init}}) + \mathsf{Ops}(Q) \\ &= JL_1 + 1 + \mathsf{Ops}(P_{\text{odd}}) + \mathsf{Ops}(P_{\text{even}}) + \mathsf{Ops}(P_{\text{init}}) + \mathsf{Ops}(Q) \end{aligned}$$

operations: i.e. at most $\mathcal{O}(J^2)$ operations. In contrast, the brute force approach used by Nason *et al.* (2000) took $\mathcal{O}(2^J)$ operations.

3.3 Algorithm for the general diagonal

The results of §3.1, particularly Proposition 2, suggest that a recursive relationship may exist for entries on other diagonals. First we define $T_{j,k,r}$ that plays a similar role to $P_{j,n}$ in (16).

Definition 6 *Let $r \in \mathbb{Z}$, $j, k \in \mathbb{N}$ with $k \geq j$, $l_r = \max\{1 - L_j, 1 - L_k + r\}$ and $u_r = \min\{L_j - 1, L_k + r - 1\}$. Then define,*

$$T_{j,k,r} = \sum_{l=l_r}^{u_r} \Psi_j(l)\Psi_k(l-r). \tag{24}$$

The support of $T_{j,k,r}$ is $[2 - L_j - L_k, L_k + L_j - 2]$ and $T_{j,k,r}$ is not generally symmetric in r, though it is easy to see that: $T_{j,k,-r} = T_{k,j,r}$. When $k = j$ we have $T_{j,j,r} = P_{j,r}$ and hence symmetry in r is restored. The following proposition establishes an efficient, recursive, approach for the construction of the $T_{j,k,r}$.

Proposition 3 *Suppose that $j, k \in \mathbb{N}$ with $k \geq j$, $j \neq 1$, $p \in \mathbb{Z}$ and set*

$$\begin{aligned}
l_{e,p} &= \max\{1 - L_1, 2 - L_{j-1} - L_{k-1} - p\}, & u_{e,p} &= \min\{L_1 - 1, L_{k-1} + L_{j-1} - 2 - p\}, \\
l_{o1,p} &= \max\{-L_1/2, 2 - L_{j-1} - L_{k-1} - p\}, & u_{o1,p} &= \min\{L_1/2 - 1, L_{k-1} + L_{j-1} - 2 - p\}, \\
l_{o2,p} &= \max\{-L_1/2, 2 + p - L_{j-1} - L_{k-1}\}, \quad \textit{and} & u_{o2,p} &= \min\{L_1/2 - 1, L_{j-1} + L_{k-1} + p - 2\}.
\end{aligned}$$

Then it can be shown that

$$T_{j,k,2p} = T_{j-1,k-1,p} + \sum_{m=l_{e,p}}^{u_{e,p}} T_{j-1,k-1,p+m} Q_m, \tag{25}$$

and

$$T_{j,k,(2p+1)} = \sum_{r=l_{01,p}}^{u_{o1,p}} \Phi_1(2r+1) T_{j-1,k-1,p+r} + \sum_{r=l_{o2,p}}^{u_{o2,p}} \Phi_1(2r+1) T_{j-1,k-1,p+r} \tag{26}$$

Combining knowledge of the $\{T_{j,k,r}\}$, together with $\{Q_r\}$ and the top row of the inner product matrix, the following recursive algorithm may be derived for elements of A.

Proposition 4 *Let $k \geq j$. Then using Q_r and $T_{j,k,r}$ as defined above, the following recursive scheme can be derived to calculate those elements which lie on a diagonal of the inner product matrix A:*

$$A_{j,k} = A_{j-1,k-1} + \sum_{r=1-L_1}^{L_1-1} T_{j-1,k-1,r} Q_r. \tag{27}$$

3.4 Effort in calculating the other diagonals

By using arguments similar to those in §3.2.2 we can develop an efficient algorithm that computes only a slighter larger number of $T_{j,k,r}$ than actually *required*. We shall only consider $A_{j,k}$ for $k > j$ since A is symmetric and we have already considered the main diagonal.

The following two quantities will be required: for $j, k \in \mathbb{N} \setminus 1$, set

$$\begin{aligned} l^e_{j,k} &= \max\{-L_1(J-k+1), 2-L_j-L_k\} \\ \text{and} \quad u^e_{j,k} &= \min\{L_1(J-k+1), L_j-L_k-2\}. \end{aligned} \tag{28}$$

3.4.1 Calculation of the initialising values

Here we consider computation of $T_{1,k,r}$ which is supported on on $[2-L_1-L_k, \ldots, L_1+L_k-2]$ as a function of r. The $\{T_{1,k,r}\}$ can be calculated directly, using the values of $\Psi_j(\tau)$ obtained from Lemma 1. However, following arguments similar to those proposed in §3.2.2, it is evident that a complicated case by case algorithm is required for the evaluation of the precise number of $\{T_{1,k,r}\}$ required for the recursive construction of the $\{A_{j,k}\}$. As before, a simpler algorithm, suitable

for all wavelet families, may be developed if we are willing to evaluate a slightly larger number of $\{T_{1,k,r}\}$ than required. An outline of the algorithm is provided below. Note how the width of the interval is dependent on k — a consequence of the assumption that $k > j$.

1. Fix $J \in \mathbb{N}$.
2. Calculate $T_{1,k,r}$ for all $r \in I_k = [l^e_{1,k}, \ldots, u^e_{1,k}]$, setting $T_{1,k,r} = 0$ for other r.

It can be shown that construction of the initialising $T_{1,k,r}$ via the above scheme takes at most $\mathsf{Ops}(T_{\text{init}}) = \mathcal{O}(J^2)$ operations (see Eckley, 2001 for further details).

3.4.2 Calculation of the $T_{j,k,2p}$

Assuming that all relevant $\{T_{j-1,k-1,r}\}$ have been evaluated, it follows from (25) that evaluation of $T_{j,k,2p}$ for any given $j, k, \in \mathbb{N}$ and $p \in \mathbb{Z}$ takes $\mathsf{Ops}(T_{j,k,2p}) = 1 + u_{e,p} - l_{e,p}$ operations. As in earlier sections, it is important to observe that for any given $j, k \in \mathbb{Z}$, it is not necessarily the case that one must evaluate $T_{j,k,2p}$ for all $2p \in [2 - L_j - L_k, L_j + L_k - 2]$. However, as in §3.4.1, the exact construction would require a cumbersome case by case algorithm. We therefore propose the following algorithm:

1. Fix $J \in \mathbb{N}$.
2. Evaluate $T_{j,k,2p}$ for all

$$p \in I^e_{j,k} = \left\{ l^e_{j,k}/2, \ldots, u^e_{j,k}/2 \right\}.$$

setting $T_{j,k,2p} = 0$ for other p.

As before this algorithm can be used for all wavelet families although it evaluates a slightly larger number of $T_{j,k,2p}$ than is actually required.

Construction of the required $\{T_{j,k,2p}\}$ via the above algorithm takes

$$\sum_{p=l^e_{j,k}/2}^{u^e_{j,k}/2} \mathsf{Ops}(T_{j,k,2p}) \quad \text{operations.} \tag{29}$$

Thus, the construction of the complete suite of $\{T_{j,k,2p}\}_{\substack{j\,=\,2,\,\ldots\,,\,J\,-\,1\\ k\,=\,j\,+\,1,\,\ldots\,,\,J}}$ for all relevant p takes

$$\begin{aligned}\mathsf{Ops}(T_{\text{even}}) &= \sum_{j=2}^{J-1}\sum_{k=j+1}^{J}\sum_{p=l^e_{j,k}/2}^{u^e_{j,k}/2}\mathsf{Ops}(T_{j,k,2p}) = \sum_{j=2}^{J-1}\sum_{k=j+1}^{J}\sum_{p=l^e_{j,k}/2}^{u^e_{j,k}/2} 1+u_{e,p}-l_{e,p};\\ &\le \sum_{j=2}^{J-1}\sum_{k=j+1}^{J}(2L_1+1)(u^e_{j,k}/2-l^e_{j,k}/2+1) \le (2L_1+1)\sum_{j=2}^{J-1}\sum_{k=j+1}^{J} L_1(J-k+1).\end{aligned}$$

Thus construction of the required $\{T_{j,k,2p+1}\}$ is, at worst, an $\mathcal{O}(J^3)$ operation. Eckley (2001) develops a similar algorithm for evaluating $T_{j,k,2p+1}$.

3.4.3 Calculation of the $A_{j,k}$

Given prior enumeration of the $\{T_{j,k,r}\}$, equation (27) shows that calculation of any given $\{A_{j,k}\}$ for $j = 2,\ldots,J-1$ and $k > j$ takes $\mathsf{Ops}(A_{j,k}) = 2L_1-1$ operations. Thus, using the schemes outlined in §3.4.1–3.4.2 , the recursive construction of those elements which lie neither upon the leading diagonal nor upon the first row of the inner product matrix, takes

$$\mathsf{Ops}(\text{Lead Diags}) = \mathsf{Ops}(T_{\text{init}})+\mathsf{Ops}(T_{\text{even}})+\mathsf{Ops}(T_{\text{odd}})+\sum_{j=2}^{J-1}\sum_{k=j+1}^{J}\mathsf{Ops}(A_{j,k}) \text{ ops.} \tag{30}$$

In other words a $\mathcal{O}(J^3)$ operation at worst. Conversely, brute force calculation would take $\mathcal{O}(2^J)$ operations. Next we consider how to compute the top row of A which seeds the whole recursive algorithm.

4 Construction of the inner product matrix top row.

Given the efficiency gains using the recursive formulae above it is natural to wonder whether the $\{A_{1,j+1}\}$ can be obtained recursively using knowledge of $A_{1,j}$ and hence seed the first row of the matrix. If such a recursive method exists is it more efficient than brute force?

The following recursive scheme seems natural but it turns out that it is not efficient.

Definition 7 *Define*

$$R^l_{j,q} = \sum_{r=\max\left\{\left\lceil \frac{-L_1-q}{2^{j-l+1}}\right\rceil, 1-L_j\right\}}^{\min\left\{\left\lfloor \frac{L_1-2-q}{2^{j-l+1}}\right\rfloor, L_j-1\right\}} \Psi_1\left(2^{j-l+1}r+q+1\right)\Psi_j(r). \tag{31}$$

The $\{R^l_{j,q}\}$ may be evaluated recursively via the following proposition.

Proposition 5 *$R^l_{j,q}$ has the following recursive form:*

$$R^l_{j,q} = R^{l-2}_{j-1,q} + \sum_p \Phi_1(2p+1) R^{l-2}_{j-1,2^{j-l+1}+2^{j-l+2}p+q}. \tag{32}$$

Furthermore, it can be shown that the support of $R^l_{j,q}$ is given by

$$\left[-L_1+(1-L_j)2^{j-l+1}, L_1+(L_j-1)2^{j-l+1}-2\right].$$

Consequently, the support of $R^j_{j,2p}$ is $[2(1-L_j)-L_1, L_1+2(L_j-2)]$.

The proposition permits the following recursive form for the top row of A.

Proposition 6 *The value of $A_{1,j+1}$ can be obtained by calculating $R^j_{j,2p}$ recursively and using Φ_1. More precisely:*

$$A_{1,j+1} = 1 + \sum_{p=-L_1/2}^{L_1/2-1} \Phi_1(2p+1) R^j_{j,2p}. \tag{33}$$

How efficient is this recursive approach? As we go from one scale to the next, the $R^l_{j,q}$ required by one scale differ from that required by the next. Hence, in effect, we have to re-calculate the $R^l_{j,q}$ for each $A_{1,j}$ which seems unattractive. Indeed, for the majority of entries along the top row of A it is more efficient to calculate the $A_{1,j}$ directly via the interpolation rules of Lemma 1, using $\Psi_1(\tau)$ and $\Psi_k(\tau)$ which can be computed using the efficient $\mathcal{O}(2^J)$ algorithm.

5 Extension to Two-Dimensions

One way of extending discrete wavelets to two dimensions is by forming tensor products of the one-dimensional ones following Mallat (1989).

Definition 8 *Let ψ_j and ϕ_j be the discrete one-dimensional wavelets from definition 1. The **two-dimensional discrete wavelets**, $\{\psi^l_j\}$, are compactly supported, of dimension L^2_j, and defined by the matrix*

$$\psi^l_j = (\psi^l_{j,(k_1,k_2)})_{k_1,k_2}$$

where $l = h$, v or d (for horizontal, vertical or diagonal) and the elements are

given by:

$$\psi^h_{j,\mathbf{k}} = \phi_{j,k_1}\psi_{j,k_2},\ \psi^v_{j,\mathbf{k}} = \psi_{j,k_1}\phi_{j,k_2},\ \psi^d_{j,\mathbf{k}} = \psi_{j,k_1}\psi_{j,k_2}, \tag{34}$$

where $\mathbf{k} = (k_1, k_2)$ *and* $k_1, k_2 = 0, \ldots, L_j - 1$. *Similarly, the two-dimensional, discrete father wavelet is defined by* $\phi_{j,\mathbf{k}} = \phi_{j,k_1}\phi_{j,k_2}$.

Two-dimensional discrete non-decimated wavelets have been used to construct locally stationary wavelet processes for the modeling and analysis of texture in Eckley (2001). Texture comprehension is closely related to the local structure of the process autocovariance function which can be represented using the following quantity:

Definition 9 *The* ***a.c. wavelet****, of a given two-dimensional discrete wavelet family* $\left\{\psi^l_{j,\mathbf{k}}\right\}$*, at scale* $j \in \mathbb{N}$ *in direction* l*, for* $\tau \in \mathbb{Z} \times \mathbb{Z}$ *is given by*

$$\Psi^l_j(\boldsymbol{\tau}) = \sum_{\mathbf{k}} \psi^l_{j,\mathbf{k}}(\mathbf{0})\ \psi^l_{j,\mathbf{k}}(\boldsymbol{\tau}) = \sum_u \sum_v \psi^l_{j,(u,v)}\psi^l_{j,(u-\tau_1,v-\tau_2)}, \tag{35}$$

where $\mathbf{k} = (u, v)$ *and* $\boldsymbol{\tau} = (\tau_1, \tau_2)$. *The two-dimensional father a.c. wavelet* $\Phi_j(\boldsymbol{\tau})$ *can be similarly defined.*

Since the two-dimensional wavelets are separable it is no surprise that the a.c. wavelets are also. In other words if $\boldsymbol{\tau} = (\tau_1, \tau_2) \in \mathbb{Z}^2$ then $\Psi^h_j(\boldsymbol{\tau}) = \Phi_j(\tau_1)\Psi_j(\tau_2)$, $\Psi^v_j(\boldsymbol{\tau}) = \Psi_j(\tau_1)\Phi_j(\tau_2)$, $\Psi^d_j(\boldsymbol{\tau}) = \Psi_j(\tau_1)\Psi_j(\tau_2)$, and $\Phi_j(\boldsymbol{\tau}) = \Phi_j(\tau_1)\Phi_j(\tau_2)$. The separability of the two-dimensional a.c. wavelets means that efficient algorithms based on the recursion in Lemma 1 can also be applied here. More details can be found in Eckley (2001).

Due to the separability we can use the earlier results to devise efficient algorithms for the inner product matrix in the 2D case. Let $B_J = (B_{j,k}) = (\langle \Phi_j, \Phi_k \rangle)$

and $C_J = (\langle \Phi_j, \Psi_l \rangle)$, note that C_J is not symmetric. To simplify notation combine the two indicates j and l into one to provide a single multi-index, η, each value of which represents a particular decomposition scale in a given direction as follows: $\eta(j,l) \equiv j + J\, I(l = h) + 2J\, I(l = d)$ for $j = 0, \ldots, J-1$ and $I(\cdot)$ is the usual indicator function. We now define the "two-dimensional" inner product matrix.

Definition 10 *The inner product matrix of a collection of discrete, two-dimensional a.c. wavelets* $\{\Psi_j^h, \Psi_j^v, \Psi_j^d\}_{j=1,\ldots,J}$ *to be the* $3J$*-dimensional matrix,* D_J *given by*

$$
\begin{aligned}
D_J &= (D_{\eta,\nu})_{\eta,\nu} = (\langle \Psi_\eta, \Psi_\nu \rangle)_{\eta,\nu} \\
&= \left[\begin{array}{c|c|c}
A_J * B_J & C_J * C_J^T & A_J * C_J \\
\hline
C_J * C_J^T & A_J * B_J & A_J * C_J \\
\hline
A_J * C_J^T & A_J * C_J^T & A_J * A_J
\end{array}\right]
\end{aligned}
$$

where η, ν are multi-indices, $A * B$ denotes a component-wise multiplication of A and B. Thus recursion schemes of the form given in §3 can be used to efficiently D_J.

6 Example

Tables 2 and 3 show the A matrix for Haar and Daubechies' compactly supported wavelet with ten vanishing moments (recall that both the extremal phase and least

Table 2: Inner product matrix of Haar discrete autocorrelation wavelets of dimension 8 (to 4 decimal places)

$$\begin{bmatrix} 1.5 & 0.75 & 0.375 & 0.1875 & 0.0938 & 0.0469 & 0.0234 & 0.0117 \\ & 1.75 & 1.125 & 0.5625 & 0.2812 & 0.1406 & 0.0703 & 0.0352 \\ & & 2.875 & 2.0625 & 1.0312 & 0.5156 & 0.2578 & 0.1289 \\ & & & 5.4375 & 4.0312 & 2.0156 & 1.0078 & 0.5039 \\ & & & & 10.7187 & 8.0156 & 4.0078 & 2.0039 \\ & & & & & 21.3594 & 16.0078 & 8.0039 \\ & & & & & & 42.6797 & 32.0039 \\ & & & & & & & 85.3398 \end{bmatrix}$$

Table 3: Inner product matrix of Daubechies' extremal phase with 10 vanishing moments discrete autocorrelation wavelets of dimension 8 (to 4 decimal places)

$$\begin{bmatrix} 1.8391 & 0.3216 & 4e-04 & 0 & 0 & 0 & 0 & 0 \\ & 3.0354 & 0.6425 & 8e-04 & 0 & 0 & 0 & 0 \\ & & 6.0704 & 1.285 & 0.0016 & 0 & 0 & 0 \\ & & & 12.1408 & 2.5701 & 0.0032 & 1e-04 & 0 \\ & & & & 24.2817 & 5.1402 & 0.0064 & 1e-04 \\ & & & & & 48.5634 & 10.2803 & 0.0127 \\ & & & & & & 97.1267 & 20.5606 \\ & & & & & & & 194.2534 \end{bmatrix}$$

asymmetric wavelets have the same autocorrelation functions). The matrix in Table 3 is not strictly a band matrix although the off diagonal elements do get small very quickly away from the main diagonal. The Haar wavelets form the beginning of the sequence of Daubechies' compactly supported wavelets and the matrices become "increasingly diagonal" and in the limit the matrix is diagonal (this is for the Shannon wavelet).

It should be mentioned that software is available to compute these and all other matrices for the Daubechies family of compactly supported wavelets using the

`ipndacw` function available from `www.stats.bris.ac.uk/~wavethresh` Prototype R routines implementing the calculation of the diagonal of the A inner product matrix, the quantities, P, Q and discrete wavelets and autocorrelation wavelets can be found at `www.stats.bris.ac.uk` at

`~magpn/Eliospub/reports/Wavelets/IPNDACW/ipndacw.html`

7 Conclusion

This article introduces a efficient recursive scheme for the construction of the inner product matrix of discrete a.c. wavelets. Our recursive scheme constructs matrices using $\mathcal{O}(J^3) = \mathcal{O}((\log N)^3)$ operations in comparison to the brute force scheme which uses $\mathcal{O}(J2^J) = \mathcal{O}(N \log N)$ operations. If our new algorithm is used in conjunction with the recursive formula for generating the a.c. wavelets then the recursive methods require $\mathcal{O}(2^J) = \mathcal{O}(N)$ operations in contrast to the expensive $\mathcal{O}(2^{2J}) = \mathcal{O}(N^2)$ operations required by the brute force approach.

The efficient computation of the discrete a.c. wavelets' inner product matrix is vital for the (asymptotically) *unbiased* estimation of the evolutionary wavelet spectra of locally stationary wavelet processes. Collection of longer time series (for better estimation) requires computation of inner product matrices of larger and larger dimension and hence demonstrates the need for efficient methods of computation.

Acknowledgements

Eckley gratefully acknowledges support from an EPSRC CASE studentship jointly funded by Unilever Research. Nason was supported by EPSRC Grants GR/M10229,

AF/001664. The authors would like to thank the Associate Editor and Referees for suggesting several improvements.

A Proofs

Proof of Lemma 1

By Definitions 1 and 2

$$\Psi_{j+1}(\tau) = \sum_l \psi_{(j+1)l}\psi_{(j+1)(l-\tau)} = \sum_l \sum_k h_{l-2k}\psi_{jk} \sum_m h_{l-\tau-2m}\psi_{jm}. \quad (36)$$

Case A: (even argument). Now

$$\begin{aligned}
\Psi_{j+1}(2\tau) &= \sum_l \sum_k h_{l-2k}\psi_{jk} \sum_m h_{l-2(\tau+m)}\psi_{jm} \quad (\text{then let } r = m+\tau) \\
&= \sum_l \sum_k h_{l-2k}\psi_{jk} \sum_r h_{l-2r}\psi_{j(r-\tau)} \\
&= \sum_k \psi_{jk} \sum_r \psi_{j(r-\tau)} \sum_m h_m h_{m+2(k-r)} \\
&= \sum_k \psi_{jk} \sum_r \psi_{j(r-\tau)}\delta_{k-r,0} = \Psi_j(\tau) \quad (\text{by (5.1.39) of Daubechies (1992)}).
\end{aligned}$$

Case B: (odd argument). Now

$$\begin{aligned}
\Psi_{j+1}(2\tau+1) &= \sum_l \sum_k h_{l-2k}\psi_{jk} \sum_m h_{l-1-2(\tau+m)}\psi_{jm} \quad (\text{again let } r = m+\tau.) \\
&= \sum_k \psi_{jk} \sum_r \psi_{j(r-\tau)} \sum_l h_{l-2k} h_{l-1-2r} \\
&= \sum_k \psi_{jk} \sum_r \psi_{j(r-\tau)} \Phi_1\{2(r-k)+1\} \quad (\text{now let } p = r-k) \\
&= \sum_p \Phi_1(2p+1)\Psi_j(\tau-p).
\end{aligned}$$

The summation limits are obtained by considering the supports of $\Phi_1(2p+1)$ and $\Psi_j(\tau - p)$. □

Proof of Proposition 1

First, consider the case of $P_{j,n}$ when n is even. By definition:

$$\begin{aligned} P_{j,2p} &= \sum_k \Psi_j(k)\Psi_j(k-2p) \\ &= \sum_{k\ (\text{even})} \Psi_j(k)\Psi_j(k-2p) + \sum_{k\ (\text{odd})} \Psi_j(k)\Psi_j(k-2p) \\ &= \sum_l \Psi_j(2l)\Psi_j(2(l-p)) + \sum_l \Psi_j(2l+1)\Psi_j(2(l-p)+1). \end{aligned}$$

Applying Lemma 1, the above expression can be re-written as

$$P_{j,2p} = P_{j-1,p} + \sum_r \sum_s \Phi_1(2r+1)\Phi_1(2s+1) \sum_l \Psi_{j-1}(l-r)\Psi_{j-1}(l-p-s).$$

Setting $q = l - r$, we obtain:

$$\begin{aligned} P_{j,2p} &= P_{j-1,p} + \sum_r \sum_s \Phi_1(2r+1)\Phi_1(2s+1) \sum_q \Psi_{j-1}(q)\Psi_{j-1}(q-(s+p-r)); \\ &= = P_{j-1,p} + \sum_r \sum_s \Phi_1(2r+1)\Phi_1(2s+1) P_{j-1,s+p-r}. \end{aligned}$$

Finally, on making the substitution $q = s - r$ and applying definition 4, the above can be simplified to

$$\begin{aligned} P_{j,2p} &= P_{j-1,p} + \sum_r \sum_q \Phi_1(2r+2q+1)\Phi_1(2r+1) P_{j-1,p+q}; \\ &= P_{j-1,p} + \sum_q P_{j-1,p+q} Q_q, \end{aligned} \tag{37}$$

as required.

The odd case, $P_{j,2p+1}$, is similar, see Eckley (2001) for further details. □

Proof of Proposition 2

Let $j \in \mathbb{Z}^+$. From Definition 3, we know that:

$$\begin{aligned} A_{j+1,j+1} &= \langle \Psi_{j+1}, \Psi_{j+1} \rangle \\ &= \sum_k \Psi_{j+1}(2k)\Psi_{j+1}(2k) + \sum_k \Psi_{j+1}(2k+1)\Psi_{j+1}(2k+1). \end{aligned}$$

Applying Lemma 1, we obtain:

$$A_{j+1,j+1} = A_{j,j} + \sum_p \sum_q \Phi_1(2p+1)\Phi_1(2q+1) \sum_k \Psi_j(k-p)\Psi_j(k-q).$$

Next, make the substitution $r = k - p$,

$$\begin{aligned} A_{j+1,j+1} &= A_{j,j} + \sum_{p,q} \Phi_1(2p+1)\Phi_1(2q+1) \sum_r \Psi_j(r)\Psi_j(r+p-q); \\ &= A_{j,j} + \sum_p \sum_q \Phi_1(2p+1)\Phi_1(2q+1) P_{j,q-p}. \end{aligned} \tag{38}$$

Finally, we make the substitution $r = q - p$ in (38). This results in

$$\begin{aligned} A_{j+1,j+1} &= A_{j,j} + \sum_r P_{j,r} \sum_p \Phi_1(2p+1)\Phi_1(2(p+r)+1); \\ &= A_{j,j} + \sum_r P_{j,r} Q_r, \quad \text{(by Definition 4.)} \end{aligned} \tag{39}$$

as required. □

Proofs of Proposition 3 and 4

Similar to the proof of Proposition 1 in that we consider $T_{j,k,2p}$ and $T_{j,k,2p+1}$ separately: the summation is divided into odd and even parts before applying Lemma 1.

For Proposition 4: similar to the proof of Proposition 2 by using the odd/even division of the summation and then applying Lemma 1

In both cases see Eckley (2001) for further details. □

Proof of Proposition 5

As with earlier proofs, this result can be shown by adopting a divide and conquer approach. By Definition 7,

$$R^l_{j,q} = \sum_r \Psi_1(2^{j-l+1}r + q + 1)\Psi_j(r) \tag{40}$$

where $j, l > 0$. Dividing the summation in (40) into odd and even valued arguments, applying Proposition 2 and re-arranging the resulting expression, we find that

$$R^l_{j,q} \quad = \quad R^{l-2}_{j-1,q} + \sum_p \Phi_1(2p+1) R^{l-2}_{j-1,2^{j-l+1}+2^{j-l+2}p+q}. \tag{41}$$

The limits of the summation in (40) can be found by considering the support of the Ψ_1 and Ψ_j term. (See Eckley, 2001 for further details).

Proof of Proposition 6

Dividing the summation in (11) into odd and even τ and using the results $\Psi_1(2\tau) = \sum_k g_k g_{k-2\tau} = \delta_{\tau,0}$ and $\Psi_j(0) = 1$ it follows that

$$A_{1,j+1} = 1 + \sum_k \Psi_1(2k+1)\Psi_{j+1}(2k+1).$$

Using Lemma 1, the above expression may be simplified to

$$A_{1,j+1} \quad = \quad 1 + \sum_p \Phi_1(2p+1) R^j_{j,p} \qquad \text{as required.} \tag{42}$$

References

Akay, M. (ed.) (1998). *Time frequency and wavelets in biomedical signal processing*, IEEE Press, Piscataway, N.J.

Clements, M. and Hendry, D. (2001). *Forecasting non-stationary economic time series*, MIT Press, Cambridge, MA.

Dahlhaus, R. (1997). Fitting time series models to nonstationary processes., *Annals of Statistics* **25**: 1–37.

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets, *Communications in Pure and Applied Mathematics* **41**: 909–996.

Daubechies, I. (1992). *Ten lectures on Wavelets*, SIAM, Philadelphia.

Deslauriers, G. and Dubuc, S. (1989). Symmetric iterative interpolation processes, *Constructive Approximation* **5**: 49–68.

Eckley, I. (2001). *Wavelet methods for time series and spatial data*, PhD thesis, Department of Mathematics, University of Bristol.

Lio, P. and Vannucci, M. (2000). Finding pathogenicity islands and gene transfer events in genome data, *Bioinformatics* **16**: 932–940.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**: 674–693.

Mallat, S. G. (1998). *A wavelet tour of signal processing*, Academic Press, San Diego.

Nason, G. and Silverman, B. (1995). The stationary wavelet transform and some statistical applications, *in* A. Antoniadis and G. Oppenheim (eds), *Lecture Notes in Statistics*, Vol. 103, Springer-Verlag, pp. 281–300.

Nason, G. and von Sachs, R. (1999). Wavelets in time series analysis, *Phil. Trans. R. Soc. Lond. A.* **357**: 2511–2526.

Nason, G., von Sachs, R. and Kroisandt, G. (2000). Wavelet processes and adaptive estimation of the evolutionary wavelet spectrum., *J. Roy. Stat. Soc. Ser. B.* **62**: 271–292.

Percival, D. and Walden, A. (2000). *Wavelet methods for time series analysis*, Cambridge University Press, Cambridge.

Priestley, M. (1981). *Spectral analysis and time series*, Academic Press, London.

Saito, N. and Beylkin, G. (1993). Multiresolution representation using the autocorrelation functions of compactly supported wavelets., *IEEE Trans. Sig. Proc.* **41**: 3584–3590.

Shensa, M. (1992). The discrete wavelet transform: wedding the à trous and mallat algorithms, *IEEE Trans. Sig. Proc.* **40**: 2464–2482.

Vidakovic, B. (1999). *Statistical modeling by Wavelets*, Wiley, New York.