Length-Preserving Wavelet Transform Algorithms for Zero-Padded and Linearly-Extended Signals

Kevin C. McGill and Carl Taswell

ABSTRACT

Since the wavelet transform is defined for infinite-length signals, finite-length signals must be extended before they can be transformed. Common extension methods include periodic or mirror-image replication, zero padding, and linear extrapolation. Zero-padding and linear extrapolation have the advantage that they are free of wrap-around and reflection problems, but they yield transform vectors that are longer than the signal vector. In this paper we show that these transform vectors can be truncated to the length of the signal vector without loss of information. This results in an invertible, length-preserving transform for zero-padded and linearly extrapolated signals. The computation of the zero-padded transform is discussed in detail and is illustrated by a practical pseudocode routine.

EDICS classification numbers: 4.1.1. and 4.2.4.
Permission to publish this abstract separately is granted.

Manuscript of March 6, 1992.

K.C. McGill is with the Rehabilitation Research and Development Center, Veterans Affairs Medical Center, Palo Alto, CA 94304-1200. C. Taswell is with Scientific Computing and Computational Mathematics, Margaret Jacks Hall 314, Stanford University, Stanford, CA 94305-2140.

Please address correspondence to: Kevin C. McGill, Ph.D., Rehabilitation R&D Center /153, VA Medical Center, Palo Alto, CA 94304-1200 (U.S.A.); email: mcgill@roses.stanford.edu; phone: 415-493-5000 ext. 4477.

I. INTRODUCTION

The wavelet transform maps a signal into a domain that is midway between the time and frequency domains [1]-[5]. This paper is concerned with wavelet transforms of finite-length signals, using the compactly supported orthogonal wavelet functions of Daubechies [1], [2].

Since the wavelet transform is defined for infinite-length signals, finite-length signals must be extended in some way before they can be transformed. Common extension methods include periodic replication (as in the fast fourier transform), mirror-image replication [4], zero padding, and linear extrapolation. Periodic replication and zero padding are appropriate for signals that begin and end on the baseline, while mirror-image replication and linear extrapolation provide continuity at the boundaries for signals that do not begin or end on the baseline.

Zero padding and linear extrapolation are natural extension methods for many types of signals. They make minimal assumptions about the behavior of the signal beyond the boundaries, so that the transform coefficients describe only the detail in the signal itself. Periodic and mirror-image replication, on the other hand, either wrap around or reflect signal detail into the region beyond the boundaries, and this can distort the interpretation of the transform coefficients near the boundaries [6].

An important drawback of zero padding, however, is that it results in a non-length-preserving transform, that is, one in which the transform vector is longer than the signal vector [6]. This is undesirable both because of the increased memory requirements and also because of the existence of a null space of the inverse transform, so that large perturbations in the transform space are not necessarily reflected by large perturbations in the signal space. Periodizing and mirroring do yield length-preserving transforms.

In this paper we present a length-preserving transform for zero-padded signals. Specifically, we show that the full transform of a zero-padded length-n signal can be truncated to n coefficients from which the signal can be exactly reconstructed. A practical transform algorithm based on an efficient method of restoring the full transform vector from the truncated transform vector is given in pseudo matlab code in the appendix.

We also consider a more general class of signal extensions consisting of a fixed component

and a component that depends linearly on the signal coefficients. This class includes linear extrapolation. We show that the transforms of signals with these types of extensions can also be truncated to length n without loss of information.

II. INFINITE-LENGTH WAVELET TRANSFORM

This section presents a matrix formulation of the infinite-length wavelet transform to serve as a framework for the finite-length case. The following notational conventions are used throughout this paper. Infinite-length vectors and matrices with infinitely many rows or columns are indicated by tildes: $\tilde{\mathbf{x}}$, $\tilde{\mathbf{A}}$. Ranges of indices in subvectors and submatrices are often indicated symbolically: thus if $\mathbf{c} = [0 : n-1]$ then $\tilde{\mathbf{x}}_{\mathbf{c}} \equiv \tilde{\mathbf{x}}_{0:n-1}$. Finite-length subvectors are also written without the tilde and with the subscript italicized: thus $\mathbf{x}_c \equiv \tilde{\mathbf{x}}_{\mathbf{c}}$. Iteration levels are indicated by superscripts: $\mathbf{A}^k$, and transposes of superscripted matrices are written either $(\mathbf{A}^k)^T$ or $\mathbf{A}^{kT}$.

The Kth-level wavelet transform of an infinite-length signal $\tilde{\mathbf{x}} = [\ldots, x_{-1}, x_0, x_1, \ldots]^T$ consists of the detail coefficients $\tilde{\mathbf{d}}^k$, $k = 1, \ldots, K$, and the approximation coefficients $\tilde{\mathbf{a}}^K$. These coefficients are computed by the forward recursions

$$\tilde{\mathbf{a}}^k = \tilde{\mathbf{I}}_{\downarrow}\tilde{\mathbf{L}}\tilde{\mathbf{a}}^{k-1}, \quad \tilde{\mathbf{d}}^k = \tilde{\mathbf{I}}_{\downarrow}\tilde{\mathbf{H}}\tilde{\mathbf{a}}^{k-1}, \quad k = 1, \ldots, K, \tag{1}$$

and the original signal can be recovered via the inverse recursion

$$\tilde{\mathbf{a}}^{k-1} = \tilde{\mathbf{L}}^T\tilde{\mathbf{I}}_{\uparrow}\tilde{\mathbf{a}}^k + \tilde{\mathbf{H}}^T\tilde{\mathbf{I}}_{\uparrow}\tilde{\mathbf{d}}^k, \quad k = K, \ldots, 1, \tag{2}$$

where $\tilde{\mathbf{a}}^0 \equiv \tilde{\mathbf{x}}$, $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{H}}$ are low- and high-pass filter matrices, and $\tilde{\mathbf{I}}_{\downarrow}$ and $\tilde{\mathbf{I}}_{\uparrow}$ are compression and dilation matrices. The filter matrices $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{H}}$ are toeplitz matrices whose rows are the impulse responses of complementary low-pass and high-pass filters, respectively. In this paper we will use the compactly supported orthogonal wavelets of Daubechies [1], [2], for which the impulse responses have finite length with an even number N of coefficients. For a basis with low-pass coefficients $l_1, \ldots, l_N$, the matrices are given by

$$\tilde{L}_{i,j} = \begin{cases} l_{j-i+N/2+1}, & \text{if } -N/2 \le j - i < N/2, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

$$\tilde{H}_{i,j} = \begin{cases} h_{j-i+N/2}, & \text{if } -N/2-1 \le j-i < N/2-1, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $h_i = (-1)^i l_{N-i+1}$. The compression and dilation matrices $\tilde{\mathbf{I}}_\downarrow$ and $\tilde{\mathbf{I}}_\uparrow$ are defined as follows:

$$(\tilde{\mathbf{I}}_\downarrow)_{i,j} = (\tilde{\mathbf{I}}_\uparrow)^T_{i,j} = \delta_{2i-j} \tag{5}$$

where $\delta_i = 1$ if $i = 0$ and 0 otherwise. So, for example, for $N = 4$, the combined filtering and compression operators are as follows:

$$\tilde{\mathbf{I}}_\downarrow \tilde{\mathbf{L}} = \left(\tilde{\mathbf{L}}^T \tilde{\mathbf{I}}_\uparrow\right)^T = \begin{bmatrix} \ddots & & & & & & & & & \\ \dots & l_3 & l_2 & l_1 & & & & & & \\ & & l_4 & l_3 & l_2 & l_1 & & & & \\ & & & & l_4 & l_3 & l_2 & l_1 & & \\ & & & & & & l_4 & l_3 & l_2 & \dots \\ & & & & & & & & & \ddots \end{bmatrix}, \tag{6}$$

$$\tilde{\mathbf{I}}_\downarrow \tilde{\mathbf{H}} = \left(\tilde{\mathbf{H}}^T \tilde{\mathbf{I}}_\uparrow\right)^T = \begin{bmatrix} \ddots & & & & & & & & & \\ \dots & h_3 & h_2 & h_1 & & & & & & \\ & & h_4 & h_3 & h_2 & h_1 & & & & \\ & & & & h_4 & h_3 & h_2 & h_1 & & \\ & & & & & & h_4 & h_3 & h_2 & \dots \\ & & & & & & & & & \ddots \end{bmatrix}, \tag{7}$$

where the leftmost, uppermost element shown in each matrix is the $0,0$ element.

In the analysis to follow, it will be convenient to combine the approximation and detail coefficients into a single combined transform vector and to write the forward and inverse transform recursions as single matrix-vector products. Thus we will write the forward recursion as:

$$\tilde{\mathbf{x}}^k = \tilde{\mathbf{A}}^{k-1} \tilde{\mathbf{x}}^{k-1}, \quad k = 1, \dots, K, \tag{8}$$

and the inverse recurion as:

$$\tilde{\mathbf{x}}^{k-1} = (\tilde{\mathbf{A}}^{k-1})^T \tilde{\mathbf{x}}^k, \quad k = K, \dots, 1, \tag{9}$$

where $\tilde{\mathbf{x}}^k$ is the combined transform vector and $\tilde{\mathbf{A}}^k$ is the combined transform matrix.

The zero-level combined transform matrix $\tilde{\mathbf{A}}^0$ is formed by interleaving the rows of $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{H}}$:

$$\tilde{\mathbf{A}}^0_{i,:} = \begin{cases} \tilde{\mathbf{L}}_{i,:} & \text{if } i \text{ is even} \\ \tilde{\mathbf{H}}_{i-1,:} & \text{if } i \text{ is odd} \end{cases} \tag{10}$$

The higher-level combined transform matrices have elements from $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{H}}$ only in the rows and columns that pertain to the kth-level approximation coefficients, and elements of the identity elsewhere to pass the detail coefficients without modification:

$$\tilde{A}^k_{i,j} = \begin{cases} \tilde{A}^0_{i/2^k,j/2^k} & \text{if } i \text{ and } j \text{ are multiples of } 2^k, \\ \delta_{i-j} & \text{otherwise.} \end{cases} \tag{11}$$

Thus for $N = 4$, the first two combined transform matrices are

$$\tilde{\mathbf{A}}^0 = \begin{bmatrix} \ddots & & & & & & & & & \\ \dots & l_3 & l_2 & l_1 & & & & & & \\ \dots & h_3 & h_2 & h_1 & & & & & & \\ & & l_4 & l_3 & l_2 & l_1 & & & & \\ & & h_4 & h_3 & h_2 & h_1 & & & & \\ & & & & l_4 & l_3 & l_2 & l_1 & & \\ & & & & h_4 & h_3 & h_2 & h_1 & & \\ & & & & & & l_4 & l_3 & l_2 & \dots \\ & & & & & & h_4 & h_3 & h_2 & \dots \\ & & & & & & & & & \ddots \end{bmatrix} \tag{12}$$

$$\tilde{\mathbf{A}}^1 = \begin{bmatrix} \ddots & & & & & & & & & \\ \dots & l_3 & & l_2 & & l_1 & & & & \\ & & 1 & & & & & & & \\ \dots & h_3 & & h_2 & & h_1 & & & & \\ & & & & 1 & & & & & \\ & & & l_4 & & l_3 & & l_2 & & \dots \\ & & & & & & 1 & & & \\ & & & h_4 & & h_3 & & h_2 & & \dots \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \end{bmatrix} \tag{13}$$

Defining the combined transform matrices in this way yields a combined transform vector

whose approximation and detail coefficients are interleaved in the following manner:

$$\tilde{\mathbf{x}}^k \equiv [\ldots,\, a_0^k,\, d_0^1,\, d_0^2,\, d_1^1,\, d_0^3,\, d_2^1,\, d_1^2,\, d_3^1,\, d_0^4 \ldots]^T, \tag{14}$$

where specifically

$$\tilde{x}_i^k = \begin{cases} d_j^p & \text{if } i = j2^p + 2^{p-1}, \quad p = 1, \ldots, k, \\ a_j^k & \text{if } i = j2^k, \end{cases} \tag{15}$$

and $\tilde{\mathbf{x}}^0 = \tilde{\mathbf{x}}$.

We remark that a family of slightly different transforms can be obtained by time-shifting the low-pass or high-pass rows of $\tilde{\mathbf{A}}^0$ or by interchanging the low-pass and high-pass rows in one or more of the $\tilde{\mathbf{A}}^k$s. For the finite-length algorithms in the next section, the best numerical conditioning is achieved when the largest elements in each row fall closest to the diagonal. Thus the $\tilde{\mathbf{A}}^0$ specified by Eq. (11) is appropriate for the "closest-to-linear-phase" wavelets of [2], while for the asymmetrical wavelets of [1] the low-pass rows should be shifted to the left, and the high-pass rows to the right, by about $N/2 - 3$ places in order to place the largest element in each row on the diagonal.

III. ALGORITHMS FOR ZERO-PADDED SIGNALS

Now consider a finite-length signal $\mathbf{x} = [x_0, \ldots, x_{n-1}]^T$. This section will consider the wavelet transform $\tilde{\mathbf{x}}^K$ of the zero-padded signal $\tilde{\mathbf{x}} = [\ldots, 0, x_0, \ldots, x_{n-1}, 0, \ldots]^T$. Because of the finite support of the rows of the $\tilde{\mathbf{A}}^k$s, $\tilde{\mathbf{x}}^K$ will have only only a finite number of non-zero coefficients, although in general this number will be greater than n. We will first give an explicit algorithm for computing the full non-zero transform vector, and then show that this vector can be truncated to length n without loss of information.

A. Length-Extending Algorithm

The actual support of the transform vector $\tilde{\mathbf{x}}^k$ depends on n, N, and k in a complicated way, but inspection shows that it is always contained in the interval $\mathbf{e}^k \equiv [-2^k m : n + r + 2^k m - 1]$, where $r = 2^K \lceil n/2^K \rceil - n$ rounds n up to an integer multiple of 2^K, and $m = (N/2 - 1)$ is the number of additional coefficients added at each end of the transform

vector per level k for large k. In fact, this interval is a tight bound on the support at all but the first few levels. (Note, however, that a different expression for $\mathbf{e}^k$ might be needed if using one of the transform variations described at the end of Section II).

The following algorithm, then, computes $\tilde{\mathbf{x}}^K_{\mathbf{e}^K}(\equiv \mathbf{x}^K_e)$, which we call the "extended" transform vector. It is simply a restatement of Eqs. (8) and (9) for the intervals $\mathbf{e}^k$. A practical implementation is given in the appendix. Note that in that implementation, the matrices $\tilde{\mathbf{A}}^{k-1}_{\mathbf{e}^k,\mathbf{e}^{k-1}}$ are never actually constructed. Rather the products $\tilde{\mathbf{A}}^{k-1}_{\mathbf{e}^k,\mathbf{e}^{k-1}}\mathbf{x}^{k-1}_e$ are computed by convolution.

Function $\mathbf{y} = \mathbf{LEWT}\ (\mathbf{x}, \mathbf{l}, K, \textit{direction})$
  if *direction* = "forward"
    $\mathbf{x}^0_e = [\mathbf{0}_{1:m};\ \mathbf{x};\ \mathbf{0}_{1:m}]$
    for $k = 1 : K$
      $\mathbf{x}^k_e = \tilde{\mathbf{A}}^{k-1}_{\mathbf{e}^k,\mathbf{e}^{k-1}}\mathbf{x}^{k-1}_e$
    end
    $\mathbf{y} = \mathbf{x}^K_e$
  elseif *direction* = "inverse"
    $\mathbf{x}^K_e = \mathbf{x}$
    for $k = K : -1 : 1$
      $\mathbf{x}^{k-1}_e = (\tilde{\mathbf{A}}^{k-1}_{\mathbf{e}^k,\mathbf{e}^{k-1}})^T\mathbf{x}^k_e$
    end
    $\mathbf{y} = [\mathbf{x}^0_e]_{m+1:m+n}$
  end

B. Length-Preserving Algorithm

The extended transform vector $\mathbf{x}^K_e$ has $\approx n + 2(K+1)m$ non-zero coefficients, but only n degrees of freedom. Therefore some of its coefficients must be redundant. It turns out that the n central coefficients $\tilde{\mathbf{x}}^K_{\mathbf{c}}(\equiv \mathbf{x}^K_c)$, where $\mathbf{c} = [0, \ldots, n-1]$, contain all the information needed to reconstruct the full vector $\mathbf{x}^K_e$. In particular, we will show in the next section that for each level k there exist an $(n + r + 2^{k+1}m) \times n$ matrix $\mathbf{P}^k$ and an $n \times n$ matrix $\mathbf{A}^k$ such

that

$$\mathbf{x}_e^k = \mathbf{P}^k \mathbf{x}_c^k, \tag{16}$$

$$\mathbf{x}_c^k = \mathbf{A}^{k-1} \mathbf{x}_c^{k-1}. \tag{17}$$

Thus $\mathbf{x}$ and $\mathbf{x}_c^K$ can be considered to be an invertible, equal-length transform pair. The following algorithm computes the length-preserving transform by truncating the length-extending transform, and then restores the length-extending transform using the matrix $\mathbf{P}^K$. A practical implementation is given in the appendix. The matrix $\mathbf{P}^K$ depends on $\mathbf{l}$ and K. Its computation is discussed in the following sections. Note that in many applications, $\mathbf{P}^K$ can be precomputed and stored in a lookup table.

Function $\mathbf{y} = \mathbf{LPWT}\ (\mathbf{x}, \mathbf{l}, K, \textit{direction})$
  if *direction* = "forward"
    $\mathbf{x}_e^K = \text{LEWT}\ (\mathbf{x}, \mathbf{l}, K, \text{"forward"})$
    $\mathbf{y} = \left[\mathbf{x}_e^K\right]_{2^{K+1}m+1:2^{K+1}m+n}$
  elseif *direction* = "inverse"
    $\mathbf{x}_e^K = \mathbf{P}^K \mathbf{x}$
    $\mathbf{y} = \text{LEWT}\ (\mathbf{x}_e^K, \mathbf{l}, K, \text{"inverse"})$
  end

C. General Formulas for P and A

General formulas for $\mathbf{P}^k$ and $\mathbf{A}^k$ can be found by induction. At level $k = 0$, $\mathbf{x}_e^k$ is obtained by simply padding $\mathbf{x}$ with zeros, so that

$$\mathbf{P}^0 = \begin{matrix} n \\ \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} \begin{matrix} m \\ n \\ r+m \end{matrix} \end{matrix} \tag{18}$$

For $k \geq 0$, the non-zero coefficients of $\tilde{\mathbf{x}}^{k+1}$ are given by

$$\mathbf{x}_e^{k+1} = \tilde{\mathbf{A}}^k_{\mathbf{e}^{k+1},\,\mathbf{e}^k} \mathbf{x}_e^k = \tilde{\mathbf{A}}^k_{\mathbf{e}^{k+1},\,\mathbf{e}^k} \mathbf{P}^k \mathbf{x}_c^k \tag{19}$$

and the central coefficients of $\tilde{\mathbf{x}}^{k+1}$ are given by

$$\mathbf{x}_c^{k+1} = \tilde{\mathbf{A}}^k_{\mathbf{c},\,\mathbf{e}^k} \mathbf{P}^k \mathbf{x}_c^k \tag{20}$$

Thus, comparing with Eq. (17) we have

$$\mathbf{A}^k = \tilde{\mathbf{A}}^k_{\mathbf{c},\mathbf{e}^k} \mathbf{P}^k \tag{21}$$

Inverting Eq. (20) to express $\mathbf{x}_c^k$ in terms of $\mathbf{x}_c^{k+1}$, and substituting back into Eq. (19) yields the following expression for $\mathbf{P}^{k+1}$:

$$\mathbf{P}^{k+1} = \tilde{\mathbf{A}}^k_{\mathbf{e}^{k+1},\mathbf{e}^k} \mathbf{P}^k (\mathbf{A}^k)^{-1} \tag{22}$$

Equations (21) and (22) are valid for all values of N and n, but they are not practical computationally for large n since they involve matrices of size $n \times n$ and larger.

D. Efficient Formulas for P and A

More efficient formulas for $\mathbf{A}^k$ and $\mathbf{P}^k$ can be derived by exploiting the block-toeplitz structure of $\tilde{\mathbf{A}}^k$. $\tilde{\mathbf{A}}^k$ is made up of blocks of size $2^k\bar{m} \times 2^k\bar{m}$, where $\bar{m} \equiv 2\lfloor N/4 \rfloor$. Note that $\bar{m}$ is the same as m (the number of additional redundant coefficients on each side per level) for odd values of $N/2$, but is one greater than m for even values of $N/2$. In order to have an integer number of blocks for all levels $1, \ldots, K$, we will assume in this section that n is an integer multiple of $2^K\bar{m}$ (which also means that the roundup constant r is zero), and we will redefine the extended interval $\mathbf{e}$ in terms of $\bar{m}$ instead of m: i.e., $\bar{\mathbf{e}}^k \equiv [-2^k\bar{m} : n + 2^k\bar{m} - 1]$.

Under these assumptions, the matrices $\tilde{\mathbf{A}}^k_{\mathbf{c},\mathbf{c}}$ and $\tilde{\mathbf{A}}^k_{\bar{\mathbf{e}}^{k+1},\bar{\mathbf{e}}^k}$ can both be evenly partitioned into square blocks of size $2^k\bar{m} \times 2^k\bar{m}$:

$$\tilde{\mathbf{A}}^k_{\mathbf{c},\mathbf{c}} = \overbrace{\begin{bmatrix} \mathbf{C}^k & \mathbf{B}^k & & \\ \mathbf{D}^k & \mathbf{C}^k & \ddots & \\ & \ddots & \ddots & \mathbf{B}^k \\ & & \mathbf{D}^k & \mathbf{C}^k \end{bmatrix}}^{n} \Bigg\} n \tag{23}$$

$$\tilde{\mathbf{A}}^k_{\bar{\mathbf{e}}^{k+1},\bar{\mathbf{e}}^k} = \begin{array}{c} \begin{matrix} 2^k\bar{m} & \overbrace{\quad\quad}^{n} & & 2^k\bar{m} \end{matrix} \\ \left[\begin{array}{cccc} \mathbf{B}^k & & & \\ \mathbf{C}^k & \mathbf{B}^k & & \\ \mathbf{D}^k & \mathbf{C}^k & \ddots & \\ & \mathbf{D}^k & \ddots & \mathbf{B}^k \\ & & \ddots & \mathbf{C}^k \\ & & & \mathbf{D}^k \end{array}\right] \end{array} \begin{array}{l} 2^k\bar{m} \\ 2^k\bar{m} \\ \Big\} n \\ \\ 2^k\bar{m} \\ 2^k\bar{m} \end{array} \tag{24}$$

where the sizes shown indicate elements, not blocks, and $\mathbf{C}^k = \tilde{\mathbf{A}}^k_{0:2^k\bar{m}-1,\,0:2^k\bar{m}-1}$, etc. For example, the zeroth-level blocks for $N = 4$ are:

$$\mathbf{B}^0 = \begin{bmatrix} l_1 & 0 \\ h_1 & 0 \end{bmatrix}, \quad \mathbf{C}^0 = \begin{bmatrix} l_3 & l_2 \\ h_3 & h_2 \end{bmatrix}, \quad \mathbf{D}^0 = \begin{bmatrix} 0 & l_4 \\ 0 & h_4 \end{bmatrix}, \tag{25}$$

and for $N = 6$ they are:

$$\mathbf{B}^0 = \begin{bmatrix} l_2 & l_1 \\ h_2 & h_1 \end{bmatrix}, \quad \mathbf{C}^0 = \begin{bmatrix} l_4 & l_3 \\ h_4 & h_3 \end{bmatrix}, \quad \mathbf{D}^0 = \begin{bmatrix} l_6 & l_5 \\ h_6 & h_5 \end{bmatrix}. \tag{26}$$

Notice from Eq. (24) that $\tilde{\mathbf{A}}^k_{\bar{\mathbf{e}}^{k+1},\bar{\mathbf{e}}^k}$ consists of a central $n \times n$ portion ($\tilde{\mathbf{A}}^k_{\mathbf{c},\mathbf{c}}$), an additional block column on the left and right corresponding to the redundant coefficients of $\tilde{\mathbf{x}}^k$, and two additional block rows on the top and bottom corresponding to the redundant coefficients of $\tilde{\mathbf{x}}^{k+1}$.

We will now show by induction that $\mathbf{P}^k$ has the following block structure

$$\mathbf{P}^k = \begin{array}{c} \begin{matrix} \overbrace{\quad\quad\quad}^{n} & & \end{matrix} \\ \left[\begin{array}{ccc} \mathbf{S}^k & \mathbf{0} & \cdots \\ \mathbf{I} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{I} \\ \cdots & \mathbf{0} & -\mathbf{S}^{kT} \end{array}\right] \end{array} \begin{array}{l} 2^k\bar{m} \\ \Big\} n \\ \\ 2^k\bar{m} \end{array} \tag{27}$$

where each block is again $2^k\bar{m} \times 2^k\bar{m}$. This is clearly the case for $k = 0$, with $\mathbf{S}^k = \mathbf{0}$.

Substituting the above equations into Eq. (21) yields the following equation for $\mathbf{A}^k$:

$$\mathbf{A}^k = \tilde{\mathbf{A}}^k_{\mathbf{c},\mathbf{c}} + \operatorname{diag}\left(\mathbf{D}^k\mathbf{S}^k,\ \mathbf{0},\ \ldots,\ \mathbf{0},\ -\mathbf{B}^k\mathbf{S}^{kT}\right). \tag{28}$$

Thus only $2^k\bar{m}$ coefficients at either end of the central interval are affected by the end effects. The inverse of $\mathbf{A}^k$ is given as follows:

$$(\mathbf{A}^k)^{-1} = \operatorname{diag}\left((\mathbf{E}^k)^{-1},\ \mathbf{I},\ \ldots,\ \mathbf{I},\ (\mathbf{F}^k)^{-1}\right)(\tilde{\mathbf{A}}^k_{\mathbf{c},\mathbf{c}})^T, \tag{29}$$

where $\mathbf{E}^k = \mathbf{C}^{kT}\mathbf{C}^k + \mathbf{D}^{kT}\mathbf{D}^k + \mathbf{C}^{kT}\mathbf{D}^k\mathbf{S}^k$ and $\mathbf{F}^k = \mathbf{C}^{kT}\mathbf{C}^k + \mathbf{B}^{kT}\mathbf{B}^k - \mathbf{C}^{kT}\mathbf{B}^k\mathbf{S}^{kT}$. Equation (29) can be verified by postmultiplying it by Eq. (23) and making use of the identities $\mathbf{B}^T\mathbf{B} + \mathbf{C}^T\mathbf{C} + \mathbf{D}^T\mathbf{D} = \mathbf{I}$ and $\mathbf{B}^T\mathbf{C} + \mathbf{C}^T\mathbf{D} = \mathbf{B}^T\mathbf{D} = \mathbf{0}$, which follow from the fact that $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} = \tilde{\mathbf{I}}$.

Substitution into Eq. (22) gives the following update formula for $\mathbf{S}$:

$$\mathbf{S}^{k+1} = \begin{bmatrix} \mathbf{B}^k\mathbf{S}^k \\ \mathbf{C}^k\mathbf{S}^k + \mathbf{B}^k \end{bmatrix} (\mathbf{E}^k)^{-1} \begin{bmatrix} \mathbf{C}^{kT} & \mathbf{D}^{kT} \end{bmatrix}. \tag{30}$$

Note that $\mathbf{S}^{k+1}$ has size $2^{k+1}\bar{m} \times 2^{k+1}\bar{m}$. To show that $\mathbf{P}^{k+1}$ has the structure of Eq. (27), i.e., that the submatrix in the lower right is the negative transpose of the submatrix in the upper left, we must also show that

$$-\mathbf{S}^{(k+1)T} = \begin{bmatrix} -\mathbf{C}^k\mathbf{S}^{kT} + \mathbf{D}^k \\ -\mathbf{D}^k\mathbf{S}^{kT} \end{bmatrix} (\mathbf{F}^k)^{-1} \begin{bmatrix} \mathbf{B}^{kT} & \mathbf{C}^{kT} \end{bmatrix}. \tag{31}$$

This can be shown as follows. Let $\mathbf{H} = \mathbf{S} + \mathbf{C}^{-1}\mathbf{B} = \mathbf{S} - \mathbf{D}^T\mathbf{C}^{-T}$. Then it follows that $\mathbf{HE} = \mathbf{F}^T\mathbf{H}$, which can be verified by multiplying out and applying the orthogonality identities. Therefore, $\mathbf{F}^{-T}\mathbf{H} = \mathbf{HE}^{-1}$ as well. Premultiplying both sides of this equation by $[\mathbf{B}\ \ \mathbf{C}]^T$ and postmultiplying by $\begin{bmatrix} \mathbf{C}^T & \mathbf{D}^T \end{bmatrix}$ yields an equation whose left-hand side can be shown to equal the right-hand side of Eq. (31), and whose right-hand side can be shown to equal the negative transpose of the right-hand side of Eq. (30).

IV. ALGORITHMS FOR GENERAL SIGNAL EXTENSIONS

In this section we will consider the transform of signals extended via the following general linear model:

$$\tilde{\mathbf{x}}^0 = \tilde{\mathbf{P}}^0 \mathbf{x} + \tilde{\mathbf{u}}^0. \tag{32}$$

Here $\mathbf{x}$ is the length-n signal to be extended, $\tilde{\mathbf{x}}^0$ is the extended signal, $\tilde{\mathbf{u}}^0$ is a known infinite-length vector that can be used to model fixed extensions, such as a-priori-known initial and final d.c. levels, and $\tilde{\mathbf{P}}^0$ is a known $\infty \times n$ matrix that can be used to model data-dependent extensions, such as providing continuity of the signal and/or its derivatives across the boundaries. For example, linear extrapolation based on the last two signal coefficients would be modelled by having $\tilde{\mathbf{u}}^0_{n:\infty} = \tilde{\mathbf{0}}$ and

$$\tilde{\mathbf{P}}^0_{n:\infty,:} = \begin{bmatrix} 0 & \dots & 0 & -1 & 2 \\ 0 & \dots & 0 & -2 & 3 \\ 0 & \dots & 0 & -3 & 4 \\ \vdots & & \vdots & \vdots & \vdots \end{bmatrix} \tag{33}$$

Note that this general framework also allows linear extensions whose slope and offset depend on a weighted sum of several signal coefficients (in case the true values are obscured by noise), as well as higher-order polynomial extensions that match higher-order derivatives.

We will again show that the infinite-length transform at level k can be computed from its n central coefficients:

$$\tilde{\mathbf{x}}^k = \tilde{\mathbf{P}}^k \tilde{\mathbf{x}}^k_{\mathbf{c}} + \tilde{\mathbf{u}}^k. \tag{34}$$

Assume that Eq. (34) is true for level k. Then from Eq. (8), we have

$$\tilde{\mathbf{x}}^{k+1} = \tilde{\mathbf{A}}^k \tilde{\mathbf{P}}^k \tilde{\mathbf{x}}^k_{\mathbf{c}} + \tilde{\mathbf{A}}^k \tilde{\mathbf{u}}^k \tag{35}$$

Selecting the central elements of $\tilde{\mathbf{x}}^{k+1}$ and solving for $\tilde{\mathbf{x}}^k_{\mathbf{c}}$ yields:

$$\tilde{\mathbf{x}}^k_{\mathbf{c}} = (\tilde{\mathbf{A}}^k_{\mathbf{c},:} \tilde{\mathbf{P}}^k)^{-1} (\tilde{\mathbf{x}}^{k+1}_{\mathbf{c}} - \tilde{\mathbf{A}}^k_{\mathbf{c},:} \tilde{\mathbf{u}}^k) \tag{36}$$

as long as $\tilde{\mathbf{A}}^k_{\mathbf{c},:} \tilde{\mathbf{P}}^k$ (which, note, is $n \times n$) is non-singular. Substituting this result back in

Eq. (35) allows us to write the update formulas for $\tilde{\mathbf{P}}^k$ and $\tilde{\mathbf{u}}^k$:

$$\tilde{\mathbf{P}}^{k+1} = \tilde{\mathbf{A}}^k \tilde{\mathbf{P}}^k (\tilde{\mathbf{A}}^k_{\mathbf{c},:} \tilde{\mathbf{P}}^k)^{-1} \tag{37}$$

$$\tilde{\mathbf{u}}^{k+1} = \left(\tilde{\mathbf{A}}^k - \tilde{\mathbf{A}}^k \tilde{\mathbf{P}}^k (\tilde{\mathbf{A}}^k_{\mathbf{c},:} \tilde{\mathbf{P}}^k)^{-1} \tilde{\mathbf{A}}^k_{\mathbf{c},:} \right) \tilde{\mathbf{u}}^k \tag{38}$$

Computational details depend on the particular extensions. If the initial (final) extension depends on at most the first (last) m elements of x, then the rows of $\tilde{\mathbf{P}}^k$ before $-2^k m$ and after $r + 2^k m$ (which are zero in the zero-padded case) are given by the corresponding rows of $\tilde{\mathbf{A}}^{k-1} \cdots \tilde{\mathbf{A}}^0 \tilde{\mathbf{P}}^0$ and can often be expressed by a simple formula. If this is the case, then the computational requirements for finding $\tilde{\mathbf{P}}^k$ are not much greater than in the zero-padded case. However, there is no guarantee that for any particular set of extensions $\tilde{\mathbf{A}}^k_{\mathbf{c},:} \tilde{\mathbf{P}}^k$ will be well conditioned or even non-singular. If $\tilde{\mathbf{A}}^k_{\mathbf{c},:} \tilde{\mathbf{P}}^k$ is ill conditioned, it may be desirable to use a different set of n coefficients of $\tilde{\mathbf{x}}^k$, rather than the central ones, that convey the information of the original signal more clearly.

V. DISCUSSION

Many signals, by their very nature, begin and end on a zero baseline. The natural way to extend these signals is by zero padding. Unfortunately, the transform of a zero-padded length-n signal has more than n non-zero coefficients, the extra ones corresponding to basis functions that overlap the boundaries from the outside. We have shown in this paper that these peripheral transform coefficients are redundant, and that only the n central transform coefficients are needed to exactly reconstruct the original signal. Thus the signal and the central transform coefficients can be considered an invertible, length-preserving transform pair.

Zero padding is preferable to periodic replication in many applications because it avoids wrap-around problems with little increase in computational cost. (In this way wavelet analysis is different from fourier analysis, in which periodic replication is necessary due to the periodic nature of the basis functions. Note also that in fourier analysis the term "zero padding" refers to extending the signal with a finite (rather than an infinite) number of

zeros.) Periodic replication does retain a certain attractiveness in wavelet analysis, however, due to its computational simplicity and its convenient analytical properties (the transform matrix $\mathbf{A}$ becomes circulant).

For signals that do not begin or end on the baseline, the appropriate methods of extension are mirroring or extrapolation, in order to prevent discontinuities at the boundaries. The relative advantages and disadvantages of extrapolation versus mirroring are less clear than those of zero padding versus periodizing. Mirroring will remain appropriate in many applications because of its simplicity, while extrapolation can be considered in applications in which it is important to minimize reflected signal detail or to avoid discontinuous derivatives at the boundaries. Extrapolation is further complicated by the need to choose a window over which to average the extrapolation parameters, while mirroring handles noisiness in boundary d.c. levels automatically.

In the appendix we present practical implementations of the algorithms LPWT, which computes the length-preserving zero-padded transform, and LEWT, which computes the length-extending transform (including redundant coefficients). The major difference these two algorithms is the tradeoff between computation and storage. LEWT requires $2nN$ operations in either direction. LPWT also requires $2nN$ operations in the forward direction, but it requires an additional $\approx K^2N^2/2$ operations in the inverse direction (not counting the precomputation of $\mathbf{S}$). On the other hand, the LPWT transform vector has length n, while the LEWT transform vector contains an additional $\approx KN$ redundant coefficients. Thus in applications with no storage constraints, LEWT might be prefered for its simplicity, while in applications in which storage is expensive, LPWT might be preferred for its parsimony.

LPWT would also be preferred to LEWT in applications in which signals are to be compared in the transform domain. For each signal vector $\mathbf{x}$, there are a multiplicity of vectors $\mathbf{y}$ for which $\mathrm{LEWT}^{-1}(\mathbf{y}) = \mathbf{x}$. (Only one of them is actually the non-zero part of the infinite-length transform of the zero-padded signal; the others are transforms of signals that match $\mathbf{x}$ over the analysis interval but are not identically zero outside. Although the latter vectors cannot arise as outputs of LEWT, they could arise after processing in the transform domain.) Since the distance (in the euclidean metric, say) between any two of the $\mathbf{y}$'s can

be arbitrarily large, comparison of signals in the non-length-preserving transform domain is unreliable.

APPENDIX
ZERO-PADDED WAVELET TRANSFORM ROUTINE

A practical routine for the finite-length wavelet transform with zero padding is given in pseudocode below. It can be used to compute either the length-preserving or the length-extending transform. The signal length is assumed to be a multiple of $2^{K+1}\lfloor N/4 \rfloor$ for the length-preserving transform, or of 2^K for the length-extending transform. A matlab version of this routine has been tested using the "nearest-to-linear-phase" wavelets of [2] (divided by $\sqrt{2}$) for values of N from 8 to 18, with zero-mean, unit-variance random signals of length $n = 384$ and with K ranging from 1 to 4. In all cases the rms error between the original signal and the signal reconstructed from the length-preserving transform was less than 10^{-10}.

The routine follows the spirit of algorithms LEWT and LPWT of the text, although it differs in several technical details. For one thing, the length-extending transform is computed using repeated convolutions rather than matrix multiplications. For another, the routine uses a different data format than the interleaved format of Eq. (14). The interleaved format, although it simplifies analysis, is inefficient for practical use. This is due to the increased spacing between the coefficients of different levels, so that $2^{k+1}\bar{m}$ storage locations are allocated to store only $2(k+1)\bar{m}$ redundant coefficients. The routine uses instead the following compact storage format:

$$\hat{\mathbf{x}}^k = [\mathbf{a}_l^k; \mathbf{d}_l^1; \mathbf{d}_l^2; \ldots; \mathbf{d}_l^k; \mathbf{x}_c^k; \mathbf{a}_r^k; \mathbf{d}_r^1; \ldots; \mathbf{d}_r^k] \tag{39}$$

where l and r refer to the left and right non-zero redundant coefficients, whose indices are $[-m, \ldots, -1]$ and $[n/(2^k m), \ldots, n/(2^k m) + m - 1]$, respectively.

The routine also uses a compacted version of the matrix $\mathbf{S}^K$, since as defined in Section III, only $(K+1)m$ of its rows (corresponding to the redundant coefficients) and $(K+1)m$ of its columns (corresponding to the first m central coefficients at each level) are non-zero.

The routine uses the compacted and permuted matrix $\hat{\mathbf{S}}^K$, which maps as follows:

$$\begin{bmatrix} \mathbf{a}_l^K \\ \mathbf{d}_l^1 \\ \vdots \\ \mathbf{d}_l^K \end{bmatrix} = \hat{\mathbf{S}}^K \begin{bmatrix} \mathbf{a}_f^K \\ \mathbf{d}_f^1 \\ \vdots \\ \mathbf{d}_f^K \end{bmatrix} \tag{40}$$

where f refers to the first m central coefficients, whose indices are $[0, \ldots, m-1]$. The matrix $\hat{\mathbf{S}}^K$ is computed by the function EdgeToRedundant, which also makes use of two other versions of $\mathbf{S}^k$. The first version, $\bar{\mathbf{S}}^k$, is based on the block size $\bar{m}$ rather than m. It is defined as in Eq. (40), but replacing K by k, l by $\bar{l}$, and f by $\bar{f}$, where $\bar{l}$ refers to indices $[-\bar{m}, \ldots, -1]$ and $\bar{f}$ refers to indices $[0, \ldots, \bar{m}-1]$. The second version, $\check{\mathbf{S}}^k$ interleaves the kth level approximation and detail coefficients:

$$\begin{bmatrix} \mathrm{mix}(\mathbf{a}_{\bar{l}}^k, \mathbf{d}_{\bar{l}}^k) \\ \mathbf{d}_{\bar{l}}^1 \\ \vdots \\ \mathbf{d}_{\bar{l}}^{k-1} \end{bmatrix} = \check{\mathbf{S}}^k \begin{bmatrix} \mathrm{mix}(\mathbf{a}_{\bar{f}}^k, \mathbf{d}_{\bar{f}}^k) \\ \mathbf{d}_{\bar{f}}^1 \\ \vdots \\ \mathbf{d}_{\bar{f}}^{k-1} \end{bmatrix} \tag{41}$$

where $\mathrm{mix}(\mathbf{a}, \mathbf{d}) \equiv [a_0, d_0, a_1, d_1, \ldots, a_{\bar{m}}, d_{\bar{m}}]^T$. The update formula for $\check{\mathbf{S}}^k$ is obtained by reordering the rows and columns of Eq. (30):

$$\check{\mathbf{S}}^{k+1} = \begin{bmatrix} \mathbf{B}^0 \bar{\mathbf{S}}_{\mathbf{i},\mathbf{j}}^k & \mathbf{0} \\ \mathbf{C}^0 \bar{\mathbf{S}}_{\mathbf{i},\mathbf{i}}^k + \mathbf{B}^0 & \mathbf{C}^0 \bar{\mathbf{S}}_{\mathbf{i},\mathbf{j}}^k \\ \bar{\mathbf{S}}_{\mathbf{j},\mathbf{i}}^k & \bar{\mathbf{S}}_{\mathbf{j},\mathbf{j}}^k \end{bmatrix} \begin{bmatrix} \mathbf{Q}^{-1} & -\mathbf{Q}^{-1}\mathbf{C}^{0T}\mathbf{D}^0\bar{\mathbf{S}}_{\mathbf{i},\mathbf{j}}^k \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{C}^{0T} & \mathbf{D}^{0T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{42}$$

where $\mathbf{i} = [1, \ldots, \bar{m}]$, $\mathbf{j} = [\bar{m}+1, \ldots, k\bar{m}]$, and

$$\mathbf{Q} = \mathbf{C}^{0T}\mathbf{C}^0 + \mathbf{D}^{0T}\mathbf{D}^0 + \mathbf{C}^{0T}\mathbf{D}^0\bar{\mathbf{S}}_{\mathbf{i},\mathbf{j}}^k \tag{42}$$

$\bar{\mathbf{S}}^{k+1}$ is then computed from $\check{\mathbf{S}}^{k+1}$ by reordering the rows and columns. Note that the matrix to be inverted in Eq. (42) is only $\bar{m} \times \bar{m}$, while the matrix to be inverted in in Eq. (30) is $2^k\bar{m} \times 2^k\bar{m}$. Finally, $\bar{\mathbf{S}}^K$ is compacted to $\hat{\mathbf{S}}^K$ by deleting every $\bar{m}$th row and column if $N/2$

is even. In many applications, of course, $\hat{\mathbf{S}}^K$ can be precomputed and stored in a look-up table.

Function $\mathbf{y} = \textbf{WaveletTransform}\ (\mathbf{x}, \mathbf{l}, K, \textit{direction}, \textit{option})$;
$N = \text{length}(\mathbf{l})$; $n = \text{length}(\mathbf{x})$; $m = N/2 - 1$;
$\mathbf{h} = \mathbf{0}_{1:N}$; for $i = 1 : N$; $\mathbf{h}_i = -1^i \mathbf{l}_{N+1-i}$; end;
if *direction* = "forward";
if *option* = "length-preserving";
$f = 2^{K+1}\lfloor N/4 \rfloor$;
elseif *option* = "length-extending";
$f = 2^K$;
endif;
$r = f\lceil n/f \rceil - n$;
$\mathbf{a} = [\mathbf{0}_{1:m};\ \mathbf{x};\ \mathbf{0}_{1:r+m}]$;
$\mathbf{y} = \mathbf{0}_{1:n+r+2(K+1)m}$;
for $k = 1 : K$;
$\mathbf{d} = \text{ConvolveCompress}(\mathbf{a}, \mathbf{h})$;
$\mathbf{a} = \text{ConvolveCompress}(\mathbf{a}, \mathbf{l})$;
$\mathbf{y}_{\text{detail}(k,n+r,N,K)} = \mathbf{d}$;
end;
$\mathbf{y}_{\text{approx}(n+r,N,K)} = \mathbf{a}$;
if *option* = "length-preserving"; $\mathbf{y} = \mathbf{y}_{1:n+r}$; endif;

elseif *direction* = "inverse";
if *option* = "length-preserving";
$\mathbf{S} = \text{EdgeToRedundant}(\mathbf{l}, \mathbf{h}, K)$;
$\mathbf{x} = [\mathbf{x};\ \mathbf{S}\mathbf{x}_{\text{left}(N,K)};\ -\mathbf{S}^T\mathbf{x}_{\text{right}(n,N,K)}]$;
elseif *option* = "length-extending";
$n = n - 2(K+1)m$;
endif;
$\mathbf{a} = \mathbf{x}_{\text{approx}(n,N,K)}$;
for $k = K : -1 : 1$;
$\mathbf{d} = \mathbf{x}_{\text{detail}(k,n,N,K)}$;
$\mathbf{a} = \text{DilateConvolve}(\mathbf{a}, \mathbf{l}) + \text{DilateConvolve}(\mathbf{d}, \mathbf{h})$;
end;
$\mathbf{y} = \mathbf{a}_{m+1:m+n}$;
endif;

Function $\mathbf{y} = \textbf{ConvolveCompress}\ (\mathbf{x}, \mathbf{f})$;
$N = \text{length}(\mathbf{f})$;
$\mathbf{y} = \text{convolve}(\mathbf{x}, \mathbf{f}_{N:-1:1})$; $\mathbf{y} = \mathbf{y}_{2:2:\text{length}(\mathbf{y})}$;

Function $\mathbf{y} = \textbf{DilateConvolve}\ (\mathbf{x}, \mathbf{f})$;
$N = \text{length}(\mathbf{f})$; $n = \text{length}(\mathbf{x})$;
$\mathbf{y} = \mathbf{0}_{1:2n-1}$; $\mathbf{y}_{1:2:2n-1} = \mathbf{x}$; $\mathbf{y} = [\text{convolve}(\mathbf{y}, \mathbf{f})]_{N-1:2n}$;

Function $\mathbf{S} = \textbf{EdgeToRedundant}\ (\mathbf{l}, \mathbf{h}, K)$;
$N = \text{length}(\mathbf{l})$; $\bar{m} = 2\lfloor N/4 \rfloor$;
$\mathbf{S} = \mathbf{0}_{1:\bar{m},\,1:\bar{m}}$; $\mathbf{A} = \mathbf{0}_{1:\bar{m},1:3\bar{m}}$;

if $\lfloor N/4 \rfloor = N/4$; $p = 1$; else; $p = 0$; endif;
for $i = 1 : 2 : \bar{m}$; $\mathbf{A}_{i:i+1,\, i+p:i+p+N-1} = [\, \mathbf{l} \quad \mathbf{h} \,]^T$; end;
$\mathbf{D} = \mathbf{A}_{:,\, 1:\bar{m}}$; $\mathbf{C} = \mathbf{A}_{:,\, \bar{m}+1:2\bar{m}}$; $\mathbf{B} = \mathbf{A}_{:,\, 2\bar{m}+1:3\bar{m}}$;
for $k = 1 : K$;
$\mathbf{i} = [1 : \bar{m}]$; $\mathbf{j} = [\bar{m} + 1 : k\bar{m}]$;
$\mathbf{p} = [[1 : 2 : 2\bar{m}],\, [2\bar{m} + 1 : (k+1)\bar{m}],\, [2 : 2 : 2\bar{m}]]$;
$\mathbf{Q} = (\mathbf{C}^T\mathbf{C} + \mathbf{D}^T\mathbf{D} + \mathbf{C}^T\mathbf{D}\mathbf{S}_{\mathbf{i},\mathbf{i}})^{-1}$;

$$\mathbf{S} = \begin{bmatrix} \mathbf{B}\mathbf{S}_{\mathbf{i},\mathbf{i}} \\ \mathbf{C}\mathbf{S}_{\mathbf{i},\mathbf{i}} + \mathbf{B} \\ \mathbf{S}_{\mathbf{j},\mathbf{i}} \end{bmatrix} \mathbf{Q} \, [\, \mathbf{C}^T \quad \mathbf{D}^T \quad -\mathbf{C}^T\mathbf{D}\mathbf{S}_{\mathbf{i},\mathbf{j}} \,] + \begin{bmatrix} \mathbf{0}_{\mathbf{i},1:2\bar{m}} & \mathbf{B}\mathbf{S}_{\mathbf{i},\mathbf{j}} \\ \mathbf{0}_{\mathbf{i},1:2\bar{m}} & \mathbf{C}\mathbf{S}_{\mathbf{i},\mathbf{j}} \\ \mathbf{0}_{\mathbf{j},1:2\bar{m}} & \mathbf{S}_{\mathbf{j},\mathbf{j}} \end{bmatrix};$$

$\mathbf{S} = \mathbf{S}_{\mathbf{p},\mathbf{p}}$;
end;
if $\lfloor N/4 \rfloor = N/4$; $\mathbf{S}_{1:\bar{m}:(K+1)\bar{m},:} = [\,]$; $\mathbf{S}_{:,\bar{m}:\bar{m}:(K+1)\bar{m}} = [\,]$; endif;

Function $\mathbf{i} = \mathbf{detail}\ (k, n, N, K)$;
% Returns indices of detail coefficients for scale k.
$m = N/2 - 1$; $i_1 = n + km$; $i_2 = n + (K + k + 1)m$;
$\mathbf{i} = [[i_1 + 1 : i_1 + m],\, [2^{k-1} + 1 : 2^k : n],\, [i_2 + 1 : i_2 + m]]$;

Function $\mathbf{i} = \mathbf{approx}\ (n, N, K)$;
% Returns indices of approximation coefficients.
$m = N/2 - 1$; $i_1 = n$; $i_2 = n + (K + 1)m$;
$\mathbf{i} = [[i_1 + 1 : i_1 + m],\, [1 : 2^K : n],\, [i_2 + 1 : i_2 + m]]$;

Function $\mathbf{i} = \mathbf{left}\ (N, K)$;
% Returns indices of left-edge central coefficients.
$m = N/2 - 1$; $\mathbf{i} = [1 : 2^K : 2^K m]$;
for $k = 1 : K$; $\mathbf{i} = [\mathbf{i},\, [2^{k-1} + 1 : 2^k : 2^k m]]$; end;

Function $\mathbf{i} = \mathbf{right}\ (n, N, K)$;
% Returns indices of right-edge central coefficients.
$m = N/2 - 1$; $\mathbf{i} = [n - 2^K m + 1 : 2^K : n]$;
for $k = 1 : K$; $\mathbf{i} = [\mathbf{i},\, [n + 2^{k-1} - 2^k m + 1 : 2^k : n]]$; end;

REFERENCES

[1] I. Daubechies, "Orthonormal bases of compactly supported wavelets." *Comm. Pure and Applied Math.*, vol. 41, pp. 909-1006, 1988.

[2] I. Daubechies, "Orthonormal bases of compactly supported wavelets. II. Variations on a theme," submitted to *SIAM J. Math. Anal.*, 1990.

[3] G. Strang, "Wavelets and dilation equations: a brief introduction," *SIAM Review*, vol. 31, pp 614-627, 1989.

[4] S.G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-691, 1989.

[5] O. Fioul and M. Vetterli, "Wavelets and signal processing," *IEEE Signal Proc. Magazine*, vol. 8, pp. 14-38, 1991.

[6] C. Taswell and K.C. McGill, "Wavelet transform algorithms for finite-duration discrete-time signals: signal-end effects," *Numerical Analysis Project*, NA-91-07, Stanford University, 1991.