Wavelet Transform Algorithms for Finite-Duration Discrete-Time Signals*

Carl Taswell† and Kevin C. McGill‡

Original 1 August 1991, Revised 1 October 1993

Abstract

The algorithms *split* for the wavelet transform and *merge* for the inverse wavelet transform are presented for finite-duration discrete-time signals of arbitrary length not restricted to a power of 2. Alternative matrix- and vector-filter implementations of alternative truncated, circulant, and extended versions are discussed. Matrix- and vector-filter implementations yield identical results and enhance, respectively, didactic conceptualization and computational efficiency. Truncated, circulant, and extended versions produce the signal-end effects of, respectively, errors, periodization, and redundancy in the transform coefficients. The use of any one of these three versions avoids the signal-end effects associated with the other two versions. Additional alternatives which eliminate all signal-end effects (albeit at the cost of increased algorithmic complexity) are discussed briefly.

Categories and Subject Descriptors:

G.1.2 Numerical Analysis: Approximations;

G.4 Mathematics of Computing: Mathematical Software;

I.4.5 Transform Methods: Reconstruction.

General Terms:

Algorithms, Signal Processing, and Waveform Analysis.

Additional Key Words and Phrases:

Wavelets, Wavelet Transform, Multiresolution Analysis.

---

*A preprint of this paper has been available as Numerical Analysis Project Manuscript NA-91-07, Department of Computer Science, Stanford University.

†C. Taswell is with Scientific Computing and Computational Mathematics, Bldg 460 Room 314, Stanford University, Stanford, CA 94305-2140; Email: taswell@sccm.stanford.edu; Phone: 415-723-4101.

‡K. C. McGill is with Rehabilitation Research and Development Center, Veterans Affairs Medical Center, Palo Alto, CA 94304-1200; Email: mcgill@roses.stanford.edu; Phone: 415-858-3991x4477.

1 Introduction

The theory of wavelet transforms and multiresolution analyses has been presented and reviewed by various authors [3, 6, 10, 5, 9]. However, these and other articles that have appeared in the literature do not provide an exposition of algorithms sufficiently detailed to expedite their use in practical applications by those not specializing in the theory. Moreover,these general algorithms have been presented for finite-duration discrete-time signals restricted to a length equal to a power of 2. Finally, the issues of errors, periodization, and redundancy in the transform coefficients at the signal ends have been mentioned but not sufficiently documented to allow the user to select one of several alternative transform algorithms most suited to the requirements of the application. In this report, we discuss these issues as they relate to specific algorithms for finite-duration discrete-time signals of arbitrary length not restricted to a power of 2. We assume that the reader is familiar (*cf.* [10]) with the basic concepts of multiresolution analysis and the wavelet transform, in particular, the decomposition of a signal into details and approximations on a dyadic scale with individual transform coefficients derived from dilates and translates of the wavelet.

2 Wavelet Transform Algorithms

The wavelet transform algorithms presented here perform multiresolution analyses based on the pyramid algorithm [1, 6, 9], with two modifications to allow transformation of signals of arbitrary length. First, appropriately sized filter matrices are used at each multiresolution scale; and second, an additional bookkeeping vector is used to track the lengths of the details at each scale. Thus, for multiresolution scales $j = 1 \ldots J$, the algorithms decompose the signal $\mathbf{x}$ into the details $\mathbf{d}^j = \mathbf{H}^j \mathbf{a}^{j-1}$ and approximations $\mathbf{a}^j = \mathbf{L}^j \mathbf{a}^{j-1}$ using the initialization $\mathbf{a}^0 = \mathbf{x}$ and the high- and low-pass filter matrices $\mathbf{H}^j$ and $\mathbf{L}^j$ corresponding to the wavelet and scaling functions ψ and ϕ, respectively.[1] The algorithms

[1]Matrices and vectors are denoted with upper- and lower-case letters, respectively. Matrices, vectors, and partitions composed of two or more elements from matrices or vectors are denoted with bold fonts. Scalars

concatenate the details (during successive cycles of the j loop) and final approximation (after completion of the j loop) in the wavelet transform vector

$$\hat{\mathbf{x}} = \begin{bmatrix} \mathbf{d}^1 \\ \vdots \\ \mathbf{d}^J \\ \mathbf{a}^J \end{bmatrix} \tag{1}$$

and store the lengths of the initial approximation and details in the bookkeeping vector

$$\mathbf{b} = \begin{bmatrix} \text{length}(\mathbf{a}^0) \\ \text{length}(\mathbf{d}^1) \\ \vdots \\ \text{length}(\mathbf{d}^J) \end{bmatrix} \tag{2}$$

With this data storage convention for the transform $\hat{\mathbf{x}}$ and bookkeeper $\mathbf{b}$, the i^{th} transform coefficient $\hat{x}_i$ is the detail coefficient d_k^j derived from the j^{th} dilation and k^{th} translation of the wavelet function ψ. Thus, $\hat{x}_i = d_k^j$ where $i = k + \sum_{l=2}^{j} b_l$ for $1 \le j \le J$ and $1 \le k \le b_{j+1}$. Using this convention together with the necessary bookkeeping operations, the algorithms then reconstruct $\mathbf{x} = \mathbf{a}^0$ from $\hat{\mathbf{x}}$ and $\mathbf{b}$ by the iteration $\mathbf{a}^{j-1} = (\mathbf{L}^j)^{\mathrm{T}}\mathbf{a}^j + (\mathbf{H}^j)^{\mathrm{T}}\mathbf{d}^j$ for scales $j = J \ldots 1$. For cases where exact reconstruction may not be possible, let $\mathbf{a} = \mathbf{a}^0$ denote the inverse transform assumed to approximate the signal so that the error $\mathbf{e} = \mathbf{x} - \mathbf{a}$ and relative mean square error $\text{rmse}(\mathbf{a}) = \|\mathbf{e}\|_2 / \|\mathbf{x}\|_2$ can be calculated for $\mathbf{a} \approx \mathbf{x}$.

The filter matrices $\mathbf{H}^j$ and $\mathbf{L}^j$ can be constructed from the filter vectors $\mathbf{h}$ and $\mathbf{l}$ composed of the coefficients specifying the wavelet and scaling functions ψ and ϕ, respectively. However, to increase computational efficiency and decrease memory storage, the necessary matrix-vector multiplications can be performed implicitly using only the filter vectors without explicitly constructing the filter matrices. The following algorithms for the discrete-time wavelet transform (*split*) and inverse discrete-time wavelet transform (*merge*) account for both possible (matrix-filter and vector-filter) implementations for which the

and single elements from matrices or vectors are denoted with nonbold fonts. All indices for matrices and vectors begin with the value 1, not 0.

various versions are selected by the switch *alt*. Three different versions of each of the two implementations are presented in the algorithms and/or discussed in the text; they are named according to the filters they use: truncated matrix filters (*tmf*), circulant matrix filters (*cmf*), extended matrix filters (*emf*), truncated vector filters (*tvf*), circulant vector filters (*cvf*), and extended vector filters (*evf*).

Algorithm 1 Given the signal $\mathbf{x}$, scaling filter $\mathbf{l}$, version switch *alt*, and desired transform depth J_{des}, the function *split* returns the transform $\hat{\mathbf{x}}$, bookkeeper $\mathbf{b}$, and computed transform depth J and length m. If J_{des} is not input, its value defaults to ∞ so that the maximum possible transform depth J is computed.

function $[\hat{\mathbf{x}}, \mathbf{b}, J, m] = \text{split}(\mathbf{x}, \mathbf{l}, alt, J_{\text{des}})$
  if #(input arguments) < 4
    $J_{\text{des}} = \infty$
  end
  $n = \text{length}(\mathbf{x})$
  $\mathbf{a} = \mathbf{x}$
  $[\mathbf{h}, \mathbf{l}^{\text{R}}, \mathbf{h}^{\text{R}}, N] = \text{l2h}(\mathbf{l})$
  $[J, m] = \text{wtdl}(n, N, alt, J_{\text{des}})$
  $\mathbf{b} = \mathbf{0}_{1:J+1}$
  $b_1 = n$
  $\hat{\mathbf{x}} = \mathbf{0}_{1:m}$
  $p = 1$
  for $j = 1 : J$
    % this if block for alternative versions of $\mathbf{d}^j = \mathbf{H}^j \mathbf{a}^{j-1}$
    if ($alt =$ '*tmf*')
      $\mathbf{d} = [\text{fvecmat}(\mathbf{h}^{\text{R}}, N, \lceil b_j/2 \rceil, b_j, 0)]\mathbf{a}$
    elseif ($alt =$ '*cmf*')
      $\mathbf{d} = [\text{fvecmat}(\mathbf{h}^{\text{R}}, N, \lceil b_j/2 \rceil, b_j, 1)]\mathbf{a}$
    elseif ($alt =$ '*emf*')
      $\mathbf{d} = [\text{fvecmat}(\mathbf{h}^{\text{R}}, N, \lfloor (N + b_j - 1)/2 \rfloor, b_j)]\mathbf{a}$
    elseif ($alt =$ '*evf*')
      $\mathbf{d} = \text{comp}(\text{conv}(\mathbf{h}, \mathbf{a}))$
    end
    $b_{j+1} = \text{length}(\mathbf{d})$
    $q = p + b_{j+1} - 1$
    $\hat{\mathbf{x}}_{p:q} = \mathbf{d}$
    $p = q + 1$
    % this if block for alternative versions of $\mathbf{a}^j = \mathbf{L}^j \mathbf{a}^{j-1}$
    if ($alt =$ '*tmf*')
      $\mathbf{a} = [\text{fvecmat}(\mathbf{l}^{\text{R}}, N, \lceil b_j/2 \rceil, b_j, 0)]\mathbf{a}$

```
        elseif (alt = 'cmf')
            a = [fvecmat(l^R, N, ⌈b_j/2⌉, b_j, 1)]a
        elseif (alt = 'emf')
            a = [fvecmat(l^R, N, ⌊(N + b_j − 1)/2⌋, b_j)]a
        elseif (alt = 'evf')
            a = comp(conv(l, a))
        end
    end
    q = p + b_{J+1} − 1
    x̂_{p:q} = a
```

Algorithm 2 Given the transform $\hat{\mathbf{x}}$, bookkeeper $\mathbf{b}$, scaling filter $\mathbf{l}$, and version switch *alt*, the function *merge* inverts the transform $\hat{\mathbf{x}}$, and returns the approximation $\mathbf{a} = \mathbf{a}^0$ of the signal $\mathbf{x}$.

```
function a = merge(x̂, b, l, alt)
    N = length(l)
    [h, l^R, h^R, N] = l2h(l)
    J = length(b) − 1
    q = length(x̂)
    p = q − b_{J+1} + 1
    a = x̂_{p:q}
    for j = J : −1 : 1
        q = p − 1
        p = q − b_{j+1} + 1
        d = x̂_{p:q}
        % this if block for alternative versions of a^{j−1} = (L^j)^T a^j + (H^j)^T d^j
        if (alt = 'tmf')
            a = [fvecmat(l^R, N, b_{j+1}, b_j, 0)]^T a + [fvecmat(h^R, N, b_{j+1}, b_j, 0)]^T d
        elseif (alt = 'cmf')
            a = [fvecmat(l^R, N, b_{j+1}, b_j, 1)]^T a + [fvecmat(h^R, N, b_{j+1}, b_j, 1)]^T d
        elseif (alt = 'emf')
            a = [fvecmat(l^R, N, b_{j+1}, b_j)]^T a + [fvecmat(h^R, N, b_{j+1}, b_j)]^T d
        elseif (alt = 'evf')
            a = [conv(l^R, dila(a)) + conv(h^R, dila(d))]_{N:N+b_j−1}
        end
    end
```

In these algorithms, $\mathbf{v}^{\mathrm{R}}$ denotes the vector $\mathbf{v}$ with elements in reverse order; $\lfloor \cdot \rfloor$ denotes the largest integer less than or equal to its argument; and $\lceil \cdot \rceil$ denotes the smallest integer greater than or equal to its argument. The vector-filter versions *evf* of *split* and *merge*

call the functions *conv*, *comp*, and *dila*, which are discussed in Section 6. Each of the matrix-filter versions *tmf*, *cmf*, and *emf* of *split* and *merge* call the following function for the construction of a matrix filter from a vector filter (*fvecmat*).

Algorithm 3 Given the vector filter $\mathbf{f}$ of length N, desired size of r rows and c columns for the matrix filter $\mathbf{F}$, the function *fvecmat* returns the extended matrix filter $\mathbf{F} = \mathbf{F}_{\text{emf}}$. If the logical switch *cir* is passed as a fifth input argument, then *fvecmat* returns the circulant matrix filter $\mathbf{F} = \mathbf{F}_{\text{cmf}}$ for $cir = 1$ and the truncated matrix filter $\mathbf{F} = \mathbf{F}_{\text{tmf}}$ for $cir = 0$.

function $\mathbf{F} = \text{fvecmat}(\mathbf{f}, N, r, c, cir)$
  $k = N - 2$
  $\mathbf{F} = \mathbf{0}_{1:r,1:2r+k}$
  for $i = 1 : r$
    $\mathbf{F}_{i,2i-1:2i+k} = \mathbf{f}$
  end
  if #(input arguments) = 5
    $k = k/2$
    if *cir*
      $r = 2r$
      $\mathbf{F}_{:,k+1:2k} = \mathbf{F}_{:,k+1:2k} + \mathbf{F}_{:,r+k+1:r+2k}$
      $\mathbf{F}_{:,r+1:r+k} = \mathbf{F}_{:,r+1:r+k} + \mathbf{F}_{:,1:k}$
    end
  end
  $\mathbf{F} = \mathbf{F}_{:,k+1:k+c}$

These algorithms assume that the filter vectors $\mathbf{h}$, $\mathbf{l}$, and $\mathbf{f}$ are implemented as row vectors whereas all other vectors (such as signal $\mathbf{x}$, detail $\mathbf{d}$, and approximation $\mathbf{a}$) in linear algebraic equations are implemented as column vectors. Furthermore, these algorithms require that the coefficient sum of the filter vector $\mathbf{l}$ is normalized to $\sqrt{2}$, that is, $\sum_{i=1}^{N} l_i = \sqrt{2}$. The family of closest-to-linear-phase compactly-supported orthogonal wavelet and scaling functions discovered by Daubechies [4] are used for the examples presented throughout this report and are denoted ψ_N and ϕ_N with corresponding filter vectors $\mathbf{h}_N$ and $\mathbf{l}_N$ where N is the number of coefficients required to specify the function and corresponding filter.[2] The wavelet filter $\mathbf{h}$ is generated from the scaling filter $\mathbf{l}$ by the following function (*l2h*).

Algorithm 4 Given the scaling filter $\mathbf{l}$, the function *l2h* returns the corresponding wavelet filter $\mathbf{h}$, the time-reversed filters $\mathbf{l}^{\text{R}}$ and $\mathbf{h}^{\text{R}}$ and their length N.

[2]The N used in this report equals 2 times the N used by Daubechies [4].

function $[\mathbf{h}, \mathbf{l}^{\mathrm{R}}, \mathbf{h}^{\mathrm{R}}, N] = \mathrm{l2h}(\mathbf{l})$
  $N = \mathrm{length}(\mathbf{l})$
  $\mathbf{l}^{\mathrm{R}} = \mathrm{rev}(\mathbf{l})$
  $\mathbf{h} = \mathbf{l}^{\mathrm{R}}$
  for $i = 2 : 2 : N$
    $h_i = -h_i$
  end
  $\mathbf{h}^{\mathrm{R}} = \mathrm{rev}(h)$

In this algorithm, the function *rev* reverses the order of the elements of its vector argument.[3] Finally, *split* also calls the following function for the wavelet transform depth and length (*wtdl*) which necessitates a definition for the computed transform depth J and length $n_{\hat{\mathbf{x}}}$. First, let J_{des} be the desired transform depth. Now for *cmf* and *emf* versions, let J be set to the maximum j such that $1 \leq \mathrm{length}(\mathbf{d}^j) < \mathrm{length}(\mathbf{d}^{j-1})$ and such that the matrix identity $(\mathbf{L}^j)^{\mathrm{T}}\mathbf{L}^j + (\mathbf{H}^j)^{\mathrm{T}}\mathbf{H}^j = \mathbf{I}$ holds true for $j \leq J_{\mathrm{des}}$. Then let $n_{\hat{\mathbf{x}}} = \mathrm{length}(\hat{\mathbf{x}})$ for the transform $\hat{\mathbf{x}}$ associated with this depth J. For the *tmf* version, let J and $n_{\hat{\mathbf{x}}}$ be the same as those obtained for the *cmf* version. The values for the vector versions (*tvf*, *cvf*, and *evf*) are always the same as those for the corresponding matrix versions (*tmf*, *cmf*, and *emf*).

Algorithm 5 Given the signal length $n = n_{\mathbf{x}}$, scaling- and wavelet-filter length N, version switch *alt*, and desired transform depth J_{des}, the function *wtdl* returns the computed transform depth J and transform length $m = n_{\hat{\mathbf{x}}}$.

function $[J, m] = \mathrm{wtdl}(n, N, alt, J_{\mathrm{des}})$
  $J = 0$
  $m = 0$
  if ($alt =$ '*tmf*') or ($alt =$ '*cmf*')
    $N = N/2$
    if (N mod 2)
      $N = N - 2$
    else
      $N = N - 1$
    end
    if ($N < 2$)

[3] In Matlab©, the functions *fliplr* and *flipud* perform this reverse-ordering operation for row and column vectors, respectively.

```
        N = 2
    end
    while (n ≥ N) and (J < J_des)
        J = J + 1
        n = ⌈n/2⌉
        m = m + n
    end
elseif (alt = 'emf') or (alt = 'evf')
    while (n ≥ N) and (J < J_des)
        J = J + 1
        n = ⌊(n + N − 1)/2⌋
        m = m + n
    end
end
m = m + n
```

3 Errors with Truncated Matrix Filters

One way to compute the wavelet transform of a finite-duration signal $\mathbf{x}$ of length $n_{\mathbf{x}}$ is to extend the signal infinitely with zero padding on both ends and then compute the infinite wavelet transform of the padded signal. This infinite transform has only a finite number of non-zero coefficients, but it is a number greater than $n_{\mathbf{x}}$. Therefore, a naive way to achieve $n_{\mathbf{x}}$ transform coefficients for the finite wavelet transform is to truncate the infinite wavelet transform to the length $n_{\mathbf{x}}$. We call the corresponding finite matrix filters "truncated matrix filters". These truncated matrix filters contain both complete and partial (left- and right-end truncated) copies of the corresponding vector filter. For example, given an input vector $\mathbf{x}$ of length $n_{\mathbf{x}} = 12$ and vector filter $\mathbf{f}$ of length $N = 6$ (which for display is

taken as $f_i = i$), then the truncated matrix filter

$$\mathbf{F}_{\text{tmf}} = \begin{bmatrix} 3 & 4 & 5 & 6 & & & & & & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & & & & & & \\ & & 1 & 2 & 3 & 4 & 5 & 6 & & & & \\ & & & & 1 & 2 & 3 & 4 & 5 & 6 & & \\ & & & & & & 1 & 2 & 3 & 4 & 5 & 6 \\ & & & & & & & & 1 & 2 & 3 & 4 \end{bmatrix} \tag{3}$$

contains 4 complete, 1 left-truncated, and 1 right-truncated copies of $\mathbf{f}$ and yields an output vector $\mathbf{y} = \mathbf{F}_{\text{tmf}}\mathbf{x}$ of length $n_{\mathbf{y}} = 6$. In general, $\mathbf{F}_{\text{tmf}}$ has $c = n_{\mathbf{x}}$ columns and $r = \lceil n_{\mathbf{x}}/2 \rceil$ rows. Now, if $\mathbf{f} = \mathbf{l}^{\text{R}}$, then $\mathbf{F}_{\text{tmf}} = \mathbf{L}_{\text{tmf}}$. Similarly, if $\mathbf{f} = \mathbf{h}^{\text{R}}$, then $\mathbf{F}_{\text{tmf}} = \mathbf{H}_{\text{tmf}}$. These matrices represent a single scale-step wavelet transform which is implemented as one cycle through the j loop of *split*. But $\mathbf{L}_{\text{tmf}}\mathbf{L}_{\text{tmf}}^{\text{T}} \neq \mathbf{I} \neq \mathbf{H}_{\text{tmf}}\mathbf{H}_{\text{tmf}}^{\text{T}}$ and $\mathbf{L}_{\text{tmf}}^{\text{T}}\mathbf{L}_{\text{tmf}} + \mathbf{H}_{\text{tmf}}^{\text{T}}\mathbf{H}_{\text{tmf}} \neq \mathbf{I}$ for all $n_{\mathbf{x}}$, so that exact reconstruction with *merge.tmf* of a signal transformed with *split.tmf* is not possible. As a consequence, significant reconstruction error can occur. Figure 1 displays an example of a random zero-mean unit-variance normal signal $\mathbf{x}$ of length $n_{\mathbf{x}} = 100$ and the reconstruction error $\mathbf{e} = \mathbf{x} - \mathbf{a}$ obtained by transforming $\mathbf{x}$ with *split.tmf* ($N = 6$) to yield $\hat{\mathbf{x}}$ of length $n_{\hat{\mathbf{x}}} = 103$ with $\mathbf{b} = [100 \quad 50 \quad 25 \quad 13 \quad 7 \quad 4 \quad 2 \quad 1]$ and then inverse transforming $\hat{\mathbf{x}}$ with *merge.tmf* ($N = 6$) to yield $\mathbf{a}$ of length $n_{\mathbf{a}} = 100$ with relative reconstruction error $\text{rmse}(\mathbf{a}) = 0.229$. Table 1 presents mean $\text{rmse}(\mathbf{a})$ values (averaged from 100 trials) at each combination of varying values of $n_{\mathbf{x}} = 2^J$ and N. In general, $\text{rmse}(\mathbf{a})$ decreases as J and thus $n_{\mathbf{x}} = 2^J$ increases for each fixed value of N. More complicated (*i.e.,*non-monotonic) patterns of results can be observed for values of $n_{\mathbf{x}} \neq 2^J$. Nevertheless, the basic result that $\text{rmse}(\mathbf{a})$ decreases as $n_{\mathbf{x}}$ increases remains valid in the sense of a decreasing trend for increasing values of $n_{\mathbf{x}} \neq 2^J$ between successive values of $n_{\mathbf{x}} = 2^J$.

4 Periodization with Circulant Matrix Filters

Exact reconstruction can be achieved at the expense of periodization. Thus, a second way to compute the wavelet transform of a finite-duration signal is to extend the signal infinitely with periodic replicas on both ends and then truncate the infinite wavelet transform to the length $n_{\mathbf{x}}$ (in a manner analogous to the finite discrete fourier transform). We call the corresponding finite matrix filters "circulant matrix filters". These circulant matrix filters contain wrapped and nonwrapped copies of the corresponding vector filter. The wrapped copies circle around from the right to left sides of the matrix. For the same example associated with $\mathbf{F}_{\text{tmf}}$ in Equation 3, the corresponding circulant matrix filter

$$\mathbf{F}_{\text{cmf}} = \begin{bmatrix} 3 & 4 & 5 & 6 & & & & & & & 1 & 2 \\ 1 & 2 & 3 & 4 & 5 & 6 & & & & & & \\ & & 1 & 2 & 3 & 4 & 5 & 6 & & & & \\ & & & & 1 & 2 & 3 & 4 & 5 & 6 & & \\ & & & & & & 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 6 & & & & & & & 1 & 2 & 3 & 4 \end{bmatrix} \tag{4}$$

contains 4 nonwrapped copies and 2 wrapped copies of $\mathbf{f}$. In general, $\mathbf{F}_{\text{cmf}}$ has the same dimensions as $\mathbf{F}_{\text{tmf}}$. Now let $n_{\min} = N/2-1$ if $N/2$ is even, and $n_{\min} = N/2-2$ if $N/2$ is odd; thus, $n_{\min}$ is always odd because N is always even. Then $\mathbf{L}_{\text{cmf}}\mathbf{L}_{\text{cmf}}^{\text{T}} \neq \mathbf{I} \neq \mathbf{H}_{\text{cmf}}\mathbf{H}_{\text{cmf}}^{\text{T}}$ for all odd $n_{\mathbf{x}}$ and $\mathbf{L}_{\text{cmf}}\mathbf{L}_{\text{cmf}}^{\text{T}} = \mathbf{I} = \mathbf{H}_{\text{cmf}}\mathbf{H}_{\text{cmf}}^{\text{T}}$ for all even $n_{\mathbf{x}} > n_{\min}$. But $\mathbf{L}_{\text{cmf}}^{\text{T}}\mathbf{L}_{\text{cmf}} + \mathbf{H}_{\text{cmf}}^{\text{T}}\mathbf{H}_{\text{cmf}} = \mathbf{I}$ for all $n_{\mathbf{x}} \geq n_{\min}$, so that exact reconstruction with *merge.cmf* of a signal transformed with *split.cmf* is possible. Thus, there is no transform error from circulant matrix filters as there is from truncated matrix filters. However, there is transform periodization that manifests itself when the transform is manipulated prior to inverse transformation. This periodization can be demonstrated by perturbing a single transform coefficient $\hat{x}_i = d_k^j$ and examining the resulting reconstruction error. Figure 2 displays the reconstruction errors obtained by transforming with *split.cmf* ($N = 6$) the same example signal displayed in Figure 1, perturbing either d_1^3 or d_{13}^3, and then inverse transforming with *merge.cmf* ($N = 6$). Since both *tmf* and *cmf* versions yield the same number of transform coefficients

for the same N, $\mathbf{b}$ is the same in both cases and thus in both Figures 1 and 2. The detail coefficients that were perturbed for Figure 2 correspond to the coefficients obtained from the first and last translations of the third dilation of the wavelet. The perturbations were adjusted such that the values of rmse($\mathbf{a}$) for the reconstruction errors in Figure 2 equaled that in Figure 1, thus enabling visual comparison on the same scale. The periodization is clearly visible for both examples: when either the left (d_1^3) or right (d_{13}^3) signal-end detail coefficient is perturbed, the resulting reconstruction error wraps around from either the left to right or right to left ends of the signal, respectively.

5 Redundancy with Extended Matrix Filters

Exact reconstruction without periodization can be achieved at the expense of redundancy resulting in additional coefficients with the transform length $n_{\hat{\mathbf{x}}}$ greater than the signal length $n_{\mathbf{x}}$. Thus, a third way to compute the wavelet transform of a finite-duration signal is to truncate the infinite wavelet transform of the zero-padded signal to the length such that all non-zero coefficients are retained. We call the corresponding finite matrix filters "extended matrix filters". These extended matrix filters contain all the translations of the corresponding vector filter that yield non-zero transform coefficients for the signal points in the index interval $i = 1 \ldots n_{\mathbf{x}}$. Thus, they contain the same number of complete but a greater number of partial (left- and right-end truncated) copies of the corresponding vector filter than do truncated matrix filters. For the same example associated with $\mathbf{F}_{\text{tmf}}$

in Equation 3 and $\mathbf{F}_{\text{cmf}}$ in Equation 4, the corresponding extended matrix filter

$$\mathbf{F}_{\text{emf}} = \begin{bmatrix} 5 & 6 & & & & & & & & & & \\ 3 & 4 & 5 & 6 & & & & & & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & & & & & & \\ & & 1 & 2 & 3 & 4 & 5 & 6 & & & & \\ & & & & 1 & 2 & 3 & 4 & 5 & 6 & & \\ & & & & & & 1 & 2 & 3 & 4 & 5 & 6 \\ & & & & & & & & 1 & 2 & 3 & 4 \\ & & & & & & & & & & 1 & 2 \end{bmatrix} \tag{5}$$

contains 4 complete, 2 left-truncated, and 2 right-truncated copies of $\mathbf{f}$. In general, $\mathbf{F}_{\text{emf}}$ has $c = n_{\mathbf{x}}$ columns and $r = \lfloor (n_{\mathbf{x}} + N - 1)/2 \rfloor > \lceil n_{\mathbf{x}}/2 \rceil$ rows. Thus, $\mathbf{F}_{\text{emf}}$ has a larger row-dimension and yields a larger number of transform coefficients than do $\mathbf{F}_{\text{tmf}}$ and $\mathbf{F}_{\text{cmf}}$. Now $\mathbf{L}_{\text{emf}}\mathbf{L}_{\text{emf}}^{\text{T}} \neq \mathbf{I} \neq \mathbf{H}_{\text{emf}}\mathbf{H}_{\text{emf}}^{\text{T}}$ for all $n_{\mathbf{x}}$ but also $\mathbf{L}_{\text{emf}}^{\text{T}}\mathbf{L}_{\text{emf}} + \mathbf{H}_{\text{emf}}^{\text{T}}\mathbf{H}_{\text{emf}} = \mathbf{I}$ for all $n_{\mathbf{x}}$, so that exact reconstruction with *merge.emf* of a signal transformed with *split.emf* is possible. Thus, there is no transform error from extended matrix filters as there is from truncated matrix filters. Moreover, there is no transform periodization from extended matrix filters as there is from circulant matrix filters. However, there is transform redundancy resulting in additional coefficients. These properties can be demonstrated by continuing the example from Figures 1 and 2. Thus, Figure 3 displays the reconstruction errors obtained by transforming with *split.emf* ($N = 6$) the same example signal displayed in Figure 1, perturbing either d_1^3 or d_{16}^3, and then inverse transforming with *merge.emf* ($N = 6$). Since *emf* versions yield a different number of transform coefficients for the same N than do *tmf* and *cmf* versions, $\mathbf{b} = [100 \quad 52 \quad 28 \quad 16 \quad 10 \quad 7 \quad 6 \quad 5]$ is different in Figure 3 than $\mathbf{b} = [100 \quad 50 \quad 25 \quad 13 \quad 7 \quad 4 \quad 2 \quad 1]$ in both Figures 1 and 2. However, the detail coefficients that were perturbed for Figure 3 also correspond to the coefficients obtained from the first and last translations of the third dilation of the wavelet as in both Figures 1 and 2. Again, the perturbations were adjusted such that the values of rmse($\mathbf{a}$) for the reconstruction errors in Figure 3 equaled those in Figures 1 and 2. The absence of any periodization is

clearly visible for both examples: when either the left (d_1^3) or right (d_{16}^3) signal-end detail coefficient is perturbed, the resulting reconstruction error does not wrap around from either end of the signal. However, there is redundancy resulting in additional coefficients with $n_{\hat{\mathbf{x}}} = 129$ for the *emf* version compared to $n_{\hat{\mathbf{x}}} = 103$ for the *tmf* and *cmf* versions.

6 Efficiency with Vector Filters

The matrix-filter versions *tmf*, *cmf*, and *emf* of *split* and *merge* are useful for didactic purposes. They lend themselves especially well to meaningful displays, such as those of the matrices $\mathbf{F}_{\text{tmf}}$, $\mathbf{F}_{\text{cmf}}$, and $\mathbf{F}_{\text{emf}}$ in Equations 3–5, which readily enable visual conceptualization of the similarities and differences between the truncated, circulant, and extended versions. However, as mentioned in Section 2, they are not as efficient in the amount of memory storage required as the corresponding vector-filter versions *tvf*, *cvf*, and *evf*. Since these vector-filter versions yield results identical to (albeit more efficiently than) those of their corresponding matrix-filter versions, the vector-filter versions inherit all of the properties of the matrix-filter versions discussed in Sections 3–5. As an example of an explicit implementation of one of the vector-filter versions, *split.evf* and *merge.evf* call the following functions *comp*, *dila*, and *conv* for the compression, dilation, and convolution of signals.

Algorithm 6 Given the input vector $\mathbf{x}$, the function *comp* returns the output vector $\mathbf{y}$ as the compressed copy of $\mathbf{x}$ obtained by downsampling from the even indices of $\mathbf{x}$. Since the downsampling rate has period 2, the length of $\mathbf{y}$ is essentially half that of $\mathbf{x}$, or more precisely, $\lfloor \text{length}(\mathbf{x})/2 \rfloor$.

function $\mathbf{y} = \text{comp}(\mathbf{x})$
  $n = \text{length}(\mathbf{x})$
  $\mathbf{y} = \mathbf{x}_{2:2:n}$

Algorithm 7 Given the input vector $\mathbf{x}$, the function *dila* returns the output vector $\mathbf{y}$ as the dilated copy of $\mathbf{x}$ obtained by upsampling from $\mathbf{x}$ to the even indices of $\mathbf{y}$ with zero filling at the odd indices of $\mathbf{y}$. Since the upsampling rate has period 2, the length of $\mathbf{y}$ is always twice that of $\mathbf{x}$.

function $\mathbf{y} = \text{dila}(\mathbf{x})$

$n = 2 \cdot \text{length}(\mathbf{x})$
$\mathbf{y} = \mathbf{0}_{1:n}$
$\mathbf{y}_{2:2:n} = \mathbf{x}$

Algorithm 8 Given the filter vector $\mathbf{f}$ and input vector $\mathbf{x}$, *conv* returns the output vector $\mathbf{y}$ as the convolution of $\mathbf{f}$ with the zero-padded extension of $\mathbf{x}$.

function $\mathbf{y} = \text{conv}(\mathbf{f}, \mathbf{x})$
$N = \text{length}(\mathbf{f})$
$n = \text{length}(\mathbf{x})$
$m = n + N - 1$
$\mathbf{x} = [\mathbf{0}_{1:N-1}^{\mathrm{T}} \quad \mathbf{x}^{\mathrm{T}} \quad \mathbf{0}_{1:N-1}^{\mathrm{T}}]^{\mathrm{T}}$
$\mathbf{y} = \mathbf{0}_{1:m}$
for $i = 1 : m$
for $j = 0 : N - 1$
$y_i = y_i + f_{N-j} x_{i+j}$
end
end

If *split* and *merge* are implemented in a particular language in which a built-in convolution filtering routine is available, then it is not necessary to code a routine for *conv*. Of course, it is still necessary to use *comp* to postprocess the output and *dila* to preprocess the input of the built-in function used instead of *conv*.[4] However, if an actual routine with a for-loop is coded for *conv*, then it should be modified so that it incorporates the downsampling of *comp* when used for *split* and the upsampling of *dila* when used for *merge*. The following algorithms *convcomp* and *dilaconv* provide an example of such an implementation.

Algorithm 9 Given the filter vector $\mathbf{f}$ and input vector $\mathbf{x}$, the function *convcomp* returns the output vector $\mathbf{y}$ as the compressed (downsampled by 2) convolution of $\mathbf{f}$ with the zero-padded extension of $\mathbf{x}$.

function $\mathbf{y} = \text{convcomp}(\mathbf{f}, \mathbf{x})$
$N = \text{length}(\mathbf{f})$
$n = \text{length}(\mathbf{x})$
$\mathbf{i} = [2 : 2 : (n + N - 1)]$
$m = \text{length}(\mathbf{i})$
$\mathbf{x} = [\mathbf{0}_{1:N-1}^{\mathrm{T}} \quad \mathbf{x}^{\mathrm{T}} \quad \mathbf{0}_{1:N-1}^{\mathrm{T}}]^{\mathrm{T}}$

[4] In Matlab©, the function *filter* can be used as an equivalent to the function *conv* described here.

$\mathbf{y} = \mathbf{0}_{1:m}$
for $j = 0 : N-1$
  $\mathbf{y} = \mathbf{y} + f_{N-j}\mathbf{x}_{\mathbf{i}+j}$
end

Algorithm 10 Given the filter vector $\mathbf{f}$, input vector $\mathbf{x}$, and desired output vector length k, the function *dilaconv* returns the output vector $\mathbf{y}$ as the convolution of $\mathbf{f}$ with the zero-padded extension of dilated (upsampled by 2) $\mathbf{x}$.

function $\mathbf{y} = \text{dilaconv}(\mathbf{f}, \mathbf{x}, k)$
  $N = \text{length}(\mathbf{f})$
  $n = \text{length}(\mathbf{x})$
  $\mathbf{y} = \mathbf{0}_{1:2(n+N-1)}$
  $\mathbf{y}_{N-1+2[1:n]} = \mathbf{x}$
  $m = 2n + N - 1$
  $\mathbf{i} = [1 : m]$
  $\mathbf{x} = \mathbf{y}$
  $\mathbf{y} = \mathbf{0}_{1:m}$
  for $j = 0 : N-1$
    $\mathbf{y} = \mathbf{y} + f_{N-j}\mathbf{x}_{\mathbf{i}+j}$
  end
  $\mathbf{y} = \mathbf{y}_{N:k+N-1}$

In these algorithms, expressions which contain both vectors and scalars evaluate to vectors. For example, the subscript expression $\mathbf{i} + j$ evaluates to a subscript vector in which each element is the sum of the corresponding element of the vector $\mathbf{i}$ plus the scalar j. All of the algorithms described in this paper have been written in pseudocode. Actual code written in the Matlab© language can be obtained via anonymous ftp from the directory /pub/taswell at simplicity.stanford.edu (36.8.0.104). The software called WavBox (Copyright © 1991-93 Carl Taswell) available at this ftp site also contains routines for the computation of wavelet transforms using biorthogonal wavelets in addition to orthogonal wavelets.

7 Selection of a Transform Algorithm

Wavelet transform algorithms, such as the pyramid algorithm described by Mallat [6] and the modification consisting of *split* and *merge* presented in this report, suffer from a variety

of annoying signal-end effects when applied to finite-duration discrete-time signals. These signal-end effects include the problems of errors (Section 3), periodization (Section 4), and redundancy (Section 5) in the transform coefficients. Each one of the three different versions presented here (truncated, circulant, and extended) eliminates two but not all three of the different end effects.[5] Truncated versions avoid periodization and redundancy but incur error and can only be used if the signal length is sufficiently long to reduce the error to tolerable amounts. Circulant versions avoid error and redundancy but incur periodization and can only be used if any potential manipulation of the transform coefficients does not lead to harmful periodization. Extended versions avoid error and periodization but incur redundancy and can only be used if the additional coefficients are sufficiently small in number to meet data storage requirements or data transmission rates. Since all three versions have essentially the same computational complexity, the user can select one of the three transform algorithms most appropriate to the requirements of the application as determined by which end effects need to be avoided. However, if increased complexity is not considered to be a determining factor in the selection of an algorithm, then it is possible to eliminate all three end effects. McGill and Taswell [7, 8] proposed a method that accomplishes this task using the usual wavelet filters to process the entire signal including the ends. Cohen, Daubechies, and Vial [2] have presented an alternative method that employs the usual wavelet filters for the interior of the signal and different boundary wavelet filters at the ends of the signal.

[5] Actually, for signals of length $n_{\mathbf{X}} = 2^J$, truncated and circulant versions accumulate 0 additional coefficients per scale, and for signals of length $n_{\mathbf{X}} \neq 2^J$, they accumulate at most 1 additional coefficient per scale but only if the approximation length is odd at that scale. In comparison, extended versions always accumulate $N - 1$ additional coefficients per scale for all signal and approximation lengths. Thus, we consider truncated and circulant versions to eliminate redundancy completely for signals of length $n_{\mathbf{X}} = 2^J$, and effectively if not completely for signals of length $n_{\mathbf{X}} \neq 2^J$.

References

[1] Burt, P. J., and Adelson, E. H. The laplacian pyramid as a compact image code. *IEEE Trans. on Com. 31*, 4 (Apr. 1983), 532–540.

[2] Cohen, A., Daubechies, I., and Vial, P. Wavelets and fast wavelet transforms on the interval. 1992 preprint.

[3] Daubechies, I. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics 41* (1988), 909–996.

[4] Daubechies, I. Orthonormal bases of compactly supported wavelets. ii. variations on a theme. *SIAM Journal on Mathematical Analysis* (1991). in preparation.

[5] Heil, C. E., and Walnut, D. F. Continuous and discrete wavelet transforms. *SIAM Review 31*, 4 (Dec. 1989), 628–666.

[6] Mallat, S. G. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 7 (July 1989), 674–693.

[7] McGill, K., and Taswell, C. Wavelet transform algorithms for finite-duration discrete-time signals: Elimination of signal-end effects. 1991 preprint.

[8] McGill, K., and Taswell, C. Wavelet transform algorithms for finite-duration discrete-time signals. In *Proceedings of the International Conference on Wavelets and Applications, Toulouse France, June 1992* (1993), Y. Meyer and S. Roques, Eds., Editions Frontieres, pp. 221–224.

[9] Rioul, O., and Vetterli, M. Wavelets and signal processing. *IEEE Signal Processing Magazine 8*, 4 (Oct. 1991), 14–38.

[10] Strang, G. Wavelets and dilation equations: A brief introduction. *SIAM Review 31*, 4 (Dec. 1989), 614–627.

Table 1: Mean rmse($\mathbf{a}$) values for random normal signals of length $n_{\mathbf{x}} = 2^J$ transformed with *split.tmf* and then inverse transformed with *merge.tmf* using wavelet and scaling filters of length N (*cf.* Section 3).

	$N=4$	$N=6$	$N=8$	$N=10$	$N=12$	$N=14$	$N=16$	$N=18$	$N=20$
$J=5$	0.1513	0.3470	0.1414	0.0373	0.1230	0.2802	0.1248	0.0408	0.1030
$J=6$	0.1142	0.2622	0.1113	0.0303	0.1015	0.2158	0.1063	0.0360	0.0877
$J=7$	0.0881	0.2007	0.0862	0.0238	0.0817	0.1641	0.0845	0.0299	0.0742
$J=8$	0.0689	0.1482	0.0686	0.0180	0.0653	0.1226	0.0653	0.0231	0.0610
$J=9$	0.0493	0.1146	0.0502	0.0147	0.0486	0.0922	0.0514	0.0183	0.0464

Figure 1: Plots of a random normal signal $\mathbf{x}$ and its reconstruction error $\mathbf{e} = \mathbf{x} - \mathbf{a}$ after transforming $\mathbf{x}$ with *split.tmf* and then inverse transforming $\hat{\mathbf{x}}$ with *merge.tmf* using wavelet and scaling filters of length $N = 6$ (*cf.* Section 3).

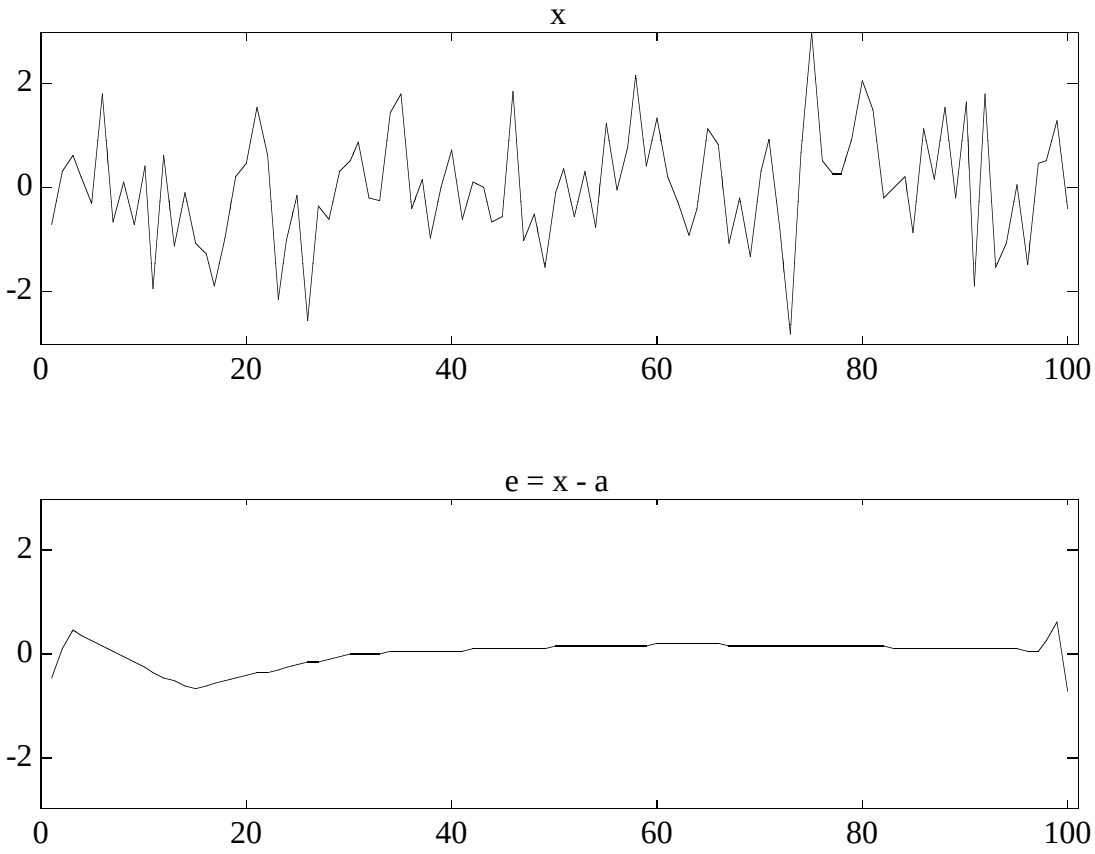

Figure 2: Plots of reconstruction errors obtained from transforming the same random normal signal displayed in Figure 1 with *split.cmf*, perturbing the indicated detail coefficient d_k^j, and then inverse transforming with *merge.cmf* using wavelet and scaling filters of length $N = 6$ (*cf.* Section 4).

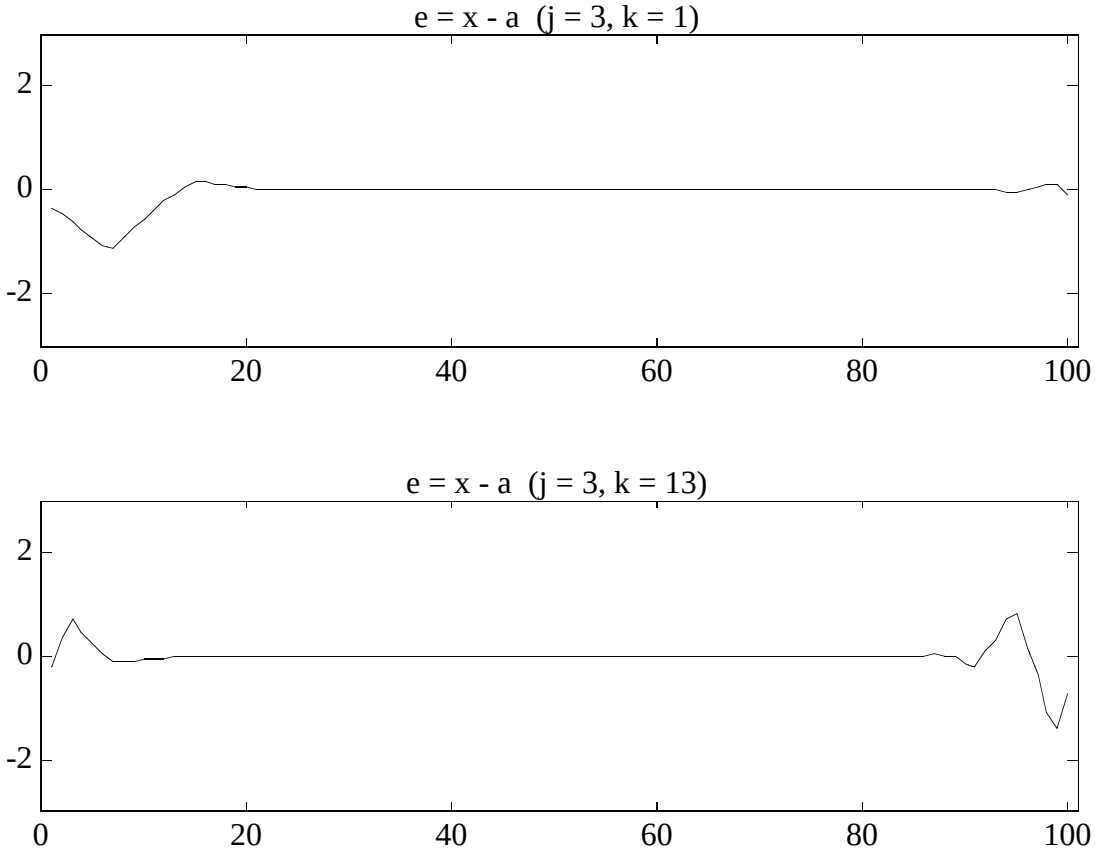

Figure 3: Plots of reconstruction errors obtained from transforming the same random normal signal displayed in Figure 1 with *split.emf*, perturbing the indicated detail coefficient d_k^j, and then inverse transforming with *merge.emf* using wavelet and scaling filters of length $N = 6$ (*cf.* Section 5).