Stanford Exploration Project, Report 75, November 17, 1997, pages 1–??

Transformation of discrete arrays by Daubechies wavelets

Matthias Schwab[1]

keywords: *wavelet, Fourier transform, wavelet transform*

ABSTRACT

Wavelet transformation, analogously to fast Fourier transformation, turns a discrete vector of length N into a vector of N coefficients, which scale the transformation's base functions. It is a linear operation which can be efficiently implemented as a recursion. Requiring the base function of the transformation to be orthogonal and approximative yields a set of constraining equations for discrete wavelets. Daubechies wavelet filter coefficients are the minimum phase solutions of this nonlinear equation set. I implemented a pyramidal algorithm which performs the transformation based on Daubechies wavelets of lengths from 4 to 20 samples. In the electronic version of this report, interactive figures allow readers to explore the transformation by submitting their own input and choosing the wavelet's length. By compressing and recovering a photograph of a face, I demonstrate the conditions for a successful wavelet compression; the matrix representation of the picture in the wavelet space is sparse and samples with small values can be neglected without affecting essential features of the uncompressed picture. Seismic data render rather densely in the wavelet domain. Finally, lacking clear criteria to define seismic data quality, a compression scheme based on data reduction does not seem to be practical. A more promising seismic application of wavelet transformation is interpolation. As with fast Fourier transformation, I utilize discrete wavelet transformation for interpolation of regularly sampled seismic traces by padding zeroes.

INTRODUCTION

Daubechies wavelets are becoming a popular image processing and data compression tool because they are base functions of finite length and represent sharp edges by a small number of coefficients. Therefore, data which show a dense Fourier representation may yield a sparse matrix when transformed by wavelets. Additionally, wavelet transformations can be implemented extremely efficiently; the cost of algorithms performing wavelet transformations increases only linearly with the length of

[1]**email:** not available

the transformed vector. The localization of its base function and its efficiency seem to make wavelet transformation a promising candidate for seismic processing. However, wavelets have not been widely used in the field of geophysics.

Daubechies (1992) gives a thorough theoretical introduction to various wavelets. Besides Daubechies wavelets, which are compact and orthogonal minimum phase filters, she discusses nonorthogonal wavelet bases, called frames, and near-zero phase filters.

Strang (1989) and Press (1991)[2] give short introductions to Daubechies wavelet transformation. Strang talks mostly about continuous Daubechies wavelets and discrete Haar transform. Press complements Strang's more theoretical paper by presenting simple algorithms for computing the wavelet transform.

Beyond its success in the compression of "edgy" data, the wavelet transform is still a tool in search of applications. Recent research in replacing standard Fourier operations by wavelet manipulations (Beylkin, 1992) may provide efficient and more physical operators than Fourier methods. Examples relevant to geophysics are interpolation, shift, and differentiation techniques.

In the geophysical literature little has been written about wavelet transformation. The case of two coefficients is known as Haar transform and was formerly discussed by Ottolini (1990). Abma and Muir (1992) discuss the Haar and sliding Fourier transform. They also visualize the positioning of the roots corresponding to different wavelet filters.

Complementing Abma's paper and generalizing Ottolini's Haar transformation this paper formulates the wavelet transform explicitly as a matrix recursion scheme. Requiring the base filter coefficients of this recursion to be compact, orthogonal, and approximative, Daubechies wavelets of any length are easily derived. A pyramidal algorithm based on this matrix recursion scheme performs the transformation based on Daubechies wavelets of lengths from 4 to 20 samples. In the electronic version of this report, interactive figures using xtpanel (Cole and Nichols, 1992) allow readers to explore the transformation by submitting their own input and choosing the wavelet's length. Finally, I apply this algorithm to compress and interpolate seismic data.

WAVELET TRANSFORM OF A VECTOR

In many respects, the discrete wavelet transform (DWT) is analogous to the fast Fourier transformation (FFT). It is a fast linear operation which transforms a one-dimensional input vector B of lenght N into a one-dimensional output vector of the same length. As with FFT, each of the N elements of the output vector represents the scale of a certain base function. In the case of FFT, these base functions are

[2]Unfortunately, Press' code implementing the DWT for wavelets of varying lengths has a serious programming error in the sorting step.

sine and cosine functions of various frequencies. The base functions of the wavelet transform are wavelets of different scale and location as seen in Figure 3.

The DWT is conveniently formulated as a recursion. Each recursion comprises a filtering by matrix multiplication and a vector permutation. The next section describes this recursion by examining the case of the 4-point Daubechies wavelet exemplarily.

The recursion step

In each recursion step we filter an input vector by multiplication with a quadratic matrix C:

$$\begin{bmatrix} m_0 \\ w_0 \\ m_1 \\ w_1 \\ m_2 \\ w_2 \\ m_3 \\ w_3 \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & & & & \\ c_3 & -c_2 & c_1 & -c_0 & & & & \\ & & c_0 & c_1 & c_2 & c_3 & & \\ & & c_3 & -c_2 & c_1 & -c_0 & & \\ & & & & c_0 & c_1 & c_2 & c_3 \\ & & & & c_3 & -c_2 & c_1 & -c_0 \\ c_2 & c_3 & & & & & c_0 & c_1 \\ c_1 & -c_0 & & & & & c_3 & -c_2 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ B_7 \end{bmatrix} \tag{1}$$

The matrix is very similar to a convolution matrix. A wraparound boundary condition is implemented in the lower right corner.

Let us imagine the result m_t of the multiplication of each odd row with the vector B as a weighted local average of adjacent vector coefficients B_t. The multiplication of each even row with the vector B results in a number w_t which measures the variation of these adjacent vector coefficients. Each pair m_t, w_t samples the same location in vector B. I refer to m_t, w_t as average or lowpass and detail or highpass coefficients, respectively. The distance between locations sampled by adjacent output pairs is two B_t samples; for example the pair m_1, w_1 is the weighted sum of the B elements 2 through 5, and the pair m_2, w_2 is the weighted sum of the B elements 4 through 7.

Obviously, not every set of c_k filter coefficients has such a highpass/lowpass characteristic. The coefficients are correspondingly chosen (see the section "The lowpass filter coefficients"). In order for the DWT to be easily invertible, the matrix C must be orthogonal. The inverse matrix is then the transpose of C:

$$\begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ B_7 \end{bmatrix} = \begin{bmatrix} c_0 & c_3 & & & & & c_2 & c_1 \\ c_1 & -c_2 & & & & & c_3 & -c_0 \\ c_2 & c_1 & c_0 & c_3 & & & & \\ c_3 & -c_0 & c_1 & -c_2 & & & & \\ & & c_2 & c_1 & c_0 & c_3 & & \\ & & c_3 & -c_0 & c_1 & -c_2 & & \\ & & & & c_2 & c_1 & c_0 & c_3 \\ & & & & c_3 & -c_0 & c_1 & -c_2 \end{bmatrix} \begin{bmatrix} m_0 \\ w_0 \\ m_1 \\ w_1 \\ m_2 \\ w_2 \\ m_3 \\ w_3 \end{bmatrix} \tag{2}$$

This orthogonality condition further constrains the choice of the wavelet filter coefficients c_k, as I discuss in the section "The low pass filter coefficients." The subroutine `dwt.rt` executes the recursion step.

```
#
#%
subroutine dwt(conj, b,n, c,nop)
integer        conj,   n,   nop, i,j, no2
real                b(n),c(nop)
temporary real tmp(n)

no2 = int(n/2)
if(conj != 0)
  Do j = 1, no2{
    tmp(2*j-1) =  b(j    )
    tmp(2*j  ) =  b(j+no2)
}
call conjnull(conj, 0, b,n, tmp,n)

Do j = 1, n - 1, 2
Do i = 1, nop
  if(conj == 0){
  tmp(j  ) = tmp(j  ) +           c(      i    ) * b(mod(j+i-2,n) + 1)
  tmp(j+1) = tmp(j+1) - (-1)**i * c(nop - i + 1) * b(mod(j+i-2,n) + 1)
  }
  else{
  b(mod(j+i-2,n) + 1) = b(mod(j+i-2,n) + 1) +          c(    i   ) * tmp(j  )
  b(mod(j+i-2,n) + 1) = b(mod(j+i-2,n) + 1) -(-1)**i * c(nop-i+1) * tmp(j+1)
}
if(conj == 0)
Do j = 1, no2{
  b(j    ) = tmp(2*j-1)
  b(j+no2) = tmp(2*j  )
}
return; end
```

The butterfly

The actual DWT is a recursion of the matrix multiplication described in the preceding section. The original vector B has the length N, which must be a power of 2. Consequently, the first recursion filter C has the dimension $N \times N$. The even $N/2$ coefficients of the output vector represent the local highpass details w_t. The odd coefficients represent the local lowpass averages m_t. The highpass coefficients are stored in the final result vector. The $N/2$ odd coefficients m_t, which are the local lowpass averages, enter the next recursion.

In the next recursion step, the matrix C has the dimensions $\frac{N}{2} \times \frac{N}{2}$. It operates entirely on the lowpass averages of the previous step. Again, its detail coefficients are stored in the output vector; its averages enter the next recursion step. Consequently, the details w_t and the averages m_t are also called wavelet and mother coefficients, respectively.

When only two coefficients remain in the recursion, the recursion ends. These final two coefficients are averages of previous averages. They complete as top coefficients the final wavelet coefficient vector w_t. Such a series of matrix multiplication and sorting is often referred to as a pyramidal or butterfly structure.

In the following diagram, the subscript of a subvector denotes the recursion of which it is a result. The subscript of each coefficient numbers pairs within the output of a particular recursion.

$$\begin{bmatrix}B_0\\B_1\\B_2\\B_3\\B_4\\B_5\\B_6\\B_7\end{bmatrix} \xrightarrow{Cmult} \begin{bmatrix}m_0\\w_0\\m_1\\w_1\\m_2\\w_2\\m_3\\w_3\end{bmatrix}_1 \xrightarrow{sort} \begin{array}{c}\begin{bmatrix}m_0\\m_1\\m_2\\m_3\end{bmatrix}_1\\ \begin{bmatrix}w_0\\w_1\\w_2\\w_3\end{bmatrix}_1\end{array} \begin{array}{c}\xrightarrow{Cmult}\\ \\ \\ \\ \\ \end{array} \begin{array}{c}\begin{bmatrix}m_0\\w_0\\m_1\\w_1\end{bmatrix}_2\\ \begin{bmatrix}w_0\\w_1\\w_2\\w_3\end{bmatrix}_1\end{array} \begin{array}{c}\xrightarrow{sort}\\ \\ \\ \\ \\ \end{array} \begin{array}{c}\begin{bmatrix}m_0\\m_1\end{bmatrix}_2\\ \begin{bmatrix}w_0\\w_1\end{bmatrix}_2\\ \begin{bmatrix}w_0\\w_1\\w_2\\w_3\end{bmatrix}_1\end{array} \tag{3}$$

The inverse DWT transform is performed by interleaving the top four coefficients of the wavelet domain vector, then multiplying them by the corresponding inverse matrix. Recursively applying the inverse operation of the forward matrix multiplication and separation yields the original input vector. The program `dwtact.rt` implements the recursion.

```
#
#%
subroutine dwtact(conj, dir, dat,n1,n2,n3, c,nop)
integer           conj, dir,     n1,n2,n3,   nop, n, i1,i2,i3
real                          dat(n1,n2,n3),c(nop)

call cofi(dir,c,nop)

Do i3 = 1,n3
Do i2 = 1,n2
  if(conj == 0){
    n = n1
    while( n >= 4 ){
      call dwt(conj, dat((i2-1)*n1+(i3-1)*n2*n1 + 1,1,1) ,n, c,nop)
      n = n/2
  } }
  else{
    n = 4
    while( n <= n1 ){
      call dwt(conj, dat((i2-1)*n1+(i3-1)*n2*n1 + 1,1,1) ,n, c,nop)
      n = 2*n
  } }
return; end
```

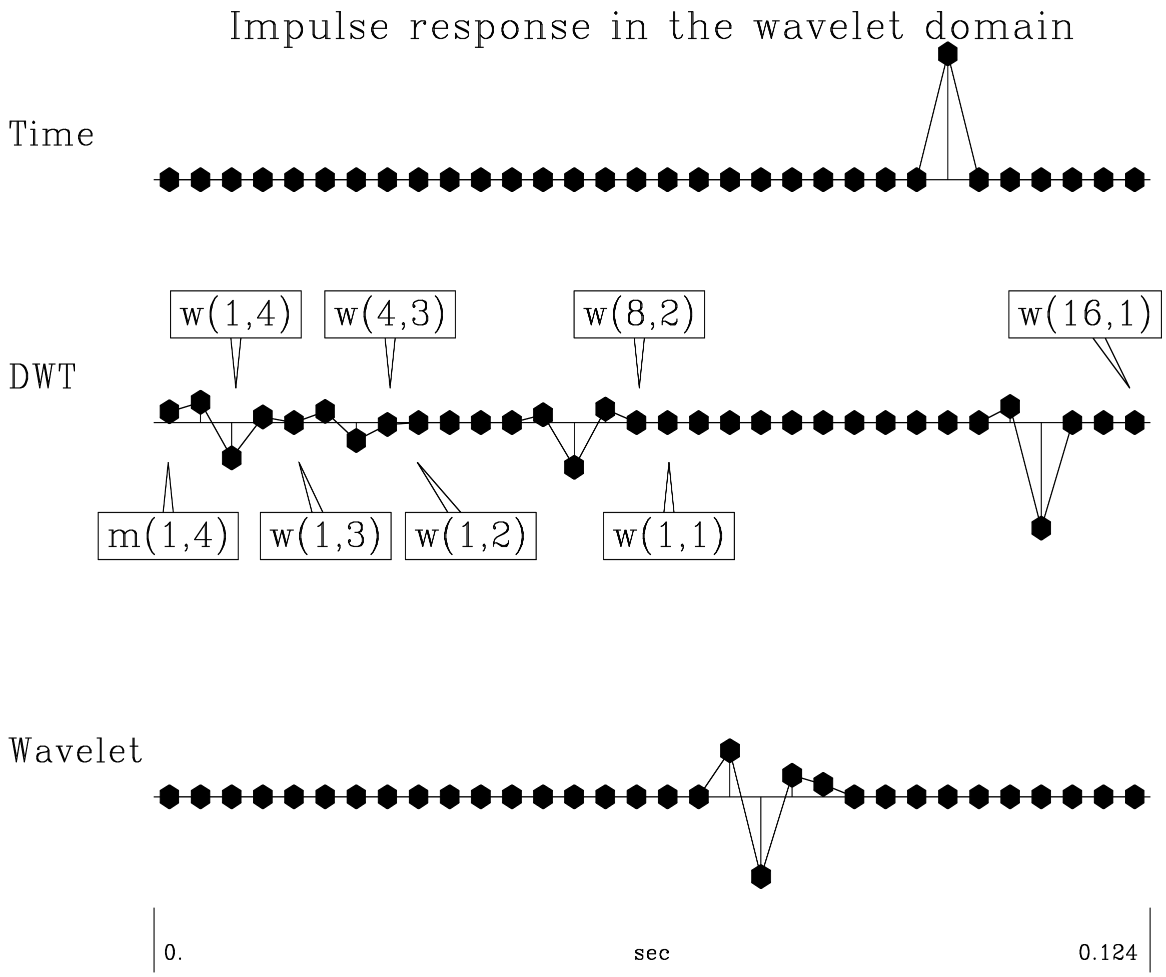

Figure 1: The Impulse response of the 4-point wavelet transformation. The input is a trace of 32 samples showing an impulse at the 26th location. The second half of the DWT trace is the filter coefficients of the 4-point filter applied to the input. The event at the 28th sample, the 12th sample in the first octave, is two samples long. These two samples are the weighted sums of the 4-point filter (bottom trace) passing the impulse in shifts by two. In each subsequent octave (increasing to the right), we can find an event sampling the impulse. These events are the highpass coefficients of the recursion step operating on the lowpass coefficients of the former octave. The active button allows you to build your own input vector and lets you apply various wavelet transforms. matt-ipret [ER]

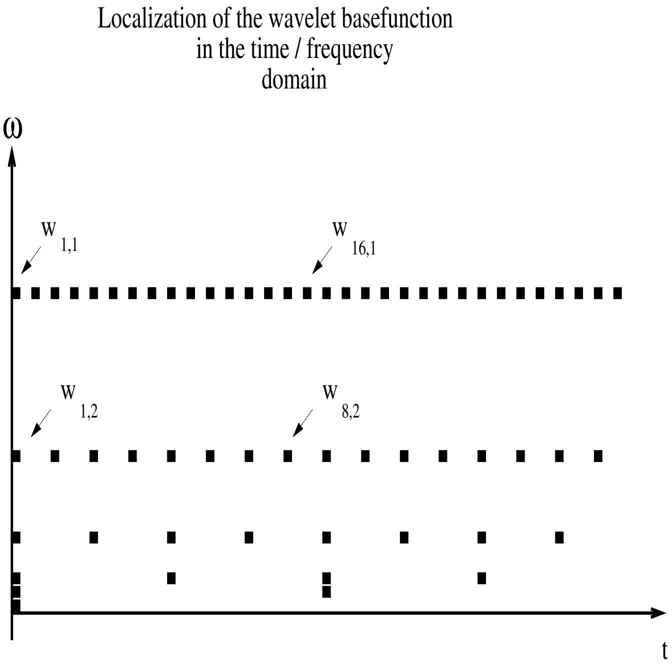

Figure 2: Localization of a base function in the frequency-time plane. Dots in the frequency-time plane represent the localization corresponding to a certain wavelet coefficient. Each horizontal line refers to coefficients of the same octave. The localization is idealized: the dot represents the first time sample of the filter whose length increases with frequency. The dot's ordinate represents the frequency peak of the filter. For example, an FFT transform would plot as a string of equally spaced dots along the ω axis. (Adapted from Daubechies) matt-loc [NR]

The lowpass filter coefficients

In the section "Wavelet transform of a vector" I assumed the matrix C to be orthogonal. Additionally, I stated lowpass and highpass characteristics of the filter coefficients c_k. In this chapter I will derive the simple nonlinear equations for c_k which guarantee these conditions.

To be orthogonal the matrix C in equation (1) multiplied by its transpose in equation (2) has to yield a unit matrix. The multiplication provides the following orthogonality relation for the wavelet filter coefficients c_k:

$$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 2 \tag{4}$$

$$c_0 c_2 + c_1 c_3 = 0 \tag{5}$$

To determine the four coefficients, two additional relations are required. The conditions

$$c_0 - c_1 + c_2 - c_3 = 0 \tag{6}$$

$$0c_0 - 1c_1 + 2c_2 - 3c_3 = 0 \tag{7}$$

ensure that any inner product of $(c_0, -c_1, c_2, -c_3)$ and a series $k_1(1,1,1,1)+k_2(1,2,3,4)$ yields 0. This annihilation is called an "approximation condition of polynomial order 1" or "the vanishing of the zeroth and first momentum." It is important to note that these rejected polynomials are the low-order and therefore smooth polynomials. These local rejections justify the imaginative visualisation of the odd row multiplications as local highpass filtering. The even row multiplication destroys high-frequency events by weighted adding.

I have coded the conditions in a mathematica script `coef.ma`[3], which solves numerically for the coefficient set c_k.

```
(* coef.ma *)
(* number of coefficients *)
(* tested for n=4 and n=6 *)
n  = 4 ;
nh = n / 2 ;
(* definitions kronecker & Power *)
kron[i_,j_] := If[ i == j, 1, 0] ;
Unprotect[Power] ;
Power[0,0] = 1 ;
Protect[Power] ;
(* approximation condition *)
Do[aus[i] = Sum[(-1)^k (n-k)^(i-1) c[k], {k,1,n}] == 0, {i,1,nh}]
(* orthogonality condition *)
Do[aus[i+nh] = Sum[ c[k] c[k-2(i-1)] , {k,2(i-1)+1,n}] ==  kron[i,1], {i,1,nh}]
(* solving Equation *)
eqsys = Table[aus[i],{i,1,n}] ;
coef = Table[c[i],{i,1,n}] ;
sol = N[Solve[eqsys,coef]] ;
(* use NSolve[] for numerical results *)
sol >>> sol.ma ;
```

The conditions (4), (5), (6), and (7) determine the coefficients c_k's amplitude but not their phase spectrum. Daubechies defines her wavelets as the minimum phase solution to these equations. The 4-point wavelet coefficients are

$$c_0 = \frac{1+\sqrt{3}}{4\sqrt{2}} \tag{8}$$

$$c_1 = \frac{3+\sqrt{3}}{4\sqrt{2}} \tag{9}$$

$$c_2 = \frac{3-\sqrt{3}}{4\sqrt{2}} \tag{10}$$

$$c_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}. \tag{11}$$

Wavelets of different orders

Daubechies' 4-point set is only one of a series of compact wavelets. The user chooses an even number of mother coefficients c_k, and the stated conditions of orthogonality

[3]This direct implementation works fine for Daubechies' 4-point and 6-point wavelets. For wavelets with more coefficients, solving the nonlinear equation system does not succeed in a reasonable amount of time.

and approximation define the wavelet transformation. The case of two coefficients leads to the Haar transform. The recursion matrix is

$$\begin{bmatrix} c_0 & c_1 & & & & \\ c_0 & -c_1 & & & & \\ & & c_0 & c_1 & & \\ & & c_0 & -c_1 & & \\ & & & & c_0 & c_1 \\ & & & & c_0 & -c_1 \end{bmatrix} \tag{12}$$

The orthogonality condition requires that

$$c_0^2 + c_1^2 = 2 \tag{13}$$

The remaining degree of freedom is used to reach a vanishing momentum of the zeroth order, as follows:

$$c_0 - c_1 = 0 \tag{14}$$

The resulting coefficients $c_0 = c_1 = 1$ define the Haar transform. Note how the vectors $(1,1)$ and $(1,-1)$ of the matrix (12) realize a simple lowpass and highpass filter. Daubechies wavelets comprise the Haar wavelet as their 2-point member. The Haar transform is much older than the Daubechies wavelets and therefore has its own name.

In general, the orthogonality condition for a Daubechies wavelet transform of order N accounts for $N/2$ relations of the form

$$\sum_k c_k c_{k-2m} = 2\delta_{0m}, \tag{15}$$

where $m = 0, 1, .., N/2 - 1$. The remaining $N/2$ degrees of freedom are used for achieving a maximum order of polynomial approximation $p = N/2 - 1$, as follows:

$$\sum_k (-1)^k k^m c_k = 0, \tag{16}$$

for $m = 0, 1, ..., p$.

Solving this system of equations and choosing the minimum phase solution yields the N coefficients c_k.[4]

READING VECTORS OF WAVELET COEFFICIENTS

Reading vectors of wavelet coefficients

The output of a wavelet transformation is difficult to read. Coefficients referring to different locations and different scales are stored in single vector. In order to

[4]All coefficients of Daubechies base functions of orders 4 to 20 can be found in Daubechies' book (1992). The coefficients I implemented in `cofi.rt` are taken from it. Daubechies' book also discusses a more efficient method of numerically computing the coefficients c_k than my implementation `coef.ma`.

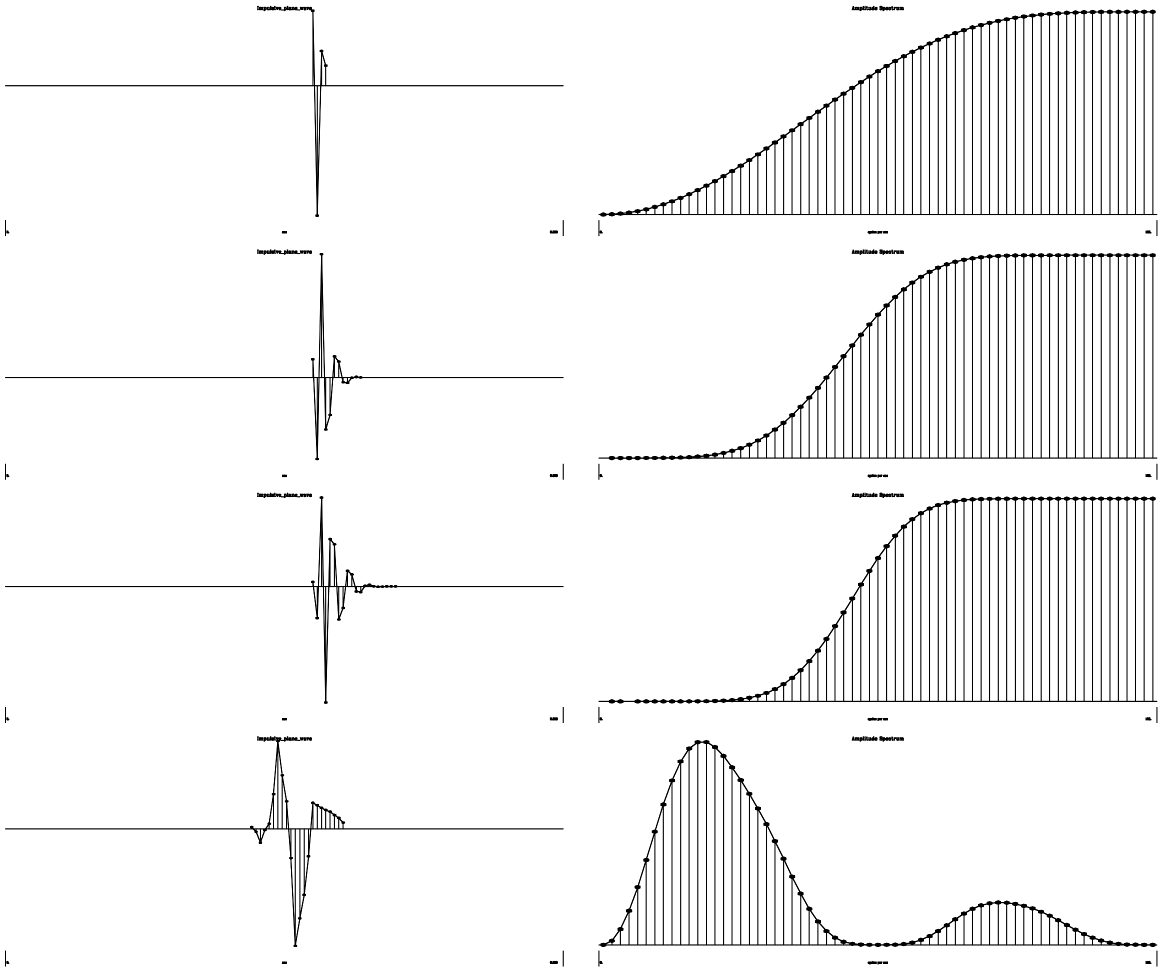

Figure 3: A table of various wavelets. The lines show the the $4, 12, 20$ point wavelet filters and their spectrum (1st through 3rd line). The spectrum slope is increased by placing more zeroes at zero frequency. The figures are created by placing impulses in the first octave space in the wavelet domain. The inverse wavelet transform and its Fourier spectrum are displayed. The bottom line depicts a wavelet corresponding to a second octave event. The spectrum is illustrated in the Figure 4. matt-copspec [ER]

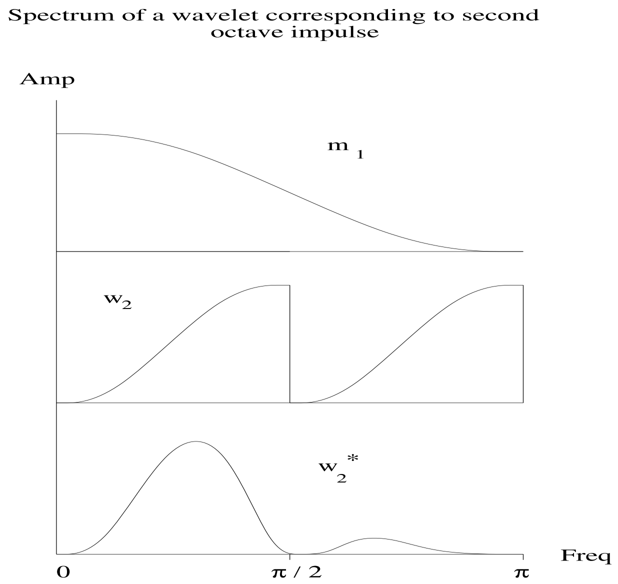

Figure 4: Amplitude spectrum of a higher scale wavelet. An impulse in the second octave is convolved by the wavelet operating on the double sample rate. In the next and final recursion, this output is convolved by a set of lowpass coefficients. Concatenation of the convolutions is equivalent to multiplication of the corresponding frequency filter. The first and second lines display the filter for the different recursion steps. The bottom line is the product of both spectra. matt-spec [NR]

separate the vector into parts representing events of the same scale, I could span a second dimension. However, I prefer instead to display a complete trace, because transforming higher dimensional arrays leads to simply interpretable matrices as seen in Figure 7. Looking at a single trace, we interpret the different octave parts of the trace as referring to different scales. The different elements in each octave subset refer to a spatial location like shown in Figure 1.

APPLICATIONS

At this preliminary stage of my research, I present a few processing examples to highlight the strengths and shortcomings of wavelet transformation as a processing tool.

Interpolation by wavelets

As in Fourier space, padding zeroes to the detail end of the wavelet vector interpolates the time series after transforming the vector back to the time domain. In contrast to interpolation in the Fourier domain, the base functions of the wavelet transform are local, and the extended spectrum contains small but noticeable energy as seen in Figure 6. In the case of a Haar transform, we would implement a next neighbour interpolation.

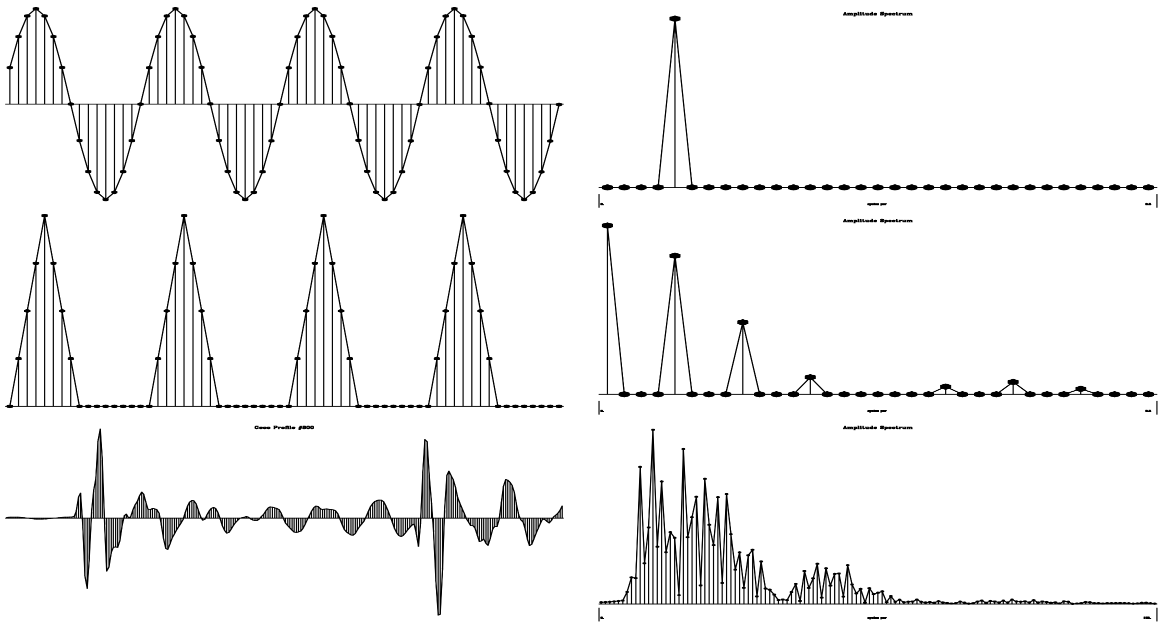

Figure 5: The original test functions and their spectra. The result of interpolating these functions is displayed in Figure 6. matt-opol [ER]

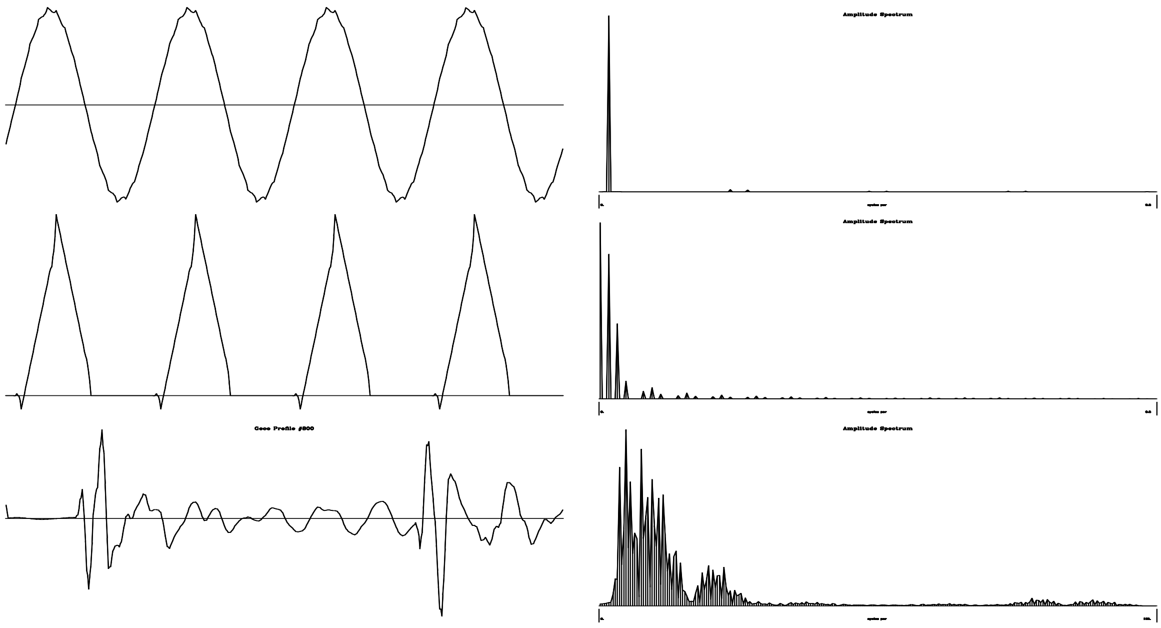

Figure 6: Interpolation by discrete wavelet transformation. Traces of 256 samples are oversampled by transforming them into the wavelet domain, padding 256 zeroes at the high-frequency end, and inverse transforming them. The triangle and sinusoid series are distorted. The spikiness in time may be realistic in the trace example. Most, but not all, of the high-frequency energy is caused by the transforms wraparound feature. The characteristics of this interpolation scheme remain to be studied. matt-ipol [ER]

Data compression

The objective of data compression is to store information efficiently. Some conservative data compression schemes achieve this compression without changing the input data. Other approaches reduce the data down to their essential features and store only those features. Obviously, for such a compression to be successful, the correct reduction or filter step must be used. The correct filter preserves the data features which we consider essential.

Data compression by wavelet transformation is accomplished by storing the location of the significant samples and omitting the insignificant values. In practice a hybrid method is often used. The values and location of the maximum samples are stored. Intermediate sized sample values are stored by one- or two-bit representation of their value. Sample values below a certain threshold are neglected. In this project I did not store the actual sample values, because I was primarily interested in the effects of filtering in the wavelet domain.

In wavelet compression the input data are transformed and small sample values are neglected. In Figures 7 and 8, I have erased all samples smaller than 10 % of the maximum value. This leaves only about 8 % of the samples as nonzero.

The purpose of a video telephone picture of a person is to be recognizable. Little details are often negligible. The reduction step can be tested by interviewing users about their satisfaction. However, for seismic data it is hard to decide which details might be of importance in latter interpretation. Without a clear definition of what we want to preserve, a data compression by reduction cannot finally be evaluated.

Data which depict a sparse wavelet transform are suited for compression by DWT. In Figure 7 the photograph's wide, smooth areas like the forehead and the cheeks are bounded by edges such as the hair line or the facial oval. The data yield a fairly sparse wavelet transform. Only few detail coefficients corresponding to the edges are of substantial size. Unfortunately, seismic data do not yield such a sparse matrix. Zeroing values below the 10 % threshold leaves the data severely distorted as seen in Figure 8. Seismic velocity spectra or well logs might be more suitable for compression by wavelet transformation.

CONCLUSIONS

Wavelet transform have become a standard tool in the field of image processing and data compression. However, it is still not widely used in seismic processing. One reason is that researchers have just started starting to discover operators in the wavelet domain doing standard Fourier operations like shifting, and differentiating time series. Additionally, DWT fails to render seismic sections as sparse matrices, which frustrates data compression. Interpolation is a more successful application. As with Fourier methods, padding of zeroes in the wavelet domain results in denser sampling

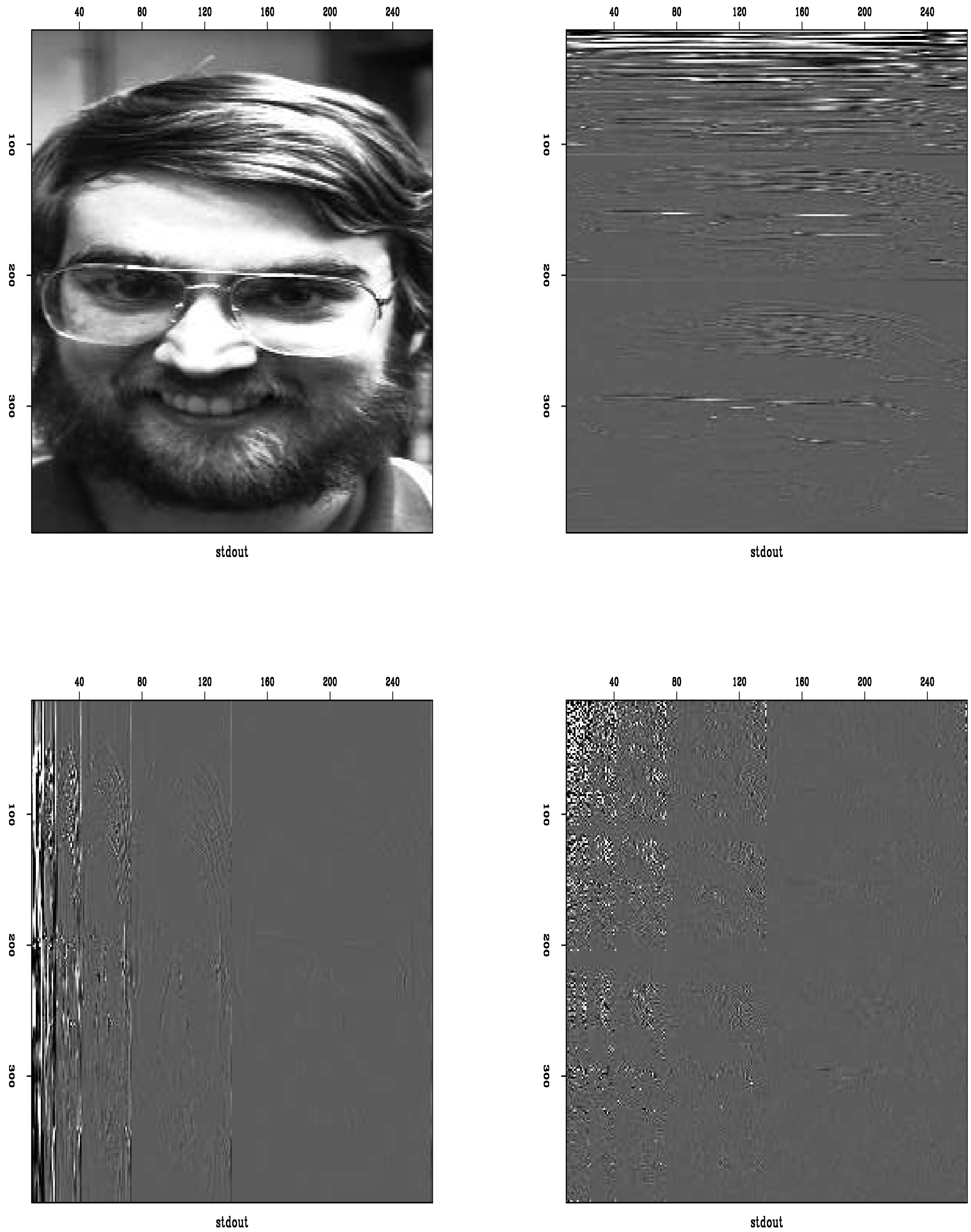

Figure 7: Two-dimensional wavelet transform of a picture. The input picture (top left) is transformed in the vertical (top right) and horizontal (bottom left) directions. Note that the face is repeated in the different octave subspaces of all transformed matrices. The lower right quarter of the matrix shows the edge details of the face. The glasses are depicted differently in each of the outer quarters of the matrix. These quarters, as well as the horizontally and vertically transformed matrix, are biased toward the horizontal and vertical edges, respectively. Unfortunately, these details are hardly visible in the paper document. matt-plane2 [ER]

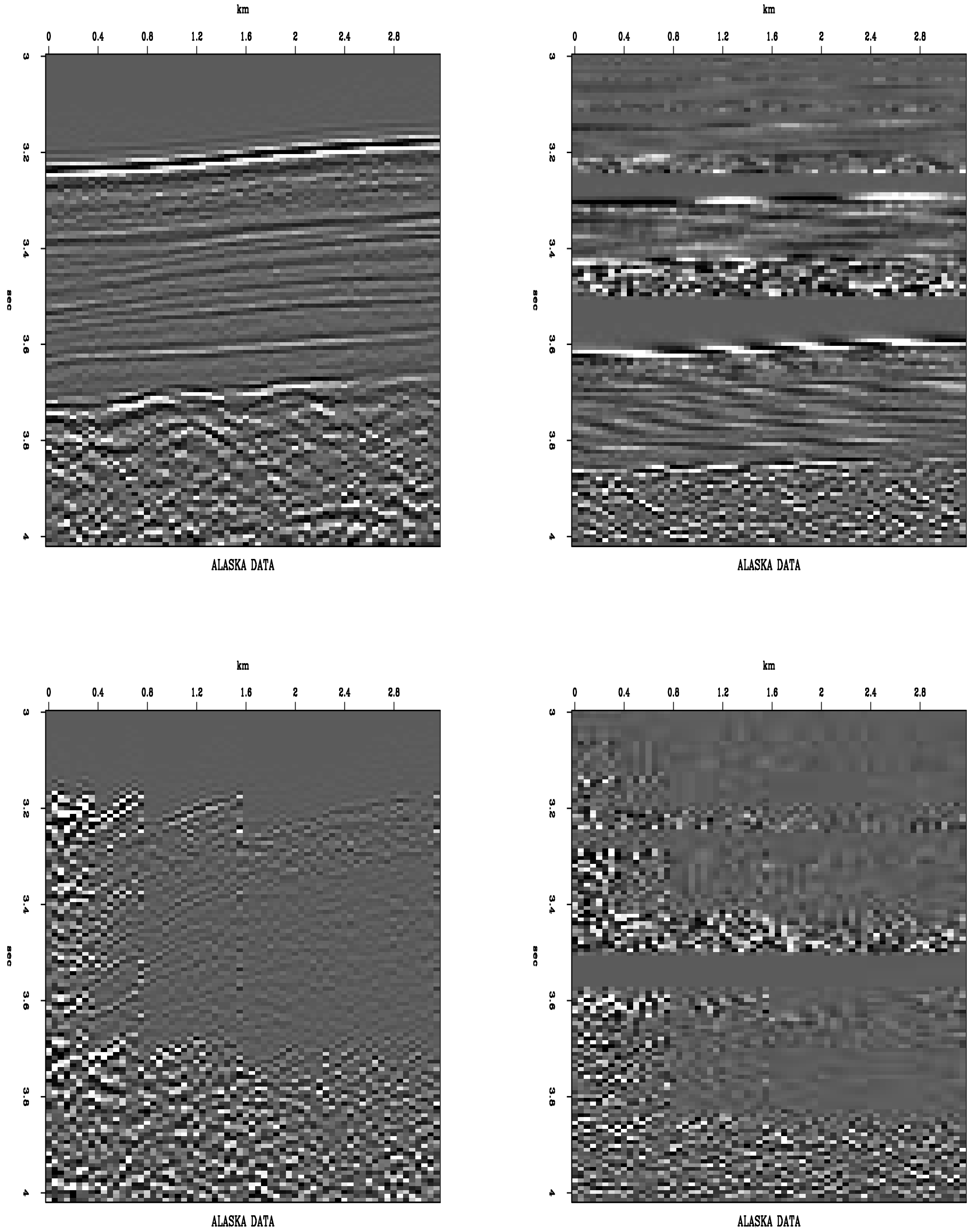

Figure 8: Two-dimensional wavelet transform of a seismic section. The same sequence of wavelet transformation as Figure 7, except that the data object is a seismic section. matt-plane1 [ER]

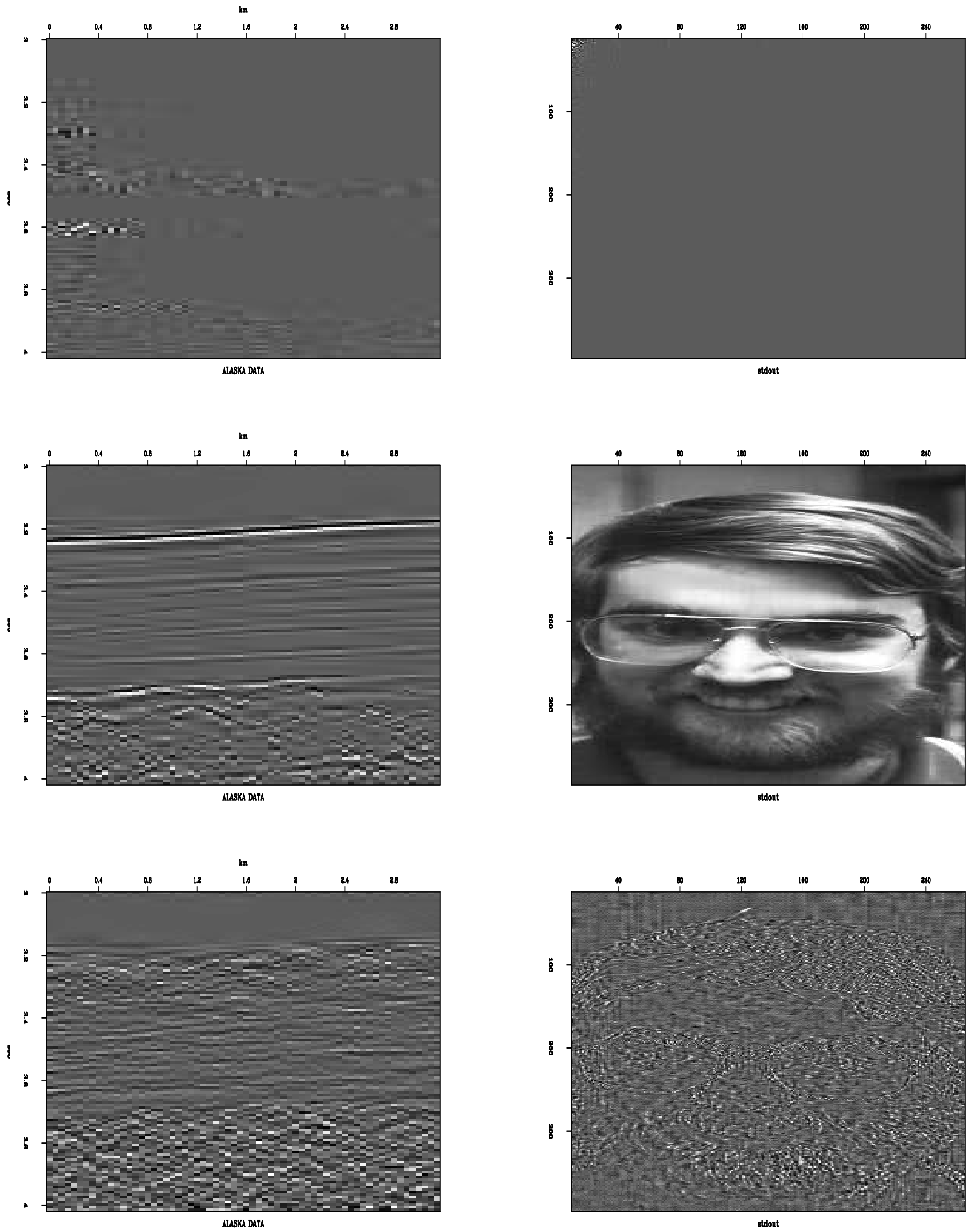

Figure 9: Data compression by wavelet transform. The 2-D transformation of the previous examples of Figure 7 and 8 is used for data reduction. In the wavelet domain every sample value smaller than a certain threshold (10 % maximum value) is erased. The top panels depict the reduced wavelet coefficient matrices. The middle panels show the reconstructed picture. The bottom row shows the difference between the inverted sections and the original data. In the electronic version of this book pushing the button let you explore data compression by wavelet transformation interactively. $\boxed{\text{matt-comp}}$ [ER]

in the time domain. In contrast to extending the Fourier transform by padding, the wavelet base function is localized and energy in the extended spectrum part is small but nonzero.

I have shown how the choice of filter coefficients implements different Daubechies wavelets. In the future wavelets other than Daubechies' may prove more applicable to seismic data.

The next goal is to implement discrete Fourier-like operators for shifting, differentiating, and interpolating time series. Comparing these operators with the traditional Fourier operators will require establishing fundamental criteria such as the specification of what accurate differentiation of discrete seismic data means.

ACKNOWLEDGEMENTS

I thank David Donoho for his introductory lecture on wavelets. Francis Muir made us all think in terms of a Z transform formulation. Martin Karrenbach introduced me to xtpanel and asked many useful questions. Ray Abma reproduced for me the picture of Steve Cole which he used in the last report. And, of course, my compliments to Steve Cole and Dave Nichols for writing xtpanel; my project would not have been as much fun without interactivity.

REFERENCES

Abma, R., and Muir, F., 1992, Wavelets and seismic processing: SEP–**75**, 373–384.

Beylkin, G., 1992, On the representation of operators in bases of compactly supported wavelets, preprint: public domain: pascal.math.yale.edu.

Cole, S., and Nichols, D., 1992, Xtpanel: An interactive panel builder: SEP–**75**, 497–520.

Daubechies, I., 1992, Ten lectures on wavelets: Society for industrial and applied mathematics.

Ottolini, R., 1990, Wavelet decomposition of seismic data: SEP–**65**, 295–304.

Press, W. H., 1991, Wavelet transforms: Harvard-Smithsonian Center for Astrophysics Preprint, **3184**.

Strang, G., 1989, Wavelets and dilation equations: a brief introduction: SIAM Review, **31**, 614–627.

APPENDIX

Corresponding to the requested wavelet length `nop`, the subroutine `dwtact.rt` initializes the vector `c(nop)` by calling the subroutine `cofi.rt`. The coefficients are taken from Daubechies' book. The `dir` parameter allows the reversal of the Daubechies coefficient set. This option is needed in order to have a wavelet of minimum phase with the standard positive time direction to the right.

```
#
#%
subroutine cofi(dir,c,nop)
integer         dir,  nop,  i
real                c(nop), tmp

if(nop == 4 ){
  c(1) = .4829629131445341 ; c(2) = .8365163037378077
  c(3) = .2241438680420134 ; c(4) =-.1294095225512603
}
else if(nop == 6 ){
  c(1) = .3326705529500825 ; c(2) = .8068915093110924
  c(3) = .4598775021184914 ; c(4) =-.1350110200102564
  c(5) =-.0854412738820267 ; c(6) = .0352262918857095
}
else if(nop == 8 ){
  c(0) = .2303778133088964 ; c(2) = .7148465705529154
  c(2) = .6308807679398587 ; c(4) =-.0279837694168599
  c(5) =-.1870348117190931 ; c(6) = .0308413818355607
  c(7) = .0328830116668852 ; c(8) =-.0105974017850690
}
else if(nop == 10){
  c(1) = .1601023979741929 ; c(2) = .6038292697971895
  c(3) = .7243085284377726 ; c(4) = .1384281459013203
  c(5) =-.2422948870663823 ; c(6) =-.0322448695846381
  c(7) = .0775714938400459 ; c(8) =-.0062414902127083
  c(9) =-.0125807519990820 ; c(10) = .0033357252854738
}
else if(nop == 12){
  c(1) = .1115407433501095 ; c(2) = .4946238903984533
  c(3) = .7511339080210959 ; c(4) = .3152503517091982
  c(5) =-.2262646939654400 ; c(6) =-.1297668675672625
  c(7) = .0975016055873225 ; c(8) = .0275228655303053
  c(9) =-.0315820393184862 ; c(10)= .0005538422011614
  c(11)= .0047772575119455 ; c(12)=-.0010773010853085
}
else if(nop == 20){
  c(1) = .0266700579005473 ; c(2) = .1881768000776347
  c(3) = .5272011889315757 ; c(4) = .6884590394534363
  c(5) = .2811723436605715 ; c(6) =-.2498464243271598
  c(7) =-.1959462743772862 ; c(8) = .1273693403357541
  c(9) = .0930573646035547 ; c(10)=-.0713941471663501
  c(11)=-.0294575368218399 ; c(12)= .0332126740593612
  c(13)= .0036065535669870 ; c(14)=-.0107331754833007
```

```
  c(15)= .0013953517470688 ; c(16)= .0019924052951925
  c(17)=-.0006858566949564 ; c(18)=-.0001164668551285
  c(19)= .0000935886703202 ; c(20)=-.0000132642028945
}
if(dir == 1)
  Do i = 1, nop, 2{
    tmp          = c(i)
    c(i)         = c(nop - i + 1)
    c(nop -i + 1) = tmp
  }

return; end
```