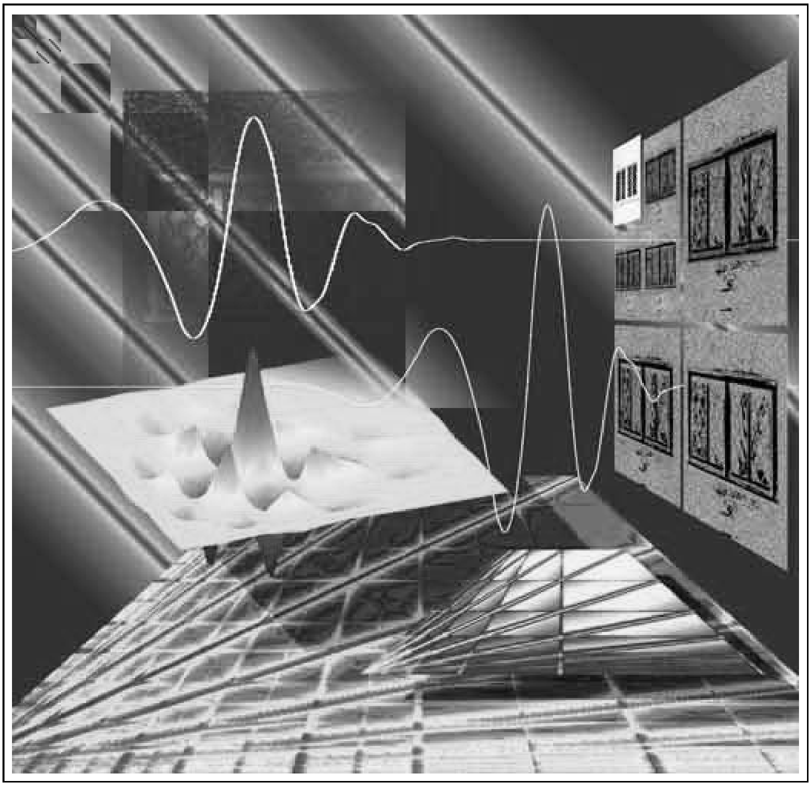
WAVEKIT: a Wavelet Toolbox for Matlab

Harri Ojanen

July 1, 1998



Contents

1 Introduction

The WAVEKIT-toolbox is a collection of functions for Matlab that implement the following wavelet and wavelet packet algorithms:

- One- and two-dimensional (periodic) fast wavelet and wavelet packet transforms and the best basis algorithm for wavelet packets.
- An implementation of the fast matrix multiplication algorithm of Beylkin, Coifman, and Rokhlin [3] for both wavelets and wavelet packets.
- Various demonstrations on visualizing wavelets, signal analysis, and the multiplication algorithm.

The programs in this toolbox are not optimized other than that the inner-most loops are written in C. They are intended for learning about wavelets, not for serious applications. The toolbox grew out of my desire to teach myself about wavelets and now I feel it has matured to the point where it might prove useful to others. I think some of the visualization tools and demos are rather unique.

All comments, criticism, bug reports, etc. are certainly welcomed!

This document is only a tutorial into using the different functions in the WAVEKIT toolbox. In particular, it is assumed that the reader is already familiar with the basics of wavelets and wavelet algorithms. There are many excellent sources for this knowledge, such as:

- Introductions to wavelets: [12, 6, 8].
- Numerical algorithms: [13].
- Applications: [6, 8, 1].
- Theory: [5, 7].

The home page for the WAVEKIT-toolbox is

```
http://www.math.rutgers.edu/˜ojanen/wavekit/
```

The program files and this document can be downloaded from the above address (this document is also available on-line there). The author can be reached with e-mail at

```
ojanen@math.rutgers.edu
```

1.1 System requirements

Matlab 5.1 or later and a compatible C-compiler[1] is required—the compiler must be able to work with Matlab to produce mex-files. At least on Unix platforms both the standard C-compiler `cc` and the GNU compiler `gcc` work with Matlab. Check the Matlab documentation for your system (the "MATLAB Application Program Interface Guide" is the relevant document).

1.2 Installation

To install the toolbox follow these steps:

1. Download the file `wavekit.zip`
2. Decide on where you want to place the files. Unpack the zip-file with the command `unzip wavekit` or `pkunzip -d wavekit`

   This generates the directory `wavekit` and some subdirectories of it.
3. Start Matlab and `cd` to the directory `wavekit` and then to `fwt`.

[1] Not necessary under Windows 95.

4. Type the command `makemex`. This compiles the C-subroutines and produces the necessary mex-files.[2] (Skip this step if you are using Windows 95.)

5. `cd` back to the `wavekit` directory (`cd ..`) and give the command `startup` (see the next section). Optionally, you may now also give the command `test`. This runs several programs included in this package and should notice if there are any problems. If `test` produces no output, everything should be fine.

Any problems in step 4 are likely caused by the fact that the mex-compiler was not properly set up when Matlab was installed on your system. To fix this, please refer to the Matlab documentation or complain loudly to your system administrators. If the problem seems to be related to the toolbox, I will be glad to hear about it. (Send me a Matlab `diary`-file that shows all error messages and also the output of the matlab `version` command.)

I have installed WAVEKIT succesfully on Solaris, AIX, and Digital Unix, but these are all Unix systems. I would like to hear from Mac and PC users and others even if there were no problems, since I don't have access to Matlab on other machines.

1.3 Getting started

The Matlab search path has to be properly set up for WAVEKIT to work correctly. An m-file is included that automatically does this (file `startup.m` in the `wavekit` directory). To start using the toolbox,

- either start matlab from the directory `wavekit` (then `startup` is executed automatically),

- or start matlab, `cd` to the directory `wavekit` and give the command `startup` yourself.

In both cases you should see the text "`Setting up paths for wavelets`".

1.4 Demonstrations

Give matlab the command `wmenu` to start the main menu to different demonstrations or type `help wmenu` or `helpwin wmenu` to see a list of all available demos. They are also individually described later in this document.

1.5 Directories and filenames

The functions in the toolbox are divided across the following directories:

- `fwt`: fast wavelet transforms and related functions.

- `packet`: one-dimensional wavelet packets.

- `packet2`: two-dimensional wavelet packets.

---

[2] `makemex` simply compiles the files `fwt1step.c`, `fwt2step.c`, `ifwt1stp.c`, and `ifwt2stp.c` (in the directory `wavekit/fwt`) using the Matlab mex compiler.

- `wdemo`: demonstrations.

Use `help fwt` (or `helpwin fwt`) etc. to view a list of functions in the corresponding directory. In this document we describe only the major programs in each category, there are many other functions that are either variations or subroutines of those described here.

The following abbreviations are used in the names of functions:

`fwt`	fast wavelet transform
`i`	inverse
`msa` or `ms`	multi-scale analysis
`ns`	non-standard
`tns`	tensor product
`wav`	wavelet
`wp`	wavelet packet
`wpa`	wavelet packet analysis
`wps`	wavelet packet synthesis

Numbers `1` and `2` refer to the number of dimensions.

2 Filter coefficients

Before doing any computations, the first step is to get the filter coefficients associated with a wavelet. The functions `wavecoef` and `selwavlt` provide access to a small database of filters. They currently know about the following wavelets:

- The standard compactly supported wavelets of Daubechies [4, 5].
- Coiflets [5].
- Symmetric biorthogonal wavelets [5, p. 271].
- Family of wavelets with optimal Sobolev-regularity constructed by the author [9, 10] (the "$n_z = 1$" family from [10]).
- "Beylkin 18", see [13, p. 444].
- A filter used in speech coding constructed by Vaidyanathan and Huong [11].

The function `wavecoef` takes as arguments a string describing the family and a number that specifies the order of the wavelet (usually the number of coeffients in the filter). `Selwavlt` is menu based, instead.

```
WAVECOEF -- returns some wavelet filter coefficients

[h,g] = wavecoef(selection,n)
[h1,g1,h2,g2] = wavecoef(selection,n)

selection  is the name of the family:
 'Haar'       Haar's wavelet
 'Beylkin'    Only one wavelet, with 18 coefficients
```

```
 'Coiflet'    Coiflets 6, 12, 18, 24, and 30 (=n)
 'Daubechies' Her compactly supported of length n (n=2,4,...,20)
 'Ojanen'     Most Sobolev-regular (see the documentation),
              lengths n=8:2:40
 'Vaidyanathan'  One wavelet with 24 coefficients
 'Symmetric biorthogonal'   of orders n= 1.3, 1.5, 2.2, 2.4, 2.6,
              2.8, 3.3, 3.5, 3.7, and 3.9
Output:
 h    Filter coefficients for the scaling function
 g    Coefficients for the corresponding wavelet

With no input arguments returns the names of the families (the
biorthogonal wavelets are listed only when there are four output
arguments).

If the user selects an orthogonal wavelet when there are four
output arguments, h2 and g2 are copies of h and g.

The first three letters are enough for  selection, which is also
case insensitive.

See also SELWAVLT, WAVDEMO.
```

Examples:

```
[h,g] = wavecoef('dau',12)
[h1,g1,h2,g2] = wavecoef('sym',3.7)
```

3 Dilation equations

A basic property of wavelets is that the scaling function φ (father wavelet) satisfies a dilation equation (also called a two-scale difference equation) of the form

$$\varphi(x) = \sum_k c_k \varphi(2x - k),$$

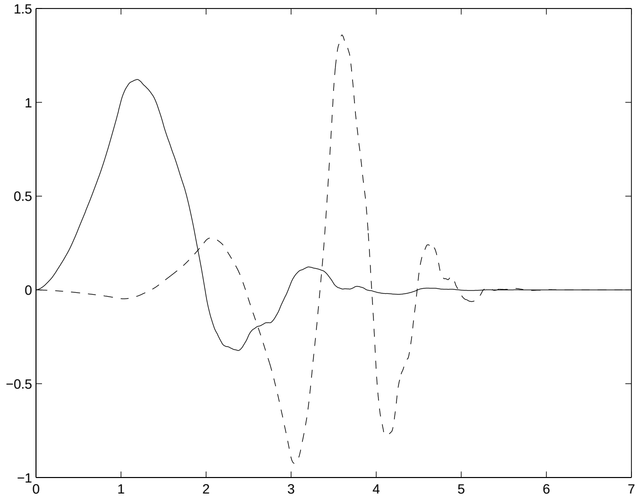
where the c_k are (up to a constant multiple), the filter coefficients `h` of the wavelet. The function `dilation` solves these equations numerically and can be used to graph scaling functions and wavelets (once a solution for the dilation equation has been obtained the function `waveletd` computes the values of the corresponding wavelet).

```
dilation -- solution to a dilation equation

f = dilation(c,levels)

Inputs:
 c        Coefficients from the dilation equation (automatically
          normalized).
 levels   How deep to iterate, result will be calculated on points
          2^(-levels) apart.

Output:
  f       The solution.
```

```
Note! This program first solves the exact values of f on
integers. This means solving an eigenvalue problem, which sometimes
fails. For discontinuous solutions, you must supply the initial data
explicitely, see below.

Optional arguments:
  [f,x] = dilation(c,levels,initf)
  initf   Iteration is started with this vector as initial data; in
          this case 'levels' gives how many new levels to calculate.
  x       Points at which f is given.

See also WAVDEMO, WAVELETD.
```

Example:

```
[h,g] = wavecoef('dau',8);
[f,x] = dilation(h,8);
plot(x,f, x,waveletd(f,x,g), '--');
```



The algorithm fails for some dilation equations (e.g., when the solution is not continuous). The Vaidyanathan wavelet is such an example.

See also `wavdemo`, which provides a graphical interface for these routines. Figure 1 shows an example.

4 Fast wavelet transforms

4.1 One-dimensional

The (periodic) one-dimensional fast wavelet transform is implemented by the functions `fwt1` and `ifwt1`.[2] Let H be the low-pass filter corresponding to the scaling function followed by down-sampling, G the high-pass filter corresponding to the wavelet (followed by down-sampling). Schematically the wavelet transform is shown in figure 2. The inverse transform essentially amounts to reversing the arrows and replacing H and G with their adjoints.

[2] The input vector must have length 2^n.

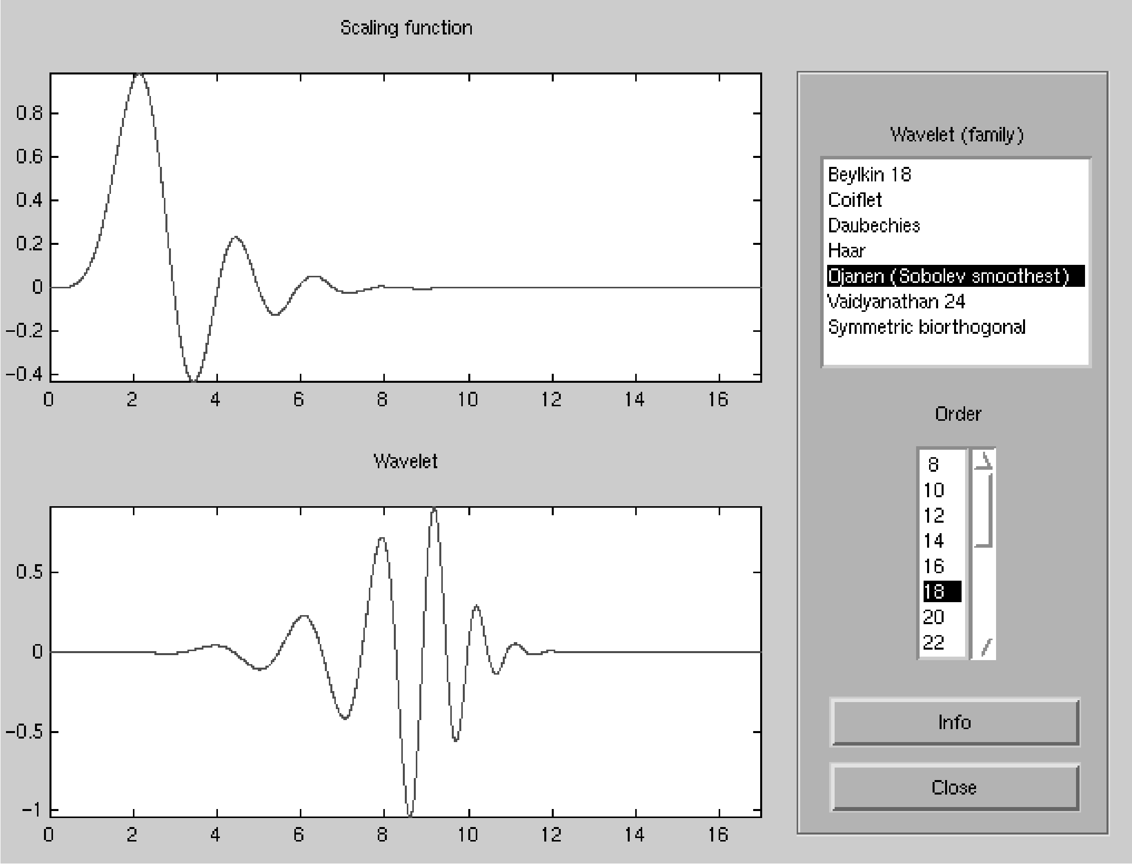


Figure 1: `wavdemo`

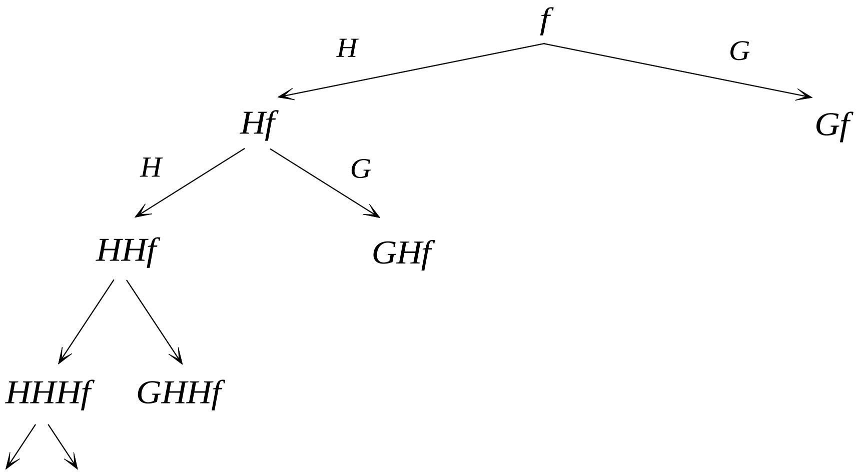


Figure 2: Wavelet transform

The output vector of `fwt1` is partitioned as follows: let (`h` and `g` contain the filter coefficients of H and G. Then

```
w = fwt1(f,h,g)
```

yields

$$w = [\ s_n \mid d_n \mid d_{n-1} \mid \cdots \mid d_2 \mid d_1\ ],$$

where $d_1 = Gf$, $d_2 = GHf$, $d_2 = GHHf$, ..., $d_n = GH \cdots Hf$, and $s_n = HH \cdots Hf$. Here $|d_i| = 2^{-i}|w|$.

```
FWT1 -- fast wavelet transform, one dimensional standard version

w = fwt1(f,h,g)

Input:
    f     Vector to transform
    h     Filter coefficients for the scaling function
    g     Filter coefficients for the wavelet

Output:
    w     contains standard multiscale analysis (column vector)

The wavelet coefficients are stored in the following order:

    w = [sn | dn | d(n-1) | d(n-2) | ... | d2 | d1 ]

where length(sn) = 1, length(di) = 2^(-i)*length(w) and
n = log2(length(w)).

See also IFWT1, FWT1NS, FWT2, FWT2TNS.
```

Inverse transform is computed by `ifwt1`:

```
IFWT1 -- inverse fast wavelet transform, standard version

result = ifwt1(msa, h, g)

h  and  g  are the filter coefficients for the scaling function
and wavelet,  msa  is a standard multiscale analysis, e.g.,
produced by fwt1.

result  is the inverse transform.

fwt1  followed by  ifwt1  is the identity.

See also FWT1, IFWT1NS, FWT2TNS, IFWT2TNS.
```

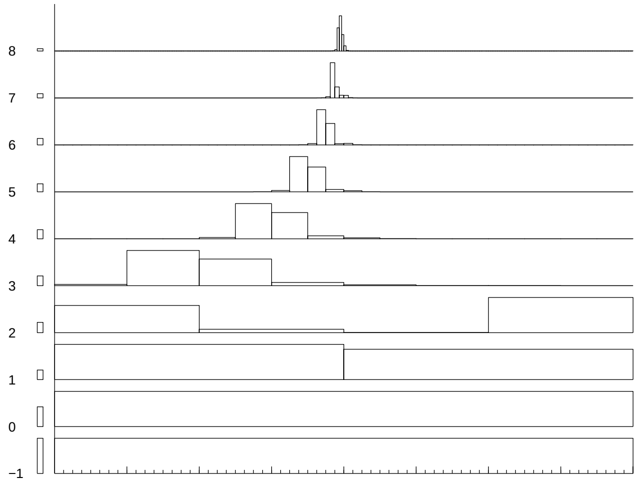
Terminology:

- The output of `fwt1` is also called a multi-resolution analysis (msa).
- The functions `fwt1` and `ifwt1` and the msa they produce are called “standard”, to distinguish them from the functions and data-structures described in section 6.

The following example computes the wavelet coefficients of the function $1/|x-0.5|$ on the interval $[0,1]$ and displays the resulting multi-resolution analysis using `showmsa`. Finally the inverse transform is computed and compared to the original vector.

```
[h,g] = wavecoef('dau',6);
x = (0:1/511:1)';
f = sqrt(1./abs(x-.5));
w = fwt1(f,h,g);
showmsa(w)
g = ifwt1(w,h,g);
norm(f-g)
```

The error was `1.40739e-13`. Here is the figure produced by `showmsa`:



The rows, from bottom to top, are bar plots of s_n, d_n, ..., d_1. Note how in finer levels the only significant coefficients are those near the singularity at 0.5 (but not exactly, since the wavelet is not centered at zero[3]).

4.2 Two-dimensional

There are two different ways to compute the wavelet transform of two-dimensional data. We first describe an approach that results in isotropic basis functions.

Let ψ be the mother wavelet and φ the scaling function, and $\psi_{jk}(x) = 2^{j/2}\psi(2^j x - k)$, $\varphi_{jk}(x) = 2^{j/2}\varphi(2^j x - k)$. The two dimensional basis is given by the collection of functions

$$\psi_{jk}(x)\psi_{jk'}(y), \quad \varphi_{jk}(x)\psi_{jk'}(y), \quad \psi_{jk}(x)\varphi_{jk'}(y).$$

Note that these functions have the same scale in both x- and y-variables (j is the same in both factors).

[3] This is really a bug in `showmsa`, it should take into account the location of the center of the filter. The function `phasepln` for wavelet packets is much better in this respect, see section 5.1 and figure 7.

Let H_1 be the low-pass filter (followed by downsampling) corresponding to φ, acting columnwise (independently on different columns), G_1 the high-pass filter corresponding to ψ; H_2 and G_2 act row-wise. Then the output of `fwt2`[4] is partitioned as follows:

$$\left[\begin{array}{c|c} \cdots & G_1H_2 \\ \hline H_1G_2 & G_1G_2 \end{array}\right],$$

where the three blocks shown have size half the size of the original matrix and the upper left corner contains the same structure recursively (the upper left *element* of the matrix is $H_1 \cdots H_1 H_2 \cdots H_2$).

```
FWT2 -- two dimensional fast wavelet transform

A = fwt2(s, h, g)

Input:
  s     original matrix
  h,g   filter coefficients for the scaling function and wavelet

Output:
  A     resulting two dimensional multi-scale analysis

See also IFWT2, FWT1, FWT1NS, FWT2TNS, FWT2NS.
```

The inverse transform is:

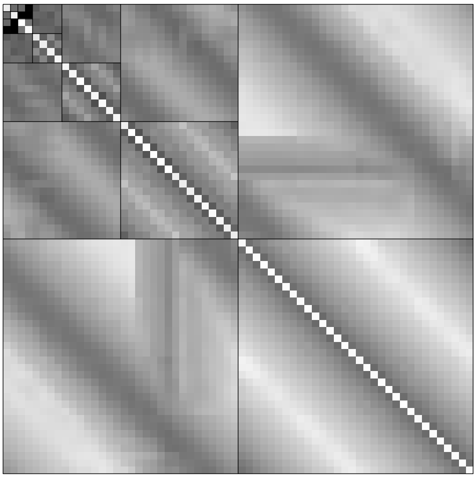
```
IFWT2 -- two dimensional inverse wavelet transform

A = ifwt2ns(w, h, g)

Input:
  w     two dimensional wavelet coefficients
  h,g   filter coefficients for the scaling function and wavelet

Output:
  A     inverse transform of w

See also FWT2, IFWT2TNS, FWT2TNS, IFWT1, IFWT1NS.
```

The function `showoper` displays the matrix containing the wavelet coefficients (actually any matrix) and `nsgrid` can be used to superimpose a grid that explains how the matrix is partitioned. The color or gray level indicates the magnitude of the corresponding element in the matrix.

Example: the transform of the matrix defined by $A_{ij} = 1/(i-j)$, $i \neq j$, and $A_{ii} = 0$.

```
[h,g] = wavecoef('coi',30);
[i,j] = meshgrid(0:63);
A = 1./(i-j);  A(1:65:64^2) = zeros(1,64);
W = fwt2(A,h,g);
showoper(W);  nsgrid
```

[4] `fwt2` is restricted to input matrices of size 2^n by 2^n.



The demo `wav2demo` allows the user to graph the basis functions that are effectively used by this algorithm (see the next section for more information). Another demo, `wavoperd`, knows a few example matrices and provides choices for which wavelets and which type of basis to use. In figure 3, the left hand plot depicts the original matrix, the right hand plot is the transformed matrix. Black indicates large magnitudes, white values close to zero (on a logarithmic scale).

The different operators correspond to the following integral operators (the matrices are simply a naive discretization of the operator):

- Conjugate function: $Cf(x) = \int \frac{1}{\tan((x-y)/2)} f(y)\, dy$, i.e., the matrix is $C_{ij} = \left[1/\tan((x_i - y_j)/2)\right]$.
- Hilbert transform: $Hf(x) = \int \frac{1}{x-y} f(y)\, dy$
- An operator like a conjugate function but with also a radial function in the kernel: $Rf(x) = \int \frac{h(|x-y|)}{\tan((x-y)/2)} f(y)\, dy$, where $h(r) = \cos(r^2)$, defined in `hfun`.
- Calderón commutator: $Sf(x) = \int \frac{a(x)-a(y)}{x-y} \frac{1}{\tan((x-y)/2)} f(y)\, dy$, $a(x) = 100\sqrt{|x|} + \sqrt{|x+\pi/2|} - \sqrt{|x-\pi/2}$, defined in `afun`.

These, and other matrices, are defined in the function `nsexampl`.

4.3 Two-dimensional: tensor products

Let $\psi_{jk}(x) = 2^{j/2}\psi(2^j x - k)$ be the one-dimensional wavelet basis as in the previous section. The tensor product basis is given by the collection of functions

$$\psi_{jk}(x)\psi_{j'k'}(y).$$

Note that the scale varies independently in the x- and y-directions (j and j').

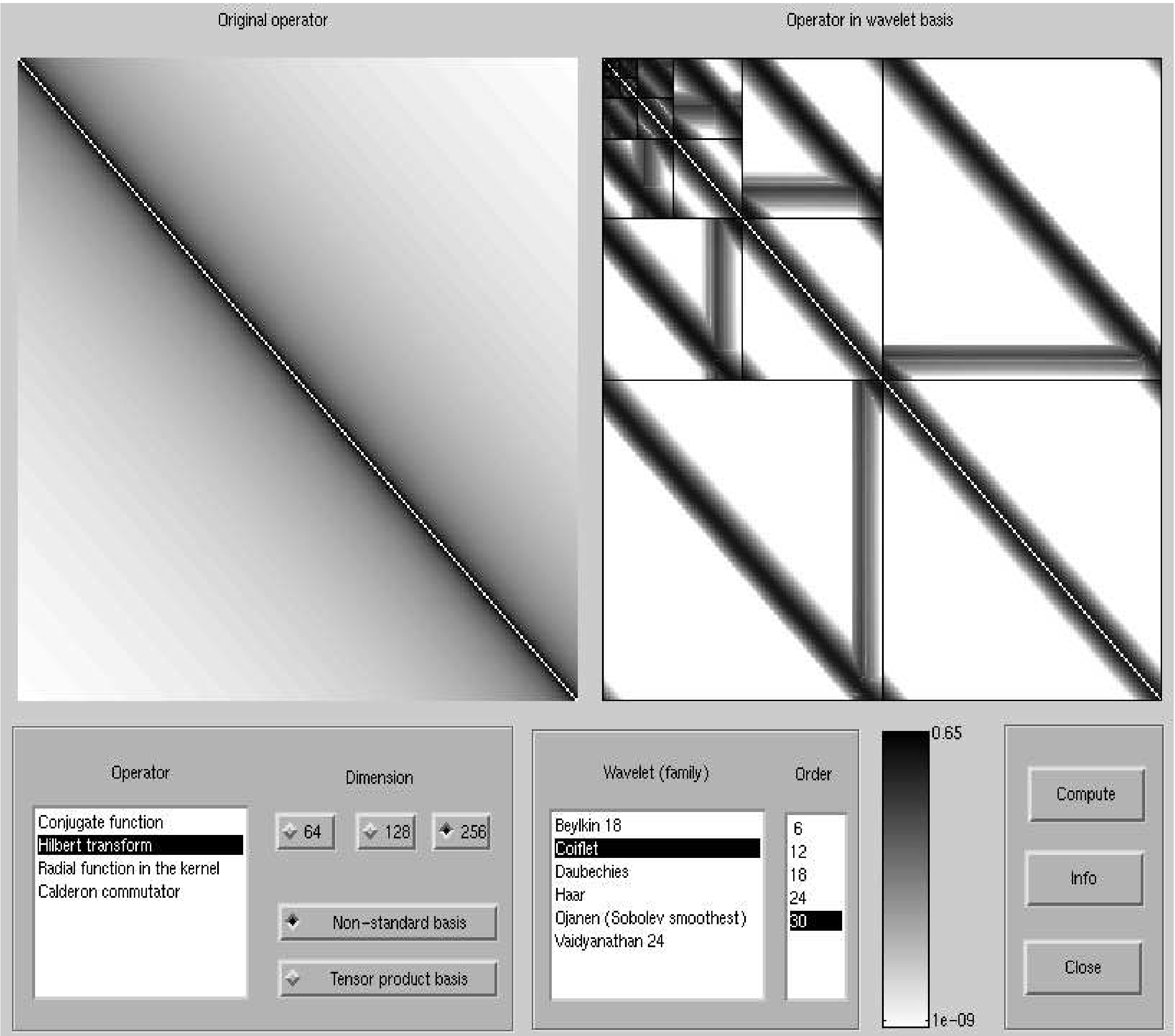


Figure 3: `wavoperd`

The tensor-product transform amounts to simply first computing one-dimensional wavelet transforms of each row of the input matrix, collecting the resulting coefficients into a new matrix, and finally computing the one-dimensional transform of each column of this matrix.

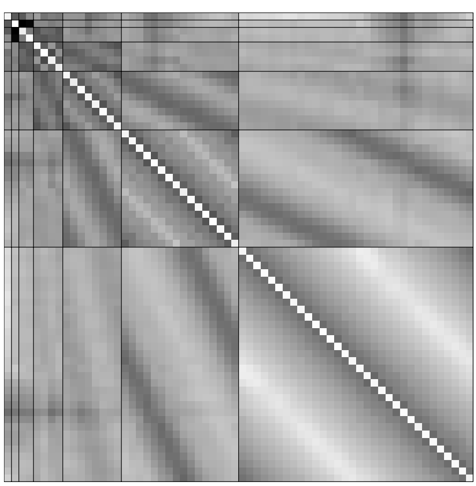
Partitions (compare with `fwt1`, section 4.1) for an 8 by 8 matrix:

$$
\left[\begin{array}{c|c|cc|cccc}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \hline
\cdot & \cdot & \cdot & \cdot & \cdot & \multicolumn{2}{c}{G_1H_1H_1G_2} & \cdot \\ \hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & & & & \multicolumn{4}{c}{G_1H_1G_2} \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & & \multicolumn{2}{c|}{G_1G_2H_2} & \multicolumn{4}{c}{G_1G_2} \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right],
$$

These operations are implemented by `fwt2tns` and the inverse transform by `fwt2tns`. They take the same arguments as `fwt2` and `ifwt2`, see section 4.2. Again, `showoper` displays the matrix containing the wavelet coefficients, `tnsgrid` can be used to superimpose a grid, and `wav2demo` to graph the basis functions.

Example (continued from section 4.2):

```
...
Wtns = fwt2tns(A,h,g);
showoper(Wtns);  tnsgrid
```



The demo `wav2demo` graphs basis functions interactively. In figure 4, the diagram on the left represents the matrix of wavelet coefficients, the small square indicates which entry is 1, others are equal to zero. The graph on the right is the corresponding wavelet. Note that the wavelet has different scales in x and y directions, which is typical of the tensor product basis.

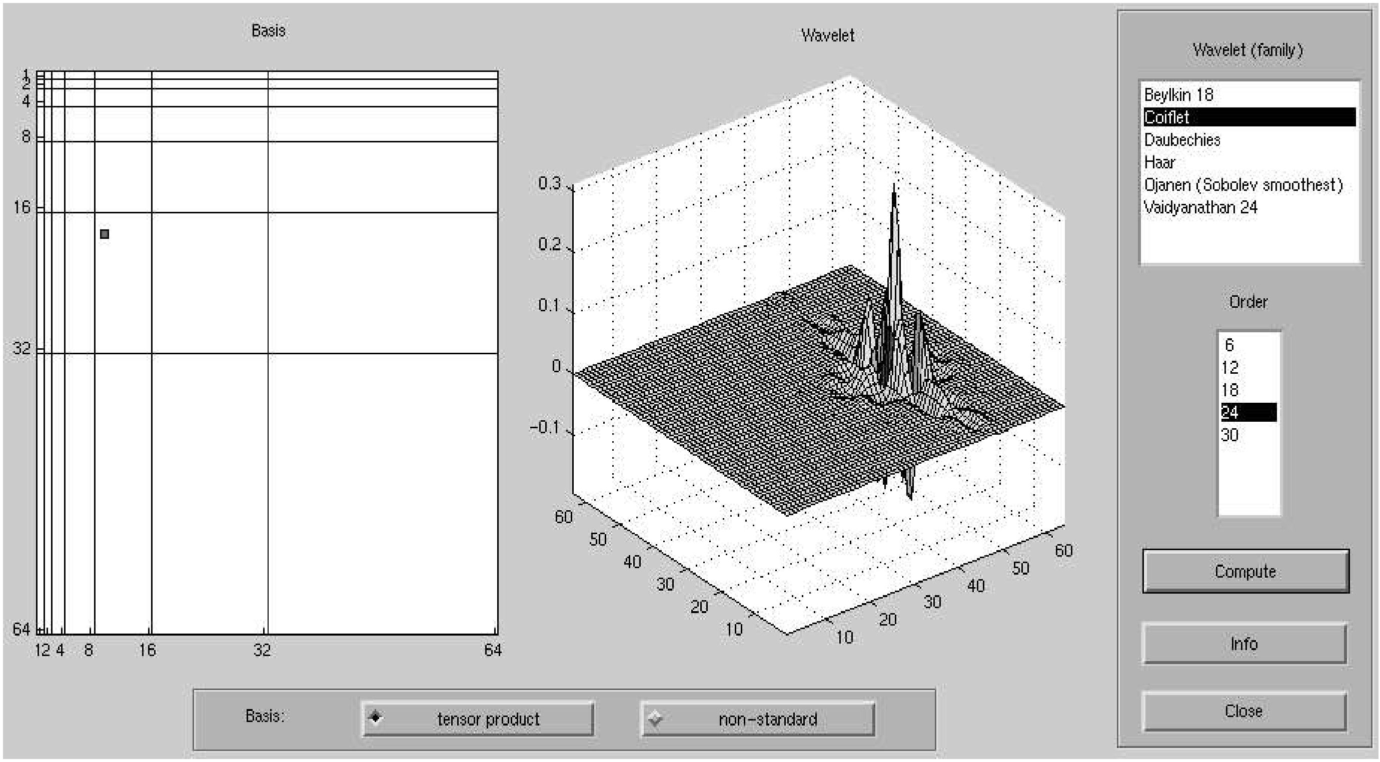


Figure 4: `wav2demo`

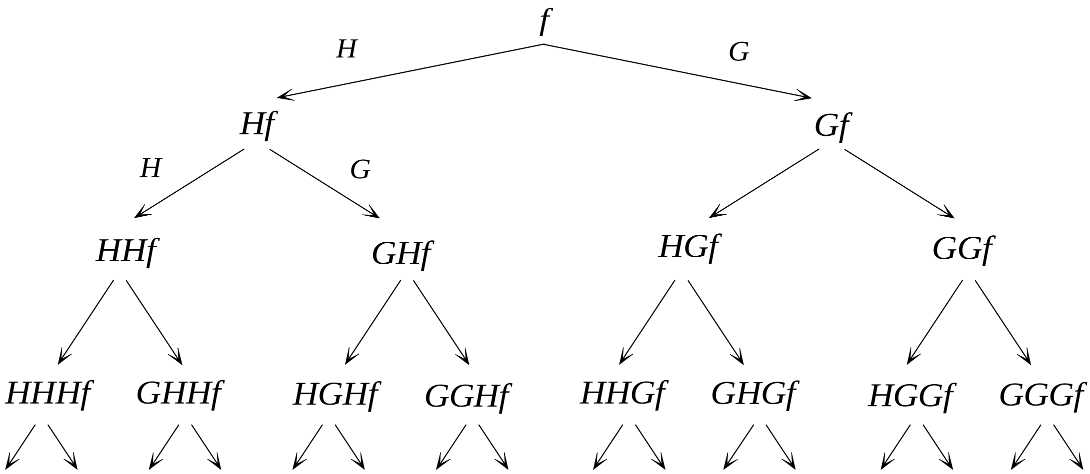


Figure 5: Wavelet packet analysis

5 Wavelet packets

5.1 One-dimensional

The wavelet packet analysis (`wpa1`) is to wavelet packets what `fwt1` is to wavelets. The algorithm is shown in figure 5 (compare with figure 2). Now a full binary tree is computed starting from the initial vector f. Left branches are the result of an application of the low-pass filter (followed by decimation by 2) H, a right branch an application of G. If $|f| = 2^n$, each row in the tree has a total of 2^n entries, for a total of $n2^n$ entries (not counting f), hence this is now an overdetermined system. The idea is to choose only parts of the tree (to make a basis) to represent f. This freedom of choice provides enormous flexibility.

Figure 6 shows examples of wavelet packets created with `wpdemo`. In these pictures it is clear that there are both mostly oscillatory and mostly transient functions in the collection. This is very different from wavelets, where each wavelet is a translate and dilate of a single function. The letters `s`, `f`, and `p` refer to the scale, frequency, and position (translate) of the packet.

Note that a wavelet basis corresponds to a fixed choice of a subtree in figure 5.

Datastructure The output of `wpa1`[5] is a Matlab structure with the following fields:

- `wp`: Contains the entries from the tree in figure 5. If the input vector has length 2^n, then `wp` is an n by 2^n matrix. The rows contain the blocks from the tree in figure 5 in the order they are shown.

- `sel`: An `uint8` (incidence) matrix the same size as `wp`, entries are either 0 or 1 (`wpa1` always returns all zeroes, the entries are changed when a basis is selected). A 1 means the corresponding entry in `wp` is to be included in the basis, 0 that it is left out.

A basis can be selected with the following functions:

[5] `wpa1` is restricted to input vectors whose length is a power of two.

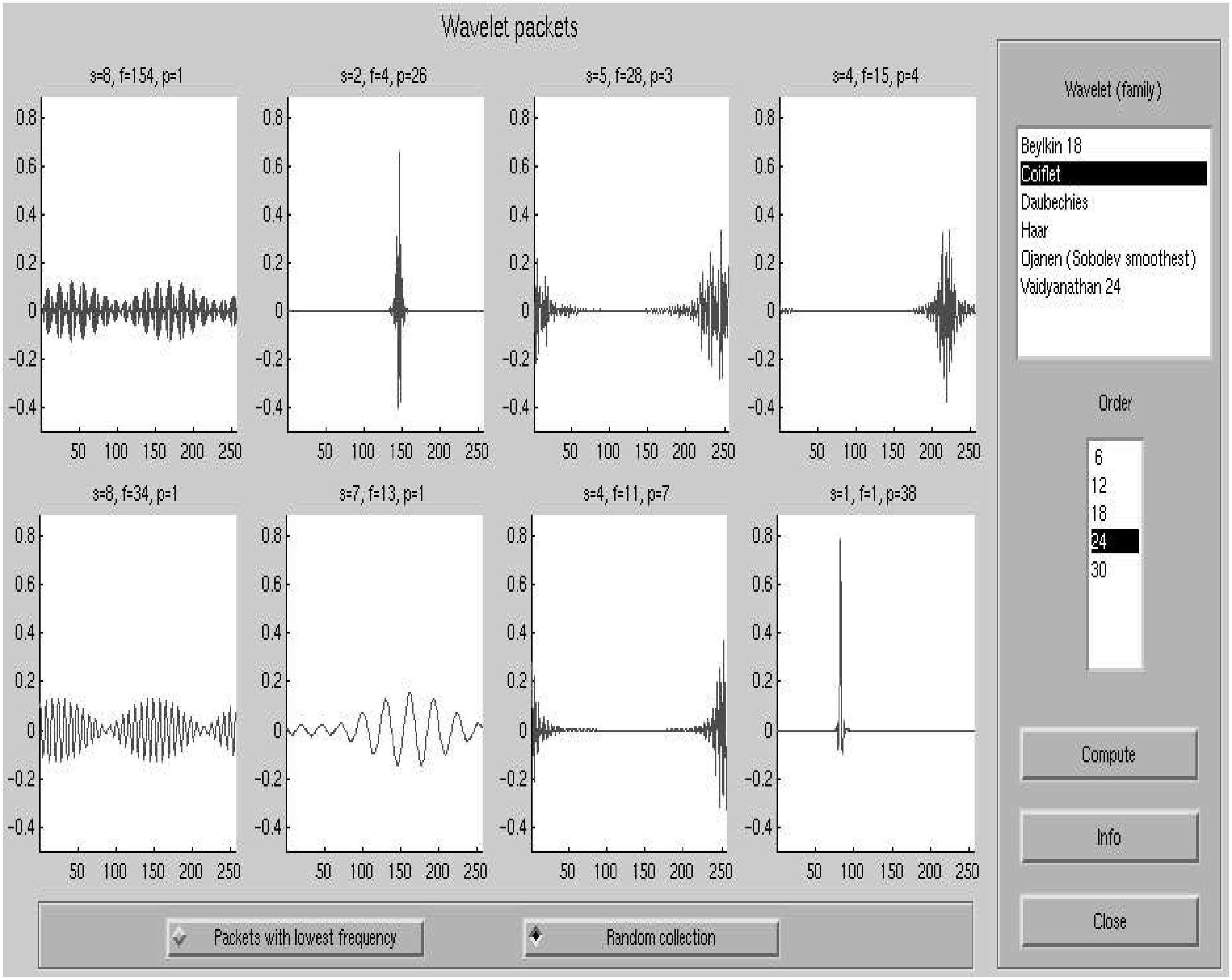


Figure 6: `wpdemo`

- `bestbase`: Best basis, i.e., lowest cost basis, with respect to a cost function (see below).
- `bestlevl`: A fixed level (with lowest cost) is chosen from the tree, i.e., all blocks are from the same row in figure 5.
- `fixlevel`: A fixed user specified level.
- `wavbasis`: Wavelet basis.
- `showwp`: This program allows the user to interactively select the basis (see below).

Cost functions The function `setcostf` is used to select the cost function for `bestbase`, `bestlevl`, `bestbas2`, and `bestlvl2`. The following functions are implemented (see also [13]):

- ℓ^p norm $\sum |x_i|^p$, useful for p close to zero.
- ℓ^2 entropy (unnormalized) $-\sum x_i^2 \log x_i^2$.
- Counting the number of elements above a threshold.

Tools for visualization The heart of the wavelet packet programs is `showwp`, which allows all of the above operations and many others to be performed interactively. An example (from the demo `wpsig`) is shown in figure 7, where a sinusoid with noise is analyzed. The top part is a tree as in figure 5, the blocks with thick boundaries are those selected in the basis. The lower half of the figure is a phaseplane plot (frequency against time with amplitude shown in different shades of gray) generated by `phasepln`. Note how the wavelet packets recognize the frequency of the sinusoid by concentrating most of the wavelet packet coefficients to the two black horizontal bands.

```
SHOWWP -- shows wavelet packet coefficients and many other things

showwp(w, h, g)
showwp(w, f, h, g)

Displays the tree of the wavelet packet coefficients in w. f is
the original vector (possibly an empty vector). If any selections
are made, the selected packets are made to stand out. Both f and
selection are optional.

The display is divided into two parts: in the top part is the tree
of the wavelet packet coefficients, the bottom part is used for
graphing other things.

The original vector is the very first row of the tree. Here white
means large values, black small (possibly) negative values.

The wavelet packet coefficients form the rest of tree. The left
branches are the result of convolution-decimation with the low
pass filter (scaling function), right branches with the high pass
filter (wavelet). Here only the absolute values of the
```

Figure 7: `showwp` and `wpsig`

```
coefficients are drawn: white means large absolute value, black
means zero.

Clicking in a box on the first row shows the original vector,
clicking in any other box shows the graph of the wavelet packet
corresponding to that box.

Select menu:
    Packet          toggle the selected/unselected state of a
                    single packet
    Group           toggle the selected/unselected state of a box
    Fixed level     click anywhere in the tree to select a level
    Wavelet basis   selects the wavelet basis
    Best level      selects the level with lowest cost
    Best basis      selects the basis with lowest cost
    Clear           clears all selections

Action menu:
    Inverse transform    computes the inverse transform using
                         current selection
    Phaseplane plot      displays the phaseplane plot for the
                         currently selected basis
    Nonincreasing rearrengement
                         displays the nonincreasing rearrengement
                         of the wavelet packet coefficients
    Compute cost         displays the cost of various things

Discard menu:
    Discards all coefficients whose absolute value is less than the
    selected percentage of the L2-norm of the whole collection of
    wavelet packet coefficients.

Options menu:
    Set cost function      allows to select the cost function
    Show cost function     shows which cost function is being used
    Natural order          switches the display to natural order
    Sequency order         switches the display to sequency order
    --> Color              switches the display to color
    --> B/W                switches the display to black and white

See also PHASEPLN, WMENU, WPDEMO, WPA1, WPS1, BESTBASE, BESTLEVL,
         FIXLEVEL, WAVBASIS.
```

5.2 Two-dimensional

Two dimensional wavelet packet analysis is done with `wpa2`,[6] synthesis with `wps2`. A basis can be selected with `bestbas2`, `bestlvl2`, `fixlvl2`, or `wavbase2` (compare with the previous section).

The data-structure returned by `wpa2` is a structure with fields `wp` and `sel` (an incidence matrix as before for `wpa1`). If the input is a 2^n by 2^n matrix, these fields are 2^n by 2^n by n, the third index being the level.

[6]The input matrix must have dimensions 2^n by 2^n.

The matrices are partitioned as follows: Let H_1 be the low-pass filter (followed by downsampling) corresponding to φ, acting columnwise (independently on different columns), G_1 the high-pass filter corresponding to ψ; H_2 and G_2 act row-wise. Then the level one submatrix (i.e., `wp(:,:,1)`) is computed as follows:

$$\left[\begin{array}{c|c} H_1 H_2 & G_1 H_2 \\ \hline H_1 G_2 & G_1 G_2 \end{array}\right],$$

where each block has size half the size of the original matrix. The level two submatrix, `wp(:,:,2)` is (by applying the above matrix to itself recursively)

$$\left[\begin{array}{c|c||c|c} H_1 H_1 H_2 H_2 & G_1 H_1 H_2 H_2 & H_1 G_1 H_2 H_2 & G_1 G_1 H_2 H_2 \\ \hline H_1 H_1 G_2 H_2 & G_1 H_1 G_2 H_2 & H_1 G_1 G_2 H_2 & \cdot \\ \hline\hline H_1 H_1 H_2 G_2 & \cdot & \cdot & \cdot \\ \hline H_1 H_1 G_2 G_2 & \cdot & \cdot & \cdot \end{array}\right],$$

where each block has size one quarter the size of the original matrix.

Figure 8 shows a two dimensional wavelet packet. The picture is from `wp2demo`, which is similar to `wav2demo`, see figure 4. The diagram on the left corresponds to how the matrix is partitioned (the resolution level is selected at the bottom). The cost function used was $\ell^{0.25}$.

Operators can be represented in wavelet packet bases, this is done in the demo `wpoperd` (compare with `wavoperd` in section 4.2). Figure 9 shows an example.

6 Fast matrix multiplication

6.1 Using wavelets: non-standard bases

The algorithm is described in many places, see e.g., [3, 2, 13]. The basic idea is to represent both the matrix and the vector in a wavelet basis and multiply the transformed matrix and transformed vector. This algorithm requires access to the intermediate results of the low-pass filter H (which are normally discarded in a wavelet transform). The function `fwt1ns` returns a vector which contains also these blocks, hence the name non-standard.

The matrix is transformed with `fwt2` as usual, the vector with `fwt1ns`, the multiplication is done by `nsmult`, and the product is inverse transformed with `ifwt1ns`. See also the demo `wavmultd` (and `nsexampl` for a source of test matrices).

The output vector of `fwt1ns` is partitioned as follows: Let H be the low-pass filter corresponding to the scaling function followed by down-sampling, G the high-pass filter corresponding to the wavelet (followed by down-sampling). Then (`h` and `g` contain the filter coefficients)

```
w = fwt1ns(f,h,g)
```

yields

$$w = [\ d_n \mid s_n \mid \ \cdots \ \mid d_2 \mid s_2 \mid d_1 \mid s_1\ ],$$

where $d_1 = Gf$, $s_1 = Hf$, $d_2 = GHf$, $s_2 = HHf$, ..., $d_n = GH\cdots Hf$, and $s_n = HH\cdots Hf$. Here $|d_i| = |s_i| = 2^{-i}|w|$.

Example: The product of $A_{ij} = 1/|i-j|$, $i \neq j$, $A_{ii} = 0$, with a random vector.

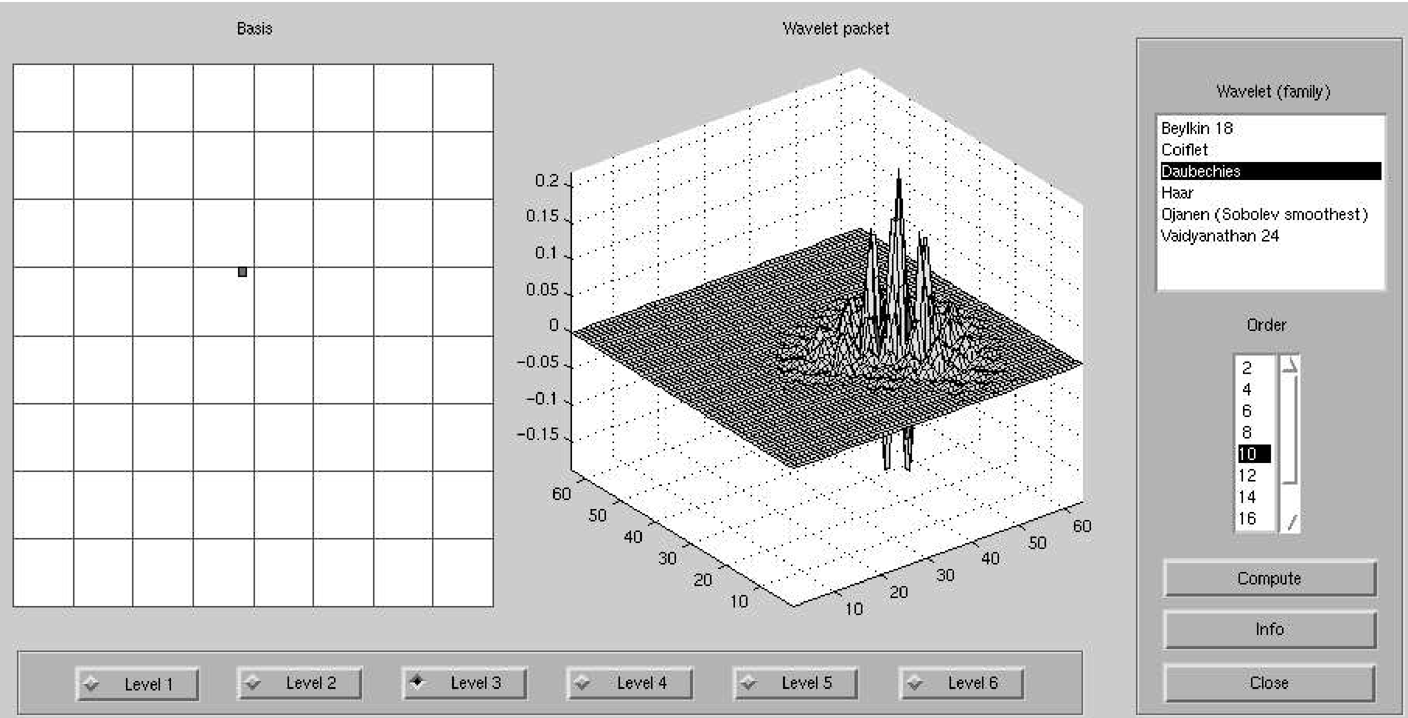


Figure 8: `wp2demo`

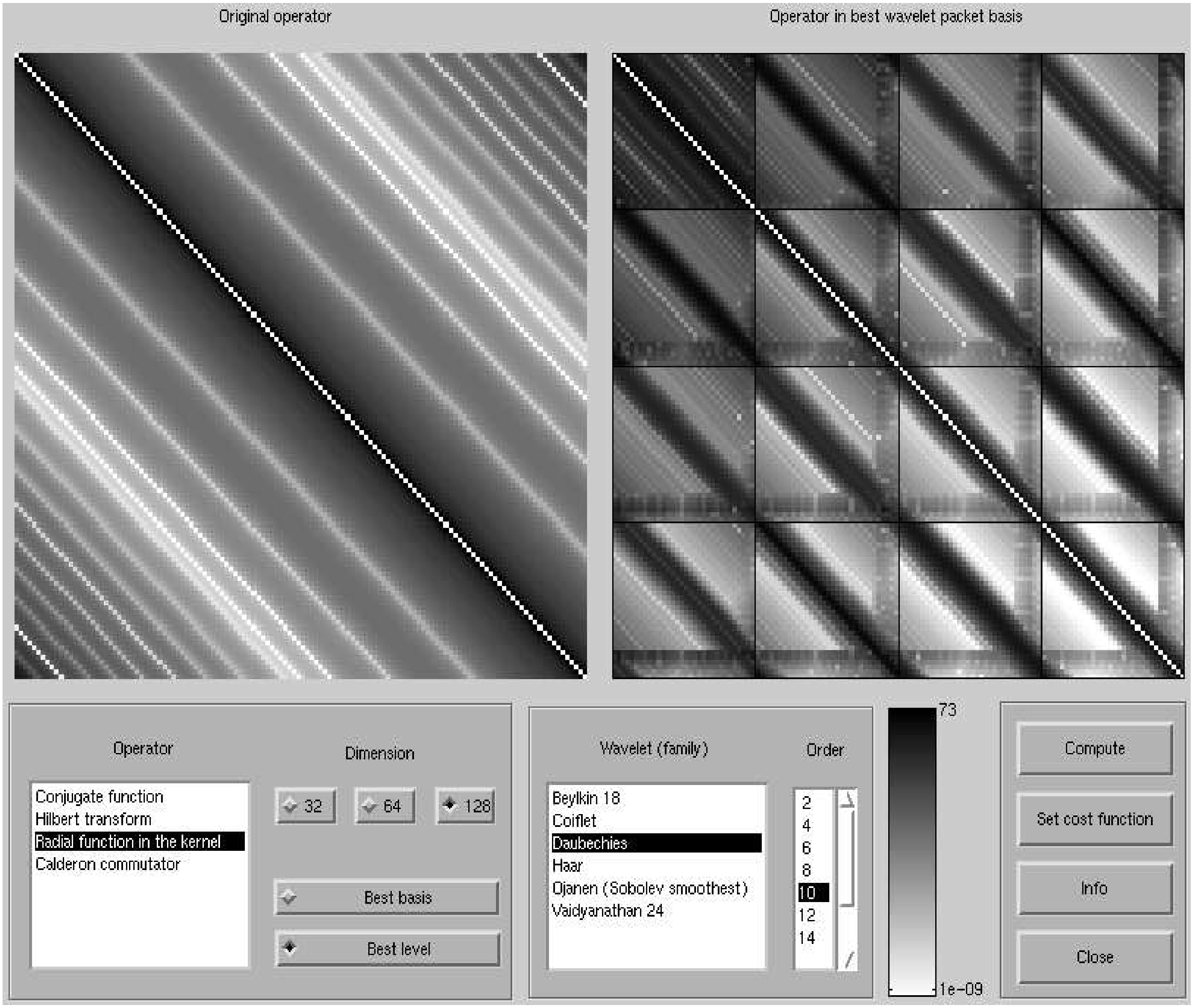


Figure 9: `wpoperd`

```
[h,g] = wavecoef('coi',30);
[i,j] = meshgrid(0:31);
A = 1./(i-j);  A(1:33:32*32) = zeros(1,32);
WA = fwt2(A,h,g);
f = rand(32,1);
wf = fwt1ns(f,h,g);
p = ifwt1ns(nsmult(WA,wf),h,g);
norm(p-A*f)
```

The error was `9.81109e-15`.

Note that `fwt1ns` followed by `ifwt1ns` is not the identity. There must be an intermediate multiplication by the transform of the identity matrix (see `help nsexampl`).

In the demo `wavmultd` the user can choose which operator to study and specify at what level the wavelet coefficients are truncated. Figure 10 shows the transformed matrix on the left. Only those matrix elements that are larger than the specified (relative) truncation level are displayed and used when computing the product. The graphs on the right are, from top to bottom, the original vector (chosen randomly), the product when there is no truncation, product computed with the truncated matrix, and the error between the last two.

6.2 Using wavelet packets

Matrix multiplication can also be done using wavelet packets. The idea is to compute the wavelet packet analysis of the matrix (`wpa2`) and choose a basis for it (`bestbas2`, `fixlvl2`, `wavbase2`). Then the vector is transformed (`wpa1`) and the multiplication is done on the transformation side (`wpmult`); the basis chosen for the matrix determines which basis is eventually used for the vector and for the product. For more information see [13].

Example (continued from section 6.1):

```
...
WPA = bestbas2(wpa2(A,h,g));
wpf = wpa1(f,h,g);
p = wps1(wpmult(WPA,wpf),h,g);
norm(p-A*f)
```

The error was `1.05801e-14`.

7 Other demonstrations

7.1 Chirps

Chirps are functions of the form $\sin(f(t)t)$, i.e., the frequence $f(t)$ changes with time. There are two demos that try to illuminate the properties of wavelets and wavelet packets by analyzing chirps:

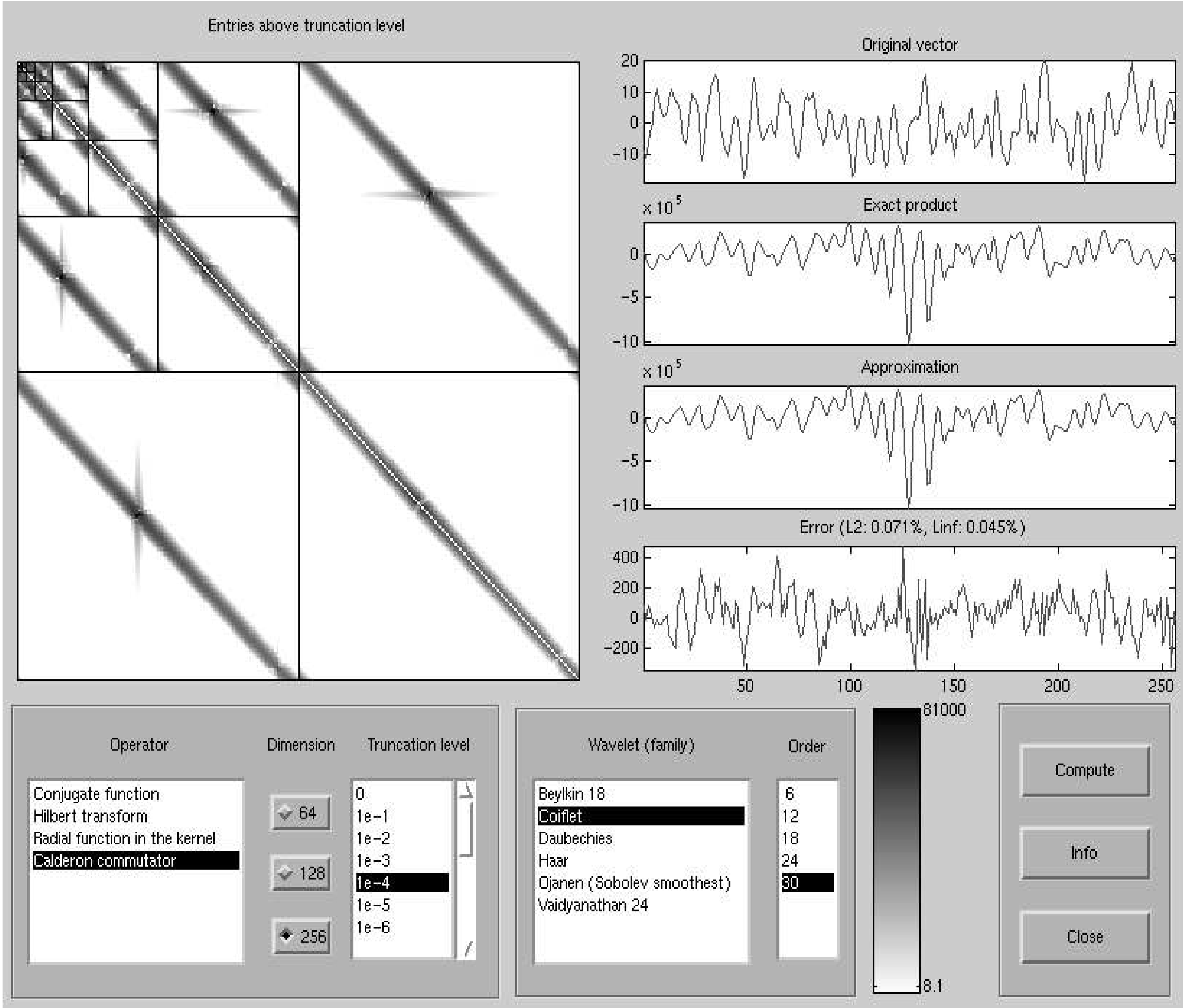


Figure 10: `wavmultd`

- `chrpdemo`: (see figure 11) this demo provides several choices for the signal (see `help chrpdemo` for more details about the signals) and provides two plots: amplitude against time and frequence against time (with amplitude shown with different shades of gray). See also `help chrpdemo` for more information.

- `chrpcomp`: (see figure 12) FFT, wavelets, and wavelet packets are used to analyze two signals, a cubic chirp and the superposition of a linear and a cubic chirp from `chrpdemo`. Wavelet packets can best find the properties of both signals, wavelets work reasonably for the simpler signal, but FFT does not give any reasonable sense out of either one.

7.2 Image processing

The demo `imgdemo` is a very naive image processing example, but it illustrates some properties of wavelets nicely. See figure 13. Note how in the wavelet transform (upper right) different blocks in the matrix recognize horizontal, vertical, or diagonal details in the image. Similar information is contained in blocks of different sizes, corresponding to different resolutions.

References

[1] J. J. Benedetto and M. W. Frazier, editors. *Wavelets: mathematics and applications*. CRC Press, 1994.

[2] G. Beylkin. Wavelets and fast numerical algorithms. In I. Daubechies, editor, *Different perspectives on wavelets (San Antonio, TX, 1993)*, Proc. Sympos. Appl. Math., 47, pages 89–117. Amer. Math. Soc., Providence, RI, 1993.

[3] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. I. *Comm. Pure Appl. Math.*, 44(2):141–183, 1991.

[4] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure and Applied Math.*, 41:909–996, 1988.

[5] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Appl. Math., 1992.

[6] I. Daubechies. Orthonormal bases of compactly supported wavelets II. Variations on a theme. *SIAM J. Math. Anal.*, 24(2):499–519, 1993.

[7] Y. Meyer. *Wavelets and operators*. Cambridge University Press, 1992.

[8] Y. Meyer. *Wavelets: algorithms and applications*. SIAM, 1993.

[9] H. Ojanen. Remarks on the Sobolev regularity of wavelets and interpolation schemes. Research Reports A305, Helsinki Univ. of Technology, Inst. of Math., 1991.

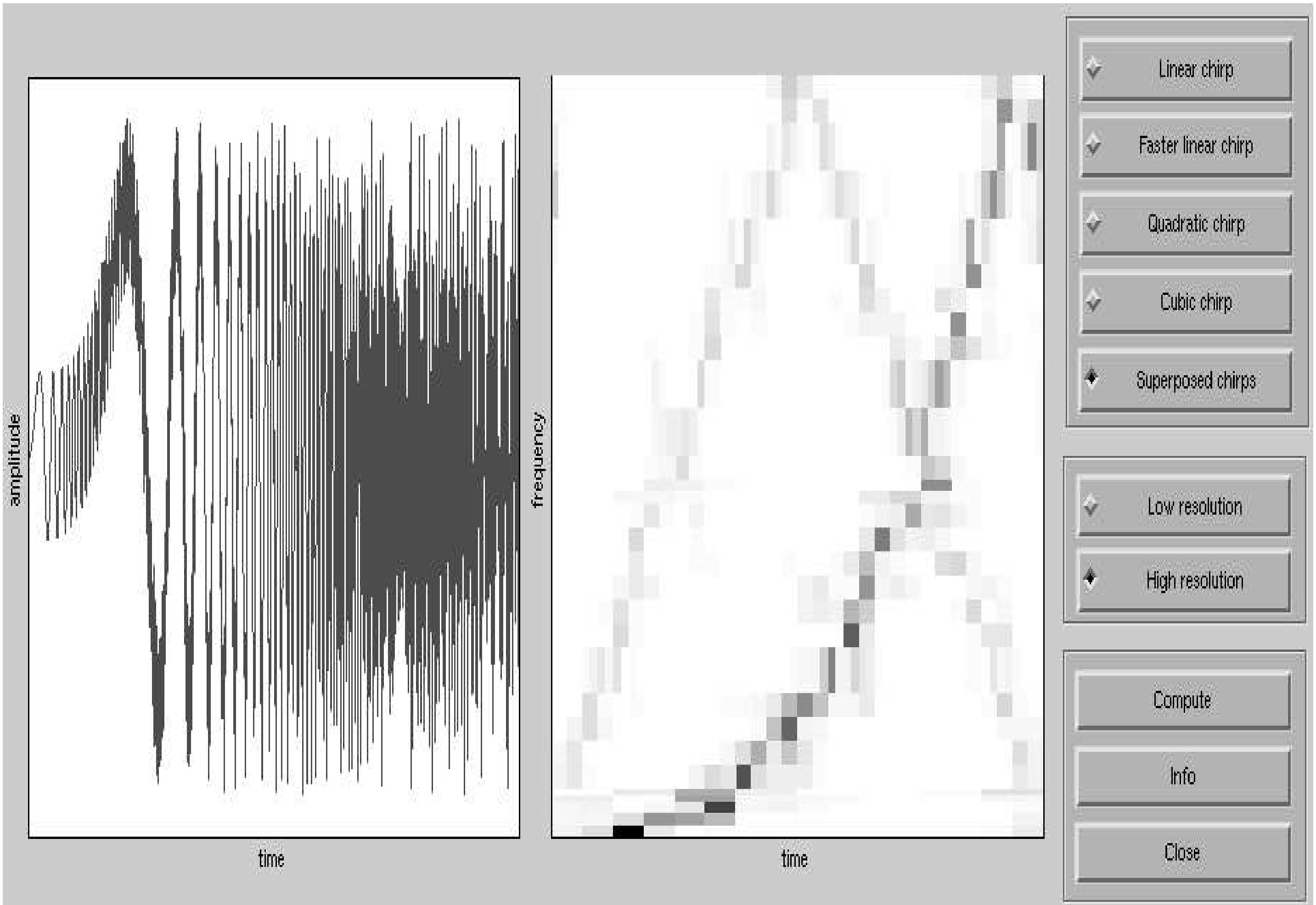


Figure 11: `chrpdemo`

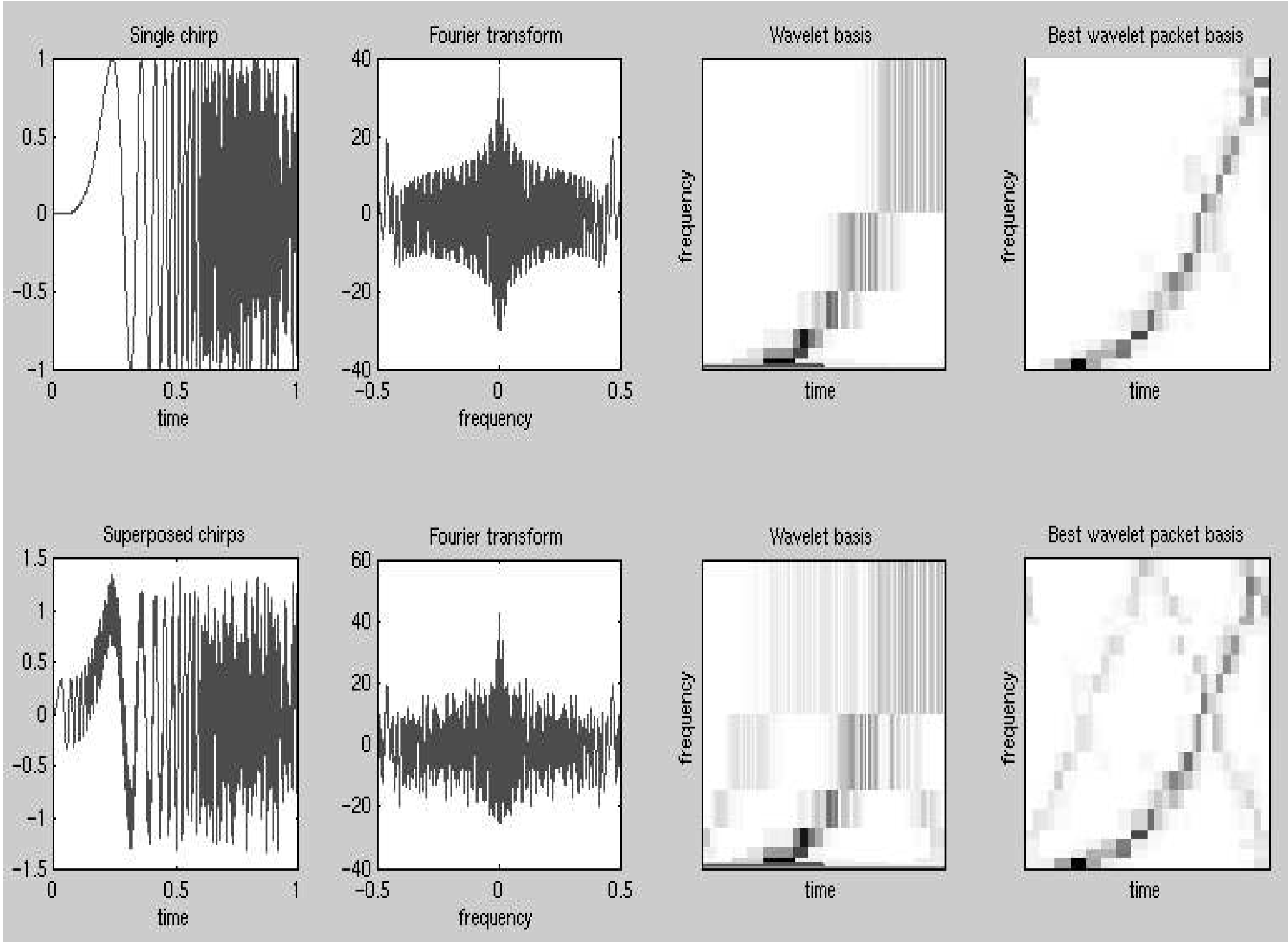


Figure 12: `chrpcomp`

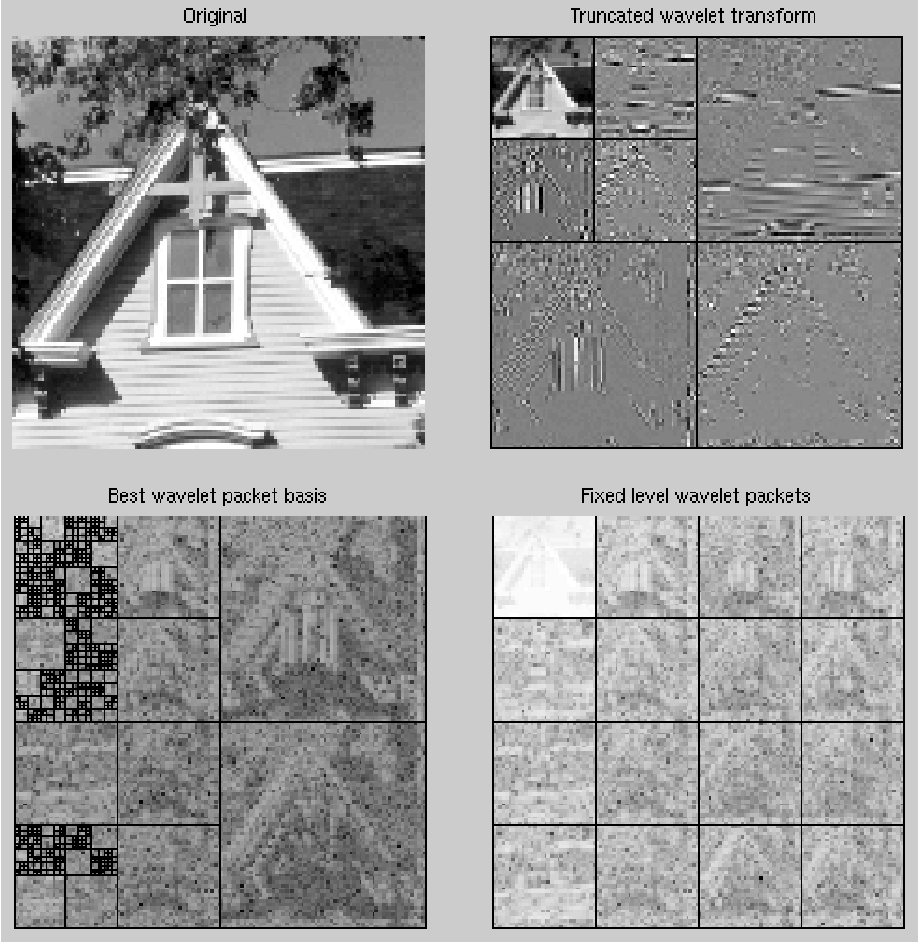


Figure 13: `imgdemo`

[10] H. Ojanen. Orthonormal compactly supported wavelets with optimal Sobolev regularity. Rutgers Univ. Math. Dept., preprint, June 1998.

[11] P. P. Vaidyanathan and P.-Q. Huong. Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 36(1):81–94, 1988.

[12] M. Vetterli and J. Kovačević. *Wavelets and subband coding*. Prentice Hall, 1995.

[13] M. V. Wickerhauser. *Adapted wavelet analysis from theory to software*. A K Peters, 1994.