WaveLab Architecture

Jonathan Buckheit and David Donoho,
Stanford University [1]

Version .850
December, 2005

[1] **Acknowledgment of Support.** This work was partially supported by NSF DMS 92-09130 and 95-05151 (Stanford), by the NASA Astrophysics Data Program (NASA-Ames), ASOR MURI, by DARPA BAA-9804, by NSF KDI, NSF DMS 00-77261, NSF DMS 01-40698 and by other sponsors.

Contents

Chapter 1

Introduction

This document describes the architecture of Wavelab version .850. It is designed for users who already have had day-to-day interaction with the package and now need specific details about the architecture of the package, for example to modify components for their own research.

For an introduction to Wavelab at an elementary level, see *About WaveLab*. This document may be accessed via WWW through the Wavelab Home Page.

Before beginning, we mention the main components of the Wavelab package, to standardize terminology. First, there are the basic "system components":

1. *Source.* There is source code, in Matlab, C, TeX, Perl.
2. *Build.* The source code is assembled into a standard release. The current release is .850.
3. *Archives.* Compressed archives of the standard release available for three platforms, Mac, Unix and PC, which users can download and install on their machines.
4. *Web Documents.* A web home page (which can be viewed using any web browser) and a series of postscript and pdf files which explain what Wavelab is and how to get it. The URL is http://www-stat.stanford.edu/~ wavelab.

Next there are the basic "user components" of an installed system:

1. *Wavelab Main Directory.* A subdirectory /Wavelab850 of the `Matlab/Toolbox` directory, containing the currently released version of Wavelab software, datasets and documentation.

2. *Scripts.* A subdirectory /Wavelab850/Papers/ of /Wavelab850 contains scripts reproducing figures in various articles and technical reports.

3. *Workouts.* A subdirectory /Wavelab850/Workouts/ of / Wavelab850 contains workouts that exercise various aspects of Wavelab.

4. *Documentation.* Both pdf and Postscript files available by WWW access.

5. *Browsers.* High-level tools which give a point-and-click interface to basic wavelet transform features in Wavelab.

6. *Datasets.* Numerical, acoustic and image data used to illustrate various aspects of wavelet analysis by the scripts and workouts.

The following document describes all these various components from a systems-level point of view. An individual needing to modify Wavelab or add to it would be interested in this information.

Chapter 2

Scripts

We briefly describe the contents and architecture of the /Wavelab850/Papers/ subdirectory of Wavelab.

2.1 Script Philosophy

The makeup of /Wavelab850/Papers/ is the whole raison d'être of the Wavelab package. The idea is that, when doing research in a computational science, one works to develop reproducible knowledge about the results of computational experiments. The `/Papers` directory is the end product of such an effort. It makes available to researchers around the world, via the Internet, the computations that produced figures which have been published in hardcopy form as technical reports at Stanford University and in forthcoming journal articles. Other researchers can obtain the Matlab code which generated these figures, and can reproduce the calculations that underly the figures. They can, if they wish, modify the calculations by editing the underlying Matlab code. They can use the algorithms on other datasets. They can try their own favorite methods on the same datasets.

Our idea is that, when doing research, long before we write an article, we prepare ourselves with the thought that *what we do on the computer will ultimately be made available to others, for their inspection, modification, re-use and criticism.* This implies several things. First, that the work product which we are aiming to create will be a subdirectory of Wavelab containing a series of scripts that will generate, from scratch, all the figures of the corresponding article. Second, that our work product is *not* the printed figures that go into the article, but the underlying algorithms and code which generate those figures, and which will be made available to others.

Thus, it is no good to print a hardcopy of a figure that we see on the screen and save that for photocopying into a final version of the paper. Once we are happy with a figure we see on the screen, we must save the code that generated the figure, and then edit the code to make it part of a system that automatically reproduces all the figures of an article.

The philosophy we are adopting can be traced to Jon Claerbout and Martin Karrenbach's article *Electronic Documents Give Reproducible Research New Meaning* (`http://sepwww.stanford.edu`). We especially like a thought of theirs which we paraphrase as follows:

> *A traditionally published article is not the end product of scholarship; it is the advertisement for the scholarship. The working software environment that produced the figures in the article is the actual end product of the scholarship.*

To work in accordance with the philosophy, we must adopt a discipline of how we structure our computational experiments in Matlab. A benefit of this discipline is, hopefully, to avoid the sloppiness and error that are ubiquitous in computational science.

2.2 Script Architecture

The architecture of the `/Papers` directory is as follows. At present, it contains these subdirectories, recreating figures in published articles:

```
AdaptDemo  - ``Adapting to Unknown Smoothness via Wavelet Shrinkage''
AsympDemo  - ``Wavelet Shrinkage: Asymptopia?''
BlockyDemo - ``Smooth Wavelet Decompositions with Blocky Coefficient Kernels''
CorrelDemo - ``Wavelet Threshold Estimators for Data with Correlated Noise''
IdealDemo  - ``Ideal Spatial Adaptation via Wavelet Shrinkage''
MESDemo    - ``Minimum Entropy Segmentation''
MIPTDemo  - ``Nonlinear Wavelet Transforms based on Median-Interpolaton''
RiskDemo  - ``Exact Risk Analysis of Wavelet Regression''
SCDemo     - ``Nonlinear Wavelet Methods for Recovery of Signals, Densities
               and Spectra from Indirect and Noisy Data''
CSpinDemo   - ``Translation-Invariant De-Noising''
TourDemo   - ``Wavelet Shrinkage and W.V.D. -- A Ten-Minute Tour''
VdLDemo    - ``WaveLab and Reproducible Research''
```

These subdirectories have been created following several rules, which should be followed in making future additions.

1. Each article gets one subdirectory of /Wavelab850/Papers/.

2. Each subdirectory contains: (a) meta-routines that run the whole figure-generating process, (b) scripts that generate individual figures, and (c) specialized tools, not present in Wavelab proper, for generating the figures.

3. The files in a subdirectory have stylized names, so that the name indicates the function of the file.

4. Stylized names are based on a *name* and a *short prefix*. The name should be short but descriptive, for example, `Adapt` for scripts associated with the paper *Adapting to Unknown Smoothness via Wavelet Shrinkage* and the prefix should be a related tag, just two-characters long, for example `ad`.

5. The subdirectory name reflects the name you have chosen, for example `/Wavelab850/Papers/Adapt`.

2.2.1 Meta-Routines

There are five meta-routines underlying the figure-generating process in the current script architecture. For example, the `Adapt` subdirectory contains:

```
AdaptDemo    - starts the Demonstration, invokes Choices
AdaptInit    - sets up data structures
AdaptFig     - called from Choices
AdaptIntro   - help file, explains contents of directories
AdaptCleanup - clears all globals created by the demo
```

Rather than a lengthy blow-by-blow at this point, it is suggested that the user who wants to understand the detailed structure of these scripts pick one of the subdirectories in the current version and inspect these files.

2.2.2 Specialized Tools

There are several tools available in the `Utilities` directory to help you with writing scripts. For example, when creating a display through several `Plot` calls, it is preferable to use Wavelab functions like `LockAxes` and `UnLockAxes` rather than to use the low-level Matlab routine `hold`. See Chapter 8 below.

2.2.3 Scripting Rules

I. One script creates one complete figure, not a series of figures, and not just a subplot of a figure.

II. If several scripts need to work with the same variables – for example, if you want a variable to be created in one script and then used in later scripts – these variables must be made global (see section 4 below).

III. No `pause`'s or `print`'s in a script.

IV. As far as possible try to use the tools in the Wavelab `Utilities` directory to perform actions like setting axes.

Inspection of existing scripts will help in following these rules. If you obey these rules, then your scripts should be upwardly compatible with script-running engines making more sophisticated use of the Matlab user interface.

2.2.4 Documenting Individual Figures

Each .m file for an individual figure contains a help header which is displayed in the command window at the time the figure is generated in the plot window. This provides a kind of on-line narrative, or caption. Here is an example from `Adapt`:

```
% adfig10 -- Adapt Figure 10: Wavelet Shrinkage of object yBlocks in Haar Basis
%
% (Panel a) depicts the noisy object yBlocks, its Haar transform (Panel c),
% wavelet shrinkage reconstruction using the Haar wavelet (Panel b), and
% the Haar Transform of the reconstruction (Panel d).
%
% The viewer is supposed to notice that in the Haar domain, the
% noise is spread out among all coefficients, while the signal is
% concentrated in only a few coefficients. Hence thresholding mostly
% affects the noise without disturbing the signal.
%
```

Note here the format of the first line of the help header. Adhering to this format helps various automatic documentation features, such as the automated *Reference Manual* build.

2.3 Adding New Scripts

To add new demonstration scripts to /Wavelab850/Papers/, having the same format and effect as `AdaptDemo, AsympDemo, BlockyDemo, CorrelDemo, IdealDemo, MESDemo, MIPTDemo, RiskDemo, SCDemo, CSpinDemo, TourDemo and VdlDemo`:

1. Decide on a *name* for your demo and a *short prefix* for files implementing your demo. For example, `MyOwnDemo` and `mo`.

2. Create a new subdirectory of /Wavelab850/Papers/. For example, `MyOwn`.

3. Create the following m-files:

```
MyOwnDemo    - starts the Demonstration, invokes Choices
MyOwnInit    - sets up data structures
MyOwnFig     - called from Choices
MyOwnIntro   - help file, explains contents of directories
MyOwnCleanup - clears all globals created by the demo
```

   Suggestion: copy the corresponding files in one of the other subdirectories of `/Papers` into your new subdirectory, giving them these names; then edit these files to refer to your own new scripts.

4. Create the scripts which implement your demo: `mofig1.m`, `mofig2.m`, etc. The scripts need to follow thee rules mentioned above in sections 2.2, 2.3 and 2.4.

2.4 Modifying Existing Scripts

You might want to modify an existing script for several reasons:

- To try it out on a different dataset;
- To try it out with different parameters;
- To insert a different method in place of the existing method, using the same dataset.

Our rules for script creation should help make this possible. Some issues to keep in mind:

First, the script that generates a certain figure might be dependent on computations done in the process of generating earlier figures. Therefore, the script cannot be assumed to work correctly in stand-alone mode. If the script refers to any global variables then, at a minimum, the corresponding `Init` script has to be run before the indicated script in order to set global variables up.

Second, in order to generate a certain effect, it might therefore be necessary to change earlier scripts, not just the script formally associated with the figure you are interested in. The change might have to be in the `Init` script (to affect global variables), and might possibly have to be in other scripts as well.

Third, when a set of scripts has been well-written, it should be necessary *only* to change the `Init` script to produce most changes of the type users will want.

As a first example, to modify the examples in `AdaptDemo` to work at a sample size 512 rather than 2048, you would edit `AdaptInit` changing the line `N=2048` to `N=512`. This would then affect every later calculation in the demonstration.

As a second example, to modify the examples in `AdaptDemo` to work with Haar wavelets rather than S8, you would edit `AdaptInit` changing the line `QMF_Filter = MakeONFilter('Symmlet',8)` to `QMF_Filter = MakeONFilter('Haar')`. This would then affect every later calculation in the demonstration.

Chapter 3

Workouts

Here we describe the contents and architecture of the `/Workouts` subdirectory of Wavelab.

3.1 Workouts Philosophy

`/Workouts` is a subdirectory of /Wavelab850 that is much like `Papers` in that it contains a variety of subdirectories, each of which contains a sequence of scripts generating figures. However, `Workouts` is different in that its primary motivation is *not* to reproduce figures in our own articles. Instead, its motivation is for more informal, exploratory purposes, both

(a) To release code which produces figures that correspond to no published papers, but instead demonstrate or test various methodologies; and

(b) To release code that approximately reproduces figures in articles by *other* researchers.

We use `/Workouts` informally in our own research group to communicate new ideas, still being prototyped, to others. We also use `/Workouts` to test out the ideas of others, so that we can understand the finer points of their work and avoid the twin dangers of, on the one hand, gullibility, and, on the other hand, the "not invented here" syndrome.

We believe that by having an organized place for "experimental" work we will do better work ourselves.

3.2 Existing Workouts

In the current release, version .850, we distribute the following workouts:

```
/BestOrthoBasis      -  Workouts for Best Ortho Basis (Coifman-Wickerhauser)

/MatchingPursuit    -  Workouts for Matching Pursuits (Mallat-Zhang)


/MultiFractal     -  Workouts illustrating some aspects
                     of the Continuous Wavelet Transform

/Toons             -  "Cartoon Guide to Wavelets"
```

3.3 Workouts Architecture

It is a good idea to follow the same naming practices and file organization as in the directory /Wavelab850/Papers/.

3.3.1 Naming

In the Best Ortho Basis workout, we use filenames like `BBFig01.m`, `BBFig02.m`, etc. In the Toons workout we use names like `toon0121.m`. We try to number figures in an obvious way and to stick with names no longer than eight characters.

3.3.2 Script Contents

Each file should generate one figure, and should avoid the use of `clg`, `figure`, `print` and `pause`. This is the same set of rules that we adhere to in /Wavelab850/Papers/.

3.3.3 Meta Routines

By following the above rules it is easy to write wrapper code to print all figures or to cycle through all figures. Such wrapper code typically has suggestive names like `BBPrintAllFigs` or `BBShowAllFigs`.

Chapter 4

Books

The Books/Wavetour directory contain a collection of scripts which reproduce the figures in the book of Stephan Mallat, *Wavelet Tour of Signal Processing*. We are hoping in the future, as part of teaching and research, to develop more scripts reproducing the work of others.

Chapter 5

Datasets

The scripts we have just discussed make use of several datasets, which are made available in the directory /Wavelab850/Datasets/. In this chapter we describe the architecture of our dataset library.

5.1 Dataset Philosophy

We make available datasets through *centralized readers*. The idea is that the knowledge of how to access a dataset should be concentrated in a single place, and that the access to any dataset should be made in a stereotyped way, through a simple function call, not through explicit input and output routines.

In this way, if a dataset is available in the system because it has been used for one script, it automatically becomes available throughout the system for any other purpose one would wish, without others needing to know the format or location of the data.

If, in the future, the dataset needs to be moved to some other location in the file system, or if it needs to be stored in some other format, no scripts that use the data for wavelet demonstrations will need to change. Instead, one changes only the code implementing the access method rather than the scripts which want to use the dataset.

(The alternative is, of course, that any such changes in the future require rewriting all existing scripts!)

The same philosophy applies for datasets which are synthetic – those created by Matlab formulas. They are accessed in a stereotyped way through access to a *centralized synthesizer*. In this way, a synthetic signal designed for one use in one script automatically becomes available for other purposes.

5.2 Dataset Directory

The Contents.m file in the `Datasets` directory contains the following information. It shows that there are several tools for accessing data, 1-d datasets and 2-d datasets.

It is possible that at some time in the future, we will also have 3-d datasets (probably movies) or collections of still images.

```
%        Data Readers
%
%   BrowseImages        -   Browser for Image Datasets
%   ImageFig            -   Called by BrowseImages
%   ReadImage           -   Uniform Interface to Image Datasets
%   ReadSignal          -   Uniform Interface to Signal Datasets
%
%
%        Data Fabricators
%
%   MakeBrownian        -   Make Fractional Brownian Motions
%   MakeFractal         -   Make fractal signals
%   MakeSignal          -   Make artificial signal
%   Make2dSignal        -   Make artificial 2d signal
%   makediag            -   Make diagonal pattern (used by Make2dSignal)
%
%
%        1-d Signals
%
%   caruso.asc          -   old recording by Enrico Caruso
%   esca.asc            -   ESCA spectrum supplied by J.P. Bib\'erian
%   greasy.asc          -   recording of the word ``greasy'' from
%                             Mallat and Zhang
%   HochNMR.asc         -   NMR Spectrum supplied by Jeff Hoch
%   laser.asc           -   Time Series competition Laser series
%   RaphNMR.asc         -   NMR Spectrum supplied by Adrian Maudsley
%   seismic.asc         -   standard PROMAX test seismic signal
%   sunspots.asc        -   sunspot numbers
%   transients.asc      -   artificial signal of Mallat and Zhang
%   tweet.asc           -   recording of a bird singing
%
%
```

```
%         2-d Images
%
%   barton.raw         -   painting of seashore compressed by
%                            Jan-Olov Stromberg
%   canaletto.raw      -   painting of Venice processed by
%                            P. Perona and J. Malik
%   coifman.raw        -   photo of R.R. Coifman
%   daubechies.raw     -   photo of Ingrid Daubechies
%   fingerprint.raw    -   someone's fingerprint
%   lenna.raw          -   Lenna
%   lincoln.raw        -   Honest Abe
%   mriscan.raw        -   someone's brain
%   phone.raw          -   someone's phone
```

5.3 Dataset Format

Datasets currently occur in one of two formats:

1. *1-d Signals.* Here the file is destined to become a 1-d signal in Wavelab, i.e. an array of n numbers, where n is dyadic. It is stored as a single column of ASCII text, one number per line. The actual file is located in the directory /Wavelab850/Datasets/, with suffix .txt.

2. *2-d Images.* Here the file is destined to become a 2-d image in Wavelab, i.e. an array of n by n numbers, where n dyadic. Due to rather large size of such arrays (e.g. 512 by 512), they are stored as arrays of bytes, which can be read in raw format using the Matlab I/O routine `fread`. The actual file is located in the directory /Wavelab850/Datasets/, with suffix .raw.

5.4 Dataset Access

Datasets currently are accessed in one of two ways:

1. *1-d Signals.* The fragment `sig = ReadSignal('Name')` causes Wavelab to look in the correct directory, read the corresponding ASCII file into an array, and shape it to the correct format for a 1-d signal. For a list of currently available names, see the documentation on this function. Examples include `'RaphaelNMR'`, `'Sunspots'` and `'Caruso'`.

2. *2-d Image.* The fragment `sig = ReadImage('Name')` causes Wavelab to look in the correct directory and read the corresponding raw format file into an n by n matrix. For a list of currently available names, see the documentation on this function. Examples include `'Daubechies'`, `'Canaletto'` and `'Fingerprint'`.

A side effect of the access methods is that the corresponding documentation file of the dataset is displayed on the Matlab console as the file is read.

5.5 Dataset Documentation

Each dataset in the system has a documentation file, with suffix .doc. Here is an example of a documentation file for a 1-d signal:

```
caruso.asc -- Digital signal of Caruso singing

Access
    Enrico = ReadSignal('Caruso');

Size
    50,000 by 1

Sampling Rate
    8192 Hz

Description
    In MATLAB, the command sound(Enrico,8192) will play this sound
    back at the right pitch.

Source
    Obtained by anonymous FTP from the xwplw package
    developed by R.R. Coifman and Fazal Majid at Yale University.
    You can get this X-windows adapted waveform analysis
    package by anonymous FTP to math.yale.edu.
```

Here is an example of a documentation file for a 2-d image:

```
canaletto.raw -- Gray-scale image of Canaletto painting
```

Access

```
Canal = ReadImage('Canaletto');
```

Size

512 by 512

Gray Levels

256

Description

This image was used in an article by P. Perona and J. Malik, "Scale-Space Filtering by Anisotropic Diffusions," IEE PAMI.

Source

Obtained from John Canny and Jitendra Malik, of EECS at U.C. Berkeley.

You will notice the following fields in the documentation:

1. *Title.* A one-line header at the start of the file, giving the filename, and, after two hyphens, descriptive text.
2. *Access.* A code fragment indicating the stereotyped access method.
3. *Size.* The size of the signal or image.
4. *Gray Levels.* Applicable for Images only.
5. *Sampling Rate.* Applicable for Sounds only.
6. *Source.* Indication of the original source of the data.
7. *Description.* Additional description of the data.

5.6 Synthetic Signals

Synthetic data are currently accessed in one of two ways:

1. *1-d Signals.* The fragment `sig = MakeSignal('Name',n)` causes Wavelab to use a built-in formula to generate a synthetic signal of length n in the correct format for a 1-d signal. For a list of currently available names, see the documentation on this function. Examples include `'Bumps'`, `'Doppler'` and `'HeaviSine'`.
2. *2-d Images.* The fragment `sig = Make2dSignal('Name',n)` causes Wavelab use a built-in formula to generate a synthetic image in an n by n matrix. For a list of currently available names, see the documentation on this function. Examples include `'Circle'`, `'StickFigure'` and `'Mondrian'`.

5.7 Adding New Datasets

To add new datasets to Wavelab, do the following:

1. *Installation.* Place the dataset, in stereotyped format, in the `Datasets` directory. Modify one of the existing access functions to read in the dataset. (You can, in a pinch, place the dataset elsewhere, or keep it in a nonstandard format).

2. *Documentation.* Insert a .doc file in the `Datasets` directory to explain the dataset.

To add a new synthetic signal or image to Wavelab, simply modify the appropriate function, `MakeSignal` or `Make2dSignal`, by inserting a new case in the "compound if"; the new case tests for a new, previously unused name, and contains a formula defining the signal in that case. It is best if the formula is designed to work the same way the other formulas work – to produce an output at any given signal length or image extent.

5.8 Dataset Sources

We would like to take this opportunity to thank the sources of our datasets. We reprint here from the file `THANKS.m` in /Wavelab850/Documentation/.

```
%  Contributors of Data
%       Jean-Paul Bib\'erian, Universite de Marseille, Luminy
%       Chris Brislawn, Los Alamos National Labs
%       John Canny, UC Berkeley
%       R.R. Coifman, Yale University
%       Ingrid Daubechies, AT&T Bell Labs
%       Paul Donoho, Chevron
%       Jeffrey Hoch, Rowland Institute
%       Doug Jones, Univ. Illinois
%       Jitendra Malik, UC Berkeley
%       Stephane Mallat, Courant Institute
%       Adrian Maudsley, VA Medical Center, San Francisco
%       Chris Raphael, Stanford University
%       Jan-Olov Stromberg, University of Tromso
%       Zhifeng Zhang, Courant Institute
```

Chapter 6

Documentation

There has been extensive concern for the documentation of the code in Wavelab. We try to use all the features of Matlab as well as other features to produce a coherent, understandable system.

6.1 Help Headers

Each function in the Wavelab system has documentation contained inside the .m file with its Matlab code. This documentation can be accessed on-line by typing `help Name` where `Name` is the name of the function. For example, typing `help BestBasis` gives:

```
function [basis,value] = BestBasis(tree,D)
% BestBasis -- Coifman-Wickerhauser Best-Basis Algorithm
%  Usage
%    [btree,vtree] = BestBasis(stree,D)
%  Inputs
%    stree    stat-tree (output by CalcStatTree)
%    D        maximum depth of tree-search
%  Outputs
%    btree    basis-tree of best basis
%    vtree    value of components of best basis
%             vtree(1) holds value of best basis
%
%  Description
%    The best-basis algorithm is used to pick out the ``best''
%    basis from all the possible bases in the packet table.
```

```
%    Here ``best'' means minimizing an additive measure of
%    information, called entropy by Coifman and Wickerhauser.
%
%    Once the stattree of entropy values is created, BestBasis
%    selects the best basis using the pruning algorithm described in
%    Wickerhauser's book.
%
%  Examples
%    [n,D] = dyadlength(signal);
%    qmf = MakeONFilter('Coiflet',3);
%    wp = WPAnalysis(signal,D, qmf);
%    stree = CalcStatTree(wp,'Entropy');
%    [btree,vtree] = BestBasis(stree,D);
%
%  Algorithm
%    Yale University has filed a patent application for this algorithm.
%    Commercial Development based on this algorithm should be cleared
%    by Yale University. Contact them for licensing information.
%
%  See Also
%    WPAnalysis, CalcStatTree, CPTour, WPTour
%
%  References
%    Wickerhauser, M.V.  _Adapted_Wavelet_Analysis_.  AK Peters (1994).
%
```

This illustrates the main components of the format we have adopted: a one-line *help header*, and sections for *Usage*, *Inputs*, *Outputs*, *Side Effects*, *Description*, *Examples*, *Algorithm*, *See Also* and *References*.

1. *Header*. The first line of the help header is called the `H1` line by the Matlab folks. It is special to Matlab, and to Wavelab. When you use the `lookfor` command, Matlab examines this line for each .m file in its path to find text matching the request. When a release of Wavelab is built, these lines are compiled and sorted in alphabetical order to make files in the documentation directory. Format: a percent sign, a single blank, the name of the function, a blank followed by double hyphens and a blank, and a short description of the function. The description should contain as many helpful keywords as possible.

2. *Usage.* Here, indicate the calling prototype. Format: the output argument(s) (enclosed within square brackets if there is more than one output argument), an equals sign, the function name followed by the input argument(s) enclosed within parentheses. Optional input arguments are enclosed within square brackets.

3. *Inputs.* Here, one line per input variable, indicating the name of the variable, the formal data type and the interpretation. Also, indicate if the input is optional by enclosing it within square brackets.

4. *Outputs.* Here, one line per output variable, indicating the name of the variable, the formal data type and the interpretation.

5. *Side Effects.* Here, indicate any side effects the routine may have (graphics, sound, etc.). Omit if the function has no side effects.

6. *Description.* Here, describe what the function does in as much detail as possible.

7. *Examples.* Here, list examples of how the function is called in practice. This field is optional.

8. *Algorithm.* Here, describe the algorithm used by the function. This field is optional.

9. *See Also.* Here, mention other routines which this routine calls or which call this one, or routines with a special relationship to this function. This field is optional.

10. *References.* Here, list references from which the user may obtain further information about the function. This field is optional.

The *WaveLab Reference Manual* is built automatically from the help headers of each Wavelab function. Thus adhering to the above format will ensure the function is properly documented in the reference manual.

6.2 Documentation Directory

The directory /Wavelab850/Documentation/ contains a variety of information about Wavelab. There are a number of general files, which describe various terms and conditions and goals. The contents of any of these files may be examined by typing its name.

```
% ADDINGNEWFEATURES       -  How to Add New Features to WaveLab
% BUGREPORT               -  How to report bugs about WaveLab
% COPYING                 -  WaveLab Copying Permissions
% DATASTRUCTURES          -  Basic data structures in WaveLab
% FEEDBACK                -  Give feedback about WaveLab
% GETTINGSTARTED          -  Ideas for getting started with WaveLab
% INSTALLATION            -  Installation of WaveLab
% LIMITATIONS             -  WaveLab known limitations
% PAYMENT                 -  No Charge for WaveLab Software
% READING                 -  Sources for further reading about wavelets
% REGISTRATION            -  WaveLab Registration
% SUPPORT                 -  WaveLab Support
% THANKS                  -  Thanks to contributors
% VERSION                 -  Part of WaveLab Version v$VERSION$
% WARRANTY                -  No Warranty on WaveLab software
```

In addition, there are several files compiled automatically during the build process:

```
% WLAlphaHelpListing      -  all help files arranged by function name
% WLAlphaSynopsisListing  -  one-line synopses arranged by function name
% WLContentsListing       -  all Contents.m files
% WLFiles                 -  listing of all WaveLab files arranged by directory
% WLHelpHeaders           -  listing of all first lines of help headers
% WLHelpListing           -  all help files arranged by directory
```

To add or modify the first group of files, very little is required. Simply add new files. The second group of files, being automatically generated at build time, should not ordinarily be modified. Instead, modify the source from which they are automatically compiled.

Because of the automatic build process, it is important to maintain the integrity of certain files. These include:

- Contents files. Every directory should have a Contents.m file. When adding a new function to a directory, be sure to add it to the directory's Contents file as well.

- H1 Lines of Help documents. Every .m file should contain a help header, and the `H1` line of the help header should follow the rules specified above.

- $VERSION$ marker. Every Contents.m file has, in the `H1` line, a description of what the directory contains, as well as a version marker. The text $VERSION$ is replaced, automatically upon build, by the current version number.

6.3 Workouts Directory

Another useful component of the system documentation is the `/Workouts` directory, which contains more than a hundred scripts that exercise the software in various ways.

The user can look through the graphics generated by this documentation and, upon seeing something interesting, inspect the corresponding script to see how the graphic was created. This gives, in effect, hundreds of working examples of how Wavelab is used.

Currently, the `/Workouts` directory contains four subdirectories:

- `BestOrthoBasis` gives examples of the Coifman-Wickerhauser "Best Orthogonal Basis" paradigm in operation, on both real and synthetic data.

- `MatchingPursuit` gives examples of Matching Pursuit in operation, on both real and synthetic data.

- `MultiFractal` gives examples of using the continuous wavelet transform tools.

- `Toons` gives more than 100 examples of wavelets and time frequency analysis in operation, both in explanatory mode (e.g. showing examples of wavelets and wavelet packets) and in analysis of signals and images.

6.4 TEX Documents

The system also comes with several documents, written in TEX, which function as manuals for system-maintenance people.

The file `WaveMacros.tex` within `WaveLab Master/Documentation` contains macros that define the current version of Wavelab, filenames, file sizes, file locations, etc. This file should be modified appropriately for new releases of Wavelab. It is included by all the documents described below.

6.4.1 About WaveLab

About WaveLab helps a new user with installing and getting started with Wavelab. The corresponding pdf and postscript documents are available at: `http://www-stat.stanford.edu/~ wavelab`. The source is written in LaTeX. It is contained within the `About WaveLab` folder in `WaveLab Master/Documentation`.

Sections that may need to be changed with a new release are: section 2.2 (the filenames in the sample session), section 2.3 (the Contents.m file), section 2.5 (the file listings of the top-level Wavelab directory), section 2.6 (the startup screen), section 3.1.1 (if `Orthogonal/Contents.m` changes), section 3.1.4 (update `WLAlphaSynopsisListing` and `WLHelpHeaders.m`), section 3.3 (if the browser changes) and section 5 (if any of the files in `Documentation` are modified). Also, although the version number of Wavelab is generally not hard-coded in this document (through the use of the `WLVersion` macro), there are certain instances (e.g. in the Contents.m files) that should be manually replaced.

6.4.2 Reference

The *WaveLab Reference* manual is generated automatically by the build process script `BuildWaveDoc`; little manual intervention is required. The corresponding postscript document is available via `http://www-stat.stanford.edu/~ wavelab`. The source is written in LaTeX, but mainly generated from the Wavelab source code by `BuildWaveDoc`. It is contained within the `WaveLab Reference` folder in `WaveLab Master/Documentation`. The `README` file in this directory outlines the manual steps that must be taken after `BuildWaveDoc` is run.

Because `BuildWaveDoc` generates LaTeX source files from Wavelab.m files that are included using the LaTeX `\begin {verbatim}` and `\end{verbatim}` directives, occasionally page breaks in files that extend more than one page are not aesthetically pleasing. The `Manual Permanent Files` folder within the `WaveLab Reference` folder contain tweaked versions of the few files that fall into this category, in LaTeX form. If any of the .m files corresponding to these documentation files are changed, the corresponding files in this directory need to be changed as well.

6.4.3 Architecture

You are currently reading the *WaveLab Architecture* document. It contains system-level information about the Wavelab distribution. The corresponding postscript document is available via `http://www-stat.stanford.edu/~`

`wavelab`. The source is written in LaTeX. It is contained within the `WaveLab Architecture` folder in `WaveLab Master/Documentation`.

Sections that may need to be changed with a new release are: beginning of section 2.2 (list of papers), beginning of section 3.2 (list of workouts), section 4.2 (the Contents.m file), section 3.8 (thanks to dataset contributors), section 5.2 (if any files in `Documentation` change), section 5.3 (list of workouts), section 7 (if any files in `Utilities` change)

Chapter 7

Browsers

In the current release of Wavelab, version .850, /Wavelab850/Browsers/ contains two subdirectories, `/One-D` and `/WaveTour`. /One-D contains the browser WLBrowser which allows point-and-click access to a number of interesting features in wavelet transforms, compression and denoising. /Wavetour contains the browser WTBrowser which invokes the scripts that reproduce the figures from the book "*Wavelet Tour of Signal Processing*" by S. Mallat.

We hope to install further browsers and systematize rules for browsers in the future.

Chapter 8

Utilities

Several utilities are available in Wavelab mainly for the purpose of centralizing various programming idioms. If Wavelab is ever to be ported to Octave, for example, these allow one to modify only the utilities to the new platform and achieve the desired effect of platform-independent scripts.

The current Contents.m file for /Wavelab850/Utilities/ goes as follows:

```
% AppendTitle       -   Utility to Build Title String
% AutoImage         -   Automatic Scaling for Image Display
% CutDyad           -   Truncate signal to Dyadic length
% GrayImage         -   Image display of Gray-scaled digital images
% HitAnyKey         -   Tool for pausing in scripts
% ifprint           -   Conditional printing to postscript file
% LockAxes          -   Version-independent axis command
% MakeTiledFigures  -   Tile the screen with figures
% PadDyad           -   Zero-fill signal to Dyadic length
% RegisterPlot      -   Add legend with file name, date, flag
% ShapeAsRow        -   Reshape 1d vector as row
% ShapeLike         -   Reshape first argument like second argument
% UnlockAxes        -   Version-independent axis command
% versaplot         -   Version-independent plot routine
% WaitUntil         -   Burn up CPU cycles until sec seconds elapse
% WhiteNoise        -   Version-independent white noise generator
```

The functions of these utilities can currently be classified into the categories: *Graphics*, *Random Numbers*, *Shaping Arrays* and *Scripting.*

8.1 Graphics

There are several graphics utilities.

For image display, the basic `image` command shipped with Matlab does no scaling of its argument, nor any special choice of colormap or axes. `AutoImage(img)` provides automatic scaling of any image and a simple colormap. For cases where memory constraints are present and the image is a gray-scale digital image taking values between 0 and `ngray-1`, `GrayImage(img,ngray)` displays the image on a gray colormap without any special scaling.

For Axis control, `LockAxes` and `UnlockAxes` provide version-independent axis control.

`versaplot` bundles together axis, subplot, and other commands into a single multi-purpose, version-independent plotting command.

`MakeTiledFigures` is a tool to fill the screen with non-overlapping figures.

8.2 Random Numbers

`WhiteNoise(x)` is a version-independent Normal(0,1) random number generator. It returns an array shaped like `x` filled with normally-distributed pseudo-random numbers.

Use of this routine avoids warning messages due to the change of conventions among different versions of Matlab for generating random numbers.

8.3 Shaping Vectors

Two routines exist to coerce vectors to have the shape expected by various algorithms in Wavelab. Most of those routines were first written assuming that signals were row vectors, which is inconvenient from some points of view. So now, at the entry of most algorithms, the argument is reshaped as a row vector, and at the end of most algorithms, the result is reshaped to be in the same form as the input had originally.

`ShapeAsRow(x)` reshapes a 1-d vector (row or column) to be a row vector. `ShapeLike(x,y)` reshapes the first argument to conform to the shape of its second argument.

Two additional routnes, `CutDyad` and `PadDyad` either truncate a signal to have dyadic length (i.e. a power of two), or add zeros to the end of it to enforce this restriction. The fast algorithms in Wavelab depend on the length of a signal being a power of two.

8.4 Scripting

There are several routines to help with scripting.

`AppendTitle` tacks on extra information to a title string, such as parameters particular to a particular figure.

`RegisterPlot` allows the tools in `/Papers` to indicate, in small print at the bottom of the page, the date a plot was created and the .m file that created it.

`HitAnyKey` pauses execution, asking the user to respond with a key stroke. Optionally, it can print the current graphic before continuing.

`WaitUntil(tics)` burns up CPU cycles so that scripts don't run by too quickly.

Chapter 9

Source and Build

This chapter describes how Wavelab source is compiled into archives for distribution.

9.1 Development System

1 The source for Wavelab development has several components in different directories:

2 *C Source* in a directory named `MEX/Mex Source` inside the `WaveLab Master` folder.

3 *TeX Source* in a directory named `Documentation` inside the `WaveLab Master` folder.

4 *Shell Source* in a directory named `shell Tools` inside the `WaveLab Master` folder.

Compilation of the master source into an archive is effected using this main tool:

- *Perl Tool* that execute scripts written in the Perl programming language.

9.2 Shell Tools

About two dozen small Shell scripts will be programmed, along with master scripts, to assist in the build process. The script `InitWaveVars` is called by the high-level scripts to initialize global variables and usually needs to be modified when modifications are made to Wavelab; for example, when new directories are added. Here is an up-to-date list of the high-level files:

```
BuildWaveDoc         -   Build Reference Manual from function headers
BuildWaveDOS         -   Build DOS version of WaveLab
BuildWaveMex         -   MPW C-compile all .mex files
BuildWaveRelease     -   Master Build
FolderCompare        -   Compare Folders to look for differences
InitWaveVars         -   Initialize build variables
List_WL_HelpHeaders  -   Compile a listing of all Help Headers
ListWaveLabFiles     -   Compile a listing of all Files
RespellBuildDir      -   Rename a function throughout Built source
RespellWaveLab       -   Rename a function throughout WaveLab source
SearchWaveLab        -   Search WaveLab source for function name
```

These can be used outside of the Master build process; for example, `SearchWaveLab` may be used to see which Wavelab functions call a certain specific function.

Here is an up-to-date listing of the low-level shell Scripts used as part of the build:

```
AlphaHelpListing     -   Build alphabetic list of all function help headers
AlphaSynopsisListing -   Build alphabetic list of all function synopsis lines
ShowHelpHeader       -   Show help header without comment markers
ShowSynopsisLine     -   Extract synopsis name from function header
```

Those scripts in turn call a variety of `streamedit` scripts. These scripts copy the standard input to the standard output, modifying it appropriately:

```
DoubleFileList       -   List files in two-column format
DropLeafName         -   Strip leaf name
Print2ndCol          -   Print second of two columns
SelectLine3          -   Print third line of the file
StripCommentMarkers  -   Strip comment markers "%"
StripNonHelpHeader   -   Strip lines above header section
StripNonHelpTail     -   Strip lines below header section
Synopsis_of_PathName -   Print pathname of file
```

In addition, there are dictionaries, used by the MPW tool `canon`, to respell source.

9.3 Compiling .mex

In the interest of execution speed, several of the core .m files have been supplanted by .mex files, which express the same algorithms as the .m files, but execute more rapidly.

The directory `WaveLab Master/MEX/Mex Source` contains the following C-language files, corresponding to Wavelab .m files:

```
dct_ii.c            Meyer/dct_ii.m
dst_ii.c            Meyer/dst_ii.m
dct_iii.c           Meyer/dct_iii.m
dst_iii.c           Meyer/dst_iii.m

IWT_PO.c            Orthogonal/IWT_PO.m
FWT_PO.c            Orthogonal/FWT_PO.m
IWT2_PO.c           Orthogonal/IWT2_PO.m
FWT2_PO.c           Orthogonal/FWT2_PO.m
UpDyadHi.c          Orthogonal/UpDyadLo.m
UpDyadLo.c          Orthogonal/UpDyadLo.m
DownDyadHi.c        Orthogonal/DownDyadHi.m
DownDyadLo.c        Orthogonal/DownDyadLo.m

dct_iv.c            Packets/dct_iv.m
WPAnalysis.c        Packets/WPAnalysis.m
CPAnalysis.c        Packets/CPAnalysis.m

FastAllSeg.c        Papers/MinEntSeg/FastAllSeg.m
FastEntProfile.c    Papers/MinEntSeg/FastEntProfile.m
off_filter.c        Papers/MinEntSeg/off_filter.m

FCPSynthesis.c      Pursuit/FCPSynthesis.m
FWPSynthesis.c      Pursuit/FWPSynthesis.m

FWT_TI.c            Stationary/FWT_TI.m
IWT_TI.c            Stationary/IWT_TI.m

FWT_PBS.c           Symmetric/FWT_PBS.m
IWT_PBS.c           Symmetric/IWT_PBS.m
```

The MPW script `BuildWaveMex` compiles all these files into .mex, invoking the Matlab MPW Script `cmex`. The compiled files are placed in the directory `MEX:Mex Fat` inside the main Wavelab directory; the directory is called `Mex Fat` because `BuildWaveMex` creates "fat binaries" which contain both 68K and PowerPC code.

The C-source files use the C `#include` directive to include the following support files, listed along with the main programs which call them:

dct	dct_ii, dst_ii
dctivsub	CPAnalysis, FCPSynthesis
dht	dct_ii, dct_iii, dst_ii, dst_iii
downhi	DownDyadHi, FWT2_PO, FWT_PO, FWT_TI, WPAnalysis
downhipbs	FWT_PBS
downlo	DownDyadLo, FWT2_PO, FWT_PO, FWT_TI, WPAnalysis
downlopbs	FWT_PBS
dst	dst, idst
idct	dst_ii, dst_iii
idst	dst_iii
maiseg	FastAllSeg
matinv	maiseg
matmpy	maiseg
mirrorfilt	DownDyadHi, FWPSynthesis, FWT2_PO, FWT_PO, FWT_TI, IWT2_PO, IWT_PO, IWT_TI, UpDyadHi, WPAnalysis
mirrosymmfilt	FWT_PBS, IWT_PBS
uphi	FWPSynthesis, IWT2_PO, IWT_PO, IWT_TI, UpDyadHi
uphipbs	IWT_PBS
uplo	FWPSynthesis, IWT2_PO, IWT_PO, IWT_TI, UpDyadLo
uplopbs	IWT_PBS

In anticipation of a UNIX build, the C-language files are stored in a directory `MEXSource`. The script, `installMEX`, stored in `WaveLab Master/MEX` are also included within this archive. Since there are no fewer than seven platforms on which the Unix version of Matlab runs – Sun-4/SPARC, HP 9000/series 300, HP 9000/series 700, DECStation, Silicon Graphics, IBM RS/6000 and NeXT – the Unix user of Wavelab will run one of them to compile and install the .mex files when he installs Wavelab. Thus the Unix distribution of Wavelab has an extra top-level directory – `MEXSource` – that the Macintosh and PC distributions do not. For the latter distributions, .mex files are pre-installed.

9.4 Standard Release

The Master Build script is `BuildWaveRelease` which builds a directory `WaveLab vers`, where `vers` is replaced by the version – supplied as the argument to `BuildWaveRelease` – of Wavelab being built. A complete copy of the Wavelab package is assembled in that directory, which is located according to the `BuildDir` variable within the `InitWaveVars` script. `BuildWaveRelease` then analyzes and processes the files and directories to produce the "standard release."

The process of building a "standard" release for the Macintosh involves:

1. Appending copyright notices and date-of-modification information to all files in the library;

2. Compiling .mex files as needed;

3. Assembling lists of all filenames into `Documentation/WLFiles`;

4. Assembling sorted lists of all one-line help headers into `Documentation/WLHelpHeaders.m`;

5. Assembling sorted lists of all one-line synopses into `Documentation/WLAlphaSynopsisListing`;

6. Assembling `Documentation/WLHelpListing`, a listing of all on-line help headers, by directory and by alphabetical order within directory;

7. Assembling `Documentation/WLAlphaHelpListing`, a listing of all help headers, by alphabetical order of the function name; and

8. Assembling `Documentation/WLContentsListing`, a listing of all directory contents files, by alphabetical order of the directory name.

9.5 Compiling .ps

9.5.1 TEX Source

The `Documentation` directory within `WaveLab Master` contains one folder for each of the Wavelab documents: *About WaveLab*, *WaveLab Reference* and *WaveLab Architecture*. These folders contain the LATEX code for the documents, which are compiled into .ps and .pdf files. These .ps and .pdf files are then made available on the WWW sites.

9.5.2 WaveLab Reference

The `WaveLab Reference` directory contains a further sub-directory, `Manual`, which contains a folder and LaTeX include file corresponding to each directory of Wavelab. Each folder contains a .tex version of each .m file in the corresponding Wavelab directory. The include file, named `dirname.tex`, where `dirname` is the corresponding Wavelab directory name, assembles all the .tex files for a given directory into a chapter of *WaveLab Reference*.

The script `BuildWaveDoc` creates this `Manual` directory automatically from the Wavelab source using the Perl script `Matlab2Tex`, found in `Tools`. A few manual steps are required before this script is run; they are outlined in the `README` file within the `WaveLab Reference` directory.

Because `BuildWaveDoc` generates LaTeX source files from Wavelab .m files that are included using the LaTeX `\begin{verbatim}` and `\end{verbatim}` directives, occasionally page breaks in files that extend more than one page are not aesthetically pleasing. The `Manual Permanent Files` folder within the `WaveLab Reference` folder contain tweaked versions of the few files that fall into this category, in LaTeX form.

The file `WaveLabRef.tex` creates the *Reference Manual* by including all the chapters from the `Manual` sub-directory. It also creates another chapter, Data Structures, using `DataStructures.tex`. `WaveLabRef.tex` must be compiled twice; after the first run, the *Textures* program *MakeIndex* must be used to generate the index that is included in the second compilation.

9.6 Distribution

The actual distribution is made by running *Winzip* to create an archive named Wavelab850.zip.

Chapter 10

Distribution and Maintenance

This chapter describes how Wavelab is distributed and maintained.

10.1 Archive Directory

The `Archive` directory within `WaveLab Master` is a depository for old versions of the software and documentation.

10.2 Developer Checklists

A collection of files meant to assist developers are kept in the `Checklists` folder of `WaveLab Master`. For example, `Adding a New WaveLab Folder` describes the series of steps that should be followed to add a new top-level directory to Wavelab. We hope that a complete library of such checklists will be developed as Wavelab evolves.

10.3 WaveLab Account

An account named `WaveLab` is maintained on rgmiller. All members of the Wavelab development team share its password. The account serves several varied purposes:

1. The sub-directory `incoming` is used as a centralized location to distribute files among members of our development group.

2. The sub-directory `public_html` holds the files used to maintain our Web page.

3. The current version of Wavelab is always present on this account in the sub-directory `WaveLab`.

4. The sub-directory `VersionBuilder` holds .mex binaries for the two most popular Unix platforms: Sparc and DEC. If someone runs into problems compiling the Wavelab .mex files using `installMEX`, we put up an archive of one of these directories for FTP. We do not include these directories with the standard distribution because .mex files tend to take up a relatively large amount of disk space.

5. Feedback – questions, comments, suggestions, etc. – may be sent to the development team by e-mailing wavelab@stat.stanford.edu. Currently a `.forward` file in the `WaveLab` home directory keeps a copy of any e-mail sent in `WaveLab`'s local mailbox as well as forwarding it to all members of the team.

10.4 Web Page

The URL of the Wavelab WWW page is http://www-stat.stanford.edu/~ wavelab. The HTML files for the home page are stored in WaveLab Master/Documentation/WWW The home page is constantly changing and evolving. New versions and updates are always announced on the home page.