WaveLab Architecture

Version 0.804
August, 1999

1 Acknowledgment of Support. This work was partially supported by NSF DM92-09150 and 95-05151 (Stanford), by the NASA Astrophysics Data Program (NASA-Ames), AFOSE MURI, by DARPA BAA-98-04, by NSF KDF.
Contents

1 Introduction 3

2 Scripts 5
1 Script Philosophy .................................................. 8
2 Script Architecture .................................................. 6
   2.1 Meta-Routines .................................................. 7
   2.2 Specialized Tools ................................................. 7
   2.3 Scripting Rules .................................................. 7
   2.4 Documenting Individual Figures ................................. 8
3 Adding New Scripts .................................................. 8
4 Modifying Existing Scripts .......................................... 9

3 Workouts 11
1 Workouts Philosophy ................................................ 11
2 Existing Workouts ................................................... 11
3 Workouts Architecture .............................................. 12
   3.1 Naming .......................................................... 12
   3.2 Script Contents ................................................. 12
   3.3 Meta-Routines .................................................. 12

4 Books 13

5 Datasets 15
1 Dataset Philosophy ................................................ 15
2 Directory Contents ................................................ 15
3 Dataset Format ..................................................... 17
4 Dataset Access .................................................... 18
5 Dataset Documentation ............................................ 18
6 Synthetic Signals .................................................. 20
7 Adding New Datasets ............................................... 20
8 Dataset Sources ................................................... 21
6 Documentation ........................................ 23
  1 Help Headers ........................................ 25
  2 Documentation Directory .......................... 26
  3 Workouts Directory ............................... 26
  4 TgX Documents ....................................... 27
     4.1 About Warelab ................................. 27
     4.2 Reference ..................................... 28
     4.3 Architecture .................................. 28

7 Browsers ............................................ 29

8 Utilities ............................................. 31
  1 Graphical .......................................... 31
  2 Random Numbers ................................... 32
  3 Shaping Vectors ................................... 32
  4 Scripting ......................................... 32

9 Source and Build .................................. 35
  1 Development System .............................. 35
  2 MPW Tools ......................................... 36
  3 Compiling .tex .................................... 37
  4 Standard Release ................................ 39
  5 Compiling .ps ..................................... 40
     5.1 TgX Source .................................... 40
     5.2 Warelab Reference ............................ 40
  6 Macintosh Distribution ........................... 40
  7 Unix Distribution ................................ 41
  8 PC Distribution .................................. 41

10 Distribution and Maintenance .................... 43
  1 Archive Directory ................................ 43
  2 Developer Checklists ............................. 43
  3 Warelab Account .................................. 43
  4 Web Page ......................................... 44
1. Introduction

This document describes the architecture of WAVELELab version 0.804. 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 WAVELELab at an elementary level, see About WaveLab. This document may be accessed via WWW through the WAVELELab Home Page.

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

1. Source. There is source code, in MATLAB, C, TyX, Perl and MPW.
2. Build. The source code is assembled into a standard release. The current release is 0.804.
3. Archive. 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 Site. A web home page (which can be viewed using any web browser) and a series of postscript files which explain what WAVELELab is and how to get it. The URL is http://www-stat.stanford.edu/~wavelelab.

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

1. WAVELELab Main Directory. A subdirectory /WaveLELab of the Matlab/Toolbox directory, containing the currently released version of WAVELELab software, datasets and documentation.
2. Scripts. A subdirectory WaveLELab/Papers of WaveLELab contains scripts reproducing figures in various articles and technical reports.
3. Workouts. A subdirectory WaveLELab/Workouts of WaveLELab contains workouts that examine various aspects of WAVELELab.
4. **Documentation.** Both ASCII text in the directory WaveLab/Documentation and Postscript files available by WWW access.

5. **Browser.** In the current version, there are two browsers: WBrowser includes High-level tools which give a point-and-click interface to basic wavelet transform features in WaveLab. WBrowser enables reproducing the figures from the book of Stéphane Mallat, *A Wavelet Tour of Signal Processing.*

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

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.
2. Scripts

We briefly describe the contents and architecture of the "WaveLab/Papers" subdirectory of WaveLab.

1. Script Philosophy

The philosophy of WaveLab/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 in 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 underlie 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 hard copy of a figure that we see on the screen and save that for photcopying 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 Kamenbook's article, "Electronic Documents Give Reproducible Research Near Meaning" (http://sepwww.stanford.edu). We especially like a thought of theirs which we para-
phrase 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 and 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. Script Architecture

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

- /Adapt: figures for "Adapting to Unknown Smoothness via Wavelet Shrinkage"
- /Asympt: figures for "Wavelet Shrinkage: Asymptopia?"
- /Blocky: figures for "Smooth Wavelet Decompositions with Blocky Coefficient Normals"
- /Correl: figures for "Wavelet Threshold Estimators for Data with Correlated Noise"
- /Ideal: figures for "Ideal Spatial Adaptation via Wavelet Shrinkage"
- /MinEntSeg: figures for "On Minimum Entropy Segmentation"
- /MIPT: figures for "Non Linear Wavelet Transforms based on Median-Interpolation"
- /Minkowski: figures for "Exact Risk Analysis of Wavelet Regression"
- /ShortCourse: figures for "Nonlinear Wavelet Methods for Recovery of Signals, Densities, Spectra and Images from Incomplete and Noisy Data"
- /SpinCycle: figures for "Translation-Invariant De-Noising"
- /Tour: figures for "Wavelet Shrinkage and W.W.B.
  — a Ten-Minute Tour"
- /VillardDeLans: figures for "WaveLab and Reproducible Research"

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

1. Each article gets one subdirectory of WaveLab/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 WaveLab/Papers/Adap.

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. Specialised 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 2 below.

2.3. Scripting Rules

1. One script creates one complete figure, not a series of figures, and not just a subplot of a figure.
CHAPTER 2. SCRIPTS

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

2.4. Documenting Individual Figures

Each m-file 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.

3. Adding New Scripts

To add new demonstration scripts to WAVELAB/Papers, having the same format and effect as AdaptDemo, AsympDemo, BlockyDemo, CorrelDemo, IdealDemo, WISDemo, NISTDemo, RinkDemo, SCDemo, TourDemo and VBLDemo:
4. **Modifying Existing Scripts**

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

2. Create a new subdirectory of `WaveLab/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 script.

4. Create the scripts which implement your demo: `fig1.m`, `fig2.m`, etc. The scripts need to follow these rules mentioned above in sections 2.2, 2.3 and 2.4.

### 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.
CHAPTER 2: SCRIPTS

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 Symlets, you would edit AdaptInit changing the line QMF = MakeONFilter('Symlets',3) to QMF = MakeONFilter('Haar'). This would then affect every later calculation in the demonstration.
3. Workouts

Here we describe the contents and architecture of the /Workouts subdirectory of WAVE-LAB.

1. Workouts Philosophy

/Workouts is a subdirectory of WAVE-LAB 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 paper, 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.

2. Existing Workouts

In the current release, version 0.804, we distribute the following workouts:

/BestOrthoBasis — Workouts for Best Ortho Basis (Coifman-Vickerhauser)

/MatchingPursuit — Workouts for Matching Pursuits (Mallat-Zhang)
CHAPTER 3: WORKOUTS

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

/Toons — "Cartoon Guide to Wavelets"

3. Workouts Architecture

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

3.1. Naming

In the BestOrthoBasis 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.

3.2. Script Contents

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

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.
4. Books

The Books/WaveTour directory contains a collection of scripts which reproduce the figures in the book of Stéphane Mallat, *A 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.
5. Datasets

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

1. Dataset Philosophy

We make available datasets through control of 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 waveform 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 non-implemented synthesizer. In this way, a synthetic signal designed for one use in one script automatically becomes available for other purposes.

2. Dataset Directory

The contents of file in the Datasets directory contains the following information. It shows that there are several tools for assessing 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.
CHAPTER 5. DATASETS

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
MakeCantor - Recursively generates a Cantor distribution.
MakeFractal - Make fractal signals
MakeImage - Make artificial 2d signal (enhancement of Make2dSignal)
MakeProcess - Make locally stationary process
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
greasy.asc - ESCA spectrum supplied by J.P. Bib/orian

ekoch.asc - DNR Spectrum supplied by Jeff Koch
lady.asc - See in Books/TimeSeries/THCh06/th06fig07.m,
        st06fig08.m, st07fig03.m, st07fig07.m
laser.asc - Time Series competition Laser series
malignal.asc - Artificial signal in the article of Mallat & Zhang
RalphNHL.asc - DNR Spectrum supplied by Adrian Maudaley
seismic.asc - standard PROMAX test seismic signal
soilmpasc.asc - Vocalises
sunspots.asc - sunspot numbers
transients.asc - artificial signal of Mallat and Zhang
tweet.asc - recording of a bird singing
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 Wavelab/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 is 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 mutina fread. The actual file is located in the directory WaveLab/Datasets, with suffix .raw.

4. Dataset Access

Datasets currently are accessed in one of two ways:

1. 1-d Signals. The fragment \texttt{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 "Raphael", "Sunspots" and "Garage".

2. 2-d Image. The fragment \texttt{sig = readImage("name")} causes WaveLab to look in the correct directory and read the corresponding raw format file into an \( n \times n \) matrix. For a list of currently available names, see the documentation on this function. Examples include "baubesie", "Galaletto" 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. 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
   4192 Hz

Description
   In MATLAB, the command sound(Enrico,4192) will play this sound
   bank at the right pitch.

Source
   Obtained by anonymous FTP from the xplor package
   developed by R.R. Coifman and Fazal Majid at Yale University.
   You can get this 1-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, " IEEE 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.

6. Synthetic Signals

Synthetic data are currently accessed in one of two ways:

1. 1-d Signals. The fragment `sig = MakeSignal(\"Yame\", n)` causes WAELAB 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 \"HariLine\".

2. 2-d Images. The fragment `sig = Make2dSignal(\"Yame\", n)` causes WAELAB uses 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\".

7. Adding New Datasets

To add new datasets to WAELAB, 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 \texttt{.doc} file in the Datasets directory to explain the dataset.
8. DATASET SOURCES

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.

8. Dataset Sources

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

% Contributors of Data
%  Jean-Paul Biberian, Universite de Marseille, Luminy
%  Chris Brislawn, Los Alamos National Labs
%  John Canny, UC Berkeley
%  A.L. Coifman, Yale University
%  Ingrid Daubechies, AT&T Bell Labs
%  Paul Doherty, Chevron
%  Jeffrey Koch, Rowland Institute
%  Doug Jones, Univ. Illinois
%  Jitendra Malik, UC Berkeley
%  Stephane Mallat, Courant Institute
%  Adrian Mandaely, VA Medical Center, San Francisco
%  Chris Raphael, Stanford University
%  Jan-Olov Stromberg, University of Tromsø
%  Zhifang Zhang, Courant Institute
6. DOCUMENTATION

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

1. Help Headers

Each function in the WAVELET 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:

```matlab
function [basis, value] = BestBasis(tree, D)
%
% BestBasis — Coifman-Wickerhauser best-basis Algorithm
% Usage
% [tree, value] = BestBasis(tree, D)
% Inputs
% tree  stat-tree (output by CalcStatTree)
% D     maximum depth of tree-search
% Outputs
% tree  basis-tree of best basis
% value value of components of best basis
% value(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 stat-tree of entropy values is created, BestBasis
% selects the best basis using the pruning algorithm described in
% Wickerhauser's book.
```

28
Examples

\[
[x,D] = dyadlength(signal);
\]

qaf = MakeQFFilter('Gauss',3);

wp = WPAnalysis(signal,D,qaf);

stress = CalcStatTree(wp,'Entropy');

[bstress,vstress] = BestBasis(stress,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, GFTour, WFTour

References

Vickerhauser, M.V. Adapted Wavelet Analysis. AI Peters (1994).

This illustrates the main components of the format we have adopted: a one-line 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 E1 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, 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. Output: 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. Example: 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.

2. Documentation Directory

The directory WaveLab/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.

% ADDINGFEATURES  -  How to Add New Features to WaveLab
% BUGREPORT      -  How to report bugs about WaveLab
% CHANGES         -  History of Changes to WaveLab
% CONCEPTS        -  Concepts used in the WaveLab package
% COPYING         -  WaveLab Copying Permissions
% DATASTRUCTURES  -  Basic data structures in WaveLab
% FEEDBACK        -  Give feedback about WaveLab
% FUTURE          -  WaveLab Future Developments
% 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 $\$VERSION$
% 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.a files
% WLFiles                      - listing of all Wavelab files arranged by directory
% WLEHelpReaders               - listing of all first lines of help headers
% WLEHelpListing               - 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.a file. When adding a new function to a directory, be sure to add it to the directory’s Contents file as well.
- HI lines of help documents. Every .a file should contain a help header, and the HI line of the help header should follow the rules specified above.
- $\$VERSION$ marker. Every Contents.a file has, in the HI 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.

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 tool.
- Toms 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.

4. \textsc{\TeX} Documents

The system also comes with several documents, written in \textsc{\TeX}, which function as manuals for users and for system-maintenance people.

We use the Macintosh program \textsc{Textures} for developing our \textsc{\TeX} code. 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. An alias to this file should be installed in the Textures directory Text Inputs. The \texttt{.st} file should also be placed in this directory since it is not included with the standard distribution of Textures.

4.1. About WaveLab

About WaveLab helps a new user with installing and getting started with WaveLab.

The corresponding postscript document is available via http://www.stat.stanford.edu/~wavelet/ftp/AboutWaveLab.ps.Z. The source is written in \LaTeX\, using the \texttt{.st} style format. 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, section 2.4 (the Contents.m file), section 2.6 (the file listings of the top-level WaveLab directory), section 2.7 (the startup screen), section 3.1.1 (if Orthogonal/Contents.m changes), section 3.1.4 (update MplusSynopsisListing and MHelpReaders.m), section 3.3 (if the browser changes) and section 3.5 (if any of the files in Documentation are modified).

Also, although the version number of WaveLab is generally not hard-coded in this
CHAPTER 6  DOCUMENTATION

document (through the use of the $\text{VERSION}$.macro), there are certain instances (e.g.
in the FTP session and Contents .am files) that should be manually replaced.

4.2. Reference

The Wavelab Reference manual is generated automatically by the build process script
BuildWavelab; little manual intervention is required. The corresponding postscript
document is available via
http://www-stat.stanford.edu/~wavelab/ftp/WavelabRef.ps.Z. The source is
written in \LaTeX, but mainly generated from the WAVELAB source code by
BuildWavelab. 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 BuildWavelab is run.

Because BuildWavelab generates \LaTeX source files from WAVELAB .am files that
are included using the \LaTeX \texttt{\begin{verbatim}} and \texttt{\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 .am file corresponding to these documentation files are changed, the corre-
spanding files in this directory need to be changed as well.

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/ftp/WavelabArch.ps.Z.
The source is written in \LaTeX, using the \texttt{\LaTeX} style format. 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 sec-
tion 2.2 (list of papers), beginning of section 3.2 (list of workouts), section 6.2 (the
Contents .am file), section 3.8 (thanks to dataset contributors), section 6.2 (if any files
in Documentation change), section 6.3 (list of workouts), section 8 (if any files in
Utilities change)
7. Browsers

In the current release of WAVELAB, version 0.804, WAVELAB/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 de-noising. /WaveTour contains the browser WTBrowser which invokes the scripts that reproduce the figures from the book "A Wavelet tour of signal processing" by S. Mallat. At the moment there is no documentation for this package.

We hope to install further browsers and systematize rules for browsers in the future.
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 WavLab/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
% LockAxes - Version-independent axis command
% MakeTiledFigures - Tile the screen with figures
% PadDyad - Zero-fill signal to dyadic length
% RegisterPlot - Add legend with file name, dates, flag
% ShapeIndex - Reshape id vector as row
% ShapeLike - Reshape first argument like second argument
% UnlockAxes - Version-independent axis command
% WaitUntil - Burn up CPU cycles until sec seconds elapse
% WhiteNoise - Version-independent white noise generator
% ifprint - Conditional printing to postscript file
% log2Lin - Transform log-scale image to linear-scale image
% pic256 - Show image of 256 gray scale
% mask - Circular right shift of i-d signal
% versmplot - Version-independent plot routine

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

1. Graphics

There are several graphics utilities.
CHAPTER 8: 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, `LockAxis` and `UnlockAxis` 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.

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.

3. Shaping Vectors

Two routines exist to coerce vectors to have the shape expected by various algorithms in WAVELAB. Most of these 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.

`ShapeInto(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 routines, `CutoffSyd` and `PadBySyd` 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.

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 *Maps* to indicate, in small print at the bottom of the page, the data a plot was created and the .jfile 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.
9. Source and Build

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

1. Development System

The source for WAVELAB development has several components in different directories on a Macintosh computer:

1. Matlab Source in a directory named WAVELAB inside the WAVELAB Master folder.
2. C Source in a directory named WAVELAB Source inside the WAVELAB Master folder.
3. Tex Source in a directory named Documentation inside the WAVELAB Master folder.
4. MPW Source in a directory named MPW Tools inside the WAVELAB Master folder.

Compilation of the master source into an archive is effected using four main tools:

1. MPW. The Macintosh Programmer's Workshop (MPW) is a UNIX-like environment in which one can write scripts to compile, copy, move, delete and rename files.
2. Perl Tool is a Macintosh application that allows MPW to execute scripts written in the Perl programming language.
3. StuffIt Deflate is a Macintosh application that allows one to build self-extracting archives that decompress and install themselves on a Mac with only a mouse click. It can also binhex those archives so they look like standard UNIX files and can be made available on a UNIX file server for access over Internet. StuffIt Deflate also allows one to directly create Unix tar archives in compressed .tar.Z format.
4. PC Exchange is a Mac application that allows one to copy Mac files to a PC floppy disk.
6. PKZIP is a DOS application that allows one to build a compressed archive on a PC.

2. MPW Tools

About two dozen small MPW scripts have been programmed, along with master scripts, to assist in the build process. The script InitVars 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
- BuildWaveDS - Build DOS version of WaveLab
- BuildWaveMac - MPW C-compile all .mac files
- BuildWaveLab - Master Build
- FolderCompare - Compare Folders to look for differences
- InitVars - Initialize build variables
- List_WL_HelpHeaders - Compile a listing of all Help Headers
- ListWaveLabFiles - Compile a listing of all files
- ReplaceAllBuildDir - Rename a function throughout built source
- ReplaceAllWaveLab - Rename a function throughout WaveLab source
- SearchWaveLab - Search WaveLab source for function name

There 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 MPW 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
- ShowSynopsisLines - Extract synopsis name from function header

These scripts in turn call a variety of streamedit scripts. Streamedit is an MPW tool with features similar to the UNIX command ed. 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 "/*"
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 WavLab Master:KEL:Mex:Source contains the following C-language 
files, corresponding to WavLab .m files:

FMPT.c               Median/FMPT.m
IKEPT.c               Median/IKEPT.m
QuadMedRef.c          Median/QuadMedRef.m

dct_ii.c               Mayer/dct_ii.m
dst_ii.c               Mayer/dst_ii.m
dct_iii.c              Mayer/dct_iii.m
dst_iii.c              Mayer/dst_iii.m

IVT_P0.c               Orthogonal/IVT_P0.m
IVT_P0.c               Orthogonal/IVT_P0.m
IVT2_P0.c              Orthogonal/IVT2_P0.m
IVT2_P0.c              Orthogonal/IVT2_P0.m
UpdyadHi.c             Orthogonal/UpdyadHi.m
UpdyadLo.c             Orthogonal/UpdyadLo.m
DowndyadHi.c           Orthogonal/DowndyadHi.m
DowndyadLo.c           Orthogonal/DowndyadLo.m

dct_iv.c               Packets/One-D/dct_iv.m
WPAnalysis.c           Packets/One-D/WPAnalysis.m
CPAnalysis.c           Packets/One-D/CPAnalysis.m

Fast11Seg.c            Papers/WinEntSeg/Fast11Seg.m
The MPW script BuildWaveMax compiles all these files into .max, invoking the MATLAB MPW script cax. The compiled files are placed in the directory WEL:MaxFat inside the main Wavelab directory; the directory is called MaxFat because BuildWaveMax 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_i1slb
 dcta, dcta

dct
 dcta, dcta

downhi
 Downhadi, FVT2_PO, FVT_PO, FVT_TI, WPA

downiphs
 FVT_PHB

downlo
 Downhadi, FVT2_PO, FVT_PO, FVT_TI, WPA

downlohs
 FVT_PHB
dသ
 dat, idat
idc
dat, dat
idat
dat_i
maissag
Fast11Seg
matinv
maissag
matmap
maissag

mirrorfilt
 Downhadi, FVT2_PO, FVT_PO, FVT_TI, INT2_PO, INT_PO, INT_TI, UpDownHi, WPA

mirrorsymfilt
 FVT_PHB, INT_PHB

uphi
 FVT2_PO, INT2_PO, INT2_PO, INT_TI, UpDownHi

upips
 INT_PHB

uiplo
 FVT2_PO, INT2_PO, INT_TI, UpDownLo

uplohs
 INT_PHB
4. **STANDARD RELEASE**

In anticipation of a UNIX build, the C-language files are stored in a directory **MElSource**. Two scripts, `installMEl` and `installMEl.old`, stored in WaveLab Master/MEl 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 (`installMEl` if he is using an older version of MATLAB) to compile and install the `.mex` files when he installs WAVELAB. Thus the Unix distribution of WAVELAB has an extra top-level directory — **MElSource** — that the Macintosh and PC distributions do not. For the latter distributions, `.mex` files are pre-installed.

4. **Standard Release**

The MPW Master Build script is **BuildWaveLab.m** which builds a directory **WaveLab** where **WaveLab** is replaced by the version — supplied as the argument to **BuildWaveLab.m** — 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 **InitWaveLab** script. **BuildWaveLab.m** then analyses 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 files into **Documentation/whelFiles**;
4. Assembling sorted lists of all one-line help headers into **Documentation/whelHelpHeaders**;
5. Assembling sorted lists of all one-line synopses into **Documentation/whelAlphaSynopsisListing**;
6. Assembling **Documentation/whel1Listing**, a listing of all on-line help headers, by directory and by alphabetical order within directory;
7. Assembling **Documentation/whelAlpha1Listing**, a listing of all help headers, by alphabetical order of the function name, and;
8. Assembling **Documentation/whelContentsListing**, a listing of all directory contents files, by alphabetical order of the directory name.
5. Compiling .ps

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 \TeX\ files for the documents, which are compiled into .ps files using the Macintosh program Publisher. These .ps files are then made available on the WWW site.

5.2. WaveLab Reference

The WaveLab Reference directory contains a further subdirectory, Manual, which contains a folder and \TeX\ include file corresponding to each directory of WaveLab. Each folder contains a .tex version of each .tex 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 MPW script BuildWaveDoc creates this Manual directory automatically from the WaveLab source using the Perl script Matlab2Tex, found in MPW 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 \TeX\ source files from WaveLab .ml files that are included using the \TeX\ \{verbatim\} and \{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 \TeX\ form.

The file WaveLabRef.tex creates the Reference Manual by including all the chapters from the Manual sub-directory. It also creates two other chapters, Data Structures, using DataStructures.tex, and Notes for the DOS Version, using DOSNotes.tex. WaveLabRef.tex must be compiled twice; after the first run, the Publisher program MakeIndex must be used to generate the index that is included in the second compilation.

6. Macintosh Distribution

The actual Macintosh distribution is made by running StuffIt to create an archive named WaveLab0804.msi. This is a binhexed self-extracting archive that may be placed on the Internet as a UNIX file, downloaded by users, and then converted by StuffIt to
7. UNIX DISTRIBUTION

A file WavLab0304.sea which is a Mac Application. When one double clicks on the corresponding icon, it uncompressed and installs itself.

The file WavLab0304.sea.tar.gz is transferred to www-stat.stanford.edu using some file transfer utility (e.g. ZMODEM) where it is made available for WWW access by placing it in the directory /home/ftp/pub/wavlab. Public file permissions need to be set for this file, e.g. chmod 774 WavLab0304.sea.tar.gz.

7. Unix Distribution

The actual Unix distribution is made by creating a compressed archive of the standard release with the addition of the KEYSources directory described in section 3 of this chapter. Using StuffIt we create a compressed tar archive named WavLab0304.tar.Z.

The file WavLab0304.tar.Z is transferred to ftp-stat.stanford.edu using some file transfer utility where it is made available for WWW access by placing it in the directory /home/ftp/pub/wavlab. Public file permissions need to be set for this file, e.g. chmod 774 WavLab0304.tar.Z.

8. PC Distribution

The DOS files are copied to a DOS floppy using Macintosh File Exchange and installed on a PC, where a compressed archive can be built using pslip. Some of the files associated with building the DOS release are listed below.

AlphaDOSIndex - Build alphabetical list of all DOS files
AlphaLeafNames - Build alphabetical list of all DOS leaf names
BuildCharNames - Map all filenames to their 8-character equivalents
BuildDOSData - Master build script for DOS version
CheckDOSRemap - Perl script to ensure no filename hits during remap
DOSData - Generated during build; contains data of DOS build
DOSIndexList - Generated from DOSOverides to move filenames
DOSOverides - Generated from DOSOverides to move filenames
DOSOverides - Supplied by operator to override certain remappings
DoubleDict - Sed script to generate double-column dictionary
List_DOS_WL_Files - List all files in the DOS version
LowerCaseDict - Generated during build; lists lower-case filenames
LowerCaseFiles - Generated during build; lists lower-case filenames
MakeLowerChar - Remaps a filename to its 8-character equivalent
Override2Kern - Generates list of filename remappings
RespellBuildDir - Apply remap to each file in DOS build
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shorthellist</td>
<td>Generate list of short filenames</td>
</tr>
<tr>
<td>StripMSuffix</td>
<td>Remove <code>.m</code> extension</td>
</tr>
<tr>
<td>StripSharpless</td>
<td>Remove percent-sign (<code>%</code>) headers</td>
</tr>
</tbody>
</table>
10. Distribution and Maintenance

This chapter describes how WAVELAB is distributed and maintained.

1. Archive Directory

The archive directory within WaveLab Master is a repository for old versions of the software and documentation. In some cases, tar.gz archives are used to organize files and save space.

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.

3. WaveLab Account

An account named waveLab is maintained on regmill. 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 .aux binaries for the two most popular Unix platforms: Sparc and DEC. If someone runs into problems compiling the WAVELAB .aux files using installMEX, we put up an archive of one of these directories for WWW access. We do not include these directories with the standard
distribution because .aux files tend to take up a relatively large amount of disk space.

6. Feedback — questions, comments, suggestions, etc. — may be sent to the development team by e-mailing mail@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.

4. Web Page

The URL of the WAVE Lab WWW page is http://www-stat.stanford.edu/~wavelab. The HTML files for the home page are stored in Wavelab:Documentation:WWW on the Macintosh. We use the program PageMill to edit our Web documents. When a HTML file is ready for publication, it is transferred from the Mac to the public_html directory of the Wavelab account on rmallier.statanford.edu.

The home page is constantly changing and evolving. New versions and updates are always announced on the home page.