WaveLab and Reproducible Research

Jonathan B. Buckheit and David L. Donoho

Stanford University, Stanford CA 94305, USA

Abstract

WaveLab is a library of Matlab routines for wavelet analysis, wavelet-packet analysis, cosine-packet analysis and matching pursuit. The library is available free of charge over the Internet. Versions are provided for Macintosh, UNIX and Windows machines.

WaveLab makes available, in one package, all the code to reproduce all the figures in our published wavelet articles. The interested reader can inspect the source code to see exactly what algorithms were used, how parameters were set in producing our figures, and can then modify the source to produce variations on our results. WaveLab has been developed, in part, because of exhortations by Jon Claerbout of Stanford that computational scientists should engage in "really reproducible" research.

1 WaveLab – Reproducible Research via the Internet

A remarkable aspect of "the wavelet community" is the wide span of intellectual activities that it makes contact with. At one extreme, wavelets are interesting to mathematicians who are interested in functional spaces, their properties and decompositions – while at the other extreme wavelets are interesting in certain commercial software development efforts, where engineers craft computer programs applying wavelets to specific problems in high-technology.

Work at Stanford on statistical applications of wavelets has, over the last five years, reflected a great deal of this range of wavelet activity. Dave Donoho and Iain Johnstone have written a number of theoretical papers; but also, a team involving Donoho, Johnstone, students Jon Buckheit, Shaobing Chen and Eric Kolaczyk, as well as Jeff Scargle of NASA-Ames, have developed a collection of software tools known as WaveLab.

The WaveLab package contains a wide range of tools for wavelet and related time-frequency transforms. As this was written, version .700 was almost complete, consisting of over 700 files – programs, data, documentation and scripts. At the moment, the package requires over two megabytes of storage in compressed form. The package is available free of charge over the Internet, using standard interfaces like FTP and WWW. The stated goal of the package, and the stated reason for its distribution, is to allow others to reproduce the figures and tables in the articles published by our group.

It of course not unusual that researchers who were initially drawn into wavelets because of interests at the mathematical end of the scale would end up also doing some software development, for example to generate figures for their articles. It is perhaps less to be expected that they would be involved in actual packaging and distribution of software to others. Release of software underlying scientific publication is the exception rather than the rule. Each year, many figures are published in the scientific literature which were generated by computer; relatively few of those figures lead to the distribution of the software which generated them. Moreover, even those who make software available would rarely take the trouble to create a relatively comprehensive computing environment around it merely to ease the task of others wishing to reproduce results. But that is exactly what we have done – constructed a whole computing environment aimed at allowing others to easily reproduce the figures in our articles.

Our main goal in this paper is to call attention to a principle we have tried to follow:

> *When we publish articles containing figures which were generated by computer, we also publish the complete software environment which generates the figures.*

We shall describe the reasons why we try to follow this principle, the software environment WaveLab we have built in trying to follow it, some of the capabilities of this environment, some of the lessons we have learned by trying to follow it, and also the implications it has for the conduct of science in general.

2 The Scandal

To avoid sounding like we are pointing the finger at anyone else, we will mention a few problems we have encountered in our own research.

- *Burning the Midnight Oil.* Once, writing an article with approximately 30 figures, we had to tweek various algorithms and display options to display clearly the effects we were looking for. As a result, after an 18-hour day we had accumulated a stack of a few hundred sheets of paper, all of which purported to be versions of the figures for the article. We gave up well after midnight.

  Returning to work eight hours later, we had a question: which were the "final" versions, the ones which should go in the article? The easy answer would have been "the nicest looking ones," but that wouldn't always be right. In fact, the correct answer would have been "the ones generated using the settings and algorithms exactly described in the paper." Those were not always the best-looking ones.

  In any event, we had a major problem sorting through the hundreds of sheets of paper to find the ones that really belonged in the article. It is possible, though not likely, that we fooled ourselves, and put the wrong version of some figures in the final copy.

- *The Stolen Briefcase.* Once, several years ago, at a conference, one of us had a briefcase stolen. The briefcase contained originals of figures which had been developed while an employee of a large commercial seismic exploration outfit. The data and data processing equipment which had generated the figures were proprietary.

There was no reasonable prospect of finding the time or opportunity to return to the seismic firm to reconstruct the figures from scratch. A manuscript had already been written. The figures were so convincing and so pivotal (the subject of the article was a specialized kind of image processing) that without them, the manuscript made no sense. The manuscript had to be abandoned.

- *Who's on First?* A Graduate Student comes into a Professor's office and says "that idea you told me to try – it doesn't work!" The Professor suggests to him some variation on the idea, and the Student returns a day later with the same response. Unfortunately, the Student's descriptions of the problems he is facing don't give the Professor much insight into what's going on, and this keeps recurring day after day. After a long period of discussion, it becomes apparent that the issue really is as follows: the student actually needs to provide the Professor with detailed information so they could explore four branches on a decision tree:

  – Is the idea itself incorrect?

  – Or is the idea okay, while the student's implementation of the idea is incorrect?

  – Or is the implementation okay, while the student's invocation of the algorithm used incorrect parameters?

  – Or is the invocation okay while the student's display of the results actually focuses on the wrong aspect of the problem?

  Mere oral communications are completely inadequate to do any of this. The student has built (whether he knows that he is doing this or not) a computing environment, and unless the Professor can enter and use the Student's environment *in situ* as he had built it, the two couldn't possibly get a fix on the answers. But since the Student had not anticipated this issue, it was very hard for him to explain the environment (algorithms, datasets, etc.) which he had constructed, and hard for the Professor to get into it.

- *A Year is a Long Time in this Business.* Once, about a year after one of us had done some work and written an article (and basically forgot the details of the work he had done), he had the occasion to apply the methods of the article on a newly-arrived dataset. When he went back to the old software library to try and do it, he couldn't remember how the software worked – invocation sequences, data structures, etc. In the end, he abandoned the project, saying he just didn't have time to get into it anymore.

- *A la Récherche des Parametres Perdues.* Once, one of us read a paper on wavelets that was very interesting. He had a vague idea of what the author of the paper was doing and wanted to try it out. Unfortunately, from the paper itself he couldn't figure out what filter coefficients, thresholds and similar tuning parameters were being used. He spoke the author of the paper, who replied, "Well, actually, the reason we didn't give many details in the paper was that we forgot which parameters gave the nice picture you see in the published article; when we tried to reconstruct that figure using parameters that we thought had been used, we only got ugly looking results. So we knew there had been some parameter settings which worked well, and perhaps one day we would stumble on them again; but we thought it best to leave things vague." (Note: this story is actually a composite of two separate true incidents).

Surely anyone reading the above recognizes the sorts of situation that we are talking about and has experienced them first-hand. It is not too much to say that these experiences are utterly common; they are the dominant experiences of researchers in those fields which rely on computational experiments. Researchers in those fields can't reproduce their own work; students in those fields can't explain to their advisers the difficulties they are having, and researchers in those fields can't reproduce the work of others.

To people who have *only* worked in such fields, this probably seems to be just the way things are, so much so that this state of affairs is unremarkable.

In the field of wavelets, where we see a mixture of researchers from several disciplines, it is easier to take a broader perspective and to see the situation as it really is: a scandal.

For a field to qualify as a science, it is important first and foremost that published work be reproducible by others. In wavelets, mathematical results are reproducible by others, who must only read and understand mathematical proofs to reproduce their validity. However, computational results are not reproducible – this is the state of affairs mentioned above. So we have a mixture of the scientific and the a-scientific, clearly visible to all.

3 The Solution

We are, of course, not the first to call attention to this type of situation. Jon Claerbout, a distinguished exploration geophysicist at Stanford, has in recent years championed the concept of *really reproducible research* in the "Computational Sciences." He has also pointed out that we have reached a point where solutions are available – it is now possible to publish computational research that is really reproducible by others. The solutions involve a convergence on several fronts.

3.1 Claerbout and Reproducibility

In order for reproducibility to become widespread, individual researchers must be convinced of the importance of reproducibility of their work, and to plan their work accordingly. For this, the ideas of Claerbout may be convincing.

Claerbout's ideas arose in exploration seismology where the goal is an image of the subsurface, and research aims to produce better images. However, Claerbout has pointed out that the research deliverable is not an image itself, but instead the software environment that, applied in the right way, produces the image, and which, hopefully, could be applied to other datasets to produce equally nice images. The scientific findings may turn out to be a *knowledge of parameter settings* for this complex software environment that seem to lead to good results on real datasets.

With this as background, reproducibility of experiments in seismic exploration requires having the complete software environment available in other laboratories and the full source code available for inspection, modification, and application under varied parameter settings.

Actually these comments apply to all fields in which mathematical and computer science heuristics may *suggest* algorithms to be tried on scientific signal processing and imaging problems, but mathematical analysis alone is *not able* to predict fully the behavior and suitability of algorithms for specific datasets. Therefore experiments are necessary and such experiments ought, in principle, to be reproducible, just as experiments in other fields of science.

In all such fields, we can distill Claerbout's insight into a slogan:

> *An article about computational science in a scientific publication is* **not** *the scholarship itself, it is merely* **advertising** *of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.*

In order to work in accordance with this slogan, Claerbout and his colleagues have developed a discipline for building their own software, so that from the start, they expect it to be made available to others as part of the publication of their work. Specifically, they publish CD-ROMs (available from Stanford University Press) which contain the text of their books along with a special viewer that makes those books *interactive documents*, where as one reads the document, each figure is accompanied by the possibility of pop-up windows which allow one to interact with the code that generated the figure, to "burn" the illustration (i.e. erase the postscript file supplied with the distribution), and to rebuild the figure from scratch performing all the signal and image processing in the software environment that the CD-ROM makes available on one's own machine. By following the discipline of planning to publish in this way from the beginning, they maintain all their work in a form which is easy to make available to others at any point in time.

While Claerbout's example is instructive, we don't think that the specifics of his approach will be widely adopted. Claerbout's project began in 1990 and much has changed in the intervening five years.

3.2 Internet

The exponential growth of Internet and of user-friendly access to information via the world-wide-web makes it possible to share information with others very efficiently. For example, we now have a wavelet digest access through WWW browsers; the Wavelet Digest has links to articles and software being made available worldwide, so that now researchers can make articles and supporting information available to others around the world twenty-four hours a day. Moreover this availability is not just theoretical; it is convenient and rapid. One can now easily locate and download megabytes of information over standard telephone lines in minutes.

3.3 Freeware

Supporting the development of Internet has been the appearance of a culture of "giving software away." A highly visible advocate of this culture is Richard Stallman, who has developed the GNU software library and the concepts of "Freeware" and "Copy-Left." These concepts have helped organize a great deal of Internet activity around the sharing and development of large bodies of software. (Incidentally, Freeware is not necessarily free of cost – it can be sold; the point of Freeware is that it can be freely redistributed, under certain conditions. Moreover, one can make a living developing Freeware, as Stallman has shown.)

3.4 Quantitative Programming Environments

The last five years have also seen an explosive growth in the ubiquity of quantitative programming environments – systems like MATLAB, MATHEMATICA and S-PLUS which manipulate and display data on personal computers and scientific workstations using high-level commands that approximate the way that engineers, statisticians and mathematicians speak about tasks they are doing (i.e. "take the fourier transform of that signal" has a fairly one-one translation into any and all of the languages). QPE's have been around for a long time, and it has long been recognized that they allow very concise expression of scientific data processing methods and that they allow very rapid prototyping of scientific algorithms. However, it was traditionally thought that they impose performance penalties which make them unsuitable for serious work – which would therefore have to be done, slowly and painfully, in low-level languages. As computer workstations have grown in power, the performance penalties from using high level languages have become less onerous. Using a QPE on a state-of-the-art workstation gives about the same level of performance as using labor-intensive low-level programming on the somewhat slower machine that was state-of-the-art twelve months ago.

From the standpoint of our major theme, reproducibility, QPE's are revolutionary because they work the same way on many different PC's and workstations, so code developed for QPE's is much more useful to others than code custom-developed in low-level languages for a single platform.

3.5 Implications

Let's discuss the implications of these developments for the wavelet community.

First, it is our perception that as we approach specific applications using wavelets and time-frequency analysis, we are becoming a computational science like seismic imaging. Performance has everything to do with specifics: exactly what was done (which wavelets, which coders, which detectors, which corpus of data) with exactly what parameters. In this setting, publishing figures or results without the complete software environment could be compared to a mathematician publishing an announcement of a mathematical theorem without giving the proof. Waveleticians ought to publish their complete computational environments.

Second, thanks to the Internet, it is easy to publish information. One simply makes it available in an automatic fashion which requires no intervention on the part of the publisher, and very little effort on the part of the user.

Third, because of QPE's, it is possible to publish ambitious computational environments in compact form. A few megabytes of code written in the language of a QPE is equivalent to hundreds of megabytes of code, makefiles and multiple-platform binaries in a low-level language. Most computational environments being developed in wavelets and related fields could be published over the Internet if implemented in QPE's.

Fourth, one can never require researchers to publish their code. But examples like the GNU project show that very bright and able people are naturally drawn to share their intellectual works with others, and so some researchers will do it. We believe that those who do will do better science than those who don't.

3.6 WaveLab

The system we have built, WaveLab, is an example of the trends we have just identified. It is a modest step in the direction of reproducible research.

It is available over the Internet via either a point-and-click Web browser or via FTP protocol. The URL's are:

`http://playfair.stanford.edu/~wavelab`

- and -

`ftp://playfair.stanford.edu/pub/wavelab`

Versions are available for Unix workstations, for Macintosh (680X0 and Power Mac) and for PC (Windows). They are compressed archives which install automatically on the user's machine using standard tools like `compress` and `tar` (Unix) `stuffit` (Mac) and `pkzip` (Windows). The complete package – code in Matlab, data, and documentation – is over two megabytes in compressed form, but takes only minutes to access and install, even over telephone lines with 14.4 modems.

The package reproduces the figures in our published articles. Our system contains a subdirectory, `WaveLab/Papers`, which contains within it one subdirectory for each article we publish. Each directory contains the code which reproduces the figures as published in hardcopy form as technical reports at Stanford University and in forthcoming journal articles. Other researchers can therefore obtain the Matlab code which generated these figures, and can reproduce the calculations that underly the figures. They can, if they wish, modify our calculations by editing the underlying Matlab code. They can use the algorithms on other datasets, or they can try their own favorite methods on the same datasets.

In accordance with Claerbout's doctrine, 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 an 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.

Claerbout, in one of his articles, claims that the approach he follows takes little effort beyond the learning to file away one's work systematically. We think his assertion grossly understates the philosophical and practical effort required to follow this path of research reproducibility. To work in accordance with this goal, we must decide on a discipline of how we will structure our computational experiments. We must also then proselytize among others in our group to get them to adopt this discipline.

On the other hand, the very effort involved may be seen to be an advantage. It practically ensures that we will reduce problems of sloppiness and self-delusion, that we will communicate more directly and frequently with our students, that our students will be raised up with better intellectual habits and that our students will do better work. The group survival value is high.

4 The WaveLab Distribution

We now describe some of the contents of WaveLab, with an eye to communicating just how much effort and attention is called for in the effort to maintain reproducibility.

4.1 Installation

WaveLab, when installed, adds the following directory structure to the user's Toolbox path:

```
WaveLab
WaveLab/Browsers
WaveLab/Browsers/One-D
WaveLab/Datasets
WaveLab/DeNoising
WaveLab/Documentation
WaveLab/Interpolating
WaveLab/Meyer
WaveLab/Orthogonal
WaveLab/Packets
WaveLab/Packets2
WaveLab/Pursuit
WaveLab/Stationary
WaveLab/Symmetric
WaveLab/Utilities
WaveLab/Papers
WaveLab/Papers/Adapt
WaveLab/Papers/Asymp
WaveLab/Papers/Blocky
WaveLab/Papers/Ideal
WaveLab/Papers/MinEntSeg
WaveLab/Papers/ShortCourse
WaveLab/Papers/Tour
WaveLab/Papers/VillardDeLans
WaveLab/Workouts
WaveLab/Workouts/BestOrthoBasis
WaveLab/Workouts/MatchingPursuit
WaveLab/Workouts/Toons
```

We now describe some of the key elements of these directories. Buried in these directories are more than 700 files of various types; due to limited space, we cannot cover them all here. (In section 5 below, we give a few examples of WaveLab in action.)

To give an idea of the contents of individual directories, we extract from the `Contents.m` file for the directory `WaveLab/Packets` of 1-d cosine packet and wavelet packet tools:

```
% Packets:Contents v.700 -- One-d Wavelet- and Cosine- Packet Tools
%
%   The routines in this directory perform wavelet packet analysis and
% cosine packet analysis of 1-d signals. The main tools for all-purpose
% use are WPTour and CPTour.  The other tools are all invoked by these.
%
%
%          Wavelet Packet Analysis/Synthesis
%
% WPAnalysis           -   Dyadic table of all Wavelet Packet coefficients
% WPSynthesis          -   Synthesize signal from Wavelet Packet coefficients
% WPTour               -   Wavelet Packet decomposition & Best Basis analysis
%
%
%          Cosine Packet Analysis/Synthesis
%
% CPAnalysis           -   Dyadic table of all Cosine Packet Coefficients
% CPSynthesis          -   Synthesize signal from Cosine Packet coefficients
% CPTour               -   Cosine Packet decomposition & Best Basis analysis
%
%
%          Search for Best Basis
%
% BestBasis            -   Coifman-Wickerhauser Best-Basis Algorithm
% CalcStatTree         -   Build tree with entropy numbers
% PlotBasisTree        -   Display basis tree with decorated branch lengths
%
%
%          Packet Table Displays
%
% PlotPacketTable      -   Display entries in wavelet, cosine packet tables
%
%
%          Phase Plane Displays
%
% ImagePhasePlane      -   Time-Frequency Display using ``image'' graphics
% PlotPhasePlane       -   Time-Frequency Display using ``plot'' graphics
%
%
%          Comparison of different bases
%
% CompareStdBases      -   calculate entropy of some standard bases
```

```
% ImageGaborPhase      -   Time-Frequency image with congruent rectangles
% ImagePhaseVarious    -   compare 4 phase plane displays
% PlotCoeffComparison  -   compare coefficients in various bases
% PlotCompressNumbers  -   plot compression numbers for a signal
% PlotGaborPhase       -   Time-Frequency plot with congruent rectangles
% PlotPhaseVarious     -   compare 4 phase plane displays
% PlotWavePhase        -   Time-Frequency plot with wavelet tiling
%
%
%         Working in a single Basis
%
% FPT_WP               -   Fast transform into specific Wavelet Packet basis
% IPT_WP               -   Fast reconstruction from specific Wavelet Packet basis
% FPT_CP               -   Fast transform from specific Cosine Packet basis
%
%
%         Synthesis of Individual Basis Elements
%
% MakeCosinePacket     -   Make cosine packet
% MakeWaveletPacket    -   Make periodized orthogonal wavelet packet
%
%
%         Cosine Packet Infrastructure
%
% fold                 -   folding projection with (+,-) polarity
% edgefold             -   folding projection with (+,-) polarity at edges
% unfold               -   undo folding projection with (+,-) polarity
% edgeunfold           -   undo folding projection with (+,-) polarity at edges
% MakeONBell           -   Make Bell for Orthonormal Local Cosine Analysis
% dct_iv               -   Type (IV) Discrete Cosine Xform
%
%
%         Data Access Functions
%
% CalcWPLocation       -   calculate location of wavelet packet entry
% node                 -   tree indexing function
% packet               -   packet table indexing
% PackBasisCoeff       -   Insert basis coefficients into packet table
% UnpackBasisCoeff     -   Extract basis coefficients from packet table
%
%
%         Utilities
%
% CalcTreeHeight       -   Utility for PlotBasisTree
% DrawHeisenberg       -   Utility for PlotPhasePlane
```

4.2 Complete Environment

Important point: the WaveLab distribution contains not only `.m` files which implement fundamental algorithms, but also a complete environment associated with the use of those algorithms.

4.2.1 Datasets As an example, we cite the inclusion of datasets and of artificial signals. The `Contents.m` file for directory `WaveLab/Documentation` lists the following contents:

```
% Datasets:Contents v.700 -- Datasets, Documentation, and Readers
%
%         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
%
%   MakeSignal          -   Make artificial signal
%   Make2dSignal        -   Make artificial 2d signal
%
%
%         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 Chris Raphael
%   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
%   daubechies.raw      -   photo of Ingrid Daubechies
```

```
%   fingerprint.raw     -   someone's fingerprint
%   lincoln.raw         -   Honest Abe
%   mriscan.raw         -   someone's brain
%   phone.raw           -   someone's phone
```

The datasets are provided in a centralized way by standard readers, so that to get an image of Ingrid Daubechies, one types:

```
>> ingrid = ReadImage('Daubechies');
```

while to hear a signal of Enrico Caruso singing, one types:

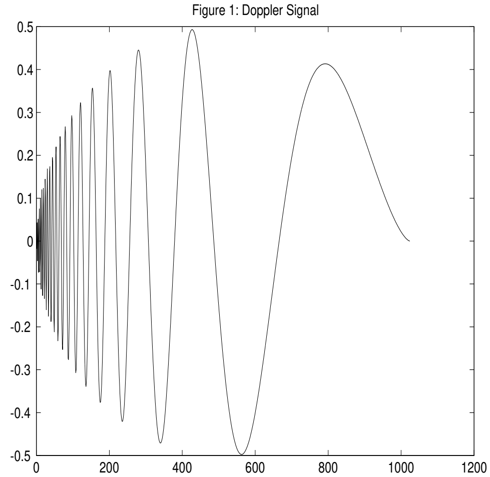
```
>> enrico = ReadSignal('Caruso');
>> sound(enrico,8192);
```

Synthetic signals are provided via centralized synthesizers.

```
doppler = MakeSignal('Doppler',1024);
plot(doppler);
```

creates the following figure:

Figure 1: Doppler Signal



Distribution of datasets is a crucial part of reproducibility. It is also important for our work together at Stanford, because it gives us some common examples that we all know about. Sadly, free distribution of datasets is far less widespread even than free distribution of software.

4.2.2 Documentation A complete computational environment includes on-line documentation. In WaveLab we handle this several ways:

1. As we have seen, `Contents.m` files summarize the contents of individual directories.

2. Each individual function contains its own help, in accordance with Matlab standards. Here is an example:

```
>> help BestBasis

  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.

     We usually consider the best basis to be the basis that most
     compactly represents the signal with respect to a given entropy.
     Once the stattree of entropy values is created, BestBasis
     selects the best basis using the pruning algorithm described in
     Wickerhauser's book.

   Examples
     n = length(signal);
     D = log(n)/log(2);
     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).
```

3. The first line of each help-header (H1 Line) gives information which is searchable by MATLAB command `lookfor`.

Items 1-3 are usual with MATLAB. The next few are less standard.

4. In the documentation directory there are files, compiled automatically as a release is built, giving alphabetical listings of all functions in WaveLab, their synopses and their H1 Lines.

5. *About WaveLab* provides a general overview of the installation and capabilities of WaveLab. Emphasis is placed upon running scripts from papers and workouts.

6. The *WaveLab Reference Manual* has documentation for each WaveLab function, presented in a manner similar to the *Matlab Reference Guide*. The *Reference Manual* is organized by WaveLab directory and contains an alphabetical index of all functions. It is generated automatically, by scripts, from the on-line help for each function in the system.

7. The *WaveLab Architecture* guide describes WaveLab from a systems-level point-of-view, including how the system is built for each platform, how the documentation is generated, how new scripts and datasets are added, etc.

8. Workouts consist of scripts that exercise various aspects of WaveLab. Currently offered are workouts for Best Orthonormal Basis, Matching Pursuit, and "The Cartoon Guide to Wavelets" or "Toons." By running these scripts and reading the code contained therein, one can see how figures that illustrate various aspects of wavelet and time-frequency analysis are generated using WaveLab. Additionally, the workouts offer us a library of figures that are easy to incorporate within our talks and classroom teaching.

If WaveLab were a commercial product, it would make sense to include tutorials and other resources for beginners. Instead, we include only documentation related to performing our *research* and sharing our research with others. Writing documentation makes our own software better – the documentation process often uncovers subtle bugs – and helps others check our work and evaluate the quality of our algorithms. Tutorials do not seem to be of much value to our research agenda.

4.3 Architecture of the Distribution

Here we summarize the architecture fully described in the *WaveLab Architecture* document.

4.3.1 Source The source for WaveLab development has several components in different directories, which we maintain on an Apple Macintosh computer. Matlab source consists of the hundreds of `.m` files that are part of the standard WaveLab distribution. For the most important of these routines, where speed is an issue, we have also provided Matlab `.mex` files which are generated from code written in the C programming language.

TeXsource generates the documentation system for WaveLab, including *About WaveLab*, the *WaveLab Reference Manual* and *WaveLab Architecture*.

Scripts written in MPW (Macintosh Programmer's Workshop) and Perl provide for the automatic build of the entire WaveLab system from the source.

4.3.2 Build The build process involves compiling the `.mex` files for the appropriate platform (i.e. Unix, Mac or PC), appending version information to each `.m` file, and automatically generating documentation from the help headers of each function. This includes synopsis listings that are included on-line in the WaveLab directory `Documentation` as well as the WaveLab Reference Manual.

Unfortunately, the build process for the PC is quite a bit more involved than Unix or Mac because of the eight-character filename limit associated with DOS and Windows. Hopefully the new version of Windows, code-named Chicago, will eliminate this restriction. In the meantime, we must map long names to eight-character versions.

4.3.3 Internet Support The WaveLab distribution, including compressed files for each platform and documentation, is provided over the Internet by both an FTP site `ftp://playfair.stanford.edu/pub/wavelab`, and a World-Wide-Web page, `http://playfair.stanford.edu/~wavelab`. Our papers about wavelets, including, of course, the papers that have directories within `WaveLab/Papers`, are provided on the same FTP site and Web server.

To access papers via FTP, use `ftp://playfair.stanford.edu/pub/lastname`, where `lastname` is the last name of the paper's first author. A more convenient interface to the same information is offered through the Stanford Statistics Department's WWW home page, `http://playfair.stanford.edu`.

5 Examples

In this section we give a brief idea of some of the capabilities of WaveLab , by illustrating some of the figures we have created as part of our recent research. A key point: all the figures in this article can be reproduced by obtaining version .700 of WaveLab and using the files in `Papers/VillardDeLans`. Many other figures can be reproduced; see the WaveLab distribution, or the published articles [17, 19, 21, 23, 25].

5.1 Wavelets

WaveLab of course offers a full complement of wavelet transforms – both the standard orthogonal periodized wavelet transforms `FWT_PO` [14], standard boundary-corrected wavelet transforms `FWT_CDJV` [9], and the standard periodized biorthogonal wavelets `FWT_PBS` [8]. It also offers less standard wavelet transforms which have been developed as part of research at Stanford. Two examples include interpolating wavelet transforms based on interpolation schemes (`FWT_DD` for what we call "Deslaurier-Dubuc wavelets" [16]) and average-interpolating wavelet transforms (`FWT_AI` [19]).

Less standard is the wavelet transform based on the Meyer Wavelet. Eric Kolaczyk has developed `FWT_YM` as part of his Thesis [26]. Figure 1 shows a Meyer wavelet with third-order polynomial window function. It was produced by the code fragment:

```
meyer = MakeWavelet(4,7,'Meyer',3,'Mother',1024);
plot(meyer);
title('Figure 2: Meyer Wavelet');
```

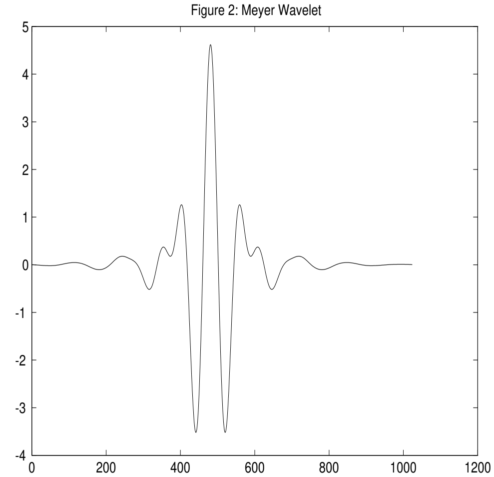


Figure 2: Meyer Wavelet

5.2 Compression Example

There has been a lot of interest recently in the use of wavelets for data compression [1, 15]. WaveLab offers a range of discrete trigonometric transforms (`dct_ii`, `dct_iii`, `dct_iv`, `dst_ii`, and `dst_iii`) [29]. It is therefore quite easy to compare standard trigonometric transforms with standard wavelet transforms as far as compression goes.

For the purposes of this article, we will call transform-based compression the act of going into the transform domain, setting to zero all but a few percent of the coefficients, and returning to the original domain. A full compression scheme would require various coders to optimally store the nonzero coefficients in the transform domain, but it is well-established that the total number of bits used and the quality of reconstruction after such processing correlate well with the performance in the simpler definition of transform compression here.

Figure 3 shows a side-by-side comparison of an seismic signal, its 95% compression using wavelets and its 95% compression using DCT.

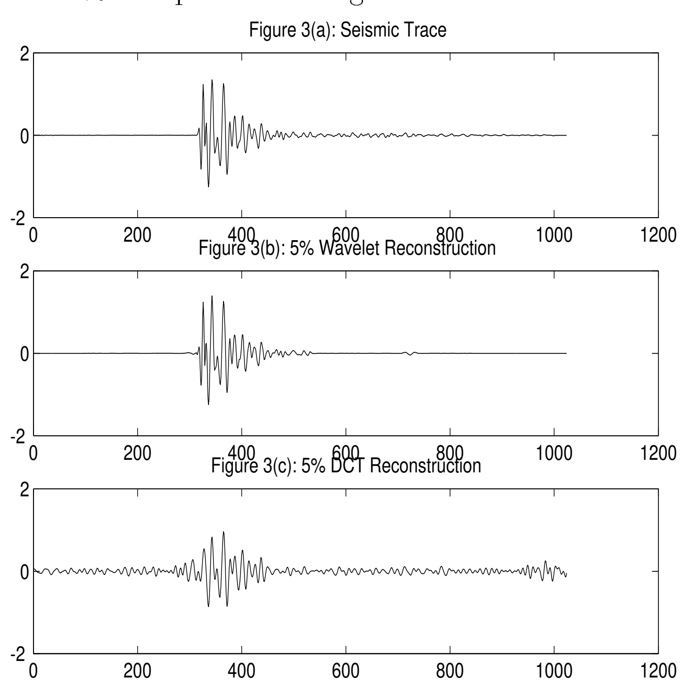


Figure 4 shows a a picture of Ingrid Daubechies and the compression curves for both the wavelet and DCT transforms.

Figure 4(a): Ingrid Daubechies

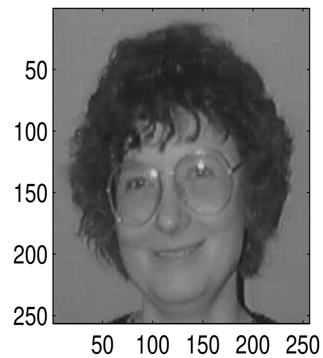
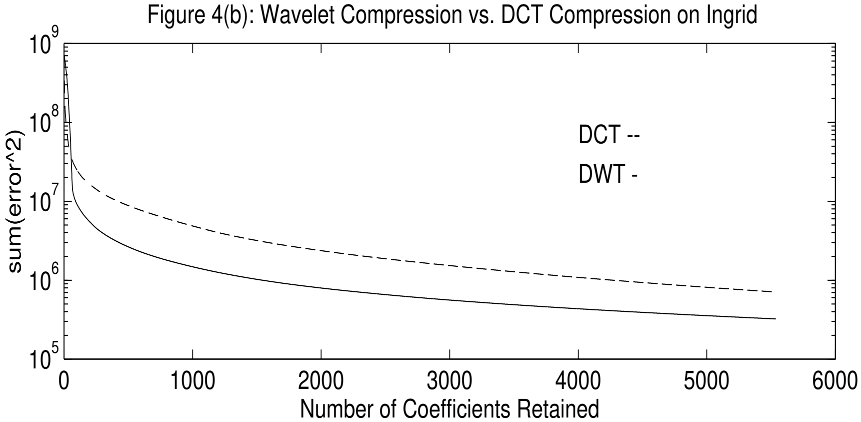


Figure 4(b): Wavelet Compression vs. DCT Compression on Ingrid

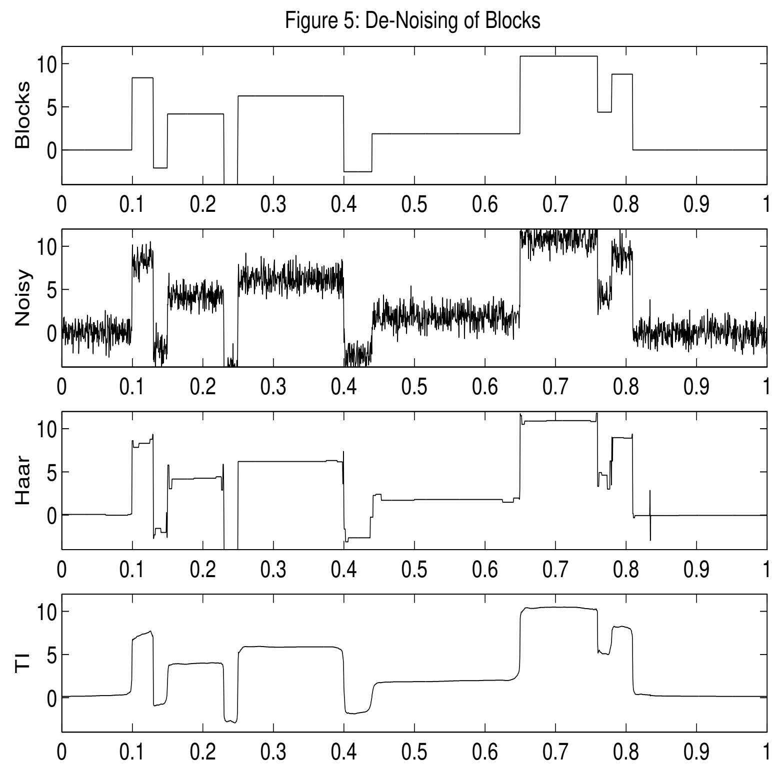


In both examples, the superiority of wavelets in compression is evident.

5.3 De-Noising

Our own research in wavelets began with interest in the applications of wavelet to removing noise from signals and images, and in particular learning how best to do this with wavelet thresholding. As our research has evolved we have tried new approaches – *second-generation de-noising*. One of theseis the use of translation-invariant approaches to de-noising [10].

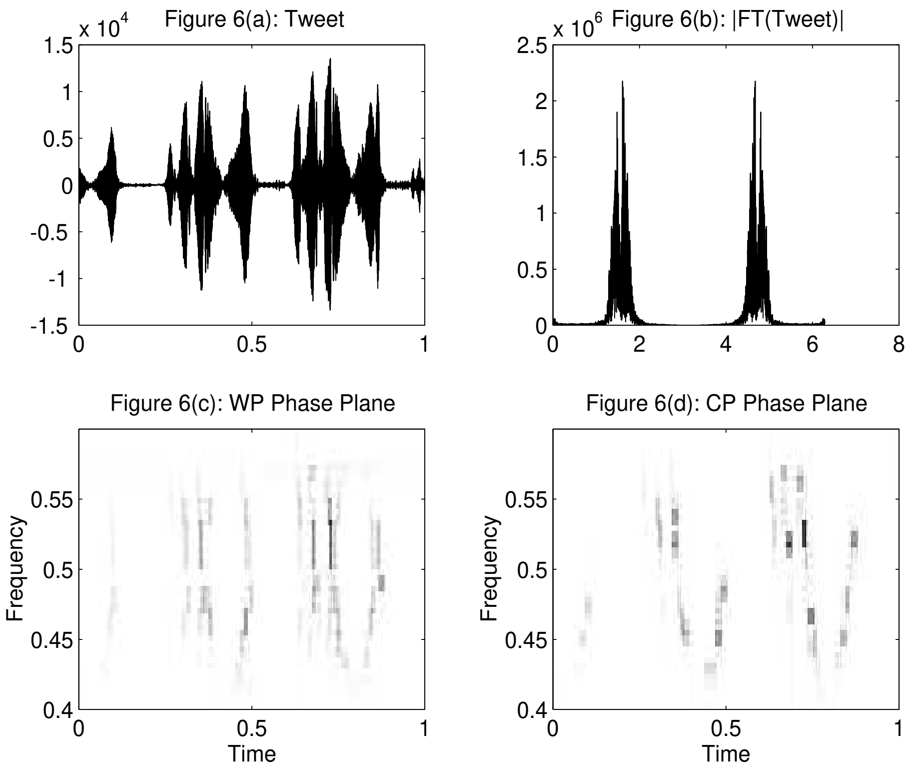
Figure 5 shows the use of a stationary, or translation-invariant, Haar transform for de-noising a noisy version the signal Blocks. For comparison, the standard Haar transform is also included. The improvement in accuracy is evident.



5.4 Wavelet Packet Artifacts

Currently, much of the interest and attention of the wavelets community is focused on the use of wavelet packets and cosine packets. The very elegant theories behind these approaches, due to Coifman, Meyer and Wickerhauser, have not yet been developed into a full-scale methodology, where the difficulties in applications are catalogued, well-understood and avoided. Using WaveLab, we have been able to identify a number of anomalies and artifacts associated with the best-basis and wavelets approaches. Others can easily reproduce and study these examples, and think of ways to avoid them.

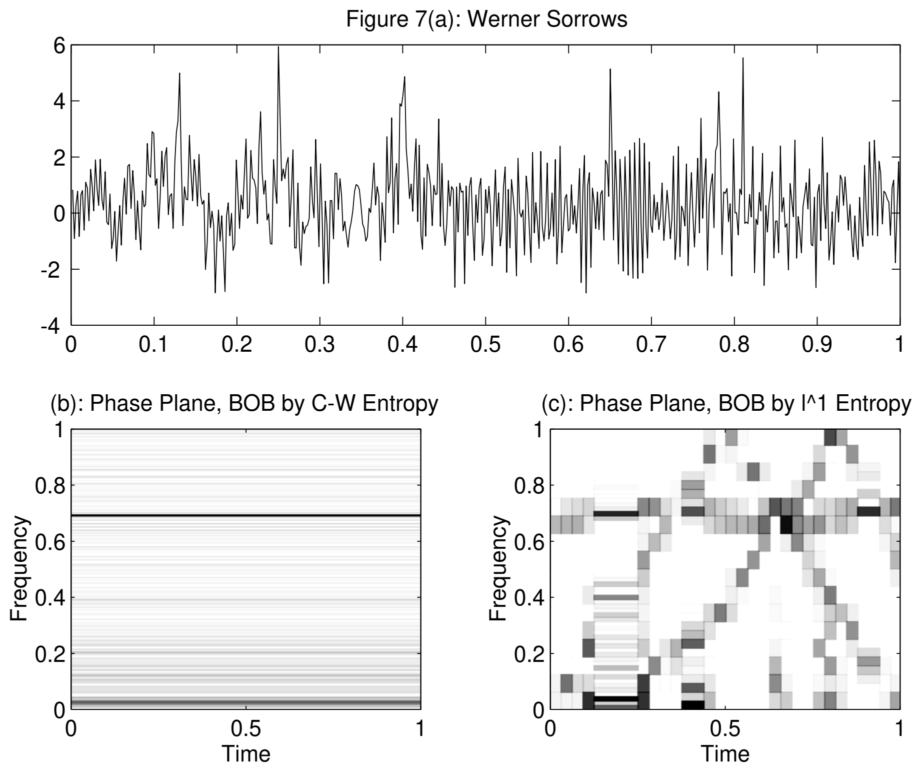
The first example has to do with an artifact of wavelet packets caused when the signal of interest concentrates near a frequency with simple dyadic structure. The signal `Tweet` that comes with WaveLab was provided by Doug Jones of the University of Illinois. When we tried to analyze it by Wavelet Packets, we got the time-frequency phase plane shown in figure 6c below. In contrast when we tried to analyze it by Cosine Packets, we got the phase plane in figure 6d.



The Cosine Packets phase plane is dramatically clearer, and shows quite clearly the chirping structure of the bird's song. Wavelet Packets fail in this case because the bird's natural pitch is nearly half the Nyquist rate for the signal sampling, which is the worst possible choice for wavelet packets.

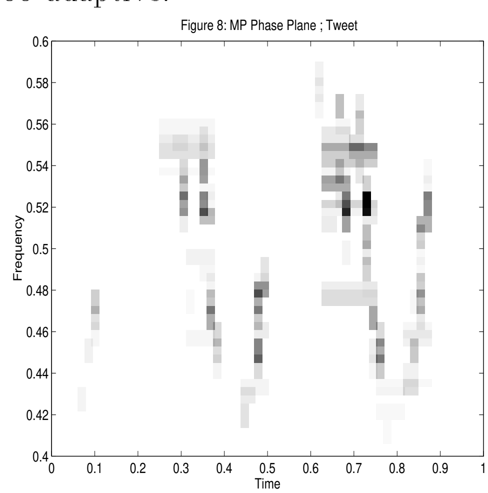
The second example has to do with choice of entropy. Most people use the Coifman-Wickerhauser original choice of entropy, or "Shannon Entropy." However, we are unaware of any specific rationale for this choice of entropy which can be tied to performance. In fact, other entropies can often perform better. Based on work in [6] we often prefer the ℓ^1 norm as an entropy.

In Figure 7 below we show the phase plane for the artificial signal `Werner Sorrows` obtained using two different entropies. The Coifman-Wickerhauser entropy chooses as best basis a global Fourier basis, and the time-varying structure of the signal is completely lost (Figure 7b). The ℓ^1 entropy chooses a time-varying basis and the resulting time-varying structure is revealed (Figure 7c).



5.5 Matching Pursuit Artifacts

Matching Pursuit is a popular method for non-orthogonal decomposition; using WaveLab we have found some interesting computational results. When MP is applied to the Tweet Signal of Figure 6 using the same Cosine Packet dictionary as in Figure 6d, we see that the nonorthogonal decomposition found by MP is markedly less clear than that found by BOB. In this case MP is too adaptive.



5.6 Minimum Entropy Focusing

We now give examples of some experiments which are easy to conduct in WaveLab. The first was based on the idea of determining if one could measure image sharpness from the wavelet transform – at least well enough to provide an auto-focus mechanism.

In order to test this idea, we took the object `Blocky`, which is an artificial signal built into WaveLab, and blurred it out. The blurring filter for this experiment was the two term autoregressive filter $y_t - 2\rho y_{t-1} + \rho^2 y_{t-2} = x_t$. The parametrized family of three term FIR filters $(b^\tau * y)_t = y_t - 2\tau y_{t-1} + \tau^2 y_{t-2}$ contains the inverse of the first filter as a special case, by taking $\tau = \rho$. How can we find, from the data alone, information guiding us to deconvolve by picking τ appropriately?

In our deblurring experiment, we tried a method of minimum wavelet-domain entropy. We set $\rho = .9$, and for each τ in a grid, we evaluated the normalized wavelet-domain entropy, searching for a minimum

$$\min_\tau \mathcal{E}(WT[b^{(\tau)}x]).$$

Here the entropy is a normalized ℓ^1 entropy of the fine scale coefficients:

$$\mathcal{E}(w) = \sum_{j \geq 3} |w_{j,k}|.$$

Figure 9 shows that when we searched through τ in the grid $\{-1, -.9, \ldots, .8, .9, 1\}$, the selected minimum was in fact .9 – just as one would hope.

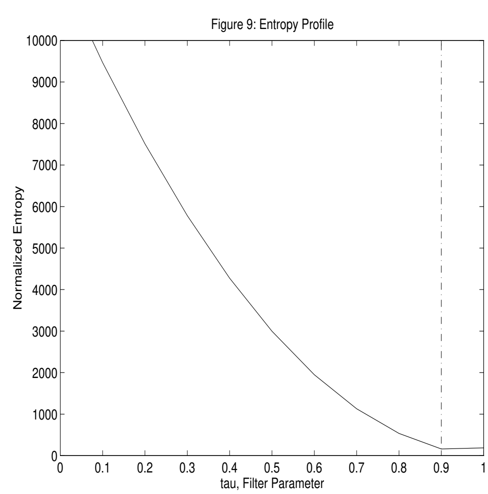
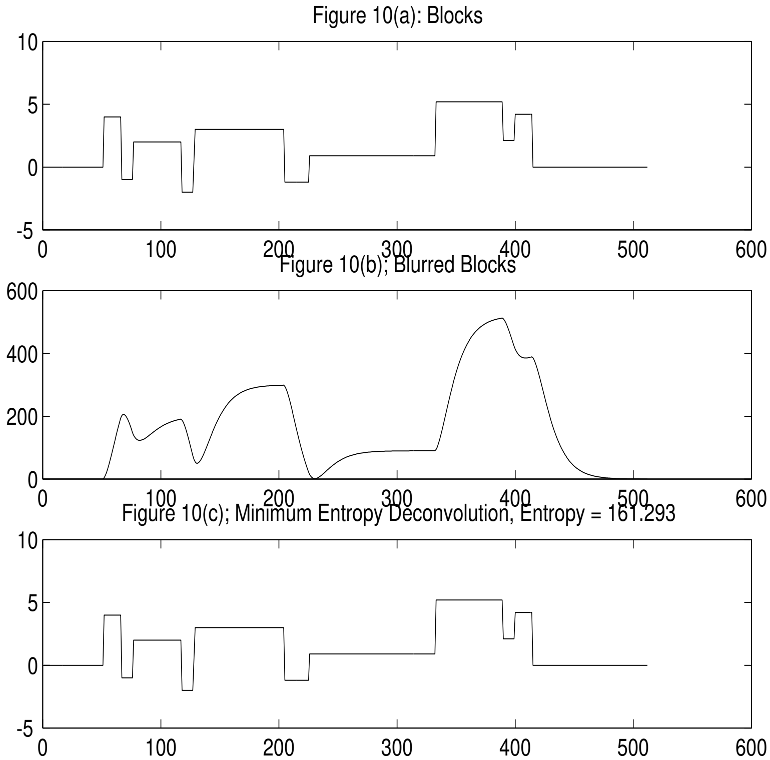


Figure 10 shows the original, blurred, and restored signal. In this experiment, a criterion of minimum entropy in the wavelet domain identified the correct deblurring filter – *blindly*.



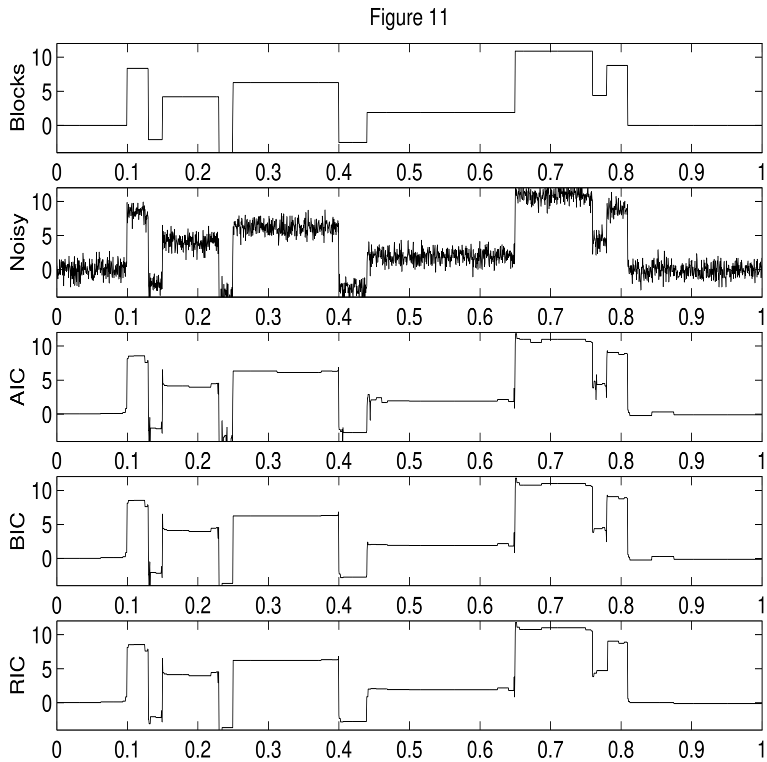
5.7 Tree-Constrained Thresholding

De-noising by wavelet thresholding acts essentially to keep or kill certain wavelet coefficients. The coefficients which survive often exhibit a certain pattern: a coefficient at a finer scale never survives thresholding unless its parent also survives.

This suggests that one might profitably *require* this hereditary pattern of surviving wavelet coefficients, i.e. *require that surviving wavelet coefficients must be in a tree pattern*. One could aim at finding the *best tree pattern* by an optimization. When using an orthonormal wavelet transform, keeping or killing according to a tree pattern corresponds to projection on a certain linear subspace associated with the tree. Finding the best projection to use (in terms of minimizing mean squared error) is what statisticians call model selection. One could therefore search for the tree which does best according to a classical model selection criterion. Letting y_T denote the least-squares projection of the data on the linear subspace associated with the tree, we define

$$CPRSS_\lambda(y,T) = \|y - y_T\|_2^2 + \lambda \cdot \#\{T\}.$$

We propose model selection by finding the tree that minimizes $CPRSS_\lambda$. Standard AIC model selection uses $\lambda = 2$, BIC uses $\lambda = \log(n)$, and RIC uses $2\log(n)$. The calculations are surprisingly easy, and involve analogs of the Coifman-Wickerhauser pruning algorithm. We illustrate the results in Figure 9 below on a noisy version of object `Blocks`, using a Haar transform.



In terms of RMS error, BIC outperforms both AIC and RIC in this case, and BIC also outperforms, in this case, the automatically adaptive thresholding (SUREShrink) of Donoho and Johnstone [22].

6 Future Issues

6.1 Reproducibility and Hypermedia

"Really reproducible research" is a very ambitious goal which is only partially addressed by our work. We have lowered the effort required for others to reproduce a specific figure or batch of figures in our papers. In the *ancien régime* this would have required several man-weeks of work – the work of programming similar algorithms, testing them for correctness and applying them to similar data – while in our brave new world it takes less than an hour – the work of locating, downloading and installing some software. The new approach is several orders of magnitude less work than the older approach.

Things can still improve by at least two more orders of magnitude.

Integrating Reproducibility into Scientific Publication. People sometimes forget that the current norms of scientific publication did not spring fully formed into widespread practice. We have read that Pasteur had the revolutionary idea to advance reproducibility in the biological sciences by adding sections to articles which gave *Materials*, *Procedures*, *Methods of Analysis*, and so on. The idea of carefully spelling out how a biological experiment was performed, and the nature of the biological specimens employed, seems so natural and automatic today, but at one time such information was not provided as part of scientific publication. After Pasteur, the accepted norms changed, and such information was furnished as a routine part of publication.

In the future, we can envision that publication in computational sciences will change so that reproducibility is integrated into process. One way to do this would be if journals were fully electronic, and if we adopted hypermedia techniques. Then every computationally-generated figure and every computationally-generated table in an article would become linked to the code and the computational environment that produced the figure. If one were interested in a figure, one would click on it with a mouse, and a new window would instantly appear, containing the code that the author of the article used to create the figure. To reproduce the figure, but perhaps change slightly the settings of the display software (for example, to view a surface from a different 3-d perspective), one would simply edit the code in the window and re-run the code; the figure would be re-computed and re-displayed.

We envision this as being two orders of magnitude easier than the current approach for these reasons:

- *Universality.* There would be a universal user interface to such electronic publications that everyone understood, and the underlying code generating figures would be in a QPE language that everyone understood. The reader would not have to learn a new QPE language in order to reproduce work of some other researcher.

- *Transparency.* The reader would only be involved in the act of reading the electronic journal and clicking on a few buttons. The reader would not have to "purchase a QPE," "download software" or "install software." The QPE would be freely distributable under a freeware arrangement like the GNU public license – one wouldn't buy it. The QPE wouldn't be consciously "installed" by the user, but instead would be automatically installed by the browser which displayed the electronic journal. The software which reproduced the figure would not have to be "located" by the user on some distant Internet host and then "downloaded" onto the user's machine; instead the browser of the electronic journal would locate the software somewhere on the Internet, download and install it.

The current scheme of reproducibility under WaveLab does not measure up to this vision.

- The code that reproduces the figures is not integrated into a viewer of the articles.

- The user must own and install Matlab (which is much more expensive than most PC software).

- Matlab is not a universal QPE; many others are in common use.

- The user must find, download and install the WaveLab system.

- The user must locate, within WaveLab, the figure he wants.

Answering all these obstacles, by meeting the goals of Transparency and Universality, will make reproducibility achievable in ten seconds or less rather than in an hour – this is the two-order of magnitude gain we wrote of above.

There are some interesting developments that point in the direction we envision.

Claerbout's group at Stanford has implemented a system that goes reasonably far in this direction – they have developed a custom TEXviewer and a set of supporting tools so that one can burn and rebuild illustrations and they have developed an interactivity facility, where dials or sliders are attached to parameters of a figure and one interactively changes the figure under the control of those dials or sliders. There are of course ways in which this pioneering effort fails to be the universal solution. The system does not use a universal QPE, doesn't work on PC's and Mac's, and lacks the full editability features we wrote of above. However, it is freely distributable and is extremely inspiring.

In another direction, commercial QPE's like MATHEMATICA and MATLAB have developed "notebook" interfaces which are somewhat in the direction of "click on a figure and see the source that generated it." Perhaps they can be more tightly integrated into scientific journal publication.

Finally, the OAK project at Sun Microsystems has the promising goal of creating a computing language which is network-based, in which one never "purchases," "downloads" or "installs" software – it is just located, installed, and run seamlessly and automatically. An electronic journal browser implemented in OAK, together with a QPE implemented in OAK, would form the foundation to realize the vision described above.

6.2 Goals for Statisticians

Historically, Statisticians have been heavily involved in the methodology of science, which includes data presentation, data visualization and conduct of scientific experiments. However, the postwar era has mostly emphasized Statistics as a branch of *applied stochastics*. Successful efforts by Statisticians to develop software packages and visualization tools show that Statistics is more than just applied stochastics. (Admittedly there would be some controversy in France and Belgium on this score, while the point would be more easily accepted in the UK and USA.) We would like to encourage Statisticians reading this article to attend to the the development of "the scientific method" in all its guises. The effort towards "really reproducible research" is worth further effort.

6.3 Acknowledgments

WAVELAB has been developed by Jonathan Buckheit, Shaobing Chen, David Donoho, Iain Johnstone and Eric Kolaczyk, all from Stanford University, and Jeffrey Scargle, NASA-Ames Research Center. Comments or questions about WAVELAB may be directed to *wavelab@playfair.stanford.edu*.

Jon Buckheit's graduate studies have been partially supported by an NSF graduate support program. David Donoho's research has been partially supported by NSF DMS 92-09130, by a university interchange agreement with the NASA Astrophysics Data Program and by other sponsors.

The authors would like to thank Anestis Antoniadis for his unstinting efforts in organizing the *XV Rencontres Franco-Belges*. Professor Antoniadis also asked us to write this article, though he bears no blame for the result. We also thank Shaobing Chen for providing three of the figures used in this paper.

jon@playfair.stanford.edu
donoho@playfair.stanford.edu

References

[1]Antonini, M., Barlaud, M., Mathieu, P. and Daubechies, I. (1991). Image coding using wavelet transforms. To appear, *IEEE Proc. Acoustics, Speech and Signal Processing.*

[2]Buckheit, J.B. and Donoho, D.L. (1995). A Cartoon Guide to Wavelets. Technical Report, Department of Statistics, Stanford University.
`ftp://playfair.stanford.edu/pub/buckheit/toons.ps`.

[3]Buckheit, J.B, Chen, S., Donoho, D.L., Johnstone, I.M. and Scargle, J.D. (1995). *About WaveLab.* `ftp://playfair.stanford.edu/pub/wavelab/AboutWaveLab.ps`.

[4]Buckheit, J.B., Donoho, D.L. and Scargle, J.D. *WaveLab Architecture.*
`ftp://playfair.stanford.edu/pub/wavelab/WaveLabArch.ps`.

[5]Buckheit, J.B, Chen, S., Donoho, D.L., Johnstone, I.M. and Scargle, J.D. (1995). *WaveLab Reference Manual.* `ftp://playfair.stanford.edu/pub/wavelab/WaveLabRef.ps`.

[6]Chen, S. and Donoho, D.L. (1994). On Basis Pursuit. Technical Report, Department of Statistics, Stanford University. `ftp://playfair.stanford.edu/pub/chen_s/asilomar.ps.Z`.

[7]Claerbout, Jon (1994). Hypertext Documents about Reproducible Research.
`http://sepwww.stanford.edu/sep/jon/blurb.html` and `nrc.html`.

[8]Cohen, A., Daubechies, I. and Feauveau, J.C. (1990). Biorthogonal bases of compactly supported wavelets. To appear, *Comm. Pure Appl. Math.*

[9]Cohen, A., Daubechies, I., Jawerth, B. and Vial, P. (1992). Multiresolution analysis, wavelets, and fast algorithms on an interval. To appear, *Comptes Rendus Acad. Sci. Paris* (A).

[10]Coifman, R.R. and Donoho, D.L. (1995). Translation-Invariant De-Noising. *This Volume.*

[11]Coifman, R.R. and Meyer, Y. (1991). Remarques sur l'analyse de Fourier à fenêtre. *Comptes Rendus Acad. Sci. Paris* (A) **312** 259-261.

[12]Coifman, R.R., Meyer, Y. and Wickerhauser, M.V. (1992). Wavelet analysis and signal processing. In *Wavelets and Their Applications*, pp. 153-178, M.B. Ruskai et al., eds., Jones and Bartlett, Boston.

[13]Coifman, R.R. and Wickerhauser, M.V. (1992). Entropy-based algorithms for best-basis selection. *IEEE Trans. Info. Theory* **38**(2)713-718.

[14]Daubechies, I. (1992) *Ten Lectures on Wavelets.* SIAM, Philadelphia.

[15]DeVore, R.A., Jawerth, B. and Lucier, B.J. (1992). Image compression through wavelet transform coding. *IEEE Trans. Info Theory*, **38**(2)719-746.

[16]Donoho, D.L. (1992). Interpolating Wavelet Transforms. Technical Report, Department of Statistics, Stanford University.
`ftp://playfair.stanford.edu/pub/donoho/interpol.ps.Z`.

[17]Donoho, D.L. (1993) Nonlinear Wavelet Methods for Recovery of Signals, Images, and Densities from Noisy and Incomplete Data. In *Different Perspectives on Wavelets*, I. Daubechies, ed. American Mathematical Society, Providence, RI. `ftp://playfair.stanford.edu/pub/donoho/ShortCourse.ps.Z`.

[18]Donoho, D.L. (1994). On Minimum Entropy Segmentation. In *Wavelets: Theory, Algorithms and Applications.* C.K. Chui, L. Montefusco and L. Puccio, eds. Academic Press, San Diego. `ftp://playfair.stanford.edu/pub/donoho/MES_TechReport.ps.Z`.

[19]Donoho, D.L. (1993). Smooth Wavelet Decompositions with Blocky Coefficient Kernels. In *Recent Advances in Wavelet Analysis*, L. Schumaker and F. Ward, eds. Academic Press. `ftp://playfair.stanford.edu/pub/donoho/blocky.ps.Z`.

[20]Donoho, D.L. (1993). Unconditional Bases are Optimal Bases for Data Compression and for Statistical Estimation. *Applied and Computational Harmonic Analysis*, **1**, 100-115. `ftp://playfair.stanford.edu/pub/donoho/UBRelease.ps.Z`.

[21]Donoho, D.L. (1993). Wavelet Shrinkage and W.V.D. – A Ten-Minute Tour. In *Progress in Wavelet Analysis and Applications,* Y. Meyer and S. Roques, eds. Éditions Frontières, Gif-sur-Yvette. `ftp://playfair.stanford.edu/pub/donoho/toulouse.ps.Z`.

[22]Donoho, D.L. and Johnstone, I.M. (1994). Adapting to Unknown Smoothness by Wavelet Shrinkage To appear, *J. Amer. Stat. Assoc.*. `ftp://playfair.stanford.edu/pub/donoho/ausws.ps.Z`.

[23]Donoho, D.L. and Johnstone, I.M. (1994). Ideal Spatial Adaptation via Wavelet Shrinkage. *Biometrika*, **81**, 425-455. `ftp://playfair.stanford.edu/pub/donoho/isaws.ps.Z`.

[24]Donoho, D.L. and Johnstone, I.M. (1994). Ideal Time-Frequency Denoising. Technical Report, Department of Statistics, Stanford University. `ftp://playfair.stanford.edu/pub/donoho/tfdenoise.ps.Z`.

[25]Donoho, D.L., Johnstone, I.M., Kerkyacharian, G. and Picard, D. (1993). Wavelet Shrinkage: Asymptopia. To appear, *J. Roy. Statist. Soc.* `ftp://playfair.stanford.edu/pub/donoho/asymp.ps.Z`.

[26]Kolaczyk, E. (1994). WVD Solution of Inverse Problems. Ph.D. Thesis, Stanford University.

[27]Mallat, S. and Zhang, S. (1993). Matching Pursuits with Time-Frequency Dictionaries. *IEEE Transactions on Signal Processing,* **41**(12):3397-3415.

[28]Meyer, Y. *Wavelets: Algorithms and Applications.* SIAM: Philadelphia, 1993.

[29]Wickerhauser, M.V. (1994). *Adapted Wavelet Analysis, from Theory to Software.* AK Peters: Boston.