WavBox 4*: A Software Toolbox for Wavelet Transforms and Adaptive Wavelet Packet Decompositions†

Carl Taswell‡

2 December 1994

Abstract

WavBox provides both a function library and a computing environment for wavelet transforms and adaptive wavelet packet decompositions. WavBox contains a collection of these transforms, decompositions, and related functions that perform multiresolution analyses of 1-D multichannel signals and 2-D images. The current version 4.1c includes overscaled pyramid transforms, discrete wavelet transforms, and adaptive wavelet and cosine packet decompositions by best level, best basis, and matching pursuit as described by Mallat, Coifman, Wickerhauser, and other authors. WavBox also implements Taswell's new search algorithms with decision criteria, called near-best basis and non-additive information costs respectively, for selecting bases in wavelet packet transforms, as well as Donoho and Johnstone's wavelet shrinkage denoising methods. Various choices of filter classes (orthogonal, biorthogonal, etc), filter families (Daubechies, Vetterli, etc), and convolution versions (interval, circular, extended, etc) exist for each transform and decomposition. The software has been designed for efficient automated computation, interactive exploratory data analysis, and pedagogy. Essential features of the design include: perfect reconstruction for multiresolution decomposition of data of arbitrary size not restricted to powers of 2; both command line and graphical user interfaces with a comprehensive set of plots and visual displays; an object property expert system with artificial intelligence for configuring valid property combinations; heirarchical modules and switch-driven function suites; vector-filter and matrix-operator implementations of convolutions; extensibility for the inclusion of other wavelet filters, convolution versions, and transforms; optional arguments with built-in defaults for most m-files; and extensive on-line help and self-running tutorial demos.

†This paper will be published as reference [4].

‡Scientific Computing and Computational Mathematics, Bldg 460 Room 314, Stanford University, Stanford, CA 94305-2140; Internet email: taswell@sccm.stanford.edu; Telephone: 415-723-4101; Facsimile: 415-723-2411.

1 Introduction

The name WavBox is a simple abbreviation of the words "WAVelet toolBOX". As a software toolbox, WavBox provides both a function library and a computing environment for wavelet transforms and adaptive wavelet packet decompositions. The "function library" aspect of WavBox refers to its use as a collection of callable subroutines for inclusion in end-user written programs. The "computing environment" aspect of WavBox refers to its use as a convenient system for the interactive exploratory analysis and manipulation of data. As such, the WavBox computing environment has two interfaces, both textual (driven by input from the keyboard) and graphical (driven by input from a pointing device), which can be used simultaneously in an integrated manner during the same work session.

WavBox contains a collection of transforms, decompositions, and related functions that perform wavelet and other multiresolution analyses of single-channel/segment one-dimensional (1-D) signals, multi-channel/segment 1-D signals, and 2-D signals. Multi-channel 1-D signals are those that have two or more data points at each time point such as two-channel music (left and right stereo channels) and twelve-channel electrocardiograms. Multi-segment 1-D signals are those that have only one data point at each time point but for which the entire signal is segmented into intervals analyzed in parallel. Both multi-channel and multi-segment 1-D signals are represented as matrices and are analyzed by 1-D algorithms. Images and other 2-D signals are also represented as matrices but are analyzed by 2-D algorithms.

The original algorithms forming the core of WavBox were written in 1991 and described in a technical report [5] available at that time. These algorithms have now been recently published by the Association for Computing Machinery [6]. Since the development of this original collection, a major new version of the toolbox has been written each year. In 1993, the three different versions available at that time (those written in 1991, 1992, and 1993) were named WavBox 1, 2, and 3, respectively. Essentially, several different software designs were tested in WavBox 1, 2, and 3 before the current design was selected for WavBox 4. As a consequence, WavBox 4 was written to combine and unify these three separate collections[1] with a common architecture supporting both a textual and graphical interface in an integrated environment. WavBox 4 was written during 1993–94 and first released in June 1994. It is the current version 4.1c of WavBox which is described in this paper. All versions have been written in the MATLAB language [1].

2 Overview of Development Goals and Software Architecture

As announced in the Wavelet Digest in July 1993 [2], the development of WavBox 4 has been guided by the following goals: 1) to provide a comprehensive collection of transforms and decompositions rather than one focussed on a specific author's work or particular perspective of the field; 2) to be useful for both exploratory data analysis as well as efficient computation in routine automated tasks; and 3) to be appropriate for instruction in educational settings. These goals have been implemented with a software architecture that incorporates the following principles: 1) heirarchical modules, 2) switch-driven function suites, 3) required arguments only for data, 4) optional arguments for parameters, all with built-in defaults, and 5) parallelized algorithms with efficient vectorization. Moreover, a requirement for generality, flexibility, and extensibility of algorithms has been imposed.

[1] WavBox 1, 2, and 3 were distinct collections and not successive versions of the same collection.

This requirement has been interpreted to mean that the transform and decomposition algorithms should be valid for a wide variety of filter classes, convolution versions, and arbitrary data sizes and dimensions, and that the overall architecture should accomodate extensions simply and efficiently.

Full implementation of these development goals in an actual programming language is necessarily restricted by the feature set of the language chosen. The MATLAB language has only one type of data structure: every variable is a matrix. It is thus not an object-oriented language with a variety of data structures and features such as the class inheritance found in the language C++. Nevertheless, some of the general principles of an object-oriented approach to programming can be emulated in MATLAB. This approach can be used to address some of the major complaints voiced by users of function libraries concerning the function arguments: how many, which ones, and in what order? WavBox resolves this problem with its system of objects and object properties. The following code example introduces and demonstrates this system and the basic paradigm used by all transforms and decompositions in WavBox.

```
>> setwb('objtyp','dwt')         % initialize ObjectType to DWT
>> load datafile                 % load x from datafile
>> setwb('datsiz',size(x))       % reset DataSize to size(x)
>> y = dwt(x);                   % forward transform
>> plotdwt(y)                    % plot transform
>> p = getwb('all');             % put all property values in p
>> save objectfile y p           % save y and p to objectfile
>>
>> clear all; clear global       % simulate restart of MATLAB
>>
>> load objectfile               % load y and p from objectfile
>> setwb('all',p)                % set properties to values stored in p
>> a = idwt(y);                  % inverse transform
>> load datafile                 % load x from datafile
>> e = esterror(x,a)             % compute estimate error
```

In this example[2] with two work sessions simulated by their separation with the `clear all` command, there are only data arguments, not parameter arguments, for the functions `dwt` and `idwt`. All parameters defining the forward and inverse transform pair, such as filter, convolution version, and transform depth, are implemented as object properties controlled by the utilities `setwb` and `getwb` with default values defined in the editable file `defwb`. Thus, the user can work with simple commands of the general form `y = transform(x)` without worrying about complicated lists of function arguments.

3 WavBox Objects and Properties

WavBox contains fixed transforms that are signal-independent and adaptive decompositions that are signal-dependent. All WavBox transforms and decompositions, however, are characterized by being multiresolution analyses of the signal. To avoid common ambiguities in terminology[3], WavBox distinguishes between mapping structures used for computation of intermediate results and object structures used for storage of the final output of a function. The object structure is not necessarily the same as the mapping structure. For example, in the case of adaptive wavelet packet

[2]The commands `load`, `save`, and `clear` are standard MATLAB commands and all others are WavBox commands. The "`wb`" in WavBox commands such as `setwb` and `getwb` is an abbreviation of "WavBox".

[3]For example, does the term "transform" refer to the function or the output of the function?

decompositions, the mapping structure may be a discrete packet table but the object structure may be a discrete packet table (a redundant representation or overcomplete set of coefficients), a discrete packet basis (a complete set), or a discrete packet list (a complete or undercomplete set). These terms are defined in another paper [3] describing in greater detail some of the algorithms contained in WavBox.

Thus, all of the WavBox multiresolution analysis functions (the fixed transforms and adaptive decompositions) map an input variable, which is the signal/image data in the data domain, to an output variable, which is the WavBox multiresolution object in the object domain. Each object has a set of properties which are the parameters (filters, convolution version, number of multiresolution levels, etc.) necessary to specify the mapping that computed the object. Without these properties appropriately managed in the environment, it would not be possible to display and manipulate the object conveniently in the object domain. More importantly, without these properties appropriately saved in a variable, it would not be possible to save the object, and then at another work session, to inverse map the object conveniently back to the data domain. Thus, each object variable has an associated property set variable; and each object must be stored, manipulated, and retrieved with its associated property set in order to obtain correct results of manipulations in the object domain as well as inversions back to the original signal/image data domain.

All parameters in the set of properties are implemented as global variables. This design eliminates the need for what would otherwise be the burdensome task of passing an additional list of arguments to various functions for the display and manipulation of objects in the object domain. When saved as a single variable, all properties in the set are packed into a record structure with each property in a field of the record. Each property has a name and value and can be accessed from the command line interface (CLI) using the functions `setwb` and `getwb` in a manner analogous to the MATLAB Handle Graphics functions `set` and `get` [1]. Object properties can also be accessed from the graphical user interface (GUI) using the popupmenus and edit boxes on the WavBox Master Control Panel displayed by the function `dispwb`. Both `dispwb` and `demowb` can be started with the command `wavbox` which also allows the user to modify screen-dependent display defaults.

All objects have a set of default properties which can be initialized by calling the function `setwb` with the name of the transform or decomposition. For example, to set the object properties to defaults for the wavelet packet decomposition by best basis, use the command `setwb('OBJTYP','WPDB')` or the command `setwb('objtyp','wpdb')` since all property names and property values are not case sensitive. To display all current property values, use the command `getwb('all')` which returns

```
DataDimension = 1
DataStructureSize = 448 x 1
MappingStructureType = WPT
MappingStructureClass = DPT
MappingStructureSize = 448 x 5
ObjectStructureType = WPDB
ObjectStructureClass = DPL
ObjectStructureSize = 448 x 4
FilterClass = ORTH
FilterFamily = DOLA
AnalysisFilterParameter = 5
SynthesisFilterParameter = NaN
ConvolutionVersion = BAF
DesiredLevel = 5
```

```
MaximumLevel = 4
YScaleLengths
   448   224   112    56    28    28
XScaleLengths
     1     1     1     1     1     1
FilterName = DOLA5
FilterLength = 10
```

although actually not all properties are necessarily displayed since the interpretation of `'all'` by `getwb` can be modified by the user to display a partial list as shown here.

To get an individual property value, use its abbreviated name such as `'conver'` for `ConvolutionVersion` in the command `getwb('conver')` which returns the information `ConvolutionVersion = BAF`. To set an individual property to a new value, use both its name and the desired value such as `'cps'` for `Circularly-Periodized Signal` in the command `setwb('conver','cps')`. To set more than one property to a new value, use pairs of property names and values as in this example where filter class is set to biorthogonal, filter family to Daubechies' biorthogonal symmetric spline, analysis parameter to 3, and synthesis parameter to 1 with the command

```
setwb('filcls','bior','filfam','dbss','anapar',3,'synpar',1).
```

To display possible values for a property, use just the name without a value. For example, when `'dwt'` is the current `ObjectStructureType`, the command `setwb('conver')` returns

```
List of ConvolutionVersions
ZES = Zero-Extended Signal
SES = Symmetrically-Extended Signal
LES = Linearly-Extended Signal
CPS = Circularly-Periodized Signal
CPF = Circularly-Periodized Filter
BAF = Boundary-Adjusted Filter
which are valid for current
ObjectStructureType = DWT
```

and thus also demonstrates an example of the artificial intelligence found in WavBox. This expert system applies to all properties which control other properties (see below). Thus, to display all possible values, use the command `setwb('all')` which returns lists of currently valid choices of property values given other current property values (example output not shown due to length).

The command just described, `setwb('all')`, first displays the list of property names and then the lists of possible values. To display just the list of property names, use the command `setwb('names')` to obtain

```
List of WavBox Object Property Names
DATDIM = DataDimension
DATSIZ = DataStructureSize
MAPTYP = MappingStructureType
MAPCLS = MappingStructureClass
MAPSIZ = MappingStructureSize
OBJTYP = ObjectStructureType
OBJCLS = ObjectStructureClass
OBJSIZ = ObjectStructureSize
FILCLS = FilterClass
FILFAM = FilterFamily
```

```
ANAPAR = AnalysisFilterParameter
SYNPAR = SynthesisFilterParameter
CONVER = ConvolutionVersion
DESLEV = DesiredLevel
MAXLEV = MaximumLevel
YSCLEN = YScaleLengths
XSCLEN = XScaleLengths
FILNAM = FilterName
FILLEN = FilterLength
ANAFIL = AnalysisFilters
SYNFIL = SynthesisFilters
LENDLO = LeftEndLowpassFilters
RENDLO = RightEndLowpassFilters
LENDHI = LeftEndHighpassFilters
RENDHI = RightEndHighpassFilters
```

Usually, the user sets only some of the properties in this list and WavBox automatically sets the rest (see below). However, advanced users may set all properties. These features are documented in detail in the help notes for `setwb`.

Current values for WavBox object properties may be changed anytime during a work session using `setwb` independently of the permanent defaults found in `defwb`. However, these permanent defaults may be changed by using a text editor to modify the m-file `defwb` itself. Actually, `setwb` calls `lrwb`. This latter function lists the valid property values and resets some of those that are invalid to valid ones. Therefore `setwb` (via `lrwb`) should correct some invalid values mistakenly passed to it as arguments as well as some invalid defaults mistakenly edited into `defwb`. However, the artificial intelligence expert system in `lrwb` is limited at the present time, and therefore some incompatible property values are detected by other error-trapping checks in other WavBox functions.

It is only necessary to set those property values as listed in the defaults found in `defwb` and WavBox automatically sets the rest. For example, the user sets the `'deslev'` or `DesiredLevel` property and WavBox automatically sets the `'maxlev'` or `MaximumLevel` property. If `getwb('maxlev')` returns zero, then the current combination of properties is invalid. Valid or invalid combinations are defined by whether or not they are capable of yielding perfect reconstruction. Additional tutorials on `setwb` and `getwb` can be found in the demonstration scripts for the various transforms and decompositions run by clicking on the pushbuttons in the Demo WavBox figure window started by the command `demowb`.

4 Summary of Key Features

Transforms and decompositions in WavBox are known as WavBox multiresolution objects. WavBox 4.1c includes overscaled pyramid transforms, 1-D and 2-D discrete wavelet transforms, 1-D and 2-D wavelet packet transforms with decompositions by best level, best basis, and near-best basis, and similarly for 1-D cosine packet transforms with analogous decompositions. In addition, WavBox 4.1c includes orthogonal and nonorthogonal matching pursuit decompositions for 1-D wavelet and cosine packets. Consult [3] for references on the various methods. The algorithms are valid for a wide variety of convolution versions including extended, periodized, symmetrized, and boundary adjusted, and a wide variety of filters including all of the well-known classes of Daubechies'

orthogonal and biorthogonal families. The algorithms are also valid for essentially arbitrary data sizes. In particular, for 2-D analyses, if `[nrows,ncols] = size(X)`, then `nrows = p*2^nlevels` and `ncols = q*2^nlevels` where `p` and `q` are integers and `nlevels` is the number of scale levels or depth of the analysis. For 1-D analyses, `nrows` and `ncols` can be any arbitrary integers but not all convolution versions may be used.

The command line interface (CLI) provides access to all forward maps performing multiresolution analyses (transforms and decompositions) with the command `y = objtyp(x)` and to all inverse maps performing multiresolution syntheses (inverse transforms and reconstructions) with the command `z = iobjtyp(y)` where `objtyp` is the ObjectStructureType string and `iobjtyp` is the letter `i` (for inverse) concatenated with the ObjectStructureType string. For example, `wpdp` and `iwpdp` are the forward and inverse wavelet packet decompositions by matching pursuit.[4] No additional arguments, other than the data argument, are required by the syntax of the commands. All parameters are implemented as object properties manipulated with the utilities `setwb` and `getwb` and defined with complete sets of defaults in `defwb` as explained in the previous section on WavBox objects and properties. The CLI also provides access to all plots of multiresolution objects with the command `plotobjcls(y)` where the name `plotobjcls` is the concatenation of the word `plot` with the ObjectStructureClass string. For example, `plotdpl` displays discrete packet lists in the time-frequency plane. In this example, the ObjectStructureClass `dpl` has several members such as the ObjectStructureTypes `wpdp`, `cpdp`, `wpdb`, `cpdb`, etc. Again, no additional arguments are required. However, various optional arguments are available and are all provided with defaults. For example, `plotdpl` has nine optional arguments which control the time-frequency distribution type (energy, magnitude, or amplitude), the plot type (image, patch, or surface), the plot title, the quantile levels displayed, the plot location, the color map and color limits, the grid size (for plot types image and surface), and the azimuth and elevation (for plot type surface).

The graphical user interface (GUI) provides pushbutton, popupmenu, and editbox access to comprehensive exploratory data analysis with WavBox transforms and decompositions. Section 3 describes the system of WavBox objects and properties which underly both the CLI and the GUI. These two interfaces are integrated to the core objects and properties in a direct manner. The function `lrwb`, which lists possible property values and resets invalid to valid ones, returns output to the command window of the CLI in the form of printed information and to the interactive control panels of the GUI in the form of popupmenus updated to reflect currently valid choices of properties. Other controls also display current property values. Thus both the CLI and GUI provide simultaneous access to property values. The command `dispwb` opens the WavBox Master Control Panel (see below) from which other multiplot GUI display panels can then be opened as desired. These auxiliary GUI display panels can also be called directly from the CLI such as with the command `dispdpt(y)` where `y` is a variable of ObjectStructureClass `dpt` which includes ObjectStructureTypes `wpt` and `cpt`. Commands for all interactive multiplot GUI displays begin with the letters `disp`.

Other than the system of objects with properties, the architectural design of WavBox is also based on several key principles. The use of heirarchical modules and switch-driven function suites are critical elements of a design providing readability and comprehensibility of code to the end-user, thereby enabling the user to modify the code as desired. Most m-files are small modules which are

[4] This pair could have been called wavelet packet pursuit decomposition (wppd) and wavelet packet pursuit reconstruction (wppr) but then this choice of names would not have conformed with the inverse and other naming conventions in WavBox.

called in a heirarchical manner. For example, `dwt2` (discrete wavelet transform 2-D) calls `dntetrad` (down tetrad) which calls `dnscal` (down scale) which depending on the ConvolutionVersion property then calls other functions such as `circonv` (circular convolution). Some key modules also have options for verbose display output which further enhances ease of learning the algorithms. In addition, many modules (ie, m-files) are actually suites of functions (ie, mathematical functions). For these modules, a particular mathematical function is selected by a switch which is usually a name (ie, a character string) for the case selected. Executing the m-file without any inputs returns a list of possible switch cases. For a simple example, see the help output below from the m-file `esterror`. Other examples include `estthrsh` with various methods of estimating threshold parameters and `thrshold` with various methods of thresholding data. However, this design approach is purposefully not used in two important areas, information cost measures and logical tests, for which all the various mathematical procedures are maintained as separate m-files rather than grouped as function suites in a single m-file. Consult the `contents` file in the Appendix A for a complete listing of WavBox m-files. Finally, another important design principle is the implementation of most filter operators in two alternative ways: either 1) to filter the input data efficiently to yield the output data, or 2) to return an appropriately sized matrix operator which can then be examined or used as desired. This design principle applies to all operators fundamental to wavelet analysis including the `dnsamp`, `dnscal`, `upsamp`, and `upscal` operators. For an example, see the help output below from the m-file `cirshift`.

Benchmark test results are an important measure of the performance of any software toolbox. Since WavBox does *not* contain any mex-files or object code of any kind, keep in mind that the results presented in Table 1 were obtained with WavBox 4.1c m-files, all of which are ASCII-text readable source-code files written in MATLAB, and run on an ALR Evolution V machine with an Intel Pentium CPU at 60 MHz clock rate and MATLAB 4.2b for Windows 3.1. In this table, the

Table 1: Benchmarks for `dwt` and `dwt2`.

`datdim`	`datsiz`	flops	time	error
2	272 x 368	4.577e+006	11.86	1.083e-010
2	256 x 256	3.006e+006	7.64	1.253e-010
1	256 x 256	2.127e+006	4.78	8.509e-011
1	256 x 16	1.374e+005	0.61	8.719e-011
1	256 x 1	1.310e+004	0.38	9.304e-011

numbers tabulated for flops and time in seconds represent *total* amounts for *both* the forward and inverse directions of the transform pairs (dwt and idwt, or dwt2 and idwt2), while error represents the error of reconstruction obtained as `Y = dwt(X); A = idwt(Y); e1 = esterror(X,A);` or as `Y = dwt2(X); A = idwt2(Y); e2 = esterror(X,A);`. Also in the table, the WavBox object properties `datdim` and `datsiz` are DataDimension and DataSize (see Section 3). Of course, the value of `datdim` is 1 for dwt and idwt, and 2 for dwt2 and idwt2. All of the benchmark tests listed in this table were run with the Daubechies' orthogonal least asymmetric filters of length 4 with boundary-adjusted edge filters (interval wavelets) to a transform depth of 4 levels.

One-dimensional analysis with `dwt` and `idwt` on matrices may at first seem strange but it can be easily interpreted as either multi-channel or multi-segment analysis as described in Section 1.

For example, the case with `datsiz = [256,16]` could be interpreted as 16 channels each with 256 time points, whereas the case with `datsiz = [256,256]` could be interpreted as 1 channel of 65536 time points analyzed in parallel as 256 segments each with 256 time points. In a one-dimensional analysis of a matrix, simple 1-D analysis is applied in parallel to each signal channel/segment (ie, each column of the matrix). In contrast, in a two-dimensional analysis of a matrix, separable 2-D analysis is applied to the entire image (ie, all rows and columns of the matrix).

5 Documentation

Each m-file has online help (see the examples below). The m-file `wbman` creates a WavBox manual concatenating the help notes from all of the m-files. In addition, there are self-running tutorial demo scripts on each transform and decomposition as well as on filters, operators, and WavBox objects and properties. These scripts demonstrate many of the features of WavBox. Finally, references for algorithms are cited in the on-line help along with complete BibTeX listings in the text file `referenc.txt`.

Help output from the m-file `esterror` demonstrates a function suite with various kinds of estimate error selected by the switch `typ` which is an optional argument with default value `'mix'`.[5]

```
esterror: ESTimate ERROR.
**********************************************************************@
Inputs:
   par, data matrix #1 considered true PARameter;
   est, data matrix #2 considered observed ESTimate;
   typ, TYPe of error to be computed;
Outputs:
   err, error of estimate;
   list, LIST of type names;
   alen, Abbreviation LENgth for each type name;
Usage:
   err = esterror(par,est);
   err = esterror(par,est,typ);
   [list,alen] = esterror;
Defaults:
   typ = 'MIX';
Notes:
   valid for vectors and matrices; performs dimension check;
   returns err = NaN for mismatched dimensions of par and est;
   'ABS' is ABSolute error: norm(par-est,'fro');
   'REL' is RELative error: norm(par-est,'fro')/norm(par,'fro');
   'MIX' is MIXed error: norm(par-est,'fro')/(1+norm(par,'fro'));
   'SSQ' is Sum of SQuares error: norm(par-est,'fro')^2;
   'MSQ' is Mean SQuares error: norm(par-est,'fro')^2/(nrows*ncols);
   'MAV' is Maximum Absolute Value error: max(max(abs(par-est)));
   'LPN' is L^P vector Norms with P = [1,2,inf];
   all error types return a scalar value except 'LPN' which
      returns a 3-vector: [norm(y,1),norm(y,2),norm(y,inf)]
      where y = x(:) and x = par - est;
Copyright (c) 1991-94 Carl Taswell.
```

[5] Switches are not case sensitive.

```
All rights reserved. Created 8/15/91, last modified 8/3/94.
**********************************************************************@
```

Help output from the m-file `cirshift` demonstrates implementation of a basic filter as a function that either 1) actually filters data efficiently without using matrices or 2) returns an operator matrix that represents the filtering operation and thus that could be used to filter the data:

```
cirshift: CIRcular SHIFT.
**********************************************************************@
Inputs:
   X, signal data matrix;
   n, signal length scalar;
   k, shift length scalar;
Outputs:
   Y, circularly shifted signal data matrix;
   C, Circular shifting matrix operator;
Usage:
   Y = cirshift(X);
   Y = cirshift(X,k);
   C = cirshift(n);
   C = cirshift(n,k);
Defaults:
   k = 1;
Notes:
   if first input is matrix X, then output is matrix Y;
   if first input is scalar n, then output is matrix C;
   if input signal X is vector, then it must be a column vector;
   if input signal X is matrix, then channels must be in columns;
   if [n,m] = size(X), then X must have n >= 2, m >= 1 where n is
      the number of time points and m is the number of channels;
   for time delay, use positive k (causal-like);
   for time advance, use negative k (anticausal-like);
References:
   Oppenheim & Schafer, pg:536;
Copyright (c) 1992-94 Carl Taswell.
All rights reserved. Created 5/8/92, last modified 2/26/94.
**********************************************************************@
```

6 Discussion

The design of any software toolbox invariably requires making choices involving trade-offs between different kinds of implementations. These choices must be made at the levels of individual modules, algorithms, architectures, and interfaces. Even the issue of how to name modules involves trade-offs. Long names with mixed upper and lower case characters along with underscore or other characters may be easy to read but not necessarily easy to type at the CLI or to port across different computing platforms. The design of WavBox has imposed the requirement of names restricted to a maximum of 8 alphanumeric (letters and digits only) characters all in lower case (see Section A).

Within a module, an implementation that uses the same variable `x` on both sides of the assignment operator is memory-efficient (because it uses a method that overwrites the same memory space) but is arguably not as easy to read and learn as a module that uses different variables `y` and

`x` on either side of the assignment operator. Ideally, algorithms could be written to call modules written both ways, one for efficient computational purposes and the other for instructional purposes. The design of WavBox at the module level has emphasized clarity of code and efficiency of computation in terms of flop counts but not necessarily memory space. A future version of WavBox will correct any remaining inefficiencies in memory use where critically important.

At the algorithm level, an implementation that uses calls to heirarchical modules and switch-driven function suites presumably must pay the penalty of slower execution speed relative to that of in-line code. However, for WavBox, this disadvantage appears to be negligible in MATLAB (see benchmarks in Table 1 and Section 4). The major advantage of the heirarchical and switch-driven design is that modules and functions within modules[6] can be studied individually and easily changed or extended to include other cases. This approach is consistent with the essential advantage of MATLAB as a language for rapid prototyping as opposed to a language such as C or FORTRAN presumably for optimized execution speed.

Finally, at the level of the overall architecture of and interfaces to the algorithms, the design trade-offs associated with the WavBox CLI and GUI interfaces and the underlying object property expert system are even more complex but include as always convenience, comprehensibility, capability, extensibility, memory requirement, and execution speed. Of significance in WavBox, however, is the expert system designed to provide as much convenience as possible to the user when used properly.[7] In particular, it *eliminates the need for the user to constantly worry about inputting parameter arguments* (such as filter, convolution version, and transform depth), *their number and order, and whether they are compatible* (ie, permit perfect reconstruction following decomposition) *with each other and for the given size of the data*. Nevertheless, some users may not like to use this expert system despite the ability to control all parameters with the `setwb` utility and then analyze data with commands such as `y = dwt(x)`. Instead, they may prefer to input the parameters simultaneously with the data directly to the transform. A future version of WavBox will allow both possibilities. However, as transforms, filters, and convolution versions multiply, use of an expert system will become increasingly important to the user who requires the information and convenience that it provides.

It is well known that toolboxes and their "personalities" characterized principally by their CLI and GUI interfaces, like languages and operating systems, acquire both ardent fans and equally ardent foes. Constructive criticisms from both fans and foes will enable WavBox to be further developed into a better software toolbox for all its users.

A WavBox Contents

This appendix contains help output from the `contents` file in WavBox; Unlike other command window output presented in this paper, the text is not displayed in verbatim typewriter font in order to save space.

WavBox 4.1 (Wavelet Toolbox for MATLAB 4.2) Version 4.1c 31-Aug-94 Copyright (c) 1993-94 Carl Taswell; All rights reserved.

Data structure convertors, modifiers, and generators: bst2bsl: Block Selection Tree to Block Selection List; dpb2dpt: Discrete Packet Basis to Discrete Packet Table; dpl2bst: Discrete Packet

[6] Here "module" refers to a MATLAB m-file and "function" refers to a mathematical function; a switch-driven function suite is an m-file module containing multiple mathematical functions selected by a switch; see Section 4.

[7] Space does not permit explanation of all of the features of the WavBox object property expert system.

List to Block Selection Tree; dpl2dpt: Discrete Packet List to Discrete Packet Table; dpt2bst: Discrete Packet Table to Basis Selection Tree; dpt2dpb: Discrete Packet Table to Discrete Packet Basis; dpt2dpl: Discrete Packet Table to Discrete Packet List; dpt2ict: Discrete Packet Table to Information Cost Tree; dwt2dpl: Discrete Wavelet Table to Discrete Packet List; ict2bst: Information Cost Tree to Basis Selection Tree; pruntree: PRUNe TREE; randdpl: RANDomly generate Discrete Packet List; sortdpl: SORT Discrete Packet List;

Data structure sizes, indices, and substructures: ailbc: Array Indices for Level-l Block-b Cell-c. clenwbms: Coordinate LENgths of WavBox Mapping Structure. hashailb: HASH table for Array Indices for Level-l Block-b. iailbc: Inverse of Array Indices for Level-l Block-b Cell-c. lbchldrn: Level Blocks' CHiLDReN blocks. lbfamily: Level Blocks' FAMILY tree. lbparent: Level Blocks' PARENT block. nlbcells: Number of Level Block CELLS. nlblocks: Number of Level BLOCKS. nlevels: Number of LEVELS. ntblocks: Number of Table/Tree BLOCKS. randlbc: RANDomly generate Level-l Block-b Cell-c indices. sizewbms: SIZE of WavBox Mapping Structure. tilb: Tree Index for Level-l Block-b. trunclen: TRUNCate LENgth. vilc: Vector Indices for Level-l Cell-c.

Demonstrations and interactive GUI displays: demcpdp: DEMo Cosine Packet Decomposition (matching Pursuit). demcpt: DEMo Cosine Packet Transform. demdwt: DEMo Discrete Wavelet Transform. demdwt2: DEMo Discrete Wavelet Transform 2-dimensional. demfcdo: DEMo Filter Coefs Daubechies Orthogonal wavelets. demintro: DEMo INTROduction to wavbox object properties. demopt: DEMo Overscaled Pyramid Transform. demowb: DEMO WavBox control panel. demscal: DEMo SCALing operators. demwpdp: DEMo Wavelet Packet Decomposition (matching Pursuit). demwpt: DEMo Wavelet Packet Transform. demwpt2: DEMo Wavelet Packet Transform 2-dimensional. dispdpt: DISPlay Discrete Packet Transform. dispisr: DISPlay Image and Selected Region. dispsss: DISPlay Signal and Selected Segment. dispwb: DISPlay WavBox master control panel. wbmcpl: WavBox Master Control Panel Layout. wbmcpu: WavBox Master Control Panel Uicontrols.

Denoising: estthrsh: ESTimate THReSHold parameter. expideal: EXPeriment for "IDEAL ..." article. fdenoise: Fourier-shrinkage DENOISE. fidenois: Fourier-shrinkage Ideal DENOISe. figadapt: FIGures for "ADAPTing ..." article. figideal: FIGures for "IDEAL ..." article. idealwgt: IDEAL WeiGhTs; thrshold: THReSHOLD data with threshold parameter. wdenoise: Wavelet-shrinkage DENOISE. widenois: Wavelet-shrinkage Ideal DENOISe.

Filter coefficients: fcbbbs: Filter Coefs Bradley-Brislawn Biorthogonal Symmetric. fcdbss: Filter Coefs Daubechies Biorthogonal Symmetric Spline. fcddns: Filter Coefficients Deslauriers-Dubuc Nonorthogonal Symmetric. fcdola: Filter Coefs Daubechies Orthogonal Least Asymmetric. fcdomp: Filter Coefs Daubechies Orthogonal Minimum Phase. fcdovm: Filter Coefs Daubechies Orthogonal Vanishing Moments. fcvhbs: Filter Coefs Vetterli-Herley Biorthogonal Symmetric. fcwssb: Filter Coefs Wickerhauser Symmetric Sine Bell. filtwbms: FILTers for WavBox Mapping Structure. typeprfb: TYPE Perfect Reconstruction Filter Banks.

Generic utilities: ctoff: Counter Timer OFF. cton: Counter Timer ON. findmstr: FIND Matrix STRing. esterror: ESTimate ERROR.

Information cost functions, norms, statistics, and other measures: dcarea: Data Compression AREA. dcnum: Data Compression NUMber. energy: ENERGY. entropy: ENTROPY. holdereg: HOLDEr REGularity. logenrgy: LOG ENeRGY. lpnorm: L-P vector NORM. quantile: QUANTILE values. tenum: Threshold Excedance NUMber. wlpnorm: Weak L-P vector NORM.

Logical tests: isbior: IS BIORthogonal matrix or filter pair. iseye: IS identity (EYE) matrix. iskron: IS KRONiker delta vector. isnorm: IS NORM function. isorth: IS ORTHogonal matrix

or filter. isperf: IS PERFect matrix set. isprfb: IS Perfect Reconstruction Filter Bank. isproj: IS PROJection matrix. isuniq: IS UNIQue matrix pair. isunit: IS UNITary matrix.

Operators for filters: cascade: iterative CASCADE auto-convolution of filter. mirror: MIRROR filter. paraconj: PARACONJugate filter. prfbank: Perfect Reconstruction Filter BANK. qmf: Quadrature Mirror Filter. wavconv: WAVeform CONVolution for scaling/wavelet filter plots.

Operators for signals: circonv: CIRcular CONVolve. cirshift: CIRcular SHIFT. dndyad: DowN DYAD. dnsamp: DowN SAMPle. dnscal: DowN SCALe. dntetrad: DowN TETRAD. extend: EXTEND signal. imidfold: Inverse of MIDdle FOLD. midfold: MIDdle FOLD. updyad: UP DYAD. upsamp: UP SAMPle. upscal: UP SCALe. uptetrad: UP TETRAD.

Plots: plotddns: PLOT Deslauriers-Dubuc Nonorthogonal Symmetric wavelet. plotdpl: PLOT Discrete Packet List in time-frequency plane. plotdpt: PLOT Discrete Packet Table. plotdpt2: PLOT Discrete Packet Table 2-dimensional. plotdwt: PLOT Discrete Wavelet Table. plotdwt2: PLOT Discrete Wavelet Table 2-dimensional. plotfreq: PLOT FREQuency response. plotiee: PLOT Image and Estimate with Error. plotlbcp: PLOT Level-l Block-b Cell-c Packet waveform. plotmpdi: PLOT Matching Pursuit Decomposition Iterations. plotsee: PLOT Signal and Estimate Error. plotspik: PLOT discrete sequence as SPIKes. plottree: PLOT TREE for discrete packet table. plotwfr: PLOT Wavelet Frequency Response. plotwir: PLOT Wavelet Impulse Response or its holder regularity.

Plot and GUI display utilities: axislims: AXIS LIMitS. colbar: COLor BAR. locplot: LOCate PLOT. makefig: MAKE FIGure. menustr: MENU STRing;

Signal generators and modifiers: addnoise: ADD white NOISE. quantize: QUANTIZE floating point numbers. scaleval: SCALE amplitude or intensity VALue. stsignal: Standard Test SIGNALs.

Transforms, decompositions, and auxiliary functions: cpdb: Cosine Packet Decomposition (best Basis). cpdl: Cosine Packet Decomposition (best Level). cpdp: Cosine Packet Decomposition (matching Pursuit). cpiipt: Cosine Packet Impulse Inner Product Table. cpt: Cosine Packet Transform. dctiv: Discrete Cosine Transform type IV. dwt: Discrete Wavelet Transform. dwt2: Discrete Wavelet Transform 2-dimensional. icpdb: Inverse Cosine Packet Decomposition (best Basis). icpdl: Inverse Cosine Packet Decomposition (best Level). icpdp: Inverse Cosine Packet Decomposition (matching Pursuit). icpt: Inverse Cosine Packet Transform. idwt: Inverse Discrete Wavelet Transform. idwt2: Inverse Discrete Wavelet Transform 2-dimensional. iopt: Inverse Overscaled Pyramid Transform. iwpdb: Inverse Wavelet Packet Decomposition (best Basis). iwpdl: Inverse Wavelet Packet Decomposition (best Level). iwpdp: Inverse Wavelet Packet Decomposition (matching Pursuit). iwpt: Inverse Wavelet Packet Transform. iwpt2: Inverse Wavelet Packet Transform 2-dimensional. mpd: Matching Pursuit Decomposition. opt: Overscaled Pyramid Transform. wpdb: Wavelet Packet Decomposition (best Basis). wpdl: Wavelet Packet Decomposition (best Level). wpdp: Wavelet Packet Decomposition (matching Pursuit). wpiipt: Wavelet Packet Impulse Inner Product Table. wpt: Wavelet Packet Transform. wpt2: Wavelet Packet Transform 2-dimensional.

WavBox object property utilities: argchkwb: ARGument CHecKs for WavBox mappings. chkwb: CHecK WavBox object properties. defwb: DEFine DEFaults for WavBox object properties. getwb: GET WavBox object properties. ipvspv: Inverse of Pack Variable Size Property Value. lrwb: List and Reset WavBox object properties. pvspv: Pack Variable Size Property Value. setwb: SET WavBox object properties.

WavBox general utilities: contents: this CONTENTS file. excerpts: EXCERPTS from demos and help output (a *.txt file). license: LICENSE information (a *.tex file). README: release up-

date and general information (a *.txt file). referenc: REFERENCes for wavbox algorithms (a *.bib file). syspecs: SYstem SPECificationS. wavbox: start WAVBOX with screen-dependent defaults. wbfigs: list WavBox FIGureS. wbman: create WavBox MANual of m-file help ('wbmanual.txt' file).

All files are executable MATLAB m-files except those marked *.txt, *.bib, or *.tex files which can be read with any ASCII text editor. MATLAB m-files are also ASCII-text readable source-code files. There are neither system-dependent mex-files nor object code of any kind. WavBox 4.1c is distributed as a single directory of 179 files which when uncompressed requires 633 kilobytes of disk space.

References

[1] The MathWorks, Inc., Natick, MA. *MATLAB Reference Guide*, August 1992.

[2] Carl Taswell. MATLAB wavelet toolbox. *Wavelet Digest*, 2(11), July 1993. Topic #7.

[3] Carl Taswell. Top-down and bottom-up tree search algorithms for selecting bases in wavelet packet transforms. In Anestis Antoniadis and Georges Oppenheim, editors, *Wavelets and Statistics*, Lecture Notes in Statistics, pages ???–??? Springer Verlag, 1995. Proceedings of the Villard de Lans Conference November 1994.

[4] Carl Taswell. WavBox 4: A software toolbox for wavelet transforms and adaptive wavelet packet decompositions. In Anestis Antoniadis and Georges Oppenheim, editors, *Wavelets and Statistics*, Lecture Notes in Statistics, pages ???–??? Springer Verlag, 1995. Proceedings of the Villard de Lans Conference November 1994.

[5] Carl Taswell and Kevin C. McGill. Wavelet transform algorithms for finite-duration discrete-time signals: Signal-end effects. Technical Report NA-91-07, Computer Science Department, Stanford University, Stanford, CA, November 1991.

[6] Carl Taswell and Kevin C. McGill. Algorithm 735: Wavelet transform algorithms for finite-duration discrete-time signals. *ACM Transactions on Mathematical Software*, 20(3):398–412, September 1994.