The *wtrans1d* package

Managing 1-dimensional wavelet transforms

and

The *extrema1d* package

Managing wavelet transform extrema representation

Emmanuel Bacry
CMAP, Ecole polytechnique, 91128 Palaiseau Cedex, France
email : lastwave@cmap.polytechnique.fr
web : http://www.cmap.polytechnique.fr/˜bacry/LastWave

The *wtrans1d* and *extrema1d* packages was co-written by

- **Benjamin Audit** *Computational Genomics Group, EMBL-European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK.*
  e-mail: audit@ebi.ac.uk
  web: http://www.ebi.ac.uk/
- **Emmanuel Bacry** *CMAP, Ecole Polytechnique, 91128 Palaiseau Cedex France.*
- **Nicolas Decoster** *CRPP Avenue Schweitzer, 33600 Pessac France.*
- **Stéphane Mallat** *CMAP, Ecole Polytechnique, 91128 Palaiseau Cedex France*
- **Jean François Muzy**, *CNRS, Université de Corse, Quartier Grossetti, 20250, Corte, France.*
  email : muzy@univ-corse.fr
- **Cédric Vaillant**, *CRPP, Avenue Schweitzer, 33600 Pessac, France.*

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Version 2, June 1991

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.
4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.
If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.
It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail

to address new problems or concerns.
Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Contents

Part I

Documentation

Chapter 1

Using the wavelet transform (`wtrans1d`) package

1.1 Loading the `wtrans1d` package

As for any package, you must load the `wtrans1d` package for using 1d-wavelet transforms. This is done by the original startup file or directly by typing `package load wtrans1d` in the terminal.

1.2 The wavelet transform structure in LastWave

1.2.1 Introduction

In LastWave, a wavelet transform corresponds to a structure that is able to store a continuous or a (bi)orthogonal wavelet transform of a sampled signal. It mainly consists in 2 two-dimensional arrays of signals.

- The `A` array. It corresponds to the approximation signals (which represent the convolution of the original signal with the smoothing function),
- The `D` array. It corresponds to the detail signals (which represent the wavelet transform).

A wavelet transform is performed on a given number of octaves `noct` using a given number of voices `nvoice` per octave. The voice number `v` of the octave number `o` corresponds to the scale `a` given by

$$a = 2^{noct} 2^{v/nvoice} \tag{1.1}$$

where `v` varies from 0 to `nvoice-1` and `o` varies from 1 to `noct`. Both `noct` and `nvoice` are fields of a wavelet transform structure in LastWave. They are both stored in the structure when you perform a wavelet transform.

The first parameter of the two-dimensional arrays `A` and `D` corresponds to the voice number and the second one corresponds to the octave number. Thus, the signal `D[o,v]` corresponds to the wavelet transform at octave `o` and voice `v` whereas the signal `A[o,v]` corresponds to the approximation at octave `o` and scale `v`. The original signal the wavelet transform is performed on is the signal `A[0,0]`.

Let us note that in the case of a (bi)orthogonal wavelet transform the parameter `nvoice` is always equal to 1 and both arrays `A` and `D` are used whereas in the case of a continuous wavelet transform `nvoice` is given by the user and only the `D` array is used. Thus, for instance, in the case of an orthogonal wavelet transform, the signal `D[1,0]` corresponds to the smallest scale of the wavelet transform whereas `A[2,0]` corresponds to the approximation at scale 4 and the signals `D[i,j]` and `A[i,j]` with `j` different from 0 are not used. On the other hand, in the case of a continous wavelet transform, the signals `D[i,j]` are used to store the wavelet coefficients for `i>0` and `0<j<nvoice` and the `A` array is not used. In any case, the signals `A[0,i]` for `i>0` and `D[0,j]` for `j` $\geq$ `0` are never used by any wavelet transforms. They can be used by the user for convenience.

1.2.2 The main fields of `&wtrans`

The main fields are

- `noct` : the number of octaves
- `nvoice` : the number of voices
- `amin`: the first scale
- `A[o,v]`: the approximation signal at octave `o` and voice `v`,
- `D[o,v]`: the detail signal at octave `o` and voice `v`,
- `name`: the name of the wavelet transform
- `size` : the `size` field of the original signal
- `x0` : the `x0` field of the original signal
- `dx` : the `dx` field of the original signal
- `wavelet` : the name of the analyzing wavelet that have been used
- `extrep` (read-only) : the `&extrep` variable it corresponds to (`null` if none) (see Chapter 2)

1.3 A simple continuous wavelet transform

In order to assign a variable to a wavelet transform, one should use the `new` command along with the `&wtrans` type

```
a> w = [new &wtrans]
= <&wtrans;0x208cf138>
a> type(w)
= '&wtrans'
```

Let us note that (as explained in the main documentation of LastWave) the variable `a` corresponds to the current object which is a `&wtrans` variable. So instead of creating a new variable `w` you could use `a` as well. It assigns the variable named `w` with a new structure that is able to store a wavelet transform.Such a variable is of type `&wtrans`.

In order to refer to the signal `A[<o>,<v>]` one can either use the "regular" syntax

w.A[<o>,<v>]

or the "abreviated syntax"

<o><v>w

Thus for instance the signal `A[0,0]` of `w` (i.e., the original signal the wavelet transform will be performed on) is accessed either by the syntax `w.A[0,0]` or by the syntax `0w` whereas the signal `A[3,4]` can be accessed using the syntax `34w`. The detail signals (i.e., the signals of `D`) are accessed in the same way except that i) for the "regular" synatx `A` is replaced by `A` and ii) for the "abreviated" syntax you must add a dot. Thus `D[3,4]` corresponds to `w.D[3,4]` or equivalently to `.34w`. Thus, for instance, let us suppose that we want to perform (in `w`) the continuous wavelet transform of a dirac function. The dirac must be put in the signal `A[0,0]`. A discrete dirac signal of size 512 can be assigned to `A[0,0]` in the following way :

```
a> 0w = I(512)==256
= <size=512;0,0,0,0,0,0,...>
```

Remark : In the Section 1.6 you will learn about what the "current wavelet transform" is. That will allow you to avoid reminding LastWave that you want to work on `w`. Thus you will be able to ommit all the references to `w` in the last command (i.e., typing 0 instead of `0w`) and in each command that follows.

Then we just need to call the continuous wavelet transform decomposition command `cwtd`

```
a> cwtd w 1 3 6 'g2'
```

which computes, starting from scale 1, the continuous wavelet transform of the signal `A[0,0]` in `w` using 3 octaves, 6 voices per each octave and the second-order Mexican hat `g2` analyzing wavelet. If you want to display the 8th scale of the decomposition (i.e., the signal `D[3,0]`), you just need to do

```
a> disp .30w
```

Actually you can use the command `disp` to display an image which represents the wavelet transform we have just computed

```
a> disp w
```

Now you can use the `disp` command in the same way as we did for signals in the last section. You can even combine signals and wavelet transforms. For instance to display the signal above the wavelet transform you just type

```
a> disp 0w w
```

Remark : Let us note that the `morlet` complex wavelet can be used in `cwtd`. The modulus of the wavelet coefficients will be stored in the `D` signals and the phase (between 0 and $2pi$) in the `A` signals.

1.4 Display and mouse interaction

To learn about display and mouse interaction, you should run the Demos of the `wtrans1d` package. You can use the mouse within the graph that displays the wavelet

transform. Dragging the left mouse button allow to perform zooms (if you hit the `z` key you change zoom mode). The middle mouse button zooms out the wavelet transform. Use the `c` key to get a cross-hair cursor. Moreover, you can use the mouse buttons along with the `shift` key to get horizontal (left mouse button) sections or vertical sections (middle mouse button for linear scale and right mouse button for logarithm scale) of the wavelet transform. These sections will be drawn in the "last" window of type `&signali` (read the `disp` manual pages of the main LastWave manual). All these behavior were defined in the script file `scripts/wtrans1d/wtrans1d.pkg`. You can modify them or create more complex actions that you can perform using the mouse. In order to do so, you should read the sections *Managing events* and *More about managing events* of the main LastWave manual.

1.4.1 The `GraphWtrans` graphic object

A `wtrans` structure is displayed using a `GraphWtrans` graphic object. The fields of that graphic class are

- `cm` : the colormap that is used to display the wavelet transform,
- `causal` : a flag (0 or 1) which indicates whether the wavelet transform coefficients affected by border effects should be displayed or not,
- `norm` : a (string) field that indicates how the coding should be done. Its value could be either
  - `'global'` : The colors are coded from the global minimum of the absolute value and the global maximum of the wavelet corfficients,
  - `'local'` : Same as `'global'` but the maximum and minimum are computed only on the portion of the image which is displayed (if the full image is displayed, the two modes are equivalent).
  - `'lglobal'` : The colors are coded separately at each scale from the global minimum of the absolute value (at each scale) and the global maximum of the wavelet coefficients,

  - `'llocal'` : Same as `'lglobal'` but the maximum and minimum are computed only on the portion of the image which is displayed (if the full image is displayed, the two modes are equivalent).

- `graph` : which allows to set/get the `wtrans` structure that will be displayed.

1.5 A simple orthogonal wavelet transform

Let's now compute the orthogonal wavelet decomposition (on 5 octaves) of the same dirac function using the *Daubechies 3* analyzing wavelet. This is done in the following way :

```
a> owtf 'D3.o'
a> owtd w 5
```

Remark : The directory were the filters are must be stored in the LastWave variable `Wtrans1dFilterDirectory` (look in the file `scripts/wtrans1d/wtrans1d.pkg` where it is set).

You can look at the coefficients (`D[1,0]`, `D[2,0]`, `D[3,0]`, `D[4,0]` and `D[5,0]`) using the `disp` command :

```
a> disp .10w .20w .30w .40w .50w
```

Then, you can reconstruct the original signal using the `owtr` command

```
a> owtr w 1w
```

The reconstructed signal is in the signal referred to as `1w` (i.e., `A[0,1]`), and you can display it using

```
a> disp 1w
```

You could also display an image corresponding to the wavelet transform by just typing

```
a> disp w
```

WARNING : The code number we gave to refer to the signals in `A` and `D` does not allow you to access the voice numbers with 2 digits. Indeed, the code `213w` will be understood as the octave number 21 and the voice number 3 and not the octave number 2 and the voice number 13. In order to do so you should use the "regular' syntax `w.D[2,13]`. Let us

threshold (using a threshold value of 0.2) all the wavelet coefficients of the noised dirac we computed just before, one could do

```
a> .10w = .10w*(abs(.10w)>.2)
```

and the same things for all the other scales (you could write a simple loop!)

```
a> .20w = .20w*(abs(.20w)>.2)
a> .30w = .30w*(abs(.30w)>.2)
a> .40w = .40w*(abs(.40w)>.2)
a> .50w = .50w*(abs(.50w)>.2)
```

then we can reconstruct and display the result

```
a> ocwtr w 1w
a> disp 1w
```

Remark : The command `wthresh` is a powerful C-command that performs thresholding of wavelet transforms. If you are not satisfied with it, it is very easy to write your own one using the script language.

1.6 The `objCur` variable and the prompt

You should first read the Section *The current object `objCur` - The prompt `&wtrans` >`a`* of the main LastWave manual A lot of LastWave commands about wavelet transforms or about any complex structures we will see later (e.g., books of Matching pursuit....) know about a "current" structure named `objCur` (in the current environment). It means that, by default, if no wavelet transform variable is specified in a command dealing with wavelet transforms (e.g., `owtr`, `owtd`, `cwtd`), or generally, if no variable pointing to a complex structure is specified in a command dealing with these complex structures, the variable `objCur` of the local environment is used. The prompt tells you what this variable (in the global environment) is pointing to. Let us see how it works on wavelet transforms.

If the `wtrans1d` package has been loaded, then 2 wavelet transform variables named `a` and `b` have been created. Moreover, the `objCur` (i.e, the current object you are working with) has been assigned to `a`. Then automatically, the prompt will be

```
(&wtrans) a>
```

Which means that the current object is the structure pointed to by the variable `a` and is of type `&wtrans`. Thus, for instance, if we assign the `objCur` variable to be the variable `b`, the prompt will automatically change :

```
(&wtrans) a> objCur=b
(&wtrans) b>
```

Let us note that this behavior of the prompt, is defined in script in the `startup` file using the `terminal prompt` command that lets you associate any script command to compute the prompt. Let us also note that in the `wtrans1d` package (look in the script file `scripts/wtrans1d/wtrans1d.pkg`) we have also created 2 commands named `a` and `b` allowing to set the `objCur` to the wavelet transforms `a` or `b`. This gives a more convenient way than typing `setvar objCur b` to change the `objCur` to `b`:

```
a> b
b> a
a>
```

In the case of wavelet transforms, if the `objCur` is set to `a`, it means that the default wavelet transform that will be used is `a`. Thus instead of typing

```
a> owtd a 5
```

one could just type

```
a> owtd 5
```

Moreover, you don't need to specify the wavelet name also in the "abreviated" syntax. Thus instead of typing

```
a> 0a = I(512)==256
a> owtd a 5
a> owtr a 1a
a> disp 1a
```

you could just type

```
a> 0= I(512)==256
a> owtd 5
a> owtr 1
a> disp 1
```

which is much simpler !

Remark : The `disp` command expects `&valobj` arguments (see LastWave main manual), i.e., anything but a number. Thus when you write `disp 1`, LastWave understands that `1` actually means `1a` **Remark** : If you want to use the `objCur` variable (implicitely or explicitely) in a script command, don't forget to import it from the calling level (if it exists there) or from the global level. A very good example is given by the loop in the `scripts/wtrans1d/wtrans1d.pkg` file that allows to create the `a` and `b` wavelet transforms along with the `a` and `b` command.

WARNING : Whenever you use LastWave evaluator if you want to access signals that belong to the current wavelet transform, you must, in every case, specify the name of the wavelet transform associated to the signal. Indeed, otherwise, the evaluator would think you just typed a regular number. Thus to compute the absolute value of the signal `0a` and put it back in `0a`, one cannot write

```
a> 0=abs(0)
```

but instead, in order to avoid any ambiguity, one must write

```
a> 0= abs(0a)
```

Remark : Be careful if you have a wavelet transform named `e10` and if you write `abs(1e10)`. There is no way to understand if you mean the number `1e10` or the signal `A[0,1]` of `e10`. As we have already explained, the `objCur` system will be used in a lot of other packages exactly in the same way as for the `wtrans1d` package. As you will see, after using LastWave a little bit, this is extremely useful.

Chapter 2

Using the Wavelet Transform Extrema (`extrema1d`) package

2.1 Loading the package

Again, the first thing you need to do is to use this package is to load it. This is done automatically by the original `startup` file or directly by typing `package load extrema1d` in the terminal.

2.2 The extrema representation structure and the extremum structure

2.2.1 Introduction

In LastWave, each wavelet transform (`&wtrans`) variable is associated to an *extrema representation structure* of type (`&extrep`). This structure allows to store all the maxima of the modulus of the wavelet transform (when the scale is fixed). Its fields are similar to the fields of the wavelet transform. It has its own `nvoice` and `noct` fields and since we want to store the maxima of the wavelet transform at each scale, the extrema representation structure involves a two-dimensionnal array `D` whose first parameter is the octave number and the second one the voice number. This array is no longer an array of signals of course but an array of extremum structures. Each extremum is of type `&ext`.

2.2.2 The main fields of `&extrep`

They are

- `noct` : the number of octaves
- `nvoice` : the number of voices
- `size` : the `size` field of the original signal
- `x0` : the `x0` field of the original signal
- `dx` : the `dx` field of the original signal

- `wavelet` : the name of the analyzing wavelet that have been used
- `D[o,v]` : the first extremum (i.e., the extremum with the smallest abscissa) at octave `o` and voice `v`
- `name` : the name of the `&extrep`
- `wtrans` (read-only) : the `&wtrans` variable it corresponds to (`null` if none)

2.2.3 The main fields of `&ext`

They are

- `x` : the abscissa of the extremum,
- `index`: the index number corresponding to the sample where the extremum is,
- `y`: the scale number of the extremum. For the sake of simplicity this number is an integer corresponding to `o*noct+v` where `o` is the octave number of the extremum, `v` is the voice number of the extremum and `nvoice` the number of voices per octave.
- `z`: (referred to in the C code as the `ordinate` field) it corresponds to the value of the wavelet transform at this maximum.

At each scale (i.e., at each octave number and voice number), these extrema are organized as chained lists. Thus from an extremum you can refer to the `next` one or to the `previous` one. Moreover, since we will need to chain the extrema along the scales, one can refer (if the extrema were chained) to the `coarser` (going to coarser scales) one and the `finer` one (going to finer scales). Thus, the other fields of `&ext` are

- `next` (read-only): the next extremum (the closest, at the same scale and with an abscissa greater than the current extremum abscissa; `null` if none)
- `previous` (read-only): the previous extremum (the closest, at the same scale and with an abscissa smaller than the current extremum abscissa; `null` if none))
- `finer` (read-only): the closest extremum at the next smaller scale (`null` if none)
- `coarser` (read-only): the closest extremum at the next greater scale (`null` if none)

2.3 A simple example

Let's compute the continuous wavelet transform of the dirac function as in the previous section :

```
a> 0=I(512)==256
a> cwtd 1 3 6 'g2'
```

in order to compute the extrema representation (on the current wavelet transform, i.e., `a`), one just need to do

```
a> extrema
= 54
```

which returns the number of extrema that was found. It AUTOMATICALLY chains them through scales AND delete all the chains that haven't been chained down to the smallest scale. This is very important in order to stabilize the extrema computation (which is a priori a very unstable computation). You can use option `-C` to keep LastWave from chaining (and deleting).

2.4 Addressing an `&extrep`

All the commands about extrema representations normally require an argument which is the extrema representation it will work on. Each wavelet transform structure is associated to an extrema representation structure. You address this extrema representation using the field `extrep`. Thus, for instance, `a.extrep` is the extrema representation of the wavelet transform `a`. Of course, if you need to you can create an extrema representation which is not associated to any wavelet transform. For that purpose you need to use the `new` command :

```
a> er = [new &extrep]
= <&extrep;0x209c2388>
a> type(er)
= '&extrep'
```

We can recompute the extrema representation of the dirac directly into the variable `er`

```
a> extrema a er
= 90
a> disp er
```

2.5 Display and mouse interaction

To display the extrema representation you again use the `disp` command! Thus, you just need to do

```
a> disp a.extrep
```

You can even combine signals, wavelet transforms and extrema representations. For instance to display the signal above the wavelet transform above the extrema representation, you just type

```
a> disp 0 a a.extrep
```

The `disp` command uses the graphic class `GraphExtrep` to display extrema representations. **To learn about interactions with extrema representations you should run the Demos of the `extrema1d` package** You should be able to use the mouse within the graph that displays the extrema representation. As for wavelet transforms, dragging the left mouse button allow to perform zooms (if you hit the `z` key you change zoom mode). The middle mouse button zooms out the the view. Use the `c` key to get a cross-hair cursor. If you hit it again, you'll get a very useful cursor which circles and gives you information about the extremum which is the closest to the mouse pointer!

Moreover, you can use the mouse buttons along with the `shift` key to get sections along a maxima line (middle button for linear scales, right button for log scales and left button just to light up the line). These sections will be drawn in the "last" window of type `&signali` (read the `disp` manual pages).

All these behavior were defined in the script file `scripts/extrema1d/extrema1d.pkg`. You can modify them or create more complex actions that you can perform using the mouse. In order to do so, you should read the Sections *Managing events* and *More about managing events* of the main LastWave manual.

2.5.1 The `GraphExtrep` graphic object

An `extrep` structure is displayed using a `GraphExtrep` graphic object. The main fields of that graphic class are

- `chain` : a flag which indicates whether the maxima lines should be displayed or not,
- `graph` : which allows to set/get the `extrep` structure that will be displayed,

2.6 Playing around with `&ext`

You can access individually each extremum of an extrema representation using the `D` field. Thus `er.D[1,5]` refers to the first extremum of the octave number 1 and voice number 5 of the extrema representation `er`. You can get the values of its field

```
a> er.D[1,5].index = 247
a> er.D[1,5].z
= -0.25079748
```

In order to loop on the extrema at a given scale you can do

```
a> for {ext = er.D[1,5]} (ext isnot null) {ext = ext.next} {
--> printf '%d\n' ext.index
--> }
247
256
265
```

You will find a lot of other examples of such loops in the script file `scripts/extrema1d/extrema1d.pkg` that defines the mouse behavior for extrema representation displays.

Part II

Reference

Chapter 3

Package `extrema1d` 2.0

Package allowing to deal with 1d wavelet transform local extrema.
*** Authors and Copyright : B.Audit, E.Bacry, J.F.Muzy and C.Vaillant*

3.1 Defined types

3.1.1 Type `&ext`

This type is used to store a local extremum of a wavelet transform. It is itself in a structure of type &extrep (i.e., an extrema representation)

- `&ext.x [= <x>]`
  Sets/Gets the abscissa of an extremum.
- `&ext.index [= <index>]`
  Sets/Gets the position of an extremum using an integer index.
- `&ext.y`
  Gets the scale of an extremum. It is an integer (0 corresponds to the smallest scale).
- `&ext.z [= <wtValue>]`
  Sets/Gets the wavelet transform value at the extremum.
- `&ext.nbExt`
  Gets the number of extrema at the same scale as this extremum.
- `&ext.next`
  Gets the next extremum (or null if none) at the same scale.
- `&ext.last`
  Gets the last extremum at the smae scale.
- `&ext.first`
  Gets the first extremum at the same scale.
- `&ext.previous`
  Gets the previous extremum (or null if none) at the same scale.

- `&ext.coarser`
  Gets the coarser extremum (or null if none) at the next larger scale on the same chain.

- `&ext.coarsest`
  Gets the coarsest extremum (or null if none) on the same chain.

- `&ext.finer`
  Gets the finer extremum (or null if none) at the next smaller scale on the same chain.

- `&ext.finest`
  Gets the finest extremum (or null if none) on the same chain.

- `&ext.extrep`
  Gets the extrema representation (type &extrep), the extremum is in.

3.1.2 Type `&extrep`

This type is used to store the local extrema of a 1d wavelet transform.

- `&extrep.D [o,v]`
  Get the first extrema at octave 'o' and voice 'v'

- `&extrep.name [= <name>]`
  Sets/Gets the name of an extrep

- `&extrep.dx [= <dx>]`
  Sets/Gets the abscissa step of the original signal of the extrema representation.

- `&extrep.x0 [= <x0>]`
  Sets/Gets the first abscissa of the original signal of the extrep.

- `&extrep.nvoice`
  Gets the number of voices per octave of the extrema representation.

- `&extrep.noct`
  Gets the number of octave of the extrema representation.

- `&extrep.size [= <size>]`
  Sets/Gets the size of the original signal of the extrema representation.

- `&extrep.wavelet [= <name>]`
  Gets/Sets the name of the analyzing wavelet used for the extrema representation.

- `&extrep.wtrans`
  Gets the wavelet transform associated to the extrema representation.

3.2 Commands related to 1d wavelet transform extrema

• **chain** [<extrep>=objCur] [-d]
Chains the extrema of the extrema representation <extrep> and deletes all the extrema which are not part of chains that reach the smallest scale (unless option '-d' is specified). It returns the number of chains found.

• **chaindelete** [<extrep>=objCur]
deletes all the extrema of the extrema representation <extrep> which are not part of chains that reach the smallest scale. It returns the number of chains found. This command is called (by default) by the 'chain' command.

• **chainmax** [<extrep>=objCur] [<exponent>=0]
Replaces the wavelet trasnform value of an extremum by the maximum value along its chain. This routine allows to perform the scale adaptative version of the WTMM method. The coefficients are first multiply by aMin*(2^ (exponent*(o-1+v/nvoice))). This is useful in the case some maxima chains do not decay when going to small scales.

• **ecopy** <extrepIn> <extrepOut>
Copies an extrema representation into another.

• **eread** [<extrep>=objCur] (<file> | <stream>)
Reads an extrema representation from a <file> or a <stream>.

• **ewrite** [<extrep>=objCur] (<file> | <stream>) [-h]
Writes an extrema representation in a <file> or in a <stream> (with no header if '-h').

• **ext**

  • **ext** <removechain> <ext>

    Removes the extrema chain the extremum <ext> belongs to.

  • **ext** remove <ext>

    Removes an extremum from its extrema representation.

• **extlistosig** <ext> <signal>
Converts the extrema list pointed by <ext> into a <signal>

• **extrema** [<wtrans>=objCur] [<extrep>=<wtrans>] [-icC] [-e <threshold>]
Computes the extrema representation associated to the wavelet transform <wtrans> and puts it in variable <extrep>. It returns the number of extrema found. Let us note that this command leads to unstable results at large sacles (i.e., too many extrema). In order to make the computation stable, by default, the 'extrema' command chain the extrema (using the 'chain' command) and delete the extrema (using the 'chaindelete' command) which have not been chained. In most cases, it works perfectly. If you want to disable this behavior you should use the '-C' option. Options are :
-i : The extremum positions are computed without using interpolation and thus always correspond to a sample of the original signal
-c : Causal effects are taken into account (The extrema are computed only between the 'firstp' and 'lastp' indexes of each wavelet coefficient signal).
-C : Disable computation of chains and deletion of extrema which do not belong to the

chains.

• **extrep** `closest [<extrep>] <x> <y>`
Gets the extremum which is closest to <x> <y> (or returns null).

• **setextrep** `OLD LASTWAVE COMMAND`
NOT TO BE USED

3.3 Graphic class `GraphExtrep` (inherits from `GObject`)

Graphic Class that allows to display 1d extrema representation
• **setg**

- **setg** `*GraphExtrep* -chained [<flagOnOff>]`
  Sets/Gets the chained flag. If 1 then it will link on the display the chained extrema.
- **setg** `*GraphExtrep* -graph [<extrep>]`
  Gets/Sets the extrema representation to be displayed by the GraphExtrep with <extrep>.

Bindings

- Shift + Middle button = Display Wavelet transform along the maxima line
- Shift + Right button = Display log(|Wavelet transform|) along the maxima line
- Shift + Left button = Show maxima line
- Type 'c' to change cursor mode
- 'z' : changes the zoom mode just type 'z'
- Left/Right/Middle button : operate the zoom

3.4 Demos

Here is a list of all the Demo files and for each of them all the corresponding Demo commands. To try a Demo command, you should first source the corresponding Demo file then run the command. (When sourcing the Demo file, LastWave tells you about all the commands included in this file).
The Demo files corresponding to this package are :

Demo file **DemoExtrema1d**

• **DemoExtrema1dSing** (in file `scripts/extrema1d/DemoExtrema1d`)

Demo command that computes the extrema of the wavelet transform of a signal which has one singularity (holder = 0.5) in white noise. It displays the signal, its wavelet transform and the extrema representation. You can use the mouse to follow the wavelet transform values along a maxima line in order to measure the Holder exponent.

Chapter 4

Package `wtrans1d` 2.0

Package allowing to perform 1d (continuous and (bi)orthogonal) wavelet transform.
*** Authors and Copyright : B.Audit, E.Bacry, N.Decoster, S.Mallat and J.F.Muzy*

4.1 Defined types

4.1.1 Type `&wtrans`

This type is the basic type for 1d wavelet transforms (both continuous, dyadic or orthogonal). It is organized as two 2d arrays of signals referred to as A (the approximation signals) and D (the detail signals). Both of them are indexed first by the octave number and then by the voice number. The signal to be analyzed is in A[0,0].

- `&wtrans.A [o,v]`
  Get the Approximation signal at octave 'o' and voice 'v'
- `&wtrans.D [o,v]`
  Get the Detail signals at octave 'o' and voice 'v'
- `&wtrans.name [= <name>]`
  Sets/Gets the name of a wtrans
- `&wtrans.dx [= <dx>]`
  Sets/Gets the abscissa step of the original signal of the wavelet transform.
- `&wtrans.x0 [= <x0>]`
  Sets/Gets the first abscissa of the original signal of the wavelet transform.
- `&wtrans.nvoice [= <nvoice>]`
  Sets/Gets the number of voices per octave of a wavelet transform.
- `&wtrans.noct [= <noct>]`
  Sets/Gets the number of octave of a wavelet transform.
- `&wtrans.size [= <size>]`
  Sets/Gets the size of the original signal of the wavelet transform.

- `&wtrans.type [= (1 | 3)]`
  Gets/Sets the type of a wavelet transform (1 for orthogonal and 3 for continuous).

- `&wtrans.amin [= <amin>]`
  Sets/Gets the smallest scale of a wavelet transform.

- `&wtrans.wavelet [= <name>]`
  Gets/Sets the name of the analyzing wavelet used for the wavelet transform.

- `&wtrans.extrep`
  Gets the extrema representation associated to the wavelet transform.

4.2 Commands for dealing with Wavelet transform structures

• **setwtrans** `Old version procedure`
Not to be used

• **wread** `[<wtrans>=objCur] (<file> | <stream>)`
Reads a wavelet transform from a <file> or a <stream>.

• **wthresh** `[<wtrans>=objCur] <threshold> [-x <xMin> <xMax>] [-o <oMin> <oMax>] [-e <alpha>] [-w <wtransOut>]`
Thresholds the wavelet coefficients of a wavelet transform <wtrans>. All the coefficients smaller than <threshold> are set to 0. The options are
-x : Only coefficients corresponding to abscissa between <xMin> and <xMax> are thresholded
-o : Only coefficients corresponding to octave between <oMin> and <oMax> are thresholded
-w : Specifies the wavelet transform the result should be stored in (by default, the initial <wtrans> is used)

• **wwrite** `[<wtrans>=objCur] (<file> | <stream>)`
Writes a wavelet transform in a <file> or in a <stream> (in binary mode).

4.3 Wavelet transform commands

• **cwtd** `[<wtrans>=objCur] <aMin> <nOct> <nVoice> [<wavelet>=g2] [-b <border> = mir] [-c] [-m] [-e <expo> = -1]`
Performs the continuous wavelet transform of the signal A[0][0] in <wtrans> from scale <aMin> using <nOct> octaves, <nVoice> voices per octave, using the nth derivative of the Gaussian function (<wavelet>=g<n> where n is in [0..4] are such that g0, -g1, g2, -g3, g4 are the gaussian function and its successive derivatives) as analyzing wavelet or the morlet wavelet (which is a complex wavelet). In the case of a complex analyzing wavelet, the modulus of the wavelet transform is stored in the D signals (e.g., .10 .20 .33) and the phase (between 0 and 2pi) in the A signals (e.g., 10 20 30). The command returns the

elapsed time in seconds. The first scale <aMin> should not be smaller than 1. There are 4 ways to deal with border effects ('per'=periodic, 'mir'=mirror, 'pad0'=padding with 0 values, 'pad'=padding with constant extremity values so that the signal is continuous).
-c: causal mode, only used when -m is used. The extrema representation iscomputed using the causal mode.

• **cwtdoctmax** <signalSize> <aMin> <nVoice> <wavelet>
It returns the largest octave number that can be used using the 'cwtd' command with the same arguments.

• **owtd** [<wtrans>=objCur] <nOct>
Performs (bi)orthogonal wavelet transform of the signal A[0][0] in <wtrans> using <nOct> octaves.

• **owtf** [<wtrans>=objCur] <filterFilename>
Sets a filter for [bi]orthogonal decomposition. Filter files are in scripts/wtrans1d/filters directory. Extension '.o' stands for orthogonal filters and '.b' for biorthogonal filters.

• **owtr** [<wtrans>=objCur] [<signalOut>=0]
Performs (bi)orthogonal wavelet reconstruction of the wavelet transform <wtrans> in signal <signalOut>.

4.4 Script Commands

4.5 Graphic class GraphWtrans (inherits from GObject)

Graphic Class that allows to display 1d wavelet transform
• **setg**

- **setg** *GraphWtrans* -causal [<flagOnOff>]

  Sets/Gets the causal flag. If 1 then it will not display all the values which were affected by border effects.

- **setg** *GraphWtrans* -cm [<colormap>]

  Sets/Gets the colormap that will be used to display the wavelet transform.

- **setg** *GraphWtrans* -graph [<wtrans>]

  Gets/Sets the wavelet transform to be displayed by the GraphWtrans. (The '-cgraph' field is equivalent to that field).

- **setg** *GraphWtrans* -norm [<type>]

  Sets/Gets the normalization used for displaying a wavelet transform. In any case, the absolute value of the wavelet transform is coded. However, there are 4 choices :
  - 'global' : The colors are coded from the global minimum of the absolute value and the global maximum
  - 'local' : The colors are coded from the local (among all the values displayed in the

View) minimum of the absolute value and the local maximum. Let us note that this mode is different from 'global' only when the image is zoomed
- 'lglobal' : The colors are coded separately at each scale from the global minimum of the absolute value (at each scale) and the global maximum

Bindings

- Shift + Middle button = vertical (linear) section
- Shift + Right button = vertical (log) section
- Shift + Left button = horizontal section
- Type 'c' to change cursor mode
- 'z' : changes the zoom mode just type 'z'
- Left/Right/Middle button : operate the zoom

4.6 Demos

Here is a list of all the Demo files and for each of them all the corresponding Demo commands. To try a Demo command, you should first source the corresponding Demo file then run the command. (When sourcing the Demo file, LastWave tells you about all the commands included in this file).
The Demo files corresponding to this package are :

Demo file **DemoWtrans1d**

• **DemoWtrans1dFrac** (in file `scripts/wtrans1d/DemoWtrans1d`)

Demo command that computes the continuous wavelet transform of a Devil Staircase (with noise). It displays the signal and its wavelet transform. It teaches you how to use the mouse.

Index