Package ‘Rwave’

October 22, 2022

Version 2.6-5
Date 2022-10-18
Title Time-Frequency Analysis of 1-D Signals
Author Rene Carmona [aut],
Bruno Torresani [aut],
Brandon Whitcher [ctb],
Andrea Wang [ctb],
Wen-Liang Hwang [ctb],
Robert Alberts [ctb],
Jonathan M. Lees [ctb, cre]
Maintainer Jonathan M. Lees <jonathan.lees@unc.edu>
Depends R (>= 2.14)
Description A set of R functions which provide an
environment for the Time-Frequency analysis of 1-D signals (and
especially for the wavelet and Gabor transforms of noisy
signals). It was originally written for Splus by Rene Carmona,
Bruno Torresani, and Wen L. Hwang, first at the University of
California at Irvine and then at Princeton University. Credit
should also be given to Andrea Wang whose functions on the
dyadic wavelet transform are included. Rwave is based on the
book: “Practical Time-Frequency Analysis: Gabor and Wavelet
Transforms with an Implementation in S”, by Rene Carmona, Wen
License GPL (>= 2)
Copyright University of California
URL https://carmona.princeton.edu/TFbook/tfbook.html,
https://r-forge.r-project.org/projects/rwave/
Repository CRAN
Date/Publication 2022-10-21 23:17:49 UTC
NeedsCompilation yes
R topics documented:

A0 ................................................................. 5
A4 ................................................................. 5
adjust.length ............................................... 6
amber7 .......................................................... 7
amber8 .......................................................... 7
amber9 .......................................................... 8
B0 ................................................................. 9
B4 ................................................................. 9
back1.000 ...................................................... 10
back1.180 ...................................................... 11
back1.220 ...................................................... 11
backscatter.1.000 ........................................... 12
backscatter.1.180 .......................................... 13
backscatter.1.220 .......................................... 13
C0 ................................................................. 14
C4 ................................................................. 15
cfamily ......................................................... 15
cgt ............................................................... 17
ch ................................................................. 18
check.maxresoln ............................................. 19
chirpm5db.dat ............................................... 19
cleanph ......................................................... 20
click ............................................................ 21
click.asc ....................................................... 21
corona ........................................................ 22
coronoid ....................................................... 23
crc .............................................................. 24
crcrc . .......................................................... 26
crfview ......................................................... 27
cwt ............................................................. 27
cwtimage ..................................................... 29
cwp ............................................................ 30
cwtpolar ...................................................... 31
cwtsquiz ...................................................... 32
cwtTh .......................................................... 33
D0 ................................................................. 34
D4 ................................................................. 34
DOG ............................................................ 35
dwinverse .................................................... 36
Ekg ............................................................... 37
epl .............................................................. 37
ext ............................................................. 38
fastgkernel .................................................. 39
fastkernel ..................................................... 40
gabor .......................................................... 41
gcrcrec ........................................................ 42
<table>
<thead>
<tr>
<th>Topics Documented</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>gkernel</td>
<td>43</td>
</tr>
<tr>
<td>gregrec</td>
<td>44</td>
</tr>
<tr>
<td>gridrec</td>
<td>45</td>
</tr>
<tr>
<td>gsampleOne</td>
<td>46</td>
</tr>
<tr>
<td>gwave</td>
<td>47</td>
</tr>
<tr>
<td>gwave2</td>
<td>47</td>
</tr>
<tr>
<td>HeartRate</td>
<td>48</td>
</tr>
<tr>
<td>HOWAREYOU</td>
<td>49</td>
</tr>
<tr>
<td>hurst.est</td>
<td>50</td>
</tr>
<tr>
<td>icm</td>
<td>51</td>
</tr>
<tr>
<td>mbtrim</td>
<td>52</td>
</tr>
<tr>
<td>mntrim</td>
<td>53</td>
</tr>
<tr>
<td>morlet</td>
<td>54</td>
</tr>
<tr>
<td>morwave</td>
<td>55</td>
</tr>
<tr>
<td>morwave2</td>
<td>55</td>
</tr>
<tr>
<td>mrecons</td>
<td>56</td>
</tr>
<tr>
<td>mw</td>
<td>57</td>
</tr>
<tr>
<td>noisy.dat</td>
<td>58</td>
</tr>
<tr>
<td>noisywave</td>
<td>59</td>
</tr>
<tr>
<td>npl</td>
<td>59</td>
</tr>
<tr>
<td>pixel_8.7</td>
<td>60</td>
</tr>
<tr>
<td>pixel_8.8</td>
<td>60</td>
</tr>
<tr>
<td>pixel_8.9</td>
<td>61</td>
</tr>
<tr>
<td>plotResult</td>
<td>62</td>
</tr>
<tr>
<td>plotwt</td>
<td>62</td>
</tr>
<tr>
<td>pure.dat</td>
<td>63</td>
</tr>
<tr>
<td>purwave</td>
<td>64</td>
</tr>
<tr>
<td>regrec</td>
<td>64</td>
</tr>
<tr>
<td>regrec2</td>
<td>66</td>
</tr>
<tr>
<td>RidgeSampling</td>
<td>67</td>
</tr>
<tr>
<td>ridrec</td>
<td>68</td>
</tr>
<tr>
<td>rkernel</td>
<td>69</td>
</tr>
<tr>
<td>rwkernel</td>
<td>70</td>
</tr>
<tr>
<td>scrcrec</td>
<td>71</td>
</tr>
<tr>
<td>signal_W_tilda.1</td>
<td>72</td>
</tr>
<tr>
<td>signal_W_tilda.2</td>
<td>72</td>
</tr>
<tr>
<td>signal_W_tilda.3</td>
<td>73</td>
</tr>
<tr>
<td>signal_W_tilda.4</td>
<td>74</td>
</tr>
<tr>
<td>signal_W_tilda.5</td>
<td>74</td>
</tr>
<tr>
<td>signal_W_tilda.6</td>
<td>75</td>
</tr>
<tr>
<td>signal_W_tilda.7</td>
<td>76</td>
</tr>
<tr>
<td>signal_W_tilda.8</td>
<td>76</td>
</tr>
<tr>
<td>signal_W_tilda.9</td>
<td>77</td>
</tr>
<tr>
<td>sig_W_tilda.1</td>
<td>78</td>
</tr>
<tr>
<td>sig_W_tilda.2</td>
<td>78</td>
</tr>
<tr>
<td>sig_W_tilda.3</td>
<td>79</td>
</tr>
<tr>
<td>sig_W_tilda.4</td>
<td>80</td>
</tr>
<tr>
<td>sig_W_tilda.5</td>
<td>80</td>
</tr>
</tbody>
</table>
R topics documented:

skeleton ................................................................. 81
skeleton2 ............................................................... 82
smooth1 ................................................................. 83
smoothwt ................................................................. 83
snake ...................................................................... 84
snakeview ............................................................... 85
snakoid ................................................................. 86
sridrec ................................................................. 87
SVD ................................................................. 88
tfgmax ................................................................. 88
tflmax ................................................................. 89
tfmean ................................................................. 90
tfpct ................................................................. 90
tfvar ................................................................. 91
Undocumented .......................................................... 92
vDOG ................................................................. 92
vecgabor ............................................................... 93
vecmorlet ............................................................ 93
vgt ................................................................. 94
vwt ................................................................. 95
wpl ................................................................. 95
wRidgeSampling ...................................................... 96
wspec.pl .............................................................. 97
WV ................................................................. 97
W_tilda.1 ............................................................. 98
W_tilda.2 ............................................................. 99
W_tilda.3 ............................................................. 99
W_tilda.4 ............................................................ 100
W_tilda.5 ............................................................ 101
W_tilda.6 ............................................................ 101
W_tilda.7 ............................................................ 102
W_tilda.8 ............................................................ 103
W_tilda.9 ............................................................ 103
yen ................................................................. 104
yendiff ............................................................. 105
YN ................................................................. 105
YNdiff ............................................................. 106
zerokernel ............................................................ 106
zeroskeleton .......................................................... 107
zeroskeleton2 .......................................................... 108

Index 110
**A0**

*Transient Signal*

---

**Description**

Transient signal.

**Usage**

```r
data(A0)
```

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(A0)
plot.ts(A0)
```

---

**A4**

*Transient Signal*

---

**Description**

Transient signal.

**Usage**

```r
data(A4)
```

**Format**

A vector containing 1024 observations.
**adjust.length**

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(A4)
plot.ts(A4)
```

---

### Description

Add zeros to the end of the data if necessary so that its length is a power of 2. It returns the data with zeros added if necessary and the length of the adjusted data.

### Usage

```r
adjust.length(inputdata)
```

### Arguments

- `inputdata`: either a text file or an S object containing data.

### Value

Zero-padded 1D array.

### References

See discussions in the text of “Practical Time-Frequency Analysis”.
amber7

Description

Pixel from amber camara.

Usage

data(amber7)

Format

A vector containing 7000 observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

    data(amber7)
    plot.ts(amber7)

amber8

Description

Pixel from amber camara.

Usage

data(amber8)

Format

A vector containing 7000 observations.
Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(amber8)
plot.ts(amber8)

amber9

Pixel from Amber Camara

Description

Pixel from amber camara.

Usage

data(amber9)

Format

A vector containing 7000 observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(amber9)
plot.ts(amber9)
**B0**  
*Transient Signal*

**Description**  
Transient signal.

**Usage**  

```r
data(B0)
```

**Format**  
A vector containing 1024 observations.

**Source**  
See discussions in the text of “Practical Time-Frequency Analysis”.

**References**  

**Examples**  

```r
data(B0)
plot.ts(B0)
```

---

**B4**  
*Transient Signal*

**Description**  
Transient signal.

**Usage**  

```r
data(B4)
```

**Format**  
A vector containing 1024 observations.
Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(B4)
plot.ts(B4)

Acoustic Returns

Description

Acoustic returns from natural underwater clutter.

Usage

data(back1.000)

Format

A vector containing 7936 observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(back1.000)
plot.ts(back1.000)
Acoustic Returns

Description

Acoustic returns from ...

Usage

data(back1.180)

Format

A vector containing 7936 observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(back1.180)
plot.ts(back1.180)

Acoustic Returns

Description

Acoustic returns from an underwater metallic object.

Usage

data(back1.220)

Format

A vector containing 7936 observations.
Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(backscatter.1.000)
plot.ts(backscatter.1.000)

data(backscatter.1.000)  # Pixel from Amber Camara

Description

Pixel from amber camara.

Usage

data(backscatter.1.000)

Format

A vector containing observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(backscatter.1.000)
plot.ts(backscatter.1.000)
**backscatter.1.180**

*Pixel from Amber Camara*

**Description**

Pixel from amber camara.

**Usage**

`data(backscatter.1.180)`

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(backscatter.1.180)
plot.ts(backscatter.1.180)
```

---

**backscatter.1.220**

*Pixel from Amber Camara*

**Description**

Pixel from amber camara.

**Usage**

`data(backscatter.1.220)`

**Format**

A vector containing observations.
**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(backscatter.1.220)
plot.ts(backscatter.1.220)
```

---

**C0**

*Transient Signal*

**Description**

Transient signal.

**Usage**

```r
data(C0)
```

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(C0)
plot.ts(C0)
```
Description

Transient signal.

Usage

data(C4)

Format

A vector containing 1024 observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(C4)
plot.ts(C4)

cfamily

Ridge Chaining Procedure

Description

Chains the ridge estimates produced by the function crc.

Usage

cfamily(ccridge, bstep=1, nbchain=100, ptiler=0.05)
Arguments

cridge: unchained ridge set as the output of the function \texttt{crc}
bstep: maximal length for a gap in a ridge.
nbchain: maximal number of chains produced by the function.
ptile: relative threshold for the ridges.

details

\texttt{crc} returns a measure in time-frequency (or time-scale) space. \texttt{cfamily} turns it into a series of one-dimensional objects (ridges). The measure is first thresholded, with a relative threshold value set to the input parameter \texttt{ptile}. During the chaining procedure, gaps within a given ridge are allowed and filled in. The maximal length of such gaps is the input parameter \texttt{bstep}.

Value

Returns the results of the chaining algorithm

\begin{verbatim}
ordered map: image containing the ridges (displayed with different colors)
chain: 2D array containing the chained ridges, according to the chain data structure
chain[,1]: first point of the ridge
chain[,2]: length of the chain
chain[,3:(chain[,2]+2)]: values of the ridge

nbchain: number of chains produced by the algorithm
\end{verbatim}

References

See discussion in text of “Practical Time-Frequency Analysis”.

See Also

\texttt{crc} for the ridge estimation, and \texttt{crcrec}, \texttt{gcrcrec} and \texttt{scrcrec} for corresponding reconstruction functions.

Examples

\begin{verbatim}
## Not run:
data(HOWAREYOU)
plot.ts(HOWAREYOU)
cgthOWAREYOU <- cgt(HOWAREYOU,70,0.01,100)
clHOWAREYOU <- crc(Mod(cgthOWAREYOU),nbclimb=1000)
cfHOWAREYOU <- cfamily(clHOWAREYOU,ptile=0.001)
image(cfHOWAREYOU$ordered > 0)
## End(Not run)
\end{verbatim}
**Description**

Computes the continuous Gabor transform with Gaussian window.

**Usage**

cgt(input, nvoice, freqstep=(1/nvoice), scale=1, plot=TRUE)

**Arguments**

- `input`: input signal (possibly complex-valued).
- `nvoice`: number of frequencies for which gabor transform is to be computed.
- `freqstep`: Sampling rate for the frequency axis.
- `scale`: Size parameter for the window.
- `plot`: logical variable set to TRUE to display the modulus of the continuous gabor transform on the graphic device.

**Details**

The output contains the (complex) values of the gabor transform of the input signal. The format of the output is a 2D array (signal_size x nb_scales).

**Value**

continuous (complex) gabor transform (2D array).

**Warning**

freqstep must be less than 1/nvoice to avoid aliasing. freqstep=1/nvoice corresponds to the Nyquist limit.

**References**

See discussion in text of “Practical Time-Frequency Analysis”.

**See Also**

cwt, cwtp, DOG for continuous wavelet transforms. cwtsquiz for synchrosqueezed wavelet transform.
Examples

```r
data(HOWAREYOU)
plot.ts(HOWAREYOU)

cgthOWAREYOU <- cgt(HOWAREYOU,70,0.01,100)
```

---

<table>
<thead>
<tr>
<th>ch</th>
<th>Chen’s Chirp</th>
</tr>
</thead>
</table>

Description

Chen’s chirp.

Usage

```r
data(ch)
```

Format

A vector containing 15,000 observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

```r
data(ch)
plot.ts(ch)
```
**check.maxresoln**

*Verify Maximum Resolution*

**Description**

Stop when \(2^{\text{maxresoln}}\) is larger than the signal size.

**Usage**

\[
\text{check.maxresoln}(\text{maxresoln}, \text{np})
\]

**Arguments**

- `maxresoln`: number of decomposition scales.
- `np`: signal size.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

`mw`, `mrecons`.

---

**chirpm5db.dat**

*Pixel from Amber Camera*

**Description**

Pixel from amber camera.

**Usage**

\[
data(\text{chirpm5db.dat})
\]

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

---
References


Examples

```r
## Not run:
data(chirpm5db.dat)

## End(Not run)
```

---

cleanph

Threshold Phase based on Modulus

Description

Sets to zero the phase of time-frequency transform when modulus is below a certain value.

Usage

```r
cleanph(tfrep, thresh=0.01, plot=TRUE)
```

Arguments

- `tfrep`: continuous time-frequency transform (2D array)
- `thresh`: (relative) threshold.
- `plot`: if set to TRUE, displays the maxima of cwt on the graphic device.

Value

thresholded phase (2D array)

References

See discussion in text of “Practical Time-Frequency Analysis”.
**click**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dolphin click data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data(click)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>A vector containing 2499 observations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>See discussions in the text of “Practical Time-Frequency Analysis”.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>References</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data(click)</code></td>
</tr>
<tr>
<td><code>plot.ts(click)</code></td>
</tr>
</tbody>
</table>

**click.asc**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel from amber camara.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data(click.asc)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>A vector containing observations.</td>
</tr>
</tbody>
</table>
Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(click.asc)
plot.ts(click.asc)

corona(tfrep, guess, tfspec=numeric(dim(tfrep)[2]), subrate=1, temprate=3, mu=1, lambda=2 * mu, iteration=1000000, seed=-7, stagnant=20000, costsub=1, plot=TRUE)

Description

Estimate a (single) ridge from a time-frequency representation, using the corona method.

Usage

corona(tfrep, guess, tfspec=numeric(dim(tfrep)[2]), subrate=1, temprate=3, mu=1, lambda=2 * mu, iteration=1000000, seed=-7, stagnant=20000, costsub=1, plot=TRUE)

Arguments

tfrep Time-Frequency representation (real valued).
guess Initial guess for the algorithm.
tfspec Estimate for the contribution of the noise to modulus.
subrate Subsampling rate for ridge estimation.
temprate Initial value of temperature parameter.
mu Coefficient of the ridge’s second derivative in cost function.
lambda Coefficient of the ridge’s derivative in cost function.
iteration Maximal number of moves.
seed Initialization of random number generator.
stagnant Maximum number of stationary iterations before stopping.
costsub Subsampling of cost function in output.
plot When set(default), some results will be shown on the display.
Details

To accelerate convergence, it is useful to preprocess modulus before running annealing method. Such a preprocessing (smoothing and subsampling of modulus) is implemented in corona. The parameter subrate specifies the subsampling rate.

Value

Returns the estimated ridge and the cost function.

ridge  1D array (of same length as the signal) containing the ridge.
cost   1D array containing the cost function.

Warning

The returned cost may be a large array, which is time consuming. The argument costsub allows subsampling the cost function.

References

See discussion in text of “Practical Time-Frequency Analysis”.

See Also

icm, coronoid, snake, snakoid.

---

coronoid  Ridge Estimation by Modified Corona Method

description

Estimate a ridge using the modified corona method (modified cost function).

Usage

coronoid(tfrep, guess, tfspec=numeric(dim(tfrep)[2]), subrate=1,
  temprate=3, mu=1, lambda=2 * mu, iteration=1000000, seed=-7,
  stagnant=20000, costsub=1, plot=TRUE)

Arguments

tfrep   Estimate for the contribution of the noise to modulus.
guess  Initial guess for the algorithm.
tfspec  Estimate for the contribution of the noise to modulus.
subrate Subsampling rate for ridge estimation.
temprate Initial value of temperature parameter.
mu Coefficient of the ridge’s derivative in cost function.
lambda          Coefficient of the ridge’s second derivative in cost function.
iteration       Maximal number of moves.
seed            Initialization of random number generator.
stagnant         Maximum number of stationary iterations before stopping.
costsub         Subsampling of cost function in output.
plot            When set (default), some results will be shown on the display.

Details
To accelerate convergence, it is useful to preprocess modulus before running annealing method. Such a preprocessing (smoothing and subsampling of modulus) is implemented in coronoid. The parameter subrate specifies the subsampling rate.

Value
Returns the estimated ridge and the cost function.
ridge           1D array (of same length as the signal) containing the ridge.
cost            1D array containing the cost function.

Warning
The returned cost may be a large array. The argument costsub allows subsampling the cost function.

References
See discussion in text of “Practical Time-Frequency Analysis”.

See Also
corona, icm, snake, snakoid.

crc          Ridge Extraction by Crazy Climbers

Description
Uses the "crazy climber algorithm" to detect ridges in the modulus of a continuous wavelet or a Gabor transform.

Usage
crc(tfrep, tfspec=numeric(dim(tfrep)[2]), bstep=3, iteration=10000, rate=0.001, seed=-7, nbclimb=10, flag.int=TRUE, chain=TRUE, flag.temp=FALSE)
Arguments

- `tfrep`: modulus of the (wavelet or Gabor) transform.
- `tfspec`: numeric vector which gives, for each value of the scale or frequency the expected size of the noise contribution.
- `bstep`: stepsize for random walk of the climbers.
- `iteration`: number of iterations.
- `rate`: initial value of the temperature.
- `seed`: initial value of the random number generator.
- `nbclimb`: number of crazy climbers.
- `flag.int`: if set to TRUE, the weighted occupation measure is computed.
- `chain`: if set to TRUE, chaining of the ridges is done.
- `flag.temp`: if set to TRUE: constant temperature.

Value

Returns a 2D array called beemap containing the (weighted or unweighted) occupation measure (integrated with respect to time)

References

See discussion in text of “Practical Time-Frequency Analysis”.

See Also

corona, icm, coronoid, snake, snakoid for ridge estimation, cfamily for chaining and crcrc, gcrcrec, scrcrec for reconstruction.

Examples

```r
data(HOWAREYOU)
plot.ts(HOWAREYOU)

cgtHOWAREYOU <- cgt(HOWAREYOU, 70, 0.01, 100)
clHOWAREYOU <- crc(Mod(cgtHOWAREYOU), nbclimb=1000)
```
**Description**
Reconstructs a real valued signal from the output of crc (wavelet case) by minimizing an appropriate quadratic form.

**Usage**
```
crcrec(siginput, inputwt, beemap, noct, nvoice, compr, minnbnodes=2, w0=2*pi, bstep=5, ptile=0.01, epsilon=0, fast=FALSE, para=5, real=FALSE, plot=2)
```

**Arguments**
- `siginput`: original signal.
- `inputwt`: wavelet transform.
- `beemap`: occupation measure, output of crc.
- `noct`: number of octaves.
- `nvoice`: number of voices per octave.
- `compr`: compression rate for sampling the ridges.
- `minnbnodes`: minimal number of points per ridge.
- `w0`: center frequency of the wavelet.
- `bstep`: size (in the time direction) of the steps for chaining.
- `ptile`: relative threshold of occupation measure.
- `epsilon`: constant in front of the smoothness term in penalty function.
- `fast`: if set to TRUE, uses trapezoidal rule to evaluate $Q_2$.
- `para`: scale parameter for extrapolating the ridges.
- `real`: if set to TRUE, uses only real constraints.
- `plot`: 1: displays signal, components, and reconstruction one after another. 2: displays signal, components and reconstruction.

**Details**
When ptile is high, boundary effects may appear. para controls extrapolation of the ridge.

**Value**
Returns a structure containing the following elements:
- `rec`: reconstructed signal.
- `ordered`: image of the ridges (with different colors).
- `comp`: 2D array containing the signals reconstructed from ridges.
crfview

See Also

crc, cfamily, scrcrec.

crfview

Display chained ridges

Description

displays a family of chained ridges, output of cfamily.

Usage

crfview(beemap, twod=TRUE)

Arguments

beemap
Family of chained ridges, output of cfamily.

twod
If set to T, displays the ridges as an image. If set to F, displays as a series of curves.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

crc, cfamily for crazy climbers and corresponding chaining algorithms.

cwt

Continuous Wavelet Transform

Description

Computes the continuous wavelet transform with for the (complex-valued) Morlet wavelet.

Usage

cwt(input, noctave, nvoice=1, w0=2 * pi, twoD=TRUE, plot=TRUE)
Arguments

- **input**  
  input signal (possibly complex-valued)
- **noctave**  
  number of powers of 2 for the scale variable
- **nvoice**  
  number of scales in each octave (i.e. between two consecutive powers of 2).
- **\( w_0 \)**  
  central frequency of the wavelet.
- **twoD**  
  logical variable set to T to organize the output as a 2D array (\( \text{signal\_size} \times \text{nb\_scales} \)), otherwise, the output is a 3D array (\( \text{signal\_size} \times \text{noctave} \times \text{nvoice} \)).
- **plot**  
  if set to T, display the modulus of the continuous wavelet transform on the graphic device.

Details

The time series is padded with zeroes to avoid problems with circular versus linear convolution. This does not affect usage, as the matrix returned has the added columns removed. (JML Sep 29, 2021).

The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be

- 2D array (\( \text{signal\_size} \times \text{nb\_scales} \))
- 3D array (\( \text{signal\_size} \times \text{noctave} \times \text{nvoice} \))

Since Morlet's wavelet is not strictly speaking a wavelet (it is not of vanishing integral), artifacts may occur for certain signals.

Value

continuous (complex) wavelet transform

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

cwt, cwtTh, DOG, gabor.

Examples

```r
x <- 1:512  
chirp <- sin(2*pi * (x + 0.002 * (x-256)^2) / 16)  
retChirp <- cwt(chirp, noctave=5, nvoice=12)
```
**Continuous Wavelet Transform Display**

**Description**
Converts the output (modulus or argument) of cwtpolar to a 2D array and displays on the graphic device.

**Usage**
cwtimage(input)

**Arguments**
- **input**: 3D array containing a continuous wavelet transform

**Details**
The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be:
- 2D array (signal_size x nb_scales)
- 3D array (signal_size x noctave x nvoice)

**Value**
2D array continuous (complex) wavelet transform

**References**
See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**
cwtpolar, cwt, DOG.

**Examples**
```r
x <- 1:512
chirp <- sin(2*pi*x + 0.002*(x-256)^2) / 16
retChirp <- cwt(chirp, noctave=5, nvoice=12, twoD=FALSE, plot=FALSE)
retPolar <- cwtpolar(retChirp)
retImageMod <- cwtimage(retPolar$modulus)
retImageArg <- cwtimage(retPolar$argument)
```
Continuous Wavelet Transform with Phase Derivative

Description

Computes the continuous wavelet transform with (complex-valued) Morlet wavelet and its phase derivative.

Usage

cwtp(input, noctave, nvoice=1, w0=2 * pi, twoD=TRUE, plot=TRUE)

Arguments

- **input**: input signal (possibly complex-valued)
- **noctave**: number of powers of 2 for the scale variable
- **nvoice**: number of scales in each octave (i.e., between two consecutive powers of 2).
- **w0**: central frequency of the wavelet.
- **twoD**: logical variable set to `T` to organize the output as a 2D array (signal size × nb scales), otherwise, the output is a 3D array (signal size × noctave × nvoice).
- **plot**: if set to `TRUE`, display the modulus of the continuous wavelet transform on the graphic device.

Value

- list containing the continuous (complex) wavelet transform and the phase derivative
  - `wt`: array of complex numbers for the values of the continuous wavelet transform.
  - `f`: array of the same dimensions containing the values of the derivative of the phase of the continuous wavelet transform.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

cgt, cwt, cwtTh, DOG for wavelet transform, and gabor for continuous Gabor transform.

Examples

```r
## discards imaginary part with error,  
## c code does not account for Im(input)
   x <- 1:512
   chirp <- sin(2*pi * (x + 0.002 * (x-256)^2 ) / 16)  
   chirp <- chirp + 1i * sin(2*pi * (x + 0.004 * (x-256)^2 ) / 16)
   retChirp <- cwtp(chirp, noctave=5, nvoice=12)
```
Conversion to Polar Coordinates

Description

Converts one of the possible outputs of the function `cwt` to modulus and phase.

Usage

`cwtpolar(cwt, threshold=0)`

Arguments

- `cwt`: 3D array containing the values of a continuous wavelet transform in the format (signal size × noctave × nvoice) as in the output of the function `cwt` with the logical flag `twodimension` set to FALSE.
- `threshold`: value of a level for the absolute value of the modulus below which the value of the argument of the output is set to $-\pi$.

Details

The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be

- 2D array (signal size × nb_scales)
- 3D array (signal size × noctave × nvoice)

Value

Modulus and Argument of the values of the continuous wavelet transform

- `output1`: 3D array giving the values (in the same format as the input) of the modulus of the input.
- `output2`: 3D array giving the values of the argument of the input.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

`cwt`, `DOG`, `cwtimage`.

Examples

```r
x <- 1:512
cirp <- sin(2*pi * (x + 0.002 * (x-256)^2 ) / 16)
retChirp <- cwt(chirp, noctave=5, nvoice=12, twoD=FALSE, plot=FALSE)
retPolar <- cwtpolar(retChirp)
```
cwtsquiz  

Squeezed Continuous Wavelet Transform

**Description**

Computes the synchrosqueezed continuous wavelet transform with the (complex-valued) Morlet wavelet.

**Usage**

```
cwtsquiz(input, noctave, nvoice=1, w0=2 * pi, twoD=TRUE, plot=TRUE)
```

**Arguments**

- `input` input signal (possibly complex-valued)
- `noctave` number of powers of 2 for the scale variable
- `nvoice` number of scales in each octave (i.e. between two consecutive powers of 2).
- `w0` central frequency of the wavelet.
- `twoD` logical variable set to T to organize the output as a 2D array (signal size × nb scales), otherwise, the output is a 3D array (signal size × noctave × nvoice).
- `plot` logical variable set to T to T to display the modulus of the squeezed wavelet transform on the graphic device.

**Details**

The output contains the (complex) values of the squeezed wavelet transform of the input signal. The format of the output can be

- 2D array (signal size × nb scales),
- 3D array (signal size × noctave × nvoice).

**Value**

synchrosqueezed continuous (complex) wavelet transform

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

`cwt, cwtp, DOG, cgt`. 
Description

Compute the continuous wavelet transform with (complex-valued) Cauchy’s wavelet.

Usage

cwtTh(input, noctave, nvoice=1, moments, twoD=TRUE, plot=TRUE)

Arguments

input       input signal (possibly complex-valued).
noctave     number of powers of 2 for the scale variable.
nvoice      number of scales in each octave (i.e. between two consecutive powers of 2).
moments     number of vanishing moments.
twoD        logical variable set to T to organize the output as a 2D array (signal size x nb scales), otherwise, the output is a 3D array (signal size x noctave x nvoice).
plot        if set to T, display the modulus of the continuous wavelet transform on the graphic device.

Details

The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be

2D array (signal size x nb scales)
3D array (signal size x noctave x nvoice)

Value

tmp          continuous (complex) wavelet transform.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

cwt, cwtp, DOG, gabor.

Examples

```
x <- 1:512
cirp <- sin(2*pi * (x + 0.002 * (x-256)^2 ) / 16)
retChirp <- cwtTh(chirp, noctave=5, nvoice=12, moments=20)
```
**D0**

**Transient Signal**

**Description**

Transient signal.

**Usage**

```r
data(D0)
```

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(D0)
plot.ts(D0)
```

---

**D4**

**Transient Signal**

**Description**

Transient signal.

**Usage**

```r
data(D4)
```

**Format**

A vector containing 1024 observations.
**DOG**

**Source**
See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

**Examples**

```r
data(D4)
plot.ts(D4)
```

---

**DOG**

*Continuous Wavelet Transform with derivative of Gaussian*

**Description**
Computes the continuous wavelet transform with for (complex-valued) derivative of Gaussian wavelets.

**Usage**

```r
DOG(input, noctave, nvoice=1, moments, twoD=TRUE, plot=TRUE)
```

**Arguments**

- **input**: input signal (possibly complex-valued).
- **noctave**: number of powers of 2 for the scale variable.
- **moments**: number of vanishing moments of the wavelet (order of the derivative).
- **nvoice**: number of scales in each octave (i.e. between two consecutive powers of 2)
- **twoD**: logical variable set to T to organize the output as a 2D array (signal_size x nb_scales), otherwise, the output is a 3D array (signal_size x noctave x nvoice)
- **plot**: if set to T, display the modulus of the continuous wavelet transform on the graphic device

**Details**
The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be

- 2D array (signal_size x nb_scales)
- 3D array (signal_size x noctave x nvoice)
dwinverse

Value

continuous (complex) wavelet transform

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

cwt, cwtp, cwtsquiz, cgt.

Examples

```r
x <- 1:512
chirp <- sin(2*pi * (x + 0.002 * (x-256)^2 ) / 16)
DOG(chirp, no octave=5, n voice=12, 3, twoD=TRUE, plot=TRUE)
```

dwinverse Inverse Dyadic Wavelet Transform

Description

Invert the dyadic wavelet transform.

Usage

dwinverse(wt, filtername="Gaussian1")

Arguments

wt dyadic wavelet transform

filtername filters used. ("Gaussian1" stands for the filters corresponds to those of Mallat and Zhong’s wavlet. And "Haar" stands for the filters of Haar basis.

Value

Reconstructed signal

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

mw, ext, mrecons.
**Ekg**

*Heart Rate Data*

**Description**

Successive beat-to-beat intervals for a normal patient.

**Usage**

```r
data(Ekg)
```

**Format**

A vector containing 16,042 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(Ekg)
plot.ts(Ekg)
```

**epl**

*Plot Dyadic Wavelet Transform Extrema*

**Description**

Plot dyadic wavelet transform extrema (output of `ext`).

**Usage**

```r
epl(dwext)
```

**Arguments**

`dwext` dyadic wavelet transform (output of `ext`).
References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

mw, ext, wpl.

---

ext (for Extrema of Dyadic Wavelet Transform)

Description

Compute the local extrema of the dyadic wavelet transform modulus.

Usage

```
ext(wt, scale=FALSE, plot=TRUE)
```

Arguments

- `wt`: dyadic wavelet transform.
- `scale`: flag indicating if the extrema at each resolution will be plotted at the same scale.
- `plot`: if set to TRUE, displays the transform on the graphics device.

Value

Structure containing:

- `original`: original signal.
- `extrema`: extrema representation.
- `Sf`: coarse resolution of signal.
- `maxresoln`: number of decomposition scales.
- `np`: size of signal.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

mw, mrecons.
**Description**

Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal, using simple trapezoidal rule for integrals.

**Usage**

```r
fastgkernel(node, phinode, freqstep, scale, x.inc=1, x.min=node[1], x.max=node[length(node)], plot=FALSE)
```

**Arguments**

- `node`: values of the variable `b` for the nodes of the ridge
- `phinode`: values of the frequency variable `ω` for the nodes of the ridge
- `freqstep`: sampling rate for the frequency axis
- `scale`: size of the window
- `x.inc`: step unit for the computation of the kernel.
- `x.min`: minimal value of `x` for the computation of $G_2$.
- `x.max`: maximal value of `x` for the computation of $G_2$.
- `plot`: if set to TRUE, displays the modulus of the matrix of $G_2$.

**Details**

Uses trapezoidal rule (instead of Romberg’s method) to evaluate the kernel.

**Value**

matrix of the $G_2$ kernel.

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

`gkernel`, `fastkernel`, `rkernel`, `zerokernel`. 
Description

Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal, using simple trapezoidal rule for integrals.

Usage

```r
fastkernel(node, phinode, nvoice, x.inc=1, x.min=node[1], x.max=node[length(node)], w0=2 * pi, plot=FALSE)
```

Arguments

- `node`: values of the variable b for the nodes of the ridge.
- `phinode`: values of the scale variable a for the nodes of the ridge.
- `nvoice`: number of scales within 1 octave.
- `x.inc`: step unit for the computation of the kernel.
- `x.min`: minimal value of x for the computation of $Q_2$.
- `x.max`: maximal value of x for the computation of $Q_2$.
- `w0`: central frequency of the wavelet.
- `plot`: if set to TRUE, displays the modulus of the matrix of $Q_2$.

Details

Uses trapezoidal rule (instead of Romberg’s method) to evaluate the kernel.

Value

Matrix of the $Q_2$ kernel.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

`kernel`, `rkernel`, `gkernel`, `zerokernel`. 

fastkernel

*Kernel for Reconstruction from Wavelet Ridges*
**gabor**

*Generate Gabor function*

**Description**

Generates a Gabor for given location and frequency.

**Usage**

`gabor(sigsize, location, frequency, scale)`

**Arguments**

- `sigsize`: length of the Gabor function.
- `location`: position of the Gabor function.
- `frequency`: frequency of the Gabor function.
- `scale`: size parameter for the Gabor function. See details.

**Details**


**Value**

complex 1D array of size `sigsize`.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

`morlet`.

**Examples**

```r
m1 = gabor(1024, 512, 2 * pi, 20 )
plot.ts(Re(m1) )
```
gcrcrec

Crazy Climbers Reconstruction by Penalization

Description

Reconstructs a real-valued signal from ridges found by crazy climbers on a Gabor transform.

Usage

gcrcrec(siginput, inputgt, beemap, nvoice, freqstep, scale, compr,
  bstep=5, ptile=0.01, epsilon=0, fast=TRUE, para=5, minnbnodes=3,
  hflag=FALSE, real=FALSE, plot=2)

Arguments

  siginput  original signal.
  inputgt   Gabor transform.
  beemap    occupation measure, output of crc.
  nvoice    number of frequencies.
  freqstep  sampling step for frequency axis.
  scale     size of windows.
  compr     compression rate to be applied to the ridges.
  bstep     size (in the time direction) of the steps for chaining.
  ptile     threshold of ridge
  epsilon   constant in front of the smoothness term in penalty function.
  fast      if set to TRUE, uses trapezoidal rule to evaluate $Q_2$.
  para      scale parameter for extrapolating the ridges.
  minnbnodes minimal number of points per ridge.
  hflag     if set to FALSE, uses the identity as first term in the kernel. If not, uses $Q_1$ instead.
  real      if set to TRUE, uses only real constraints.
  plot      1 displays signal, components, and reconstruction one after another.
             2 displays signal, components and reconstruction.

Details

When ptile is high, boundary effects may appear. para controls extrapolation of the ridge.

Value

Returns a structure containing the following elements:

  rec     reconstructed signal.
  ordered image of the ridges (with different colors).
  comp    2D array containing the signals reconstructed from ridges.
gkernel

Kernel for Reconstruction from Gabor Ridges

Description

Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal.

Usage

gkernel(node, phinode, freqstep, scale, x.inc=1, x.min=node[1], x.max=node[length(node)], plot=FALSE)

Arguments

- node: values of the variable b for the nodes of the ridge.
- phinode: values of the scale variable a for the nodes of the ridge.
- freqstep: sampling rate for the frequency axis.
- scale: size of the window.
- x.inc: step unit for the computation of the kernel.
- x.min: minimal value of x for the computation of $Q_2$.
- x.max: maximal value of x for the computation of $Q_2$.
- plot: if set to TRUE, displays the modulus of the matrix of $Q_2$.

Value

matrix of the $Q_2$ kernel

References

See discussions in the text of “Time-Frequency Analysis”.

See Also

fastgkernel, kernel, rkernell, fastkernel, zerokernel.
gregrec  Reconstruction from a Ridge

Description

Reconstructs signal from a “regularly sampled” ridge, in the Gabor case.

Usage

```r
gregrec(siginput, gtinput, phi, nbnodes, nvoice, freqstep, scale, epsilon=0, fast=FALSE, plot=FALSE, para=0, hflag=FALSE, real=FALSE, check=FALSE)
```

Arguments

- `siginput`: input signal.
- `gtinput`: Gabor transform, output of `cgt`.
- `phi`: unsampled ridge.
- `nbnodes`: number of nodes used for the reconstruction.
- `nvoice`: number of different scales per octave
- `freqstep`: sampling rate for the frequency axis
- `scale`: size parameter for the Gabor function.
- `epsilon`: coefficient of the $Q_2$ term in reconstruction kernel
- `fast`: if set to T, the kernel is computed using trapezoidal rule.
- `plot`: if set to TRUE, displays original and reconstructed signals
- `para`: scale parameter for extrapolating the ridges.
- `hflag`: if set to TRUE, uses $Q_1$ as first term in the kernel.
- `real`: if set to TRUE, uses only real constraints on the transform.
- `check`: if set to TRUE, computes `cwt` of reconstructed signal.

Value

Returns a list containing:

- `sol`: reconstruction from a ridge.
- `A`: `<gaborlets,dualgaborlets>` matrix.
- `lam`: coefficients of dual wavelets in reconstructed signal.
- `dualwave`: array containing the dual wavelets.
- `gaborets`: array containing the wavelets on sampled ridge.
- `solskel`: Gabor transform of sol, restricted to the ridge.
- `inputskel`: Gabor transform of signal, restricted to the ridge.
- `Q2`: second part of the reconstruction kernel.
References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

regrec.

---

gridrec Reconstruction from a Ridge

Description

Reconstructs signal from sample of a ridge, in the Gabor case.

Usage

gridrec(gtinput, node, phinode, nvoice, freqstep, scale, Qinv, epsilon, np, real=FALSE, check=FALSE)

Arguments

gtinput
node
phinode
nvoice
freqstep
scale
Qinv
epsilon
np
real
check

Value

Returns a list containing the reconstructed signal and the chained ridges.

sol
A
lam
dualwave
gaborlets
solskel
inputskskel
References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

sridrec, gregrec, regrec, regrec2.

---

gsampleOne  Sampled Identity

Description

Generate a sampled identity matrix.

Usage

gsampleOne(node, scale, np)

Arguments

node location of the reconstruction gabor functions.

scale scale of the gabor functions.

np size of the reconstructed signal.

Value

diagonal of the “sampled” $Q_1$ term (1D vector)

References

See discussions in the text of “Time-Frequency Analysis”.

See Also

kernel, gkernel.
**gwave**

*Gabor Functions on a Ridge*

**Description**

Generation of Gabor functions located on the ridge.

**Usage**

```r
gwave(bridge, omegaridge, nvoice, freqstep, scale, np, N)
```

**Arguments**

- `bridge`: time coordinates of the ridge samples
- `omegaridge`: frequency coordinates of the ridge samples
- `nvoice`: number of different scales per octave
- `freqstep`: sampling rate for the frequency axis
- `scale`: scale of the window
- `np`: size of the reconstruction kernel
- `N`: number of complex constraints

**Value**

Array of Gabor functions located on the ridge samples

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

- `gwave2`, `morwave`, `morwave2`.

---

**gwave2**

*Real Gabor Functions on a Ridge*

**Description**

Generation of the real parts of gabor functions located on a ridge. (modification of `gwave`.)

**Usage**

```r
gwave2(bridge, omegaridge, nvoice, freqstep, scale, np, N)
```

---
Arguments

- bridge: time coordinates of the ridge samples
- omegaridge: frequency coordinates of the ridge samples
- nvoice: number of different scales per octave
- freqstep: sampling rate for the frequency axis
- scale: scale of the window
- np: size of the reconstruction kernel
- N: number of complex constraints

Value

Array of real Gabor functions located on the ridge samples

References

See discussions in the text of “Time-Frequency Analysis”.

See Also

gwave, morwave, morwave2.

HeartRate

Pixel from Amber Camara

Description

Pixel from amber camara.

Usage

data(HeartRate)

Format

A vector containing observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References

Examples

```
data(HeartRate)
plot.ts(HeartRate)
```

Description

Example of speech signal.

Usage

```
data(HOWAREYOU)
```

Format

A vector containing 5151 observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

```
data(HOWAREYOU)
plot.ts(HOWAREYOU)
```
hurst.est  

Estimate Hurst Exponent

Description

Estimates Hurst exponent from a wavelet transform.

Usage

hurst.est(wspec, range, nvoice, plot=TRUE)

Arguments

wspec  
wavelet spectrum (output of tfmean)

range  
range of scales from which estimate the exponent.

nvoice  
number of scales per octave of the wavelet transform.

plot  
if set to TRUE, displays regression line on current plot.

Value

complex 1D array of size sigsize.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

tfmean, wspec.pl.

Examples

# White Noise Hurst Exponent: The plots on the top row of Figure 6.8  
# were produced by the falling S-commands. These make use of the two  
# functions Hurst.est (estimation of Hurst exponent from CWT) and  
# wspec.pl (display wavelet spectrum).

# Compare the periodogram and the wavelet spectral estimate.
wnoise <- rnorm(8192)
plot.ts(wnoise)
spwnoise <- fft(wnoise)
spwnoise <- Mod(spwnoise)
spwnoise <- spwnoise*spwnoise
plot(spwnoise[1:4096], log="xy", type="l")
lswnoise <- lsfit(log10(1:4096), log10(spwnoise[1:4096]))
abline(lswnoise$coef)
cwtwnoise <- DOG(wnoise, 10, 5, 1, plot=FALSE)
mcwtwnoise <- Mod(cwtwnoise)
mcwtwnoise <- mcwtwnoise * mcwtwnoise
wspwnoise <- tfmean(mcwtwnoise, plot=FALSE)
wspec.pl(wspwnoise, 5)
hurst.est(wspwnoise, 1:50, 5)

---

**Description**

Estimate a (single) ridge from a time-frequency representation, using the ICM minimization method.

**Usage**

```r
icm(modulus, guess, tfspec=numeric(dim(modulus)[2]), subrate=1,
    mu=1, lambda=2 * mu, iteration=100)
```

**Arguments**

- **modulus**: Time-Frequency representation (real valued).
- **guess**: Initial guess for the algorithm.
- **tfspec**: Estimate for the contribution of the noise to modulus.
- **subrate**: Subsampling rate for ridge estimation.
- **mu**: Coefficient of the ridge’s second derivative in cost function.
- **lambda**: Coefficient of the ridge’s derivative in cost function.
- **iteration**: Maximal number of moves.

**Details**

To accelerate convergence, it is useful to preprocess modulus before running annealing method. Such a preprocessing (smoothing and subsampling of modulus) is implemented in `icm`. The parameter `subrate` specifies the subsampling rate.

**Value**

Returns the estimated ridge and the cost function.

- **ridge**: 1D array (of same length as the signal) containing the ridge.
- **cost**: 1D array containing the cost function.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

- `corona`, `coronoid`, and `snake`, `snakoid`.
mbtrim  
*Trim Dyadic Wavelet Transform Extrema*

**Description**
Trimming of dyadic wavelet transform local extrema, using bootstrapping.

**Usage**
`mbtrim(extrema, scale=FALSE, prct=0.95)`

**Arguments**
- `extrema`: dyadic wavelet transform extrema (output of `ext`).
- `scale`: when set, the wavelet transform at each scale will be plotted with the same scale.
- `prct`: percentage critical value used for thresholding

**Details**
The distribution of extrema of dyadic wavelet transform at each scale is generated by bootstrap method, and the 95% critical value is used for thresholding the extrema of the signal.

**Value**
Structure containing
- `original`: original signal.
- `extrema`: trimmed extrema representation.
- `Sf`: coarse resolution of signal.
- `maxresoln`: number of decomposition scales.
- `np`: size of signal.

**References**
See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**
mntrim, mrecons, ext.
**mntrim**

**Trim Dyadic Wavelet Transform Extrema**

**Description**

Trimming of dyadic wavelet transform local extrema, assuming normal distribution.

**Usage**

```r
mntrim(extrema, scale=FALSE, prct=0.95)
```

**Arguments**

- `extrema`: dyadic wavelet transform extrema (output of `ext`).
- `scale`: when set, the wavelet transform at each scale will be plotted with the same scale.
- `prct`: percentage critical value used for thresholding

**Details**

The distribution of extrema of dyadic wavelet transform at each scale is generated by simulation, assuming a normal distribution, and the 95% critical value is used for thresholding the extrema of the signal.

**Value**

Structure containing

- `original`: original signal.
- `extrema`: trimmed extrema representation.
- `Sf`: coarse resolution of signal.
- `maxresoln`: number of decomposition scales.
- `np`: size of signal.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

`mbtrim, mrecons, ext`
morlet  Morlet Wavelets

Description

Computes a Morlet wavelet at the point of the time-scale plane given in the input

Usage

morlet(sigsize, location, scale, w0=2 * pi)

Arguments

- sigsize: length of the output.
- location: time location of the wavelet.
- scale: scale of the wavelet.
- w0: central frequency of the wavelet.

Details

The details of this construction (including the definition formulas) are given in the text.

Value

Returns the values of the complex Morlet wavelet at the point of the time-scale plane given in the input

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

gabor.

Examples

m1 = morlet(1024, 512, 20, w0=2 * pi)
plot.ts(Re(m1) )
**morwave**

*Ridge Morvelets*

**Description**
Generates the Morlet wavelets at the sample points of the ridge.

**Usage**
```
morwave(bridge, aridge, nvoice, np, N, w0=2 * pi)
```

**Arguments**
- **bridge**: time coordinates of the ridge samples.
- **aridge**: scale coordinates of the ridge samples.
- **nvoice**: number of different scales per octave.
- **np**: number of samples in the input signal.
- **N**: size of reconstructed signal.
- **w0**: central frequency of the wavelet.

**Value**
Returns the Morlet wavelets at the samples of the time-scale plane given in the input: complex array of Morlet wavelets located on the ridge samples.

**References**
See discussions in the text of “Time-Frequency Analysis”.

**See Also**
- `morwave2`, `gwave`, `gwave2`.

---

**morwave2**

*Real Ridge Morvelets*

**Description**
Generates the real parts of the Morlet wavelets at the sample points of a ridge.

**Usage**
```
morwave2(bridge, aridge, nvoice, np, N, w0=2 * pi)
```
Arguments

bridge  time coordinates of the ridge samples.
ridge  scale coordinates of the ridge samples.
nvoice  number of different scales per octave.
np  number of samples in the input signal.
N  size of reconstructed signal.
w0  central frequency of the wavelet.

Value

Returns the real parts of the Morlet wavelets at the samples of the time-scale plane given in the input: array of Morlet wavelets located on the ridge samples.

References

See discussions in the text of “Time-Frequency Analysis”.

See Also

morwave, gwave, gwave2.

mrecons  Reconstruct from Dyadic Wavelet Transform Extrema

Description

Reconstruct from dyadic wavelet transform modulus extrema. The reconstructed signal preserves locations and values at extrema.

Usage

mrecons(extrema, filtername="Gaussian1", readflag=FALSE)

Arguments

extrema  the extrema representation.
filtername  filter used for dyadic wavelet transform.
readflag  if set to T, read reconstruction kernel from precomputed file.

Details

The reconstruction involves only the wavelet coefficients, without taking care of the coarse scale component. The latter may be added a posteriori.
**Description**

Dyadic wavelet transform, with Mallat’s wavelet. The reconstructed signal preserves locations and values at extrema.

**Usage**

`mw(inputdata, maxresoln, filtername="Gaussian1", scale=FALSE, plot=TRUE)`

**Arguments**

- `inputdata`: either a text file or an R object containing data.
- `maxresoln`: number of decomposition scales.
- `filtername`: name of filter (either Gaussian1 for Mallat and Zhong’s wavelet or Haar wavelet).
- `scale`: when set, the wavelet transform at each scale is plotted with the same scale.
- `plot`: indicate if the wavelet transform at each scale will be plotted.

**Details**

The decomposition goes from resolution 1 to the given maximum resolution.

**Value**

Structure containing

- `original`: original signal.
- `Wf`: dyadic wavelet transform of signal.
- `Sf`: multiresolution of signal.
- `maxresoln`: number of decomposition scales.
- `np`: size of signal.
noisy.dat

References
See discussions in the text of “Practical Time-Frequency Analysis”.

See Also
dwinverse, mrecons, ext.

noisy.dat       Pixel from Amber Camara

Description
Pixel from amber camara.

Usage
data(noisy.dat)

Format
A vector containing observations.

Source
See discussions in the text of “Practical Time-Frequency Analysis”.

References

Examples
data(noisy.dat)
plot.ts(noisy.dat)
noisywave

Description

Noisy gravitational wave.

Usage

data(noisywave)

Format

A vector containing 8192 observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(noisywave)
pplot.ts(noisywave)

npl

Prepare Graphics Environment

Description

Splits the graphics device into prescribed number of windows.

Usage

npl(nbrow)

Arguments

nbrow number of plots.
**pixel_8.7**

*Pixel from Amber Camara*

**Description**

Pixel from amber camara.

**Usage**

```r
data(pixel_8.7)
```

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(pixel_8.7)
plot.ts(pixel_8.7)
```

---

**pixel_8.8**

*Pixel from Amber Camara*

**Description**

Pixel from amber camara.

**Usage**

```r
data(pixel_8.8)
```

**Format**

A vector containing observations.
Pixel from Amber Camara

Description

Pixel from amber camara.

Usage

data(pixel_8.9)

Format

A vector containing observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(pixel_8.9)
plot.ts(pixel_8.9)
**plotwt**

*Plot Dyadic Wavelet Transform*

**Description**

Plot dyadic wavelet transform.

**Usage**

plotwt(original, psi, phi, maxresoln, scale=FALSE, yaxtype="s")

**Arguments**

- original: input signal.
- psi: dyadic wavelet transform.
- phi: scaling function transform at last resolution.
- maxresoln: number of decomposition scales.
- scale: when set, the wavelet transform at each scale is plotted with the same scale.
- yaxtype: axis type (see R manual).

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

plotwt, epl, wpl.

---

**plotResult**

*Plot Dyadic Wavelet Transform Extrema*

**Description**

Plot extrema of dyadic wavelet transform.

**Usage**

plotResult(result, original, maxresoln, scale=FALSE, yaxtype="s")

**Arguments**

- result: result.
- original: input signal.
- maxresoln: number of decomposition scales.
- scale: when set, the extrema at each scale is plotted with the same scale.
- yaxtype: y axis type (see R manual).

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

plotwt, epl, wpl.
pure.dat

References

See discussions in the text of “Time-Frequency Analysis”.

See Also

plotResult, epl, wpl.

pure.dat

Pixel from Amber Camara

Description

Pixel from amber camara.

Usage

data(pure.dat)

Format

A vector containing observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(pure.dat)
plot.ts(pure.dat)
**purwave**

*Pure Gravitational Wave*

**Description**

Pure gravitational wave.

**Usage**

```r
data(purwave)
```

**Format**

A vector containing 8192 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(purwave)
plot.ts(purwave)
```

---

**regrec**

*Reconstruction from a Ridge*

**Description**

Reconstructs signal from a “regularly sampled” ridge, in the wavelet case.

**Usage**

```r
regrec(siginput, cwtinput, phi, compr, noct, nvoice, epsilon=0, w0=2 * pi, fast=FALSE, plot=FALSE, para=0, hflag=FALSE, check=FALSE, minnbnodes=2, real=FALSE)
```
Arguments

- `siginput`: input signal.
- `cwtinput`: wavelet transform, output of `cwt`.
- `phi`: unsampled ridge.
- `compr`: subsampling rate for the wavelet coefficients (at scale 1).
- `noct`: number of octaves (powers of 2).
- `nvoice`: number of different scales per octave.
- `epsilon`: coefficient of the $Q_2$ term in reconstruction kernel.
- `w0`: central frequency of Morlet wavelet.
- `fast`: if set to TRUE, the kernel is computed using trapezoidal rule.
- `plot`: if set to TRUE, displays original and reconstructed signals.
- `para`: scale parameter for extrapolating the ridges.
- `hflag`: if set to TRUE, uses $Q_1$ as first term in the kernel.
- `check`: if set to TRUE, computes `cwt` of reconstructed signal.
- `minnbnodes`: minimum number of nodes for the reconstruction.
- `real`: if set to TRUE, uses only real constraints on the transform.

Value

Returns a list containing:

- `sol`: reconstruction from a ridge.
- `A`: `<wavelets,dualwavelets>` matrix.
- `lam`: coefficients of dual wavelets in reconstructed signal.
- `dualwave`: array containing the dual wavelets.
- `morvelets`: array containing the wavelets on sampled ridge.
- `solskel`: wavelet transform of `sol`, restricted to the ridge.
- `inputskel`: wavelet transform of signal, restricted to the ridge.
- `Q2`: second part of the reconstruction kernel.
- `nbnodes`: number of nodes used for the reconstruction.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

`regrec2`, `ridrec`, `gregrec`, `gridrec`. 
Reconstruction from a Ridge

Description
Reconstructs signal from a “regularly sampled” ridge, in the wavelet case, from a precomputed kernel.

Usage
`regrec2(siginput, cwtinput, phi, nbnodes, noct, nvoice, Q2, epsilon=0.5, w0=2 * pi, plot=FALSE)`

Arguments
- `siginput`: input signal.
- `cwtinput`: wavelet transform, output of `cwt`.
- `phi`: unsampled ridge.
- `nbnodes`: number of samples on the ridge.
- `noct`: number of octaves (powers of 2).
- `nvoice`: number of different scales per octave.
- `Q2`: second term of the reconstruction kernel.
- `epsilon`: coefficient of the \( Q_2 \) term in reconstruction kernel.
- `w0`: central frequency of Morlet wavelet.
- `plot`: if set to TRUE, displays original and reconstructed signals.

Details
The computation of the kernel may be time consuming. This function avoids recomputing it if it was computed already.

Value
Returns a list containing:
- `sol`: reconstruction from a ridge.
- `A`: <wavelets,dualwavelets> matrix.
- `lam`: coefficients of dual wavelets in reconstructed signal.
- `dualwave`: array containing the dual wavelets.
- `morselets`: array containing the wavelets on sampled ridge.
- `solskel`: wavelet transform of sol, restricted to the ridge.
- `inputskel`: wavelet transform of signal, restricted to the ridge.
- `nbnodes`: number of nodes used for the reconstruction.
RidgeSampling

References
See discussions in the text of “Practical Time-Frequency Analysis”.

See Also
regrec, gregrec, ridrec, sridrec.

Description
Given a ridge phi (for the Gabor transform), returns a (regularly) subsampled version of length nbnodes.

Usage
RidgeSampling(phi, nbnodes)

Arguments
phi ridge (1D array).
nbnodes number of samples.

Details
Gabor ridges are sampled uniformly.

Value
Returns a list containing the discrete values of the ridge.

node time coordinates of the ridge samples.
phinode frequency coordinates of the ridge samples.

References
See discussions in the text of “Time-Frequency Analysis”.

See Also
wRidgeSampling.
ridrec  Reconstruction from a Ridge

Description
Reconstructs signal from sample of a ridge, in the wavelet case.

Usage
ridrec(cwtinput, node, phinode, noct, nvoice, Qinv, epsilon, np,
w0=2 * pi, check=FALSE, real=FALSE)

Arguments
cwtinput wavelet transform, output of \textit{cwt}.
node time coordinates of the ridge samples.
phinode scale coordinates of the ridge samples.
noct number of octaves (powers of 2).
nvoice number of different scales per octave.
Qinv inverse of the matrix $Q$ of the quadratic form.
epsilon coefficient of the $Q_2$ term in reconstruction kernel
np number of samples of the reconstructed signal.
w0 central frequency of Morlet wavelet.
check if set to TRUE, computes \textit{cwt} of reconstructed signal.
real if set to TRUE, uses only constraints on the real part of the transform.

Value
Returns a list containing the reconstructed signal and the chained ridges.
sol reconstruction from a ridge
A <\text{wavelets,dualwavelets}> matrix
lam coefficients of dual wavelets in reconstructed signal.
dualwave array containing the dual wavelets.
morvelets array of morlet wavelets located on the ridge samples.
solskel wavelet transform of sol, restricted to the ridge
inputskel wavelet transform of signal, restricted to the ridge

References
See discussions in the text of “Practical Time-Frequency Analysis”.

See Also
\texttt{sridrec, regrec, regrec2}.
rkernel

Kernel for Reconstruction from Wavelet Ridges

Description
Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal, in the case of real constraints. Modification of the function kernel.

Usage
rkernel(node, phinode, nvoice, x.inc=1, x.min=node[1], x.max=node[length(node)], w0=2 * pi, plot=FALSE)

Arguments
- node: values of the variable b for the nodes of the ridge.
- phinode: values of the scale variable a for the nodes of the ridge.
- nvoice: number of scales within 1 octave.
- x.inc: step unit for the computation of the kernel.
- x.min: minimal value of x for the computation of $Q_2$.
- x.max: maximal value of x for the computation of $Q_2$.
- w0: central frequency of the wavelet.
- plot: if set to TRUE, displays the modulus of the matrix of $Q_2$.

Details
Uses Romberg’s method for computing the kernel.

Value
matrix of the $Q_2$ kernel

References
See discussions in the text of "Time-Frequency Analysis".

See Also
kernel, fastkernel, gkernel, zerokernel.
rwkernel  

\textit{Kernel for Reconstruction from Wavelet Ridges}

\textbf{Description}

Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal

\textbf{Usage}

\begin{verbatim}
rwkernel(node, phinode, nvoice, x.inc=1, x.min=node[1], x.max=node[length(node)], w0=2 * pi, plot=FALSE)
\end{verbatim}

\textbf{Arguments}

- \texttt{node} values of the variable \(b\) for the nodes of the ridge.
- \texttt{phinode} values of the scale variable \(a\) for the nodes of the ridge.
- \texttt{nvoice} number of scales within 1 octave.
- \texttt{x.inc} step unit for the computation of the kernel.
- \texttt{x.min} minimal value of \(x\) for the computation of \(Q_2\).
- \texttt{x.max} maximal value of \(x\) for the computation of \(Q_2\).
- \texttt{w0} central frequency of the wavelet.
- \texttt{plot} if set to TRUE, displays the modulus of the matrix of \(Q_2\).

\textbf{Details}

The kernel is evaluated using Romberg’s method.

\textbf{Value}

matrix of the \(Q_2\) kernel

\textbf{References}

See discussions in the text of "Time-Frequency Analysis".

\textbf{See Also}

\texttt{gkernel}, \texttt{rkernel}, \texttt{zerokernel}. 

**Description**

Reconstructs signal from ridges obtained by *crc*, using the restriction of the transform to the ridge.

**Usage**

```
scrcrc(siginput, tfinput, beemap, bstep=5, ptile=0.01, plot=2)
```

**Arguments**

- **siginput**: input signal.
- **tfinput**: time-frequency representation (output of *cwt* or *cgt*).
- **beemap**: output of crazy climber algorithm.
- **bstep**: used for the chaining (see *cfamily*).
- **ptile**: threshold on the measure beemap (see *cfamily*).
- **plot**: 1: displays signal, components, and reconstruction one after another. 2: displays signal, components and reconstruction. Else, no plot.

**Value**

Returns a list containing the reconstructed signal and the chained ridges.

- **rec**: reconstructed signal
- **ordered**: image of the ridges (with different colors)
- **comp**: 2D array containing the signals reconstructed from ridges

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

*crc*, *cfamily* for crazy climbers method, *crcrec* for reconstruction methods.
**signal_W_tilda.1**  
*Pixel from Amber Camara*

**Description**  
Pixel from amber camara.

**Usage**  
```r  
data(signal_W_tilda.1)  
```

**Format**  
A vector containing observations.

**Source**  
See discussions in the text of “Practical Time-Frequency Analysis”.

**References**  

**Examples**  
```r  
data(signal_W_tilda.1)  
plot.ts(signal_W_tilda.1)  
```

---

**signal_W_tilda.2**  
*Pixel from Amber Camara*

**Description**  
Pixel from amber camara.

**Usage**  
```r  
data(signal_W_tilda.2)  
```

**Format**  
A vector containing observations.
Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(signal_W_tilda.2)
plot.ts(signal_W_tilda.2)

data(signal_W_tilda.3)
plot.ts(signal_W_tilda.3)

Description

Pixel from amber camara.

Usage

data(signal_W_tilda.3)

Format

A vector containing observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(signal_W_tilda.3)
plot.ts(signal_W_tilda.3)
**signal_W_tilda.4**

*Pixel from Amber Camara*

**Description**

Pixel from amber camara.

**Usage**

```r
data(signal_W_tilda.4)
```

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(signal_W_tilda.4)
plot.ts(signal_W_tilda.4)
```

---

**signal_W_tilda.5**

*Pixel from Amber Camara*

**Description**

Pixel from amber camara.

**Usage**

```r
data(signal_W_tilda.5)
```

**Format**

A vector containing observations.
signal_W_tilda.6

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(signal_W_tilda.5)
plot.ts(signal_W_tilda.5)

signal_W_tilda.6  Pixel from Amber Camara

Description

Pixel from amber camara.

Usage

data(signal_W_tilda.6)

Format

A vector containing observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(signal_W_tilda.6)
plot.ts(signal_W_tilda.6)
**signal_W_tilda.7**  
*Pixel from Amber Camara*

**Description**

Pixel from amber camara.

**Usage**

```r
data(signal_W_tilda.7)
```

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(signal_W_tilda.7)
plot.ts(signal_W_tilda.7)
```

**signal_W_tilda.8**  
*Pixel from Amber Camara*

**Description**

Pixel from amber camara.

**Usage**

```r
data(signal_W_tilda.8)
```

**Format**

A vector containing observations.
**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(signal_W_tilda.8)
plot.ts(signal_W_tilda.8)
```

---

**Description**

Pixel from amber camara.

**Usage**

```r
data(signal_W_tilda.9)
```

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(signal_W_tilda.9)
plot.ts(signal_W_tilda.9)
```
**Description**

Pixel from amber camara.

**Usage**

data(sig_W_tilda.1)

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

data(sig_W_tilda.1)
plot.ts(sig_W_tilda.1)

---

**Description**

Pixel from amber camara.

**Usage**

data(sig_W_tilda.2)

**Format**

A vector containing observations.
Source
See discussions in the text of “Practical Time-Frequency Analysis”.

References

Examples
```r
data(sig_W_tilda.2)
plot.ts(sig_W_tilda.2)
```

<table>
<thead>
<tr>
<th>sig_W_tilda.3</th>
<th>Pixel from Amber Camara</th>
</tr>
</thead>
</table>

Description
Pixel from amber camara.

Usage
```r
data(sig_W_tilda.3)
```

Format
A vector containing observations.

Source
See discussions in the text of “Practical Time-Frequency Analysis”.

References

Examples
```r
data(sig_W_tilda.3)
plot.ts(sig_W_tilda.3)
```
**Description**

Pixel from amber camara.

**Usage**

```r
data(sig_W_tilda.4)
```

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(sig_W_tilda.4)
plot.ts(sig_W_tilda.4)
```
Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References

Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S,

Examples

data(sig_W_tilda.5)
plot.ts(sig_W_tilda.5)

Description

Computes the reconstructed signal from the ridge, given the inverse of the matrix Q.

Usage

skeleton(cwtinput, Qinv, morvelets, bridge, aridge, N)

Arguments

cwtinput continuous wavelet transform (as the output of cwt)
Qinv inverse of the reconstruction kernel (2D array)
morvelets array of Morlet wavelets located at the ridge samples
bridge time coordinates of the ridge samples
aridge scale coordinates of the ridge samples
N size of reconstructed signal

Value

Returns a list of the elements of the reconstruction of a signal from sample points of a ridge

sol reconstruction from a ridge
A matrix of the inner products
lam coefficients of dual wavelets in reconstructed signal. They are the Lagrange multipliers λ's of the text.
dualwave array containing the dual wavelets.
References
See discussions in the text of “Practical Time-Frequency Analysis”.

See Also
skeleton2, zeroskeleton, zeroskeleton2.

skeleton2

Reconstruction from Dual Wavelet

Description
Computes the reconstructed signal from the ridge in the case of real constraints.

Usage
skeleton2(cwtinput, Qinv, morvelets, bridge, aridge, N)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwtinput</td>
<td>continuous wavelet transform (as the output of cwt).</td>
</tr>
<tr>
<td>Qinv</td>
<td>inverse of the reconstruction kernel (2D array).</td>
</tr>
<tr>
<td>morvelets</td>
<td>array of Morlet wavelets located at the ridge samples.</td>
</tr>
<tr>
<td>bridge</td>
<td>time coordinates of the ridge samples.</td>
</tr>
<tr>
<td>aridge</td>
<td>scale coordinates of the ridge samples.</td>
</tr>
<tr>
<td>N</td>
<td>size of reconstructed signal.</td>
</tr>
</tbody>
</table>

Value
Returns a list of the elements of the reconstruction of a signal from sample points of a ridge

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sol</td>
<td>reconstruction from a ridge.</td>
</tr>
<tr>
<td>A</td>
<td>matrix of the inner products.</td>
</tr>
<tr>
<td>lam</td>
<td>coefficients of dual wavelets in reconstructed signal. They are the Lagrange multipliers λ’s of the text.</td>
</tr>
<tr>
<td>dualwave</td>
<td>array containing the dual wavelets.</td>
</tr>
</tbody>
</table>

References
See discussions in the text of “Practical Time-Frequency Analysis”.

See Also
skeleton.
smoothts

**Smoothing Time Series**

**Description**
Smooth a time series by averaging window.

**Usage**
smoothts(ts, windowsize)

**Arguments**
- ts: Time series.
- windowsize: Length of smoothing window.

**Value**
Smoothed time series (1D array).

**References**
See discussions in the text of “Time-Frequency Analysis”.

smoothwt

**Smoothing and Time Frequency Representation**

**Description**
smooth the wavelet (or Gabor) transform in the time direction.

**Usage**
smoothwt(modulus, subrate, flag=FALSE)

**Arguments**
- modulus: Time-Frequency representation (real valued).
- subrate: Length of smoothing window.
- flag: If set to TRUE, subsample the representation.

**Value**
2D array containing the smoothed transform.
References

See discussions in the text of “Time-Frequency Analysis”.

See Also

corona, coronoid, snake, snakoid.

snake Ridge Estimation by Snake Method

Description

Estimate a ridge from a time-frequency representation, using the snake method.

Usage

snake(tfrep, guessA, guessB, snakesize=length(guessB),
tfspec=numeric(dim(modulus)[2]), subrate=1, temprate=3, muA=1,
muB=muA, lambdaB=2 * muB, lambdaA=2 * muA, iteration=1000000,
seed=-7, costsub=1, stagnant=20000, plot=TRUE)

Arguments

tfrep Time-Frequency representation (real valued).
guessA Initial guess for the algorithm (frequency variable).
guessB Initial guess for the algorithm (time variable).
snakesize the length of the initial guess of time variable.
tfspec Estimate for the contribution of the noise to modulus.
subrate Subsampling rate for ridge estimation.
temprate Initial value of temperature parameter.
muA Coefficient of the ridge’s derivative in cost function (frequency component).
muB Coefficient of the ridge’s derivative in cost function (time component).
lambdaB Coefficient of the ridge’s second derivative in cost function (time component).
lambdaA Coefficient of the ridge’s second derivative in cost function (frequency component).
iteration Maximal number of moves.
seed Initialization of random number generator.
costsub Subsampling of cost function in output.
stagnant maximum number of steps without move (for the stopping criterion)
plot when set (by default), certain results will be displayed
snakeview

Value

Returns a structure containing:

ridge  1D array (of same length as the signal) containing the ridge.

cost   1D array containing the cost function.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

corona, coronoid, icm, snakoid.

---

snakeview               Restriction to a Snake

Description

Restrict time-frequency transform to a snake.

Usage

    snakeview(modulus, snake)

Arguments

    modulus   Time-Frequency representation (real valued).
    snake     Time and frequency components of a snake.

Details

Recall that a snake is a (two components) \( \mathbb{R} \) structure.

Value

2D array containing the restriction of the transform modulus to the snake.

References

See discussions in the text of “Time-Frequency Analysis”.

Modified Snake Method

**Description**

Estimate a ridge from a time-frequency representation, using the modified snake method (modified cost function).

**Usage**

```r
snakoid(modulus, guessA, guessB, snakesize=length(guessB),
        tfspec=numeric(dim(modulus)[2]), subrate=1, temprate=3, muA=1,
        muB=muA, lambdaB=2 * muB, lambdaA=2 * muA, iteration=1000000,
        seed=-7, costsub=1, stagnant=20000, plot=TRUE)
```

**Arguments**

- `modulus`: Time-Frequency representation (real valued).
- `guessA`: Initial guess for the algorithm (frequency variable).
- `guessB`: Initial guess for the algorithm (time variable).
- `snakesize`: The length of the first guess of time variable.
- `tfspec`: Estimate for the contribution of the noise to modulus.
- `subrate`: Subsampling rate for ridge estimation.
- `temprate`: Initial value of temperature parameter.
- `muA`: Coefficient of the ridge’s derivative in cost function (frequency component).
- `muB`: Coefficient of the ridge’s derivative in cost function (time component).
- `lambdaB`: Coefficient of the ridge’s second derivative in cost function (time component).
- `lambdaA`: Coefficient of the ridge’s second derivative in cost function (frequency component).
- `iteration`: Maximal number of moves.
- `seed`: Initialization of random number generator.
- `costsub`: Subsampling of cost function in output.
- `stagnant`: Maximum number of stationary iterations before stopping.
- `plot`: when set (default), some results will be displayed

**Value**

Returns a structure containing:

- `ridge`: 1D array (of same length as the signal) containing the ridge.
- `cost`: 1D array containing the cost function.
- `plot`: when set (default), some results will be displayed.
sridrec

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

corona, coronoid, icm, snake.

| sridrec | Simple Reconstruction from Ridge |

Description

Simple reconstruction of a real valued signal from a ridge, by restriction of the transform to the ridge.

Usage

sridrec(tinput, ridge)

Arguments

tinput time-frequency representation.
ridge ridge (1D array).

Value

(real) reconstructed signal (1D array)

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

ridrec, gridrec.
SVD  
*Singular Value Decomposition*

**Description**

Computes singular value decomposition of a matrix.

**Usage**

`SVD(a)`

**Arguments**

`a`  
input matrix.

**Details**

R interface for Numerical Recipes singular value decomposition routine.

**Value**

a structure containing the 3 matrices of the singular value decomposition of the input.

**References**

See discussions in the text of “Time-Frequency Analysis”.

**Examples**

```r
hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, "+") }
X <- hilbert(6)
z = SVD(X)
z
```

---

**tfgmax**  
*Time-Frequency Transform Global Maxima*

**Description**

Computes the maxima (for each fixed value of the time variable) of the modulus of a continuous wavelet transform.

**Usage**

`tfgmax(input, plot=TRUE)`
Arguments

input wavelet transform (as the output of the function `cwt`)
plot if set to TRUE, displays the values of the energy as a function of the scale.

Value

output values of the maxima (1D array)
pos positions of the maxima (1D array)

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

`tflmax`,

---

`tflmax` *Time-Frequency Transform Local Maxima*

Description

Computes the local maxima (for each fixed value of the time variable) of the modulus of a time-frequency transform.

Usage

`tflmax(input, plot=TRUE)`

Arguments

input time-frequency transform (real 2D array).
plot if set to T, displays the local maxima on the graphic device.

Value

values of the maxima (2D array).

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

`tfgmax`. 
tfmean

**Description**
Compute the mean of time-frequency representation frequency by frequency.

**Usage**
```
   tfmean(input, plot=TRUE)
```

**Arguments**
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>time-frequency transform (output of <code>cwt</code> or <code>cgt</code>).</td>
</tr>
<tr>
<td>plot</td>
<td>if set to T, displays the values of the energy as a function of the scale (or frequency).</td>
</tr>
</tbody>
</table>

**Value**
1D array containing the noise estimate.

**References**
See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**
- `tfpct`, `tfvar`.


tfpct

**Description**
Compute a percentile of time-frequency representation frequency by frequency.

**Usage**
```
   tfpct(input, percent=0.8, plot=TRUE)
```

**Arguments**
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>time-frequency transform (output of <code>cwt</code> or <code>cgt</code>).</td>
</tr>
<tr>
<td>percent</td>
<td>percentile to be retained.</td>
</tr>
<tr>
<td>plot</td>
<td>if set to T, displays the values of the energy as a function of the scale (or frequency).</td>
</tr>
</tbody>
</table>
tfvar

Value

1D array containing the noise estimate.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

tfmean, tfvar.

tfvar

Variance frequency by frequency

Description

Compute the variance of time-frequency representation frequency by frequency.

Usage

tfvar(input, plot=TRUE)

Arguments

input time-frequency transform (output of cwt or cgt).
plot if set to T, displays the values of the energy as a function of the scale (or frequency).

Value

1D array containing the noise estimate.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

tfmean, tfpct.
**Undocumented Functions in Rwave**

**Description**
Numerous functions were not documented in the original Swave help files.

**References**
See discussions in the text of “Practical Time-Frequency Analysis”.

**vDOG**

**DOG Wavelet Transform on one Voice**

**Description**
Compute DOG wavelet transform at one scale.

**Usage**

vDOG(input, scale, moments)

**Arguments**

- **input** Input signal (1D array).
- **scale** Scale at which the wavelet transform is to be computed.
- **moments** number of vanishing moments.

**Value**
1D (complex) array containing wavelet transform at one scale.

**References**
See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**
vgt, vwt.
vecgabor  

**Gabor Functions on a Ridge**

**Description**

Generate Gabor functions at specified positions on a ridge.

**Usage**

```r
vecgabor(sigsize, nbnodes, location, frequency, scale)
```

**Arguments**

- `sigsize`: Signal size.
- `nbnodes`: Number of wavelets to be generated.
- `location`: b coordinates of the ridge samples (1D array of length nbnodes).
- `frequency`: frequency coordinates of the ridge samples (1D array of length nbnodes).
- `scale`: size parameter for the Gabor functions.

**Value**

size parameter for the Gabor functions.

**See Also**

`vecmorlet`.

vecmorlet  

**Morlet Wavelets on a Ridge**

**Description**

Generate Morlet wavelets at specified positions on a ridge.

**Usage**

```r
vecmorlet(sigsize, nbnodes, bridge, aridge, w0=2 * pi)
```

**Arguments**

- `sigsize`: Signal size.
- `nbnodes`: Number of wavelets to be generated.
- `bridge`: b coordinates of the ridge samples (1D array of length nbnodes).
- `aridge`: a coordinates of the ridge samples (1D array of length nbnodes).
- `w0`: Center frequency of the wavelet.
Value

2D (complex) array containing wavelets located at the specific points.

See Also

vecgabor.

---

vgt  

*Gabor Transform on one Voice*

Description

Compute Gabor transform for fixed frequency.

Usage

vgt(input, frequency, scale, plot=FALSE)

Arguments

- **input**: Input signal (1D array).
- **frequency**: frequency at which the Gabor transform is to be computed.
- **scale**: frequency at which the Gabor transform is to be computed.
- **plot**: if set to TRUE, plots the real part of cgt on the graphic device.

Value

1D (complex) array containing Gabor transform at specified frequency.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

vwt, vDOG.
vwt

Voice Wavelet Transform

Description
Compute Morlet's wavelet transform at one scale.

Usage
vwt(input, scale, w0=2 * pi)

Arguments
- input: Input signal (1D array).
- scale: Scale at which the wavelet transform is to be computed.
- w0: Center frequency of the wavelet.

Value
1D (complex) array containing wavelet transform at one scale.

References
See discussions in the text of “Practical Time-Frequency Analysis”.

See Also
vgt, vDOG.

wpl
Plot Dyadic Wavelet Transform.

Description
Plot dyadic wavelet transform(output of mw).

Usage
wpl(dwtrans)

Arguments
- dwtrans: dyadic wavelet transform (output of mw).

See Also
mw, ext, epl.
Description

Given a ridge $\phi$ (for the wavelet transform), returns a (appropriately) subsampled version with a
given subsampling rate.

Usage

```wRidgeSampling(phi, compr, nvoice)```

Arguments

- **phi**: ridge (1D array).
- **compr**: subsampling rate for the ridge.
- **nvoice**: number of voices per octave.

Details

To account for the variable sizes of wavelets, the sampling rate of a wavelet ridge is not uniform,
and is proportional to the scale.

Value

Returns a list containing the discrete values of the ridge.

- **node**: time coordinates of the ridge samples.
- **phinode**: scale coordinates of the ridge samples.
- **nbnodes**: number of nodes of the ridge samples.

See Also

`RidgeSampling`.
WSPEC.PL

Log of Wavelet Spectrum Plot

Description
Displays normalized log of wavelet spectrum.

Usage
wspec.pl(wspec, nvoice)

Arguments
wspec wavelet spectrum.
nvoice number of voices.

References
See discussions in the text of “Practical Time-Frequency Analysis”.

See Also
hurst.est.

WV
Wigner-Ville function

Description
Compute the Wigner-Ville transform, without any smoothing.

Usage
WV(input, nvoice, freqstep = (1/nvoice), plot = TRUE)

Arguments
input input signal (possibly complex-valued)
nvoice number of frequency bands
freqstep sampling rate for the frequency axis
plot if set to TRUE, displays the modulus of CWT on the graphic device.

Value
(complex) Wigner-Ville transform.
**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

---

**W_tilda.1**  
*Pixel from Amber Camara*

---

**Description**

Pixel from amber camara.

**Usage**

```r
data(W_tilda.1)
```

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

*Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*,  

**Examples**

```r
data(W_tilda.1)
plot.ts(W_tilda.1)
```
**W_tilda.2**  
*Pixel from Amber Camara*

### Description
Pixel from amber camara.

### Usage
```r
data(W_tilda.2)
```

### Format
A vector containing observations.

### Source
See discussions in the text of “Practical Time-Frequency Analysis”.

### References
*Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S.*  

### Examples
```r
data(W_tilda.2)  
plot.ts(W_tilda.2)
```

---

**W_tilda.3**  
*Pixel from Amber Camara*

### Description
Pixel from amber camara.

### Usage
```r
data(W_tilda.3)
```

### Format
A vector containing observations.
Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(W_tilda.3)
plot.ts(W_tilda.3)

---

W_tilda.4  
*Pixel from Amber Camara*

Description

Pixel from amber camara.

Usage

data(W_tilda.4)

Format

A vector containing observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(W_tilda.4)
plot.ts(W_tilda.4)
Description
Pixel from amber camara.

Usage
data(W_tilda.5)

Format
A vector containing observations.

Source
See discussions in the text of “Practical Time-Frequency Analysis”.

References

Examples

data(W_tilda.5)
plot.ts(W_tilda.5)

Description
Pixel from amber camara.

Usage
data(W_tilda.6)

Format
A vector containing observations.
Source
See discussions in the text of “Practical Time-Frequency Analysis”.

References

Examples
```
data(W_tilda.6)
plot.ts(W_tilda.6)
```

---

**W_tilda.7**

*Pixel from Amber Camara*

Description
Pixel from amber camara.

Usage
```
data(W_tilda.7)
```

Format
A vector containing observations.

Source
See discussions in the text of “Practical Time-Frequency Analysis”.

References

Examples
```
data(W_tilda.7)
plot.ts(W_tilda.7)
```
**W_tilda.8**  
*Pixel from Amber Camara*

**Description**  
Pixel from amber camara.

**Usage**  
data(W_tilda.8)

**Format**  
A vector containing observations.

**Source**  
See discussions in the text of “Practical Time-Frequency Analysis”.

**References**  

**Examples**  
data(W_tilda.8)  
plot.ts(W_tilda.8)

**W_tilda.9**  
*Pixel from Amber Camara*

**Description**  
Pixel from amber camara.

**Usage**  
data(W_tilda.9)

**Format**  
A vector containing observations.
Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(W_tilda.9)
plot.ts(W_tilda.9)

data(yen)

Pixel from Amber Camara

Description

Pixel from amber camara.

Usage

data(yen)

Format

A vector containing observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(yen)
plot.ts(yen)
**yendiff**

*Pixel from Amber Camara*

**Description**

Pixel from amber camara.

**Usage**

```r
data(yendiff)
```

**Format**

A vector containing observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**


**Examples**

```r
data(yendiff)
plot.ts(yendiff)
```

---

**YN**

*Logarithms of the Prices of Japanese Yen*

**Description**

Logarithms of the prices of a contract of Japanese yen.

**Usage**

```r
data(YN)
```

**Format**

A vector containing 500 observations.
Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(YN)
plot.ts(YN)

YNdiff Daily differences of Japanese Yen

Description

Daily differences of YN.

Usage

data(YNdiff)

Format

A vector containing 499 observations.

Source

See discussions in the text of “Practical Time-Frequency Analysis”.

References


Examples

data(YNdiff)
plot.ts(YNdiff)
zerokernel  

**Reconstruction from Wavelet Ridges**

**Description**

Generate a zero kernel for reconstruction from ridges.

**Usage**

```r
zerokernel(x.inc=1, x.min, x.max)
```

**Arguments**

- `x.min`: minimal value of `x` for the computation of $Q_2$.
- `x.max`: maximal value of `x` for the computation of $Q_2$.
- `x.inc`: step unit for the computation of the kernel.

**Value**

matrix of the $Q_2$ kernel

**See Also**

`kernel`, `fastkernel`, `gkernel`, `gkernel`.

zeroskeleton  

**Reconstruction from Dual Wavelets**

**Description**

Computes the reconstructed signal from the ridge when the epsilon parameter is set to zero.

**Usage**

```r
zeroskeleton(cwtinput, Qinv, morvelets, bridge, aridge, N)
```

**Arguments**

- `cwtinput`: continuous wavelet transform (as the output of `cwt`).
- `Qinv`: inverse of the reconstruction kernel (2D array).
- `morvelets`: array of Morlet wavelets located at the ridge samples.
- `bridge`: time coordinates of the ridge samples.
- `aridge`: scale coordinates of the ridge samples.
- `N`: size of reconstructed signal.
Details

The details of this reconstruction are the same as for the function skeleton. They can be found in the text.

Value

Returns a list of the elements of the reconstruction of a signal from sample points of a ridge:

- `sol`: reconstruction from a ridge.
- `A`: matrix of the inner products.
- `lam`: coefficients of dual wavelets in reconstructed signal. They are the Lagrange multipliers $\lambda$'s of the text.
- `dualwave`: array containing the dual wavelets.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

`skeleton, skeleton2, zeroskeleton2`.

---

**zeroskeleton2**  
*Reconstruction from Dual Wavelets*

Description

Computes the reconstructed signal from the ridge when the epsilon parameter is set to zero, in the case of real constraints.

Usage

`zeroskeleton2(cwtinput, Qinv, morvelets, bridge, aridge, N)`

Arguments

- `cwtinput`: continuous wavelet transform (output of `cwt`).
- `Qinv`: inverse of the reconstruction kernel (2D array).
- `morvelets`: array of Morlet wavelets located at the ridge samples.
- `bridge`: time coordinates of the ridge samples.
- `aridge`: scale coordinates of the ridge samples.
- `N`: size of reconstructed signal.
Details

The details of this reconstruction are the same as for the function skeleton. They can be found in the text

Value

Returns a list of the elements of the reconstruction of a signal from sample points of a ridge

- sol: reconstruction from a ridge.
- A: matrix of the inner products.
- lam: coefficients of dual wavelets in reconstructed signal. They are the Lagrange multipliers $\lambda$'s of the text.
- dualwave: array containing the dual wavelets.

References

See discussions in the text of “Practical Time-Frequency Analysis”.

See Also

skeleton, skeleton2, zeroskeleton.
Index

* datasets
  A0.5
  A4.5
  amber7.7
  amber8.7
  amber9.8
  B0.9
  B4.9
  back1.000.10
  back1.180.11
  back1.220.11
  backscatter.1.000.12
  backscatter.1.180.13
  backscatter.1.220.13
  C0.14
  C4.15
  ch.18
  chirpm5db.dat.19
  click.21
  click.asc.21
  D0.34
  D4.34
  Ekg.37
  HeartRate.48
  HOWAREYOU.49
  noisy.dat.58
  noisywave.59
  pixel_8.7.60
  pixel_8.8.60
  pixel_8.9.61
  pure.dat.63
  purwave.64
  sig_W_tilda.1.78
  sig_W_tilda.2.78
  sig_W_tilda.3.79
  sig_W_tilda.4.80
  sig_W_tilda.5.80
  signal_W_tilda.1.72
  signal_W_tilda.2.72
  signal_W_tilda.3.73
  signal_W_tilda.4.74
  signal_W_tilda.5.74
  signal_W_tilda.6.75
  signal_W_tilda.7.76
  signal_W_tilda.8.76
  signal_W_tilda.9.77
  W_tilda.1.98
  W_tilda.2.99
  W_tilda.3.99
  W_tilda.4.100
  W_tilda.5.101
  W_tilda.6.101
  W_tilda.7.102
  W_tilda.8.103
  W_tilda.9.103
  yen.104
  yendiff.105
  YN.105
  YNdiff.106

* hplot
  wspec.pl.97

* ts
  adjust.length.6
  cfamily.15
  cgt.17
  check.maxresoln.19
  cleanph.20
  corona.22
  coronoid.23
  crc.24
  crcrc.26
  crfview.27
  cwt.27
  cwtimage.29
  cwtplot.30
  cwtplot.31
  cwtquiz.32
  cwtTh.33
INDEX

DOG, 35

dwinverse, 36

epl, 37
 ext, 38
 fastgkernel, 39
 fastkernel, 40
 gabor, 41
gcrcrc, 42
gkernel, 43
gregrec, 44
ggridrec, 45
gsampleOne, 46
gwave, 47
gwave2, 47
 hurst.est, 50
 icm, 51
 mbtrim, 52
 mntrim, 53
 morlet, 54
 morwave, 55
 morwave2, 55
 mrecons, 56
 mw, 57
 npl, 59
 plotResult, 62
 plotwt, 62
 regrec, 64
 regrec2, 66
 RidgeSampling, 67
 ridrec, 68
 rkernel, 69
 rwkernel, 70
 scrcrc, 71
 skeleton, 81
 skeleton2, 82
 smoothts, 83
 smoothwt, 83
 snake, 84
 snakeview, 85
 snakoid, 86
 sridrec, 87
 SVD, 88
 tfgmax, 88
 tflmax, 89
 tfmean, 90
 tfpct, 90
 tfvar, 91

Undocumented, 92

vDOG, 92
 vecgabor, 93
 vecmorlet, 93
 vgt, 94
 vwt, 95
 wpl, 95
 wRidgeSampling, 96
 wspec.pl, 97
 WV, 97
 zerokernel, 107
 zeroskeleton, 107
 zeroskeleton2, 108

A0, 5
 A4, 5
 adjust.length, 6
 amber7, 7
 amber8, 7
 amber9, 8

B0, 9
 B4, 9
 back1.000, 10
 back1.180, 11
 back1.220, 11
 backscatter.1.000, 12
 backscatter.1.180, 13
 backscatter.1.220, 13
 band (Undocumented), 92

C0, 14
 C4, 15
 cfamily, 15, 16, 25, 27, 43, 71
 cgtradar (Undocumented), 92
 ch, 18
 check.maxresoln, 19
 chirpm5db.dat, 19
 cleanph, 20
 click, 21
 click.asc, 21
 cloudXYZ (Undocumented), 92
 confident (Undocumented), 92
 corona, 22, 23–25, 51, 84, 85, 87
 coronoid, 23, 23, 24, 25, 51, 84, 85, 87
 crc, 15, 16, 24, 26, 27, 42, 43, 71
 crcirgrec (Undocumented), 92
 crcirirc (Undocumented), 92
 crcirc (16, 25, 26, 43, 71
crfview, 27


cwtimage, 29, 31

cwtp, 17, 28, 30, 32, 33, 36

cwtpolar, 29, 31

cwtsquiz, 17, 32, 36

cwtTh, 28, 30, 33

D0, 34
D4, 34

ddw (Undocumented), 92

dOG, 17, 28–33, 35

dw (Undocumented), 92

dwinverse, 36, 58

Ekg, 37

epl, 37, 62, 63, 95

ext, 36–38, 38, 52, 53, 57, 58, 95

fastgkernel, 39, 43

fastkernel, 39, 40, 43, 69, 107

fftshift (Undocumented), 92

gabor, 28, 30, 33, 41, 54

gccrcrc, 16, 25, 42

girregrec (Undocumented), 92

gkernel, 39, 40, 43, 46, 69, 70, 107

gregrec, 44, 46, 65, 67

ggridrec, 45, 65, 87

gsampleOne, 46

gwave, 47, 47, 48, 55, 56

gwave2, 47, 47, 55, 56

HeartRate, 48

hescrc (Undocumented), 92

HOWAREYOU, 49

hurst.est, 50, 97

icm, 23–25, 51, 51, 85, 87

irregrec (Undocumented), 92

kernel, 40, 43, 46, 69, 107

mbpval (Undocumented), 92

mbtrim, 52, 53

mcgt (Undocumented), 92

mnpval (Undocumented), 92

mntrim, 52, 53

morlet, 41, 54

morwave, 47, 48, 55, 56

morwave2, 47, 48, 55, 55

mrecons, 19, 36, 38, 52, 53, 56, 58

mw, 19, 36, 38, 57, 57, 95

noisy.dat, 58

noisyswave, 59

npl, 59

pcacrc (Undocumented), 92

pcafamily (Undocumented), 92

pcamaxima (Undocumented), 92

pcamorwave (Undocumented), 92

pcarec (Undocumented), 92

pcaregrec (Undocumented), 92

PcaRidgeSampling (Undocumented), 92

pcaidnrec (Undocumented), 92

pcazeroskeleton (Undocumented), 92

pixel_8.7, 60

pixel_8.8, 60

pixel_8.9, 61

plotResult, 62, 63

plotwt, 62, 62

pure.dat, 63

purwave, 64

regrec, 45, 46, 64, 67, 68

regrec2, 46, 65, 66, 68

RidgeDist (Undocumented), 92

RidgeIrregSampling (Undocumented), 92

RidgeSampling, 67, 96

ridrec, 65, 67, 68, 87

rkernel, 39, 40, 43, 69, 70

robustrec (Undocumented), 92

RunRec (Undocumented), 92

rwkernel, 70

SampleGen (Undocumented), 92

Sausage (Undocumented), 92

scrcrc, 16, 25, 27, 43, 71

showRadar (Undocumented), 92

sig_W_tilda.1, 78

sig_W_tilda.2, 78

sig_W_tilda.3, 79

sig_W_tilda.4, 80

sig_W_tilda.5, 80

signal_W_tilda.1, 72

signal_W_tilda.2, 72

signal_W_tilda.3, 73
INDEX

113

signal_W_tilda.4, 74
signal_W_tilda.5, 74
signal_W_tilda.6, 75
signal_W_tilda.7, 76
signal_W_tilda.8, 76
signal_W_tilda.9, 77
simplepcarec (Undocumented), 92
skeleton, 81, 82, 108, 109
skeleton2, 82, 82, 108, 109
smoothths, 83
smoothwt, 83
snake, 23–25, 51, 84, 84, 87
snakeview, 85
snakoid, 23–25, 51, 84, 85, 86
SpecGen (Undocumented), 92
sridrec, 46, 67, 68, 87
SVD, 88
tfgmax, 88, 89
tflmax, 89, 89
tfmean, 50, 90, 91
tfpct, 90, 90, 91
tfvar, 90, 91, 91
Undocumented, 92
vDOG, 92, 94, 95
vecgabor, 93, 94
vecmorlet, 93, 93
vgt, 92, 94, 95
vwt, 92, 94, 95
vwtTh (Undocumented), 92
W_tilda.1, 98
W_tilda.2, 99
W_tilda.3, 99
W_tilda.4, 100
W_tilda.5, 101
W_tilda.6, 101
W_tilda.7, 102
W_tilda.8, 103
W_tilda.9, 103
wpl, 38, 62, 63, 95
wRidgeSampling, 67, 96
wspec.pl, 50, 97
WV, 97
yen, 104
yendiff, 105

YN, 105, 106
YNdif, 106
zeroskeleton, 82, 107, 109
zeroskeleton2, 82, 108, 108
zerokernel, 39, 40, 43, 69, 70, 107
zerokernel, 39, 40, 43, 69, 70, 107
zeroskeleton, 82, 107, 109
zeroskeleton2, 82, 108, 108