BoundaryWavelets Documentation

Josefine Holm, Steffen Lønsmann Nielsen

Feb 12, 2019

Contents:

1 BoundaryWavelets package **1**
- 1.1 Submodules . . . 1
  - 1.1.1 boundwave.BoundaryWavelets module . . . 1
  - 1.1.2 boundwave.FourierBoundaryWavelets module . . . 2
  - 1.1.3 boundwave.Orthonormal module . . . 3

2 Test and example scripts **5**
- 2.1 DataTest module . . . 5
- 2.2 ReconFunctions module . . . 6
- 2.3 ReconTest module . . . 7
- 2.4 WaveletTest module . . . 7

3 Indices and tables **9**

Python Module Index **11**

Index **13**

CHAPTER 1

BoundaryWavelets package

1.1 Submodules

1.1.1 boundwave.BoundaryWavelets module

This module is used for calculations of the boundary wavelets in the time domain.

boundwave.BoundaryWavelets.**BoundaryWavelets**(*phi*, *J*, *WaveletCoef*, *AL=None*, *AR=None*)

This function evaluates the left boundary functions.

INPUT:

phi [numpy.float64] The scaling function at scale 0. (1d array)

J [int] The scale the scaling function has to have.

WaveletCoef [numpy.float64] The wavelet coefficients must sum to $\sqrt{2}$. For Daubechies 2 they can be found using *np.flipud(pywt.Wavelet('db2').dec_lo)*.

AL=None [numpy.float64] The left orthonormalisation matrix, if this is not suplied the functions will not be orthonormalized. Can be computed using *boundwave.Orthonormal.OrthoMatrix()*.

AR=None [numpy.float64] The right orthonormalisation matrix, if this is not suplied the functions will not be orthonormalized. Can be computed using *boundwave.Orthonormal.OrthoMatrix()*.

OUTPUT:

x [numpy.float64] 2d numpy array with the boundary functions in the columns; orthonormalised if *AL* and *AR* given.

boundwave.BoundaryWavelets.**DownSample**(*x*, *Shift*, *N*, *J*, *zero=True*)

This is a function which dilates a signal. Make sure there are enough samples for it to make sense.

INPUT:

x [numpy.float64] 1d numpy array, signal to be dilated.

Shift [int] Time shift before dilation (for now it only supports integers).

N [int] Number of samples per 'time' unit.

J [int] The scale to make. Non-negative integer.

zero=True [bool] If true, it concatenates zeros on the signal to retain the original length.

OUTPUT:

y [numpy.float64] The scaled version of the scaling function.

boundwave.BoundaryWavelets.**InnerProductPhiX**(*alpha*, *J*, *k*, *Moments*)

This function calculates the inner product between x^α and $\phi_{J,k}$.

INPUT:

alpha [int] The power of x

J, k [int] The indices for ϕ.

Moments [numpy.float64] A 1d array of moments for ϕ, up to power *alpha*. Can be calculated using the function *Moments()*.

OUTPUT:

i [numpy.float64] The inner product.

boundwave.BoundaryWavelets.**Moments**(*WaveletCoef*, *n*)

This function calculates the moments of ϕ up to power *n*, i.e. $\langle x^l, \phi \rangle$, for $0 \leq l \leq n$.

INPUT:

WaveletCoef [numpy.float64] The wavelet coefficients, must sum to $\sqrt{2}$. For Daubechies 2 they can be found using *np.flipud(pywt.Wavelet('db2').dec_lo)*.

n [int] The highest power moment to calculate.

OUTPUT:

moments [numpy.float64] A 1d array with the moments.

1.1.2 boundwave.FourierBoundaryWavelets module

This module is used for calculations of the boundary wavelets in the frequency domain.

The BoundaryWavelets.py package is licensed under the MIT "Expat" license.

boundwave.FourierBoundaryWavelets.**FourierBoundaryWavelets**(*J*, *Scheme*, *WaveletCoef*, *AL=None*, *AR=None*, *Win=<function Rectangle>*)

This function evaluates the Fourier transformed boundary functions for db2.

INPUT:

J [int] The scale.

Scheme [numpy.float64] The sampling scheme in the Fourier domain.

WaveletCoef [numpy.float64] The wavelet coefficients, must sum to $\sqrt{2}$. For Daubeshies 2 they can be found using *np.flipud(pywt.Wavelet('db2').dec_lo)*.

AL=None [numpy.float64] The left orthonormalisation matrix, if this is not supplied the functions will not be orthonormalized. Can be computed using *boundwave.Orthonormal.OrthoMatrix()*.

AR=None [numpy.float64] The right orthonormalisation matrix, if this is not supplied the functions will not be orthonormalized. Can be computed using *boundwave.Orthonormal.OrthoMatrix()*.

Win= ***Rectangle()*** [numpy.complex128] The window to use on the boundary functions.

OUTPUT:

x [numpy.complex128] 2d numpy array with the boundary functions in the columns; orthonormalised if *AL* and *AR* given.

boundwave.FourierBoundaryWavelets.**Rectangle**(*Scheme*)

The Fourier transform of a rectangular window function.

INPUT:

Scheme [numpy.float64] A numpy array with the frequencies in which to sample.

OUTPUT:

chi [numpy.complex128] A numpy array with the window function sampled in the freqency domain.

boundwave.FourierBoundaryWavelets.**ScalingFunctionFourier**(*WaveletCoef*, *J*, *k*, *Scheme*, *Win*, *P=20*)

This function evaluates the Fourier transform of the scaling function, $\phi_{j,k}$, sampled in scheme.

INPUT:

WaveletCoef [numpy.float64] The wavelet coefficients, must sum to $\sqrt{2}$. For Daubechies 2 they can be found using *np.flipud(pywt.Wavelet('db2').dec_lo)*.

J [int] The scale.

k [int] The translation.

Scheme [numpy.float64] The points in which to evaluate.

Window=Rectangle [numpy.complex128] The window to use on the boundary functions.

P=20 [int] The number of factors to include in the infinite product in the Fourier transform of phi.

OUTPUT:

phi [numpy.complex128] $\hat{\phi}_{j,k}$

1.1.3 boundwave.Orthonormal module

This module is used for calculations of the orthonormalization matrix for the boundary wavelets.

The BoundaryWavelets.py package is licensed under the MIT "Expat" license.

Copyright (c) 2019: Josefine Holm and Steffen L. Nielsen.

boundwave.Orthonormal.**Integral**(*J*, *k*, *l*, *WaveletCoef*, *phi*)

This function calculates the integral (16) numerically.

INPUT:

J [int] The scale.

k [int] The translation for the first function.

l [int] The translation for the second function.

WaveletCoef [numpy.float64] The wavelet coefficients, must sum to $\sqrt{2}$. For Daubechies 2 they can be found using *np.flipud(pywt.Wavelet('db2').dec_lo)*.

phi [numpy.float64] The phi function, can be made with *pywt.Wavelet(wavelet).wavefun(level=15)*.

OUTPUT:

out [int] The value of the integral.

boundwave.Orthonormal.**M_AlphaBeta**(*alpha*, *beta*, *J*, *WaveletCoef*, *InteMatrix*, *Side*)

This function calculates an entry in the martix M (15).

INPUT:

alpha [int] alpha

beta [int] beta

J [int] The scale.

WaveletCoef [numpy.float64] The wavelet coefficients, must sum to $\sqrt{2}$. For Daubechies 2 they can be found using *np.flipud(pywt.Wavelet('db2').dec_lo)*.

InteMatrix [numpy.float64] A matrix with the values for the integrals calculated with the function *Integral()* for k and l in the interval [-2*a+2,0] or [2**J-2*a+1,2**J-1].

Side [str] *'L'* for left interval boundary and *'R'* for right interval boundary.

OUTPUT:

M [numpy.float64] Entry (alpha,beta) of the martix M

boundwave.Orthonormal.**OrthoMatrix**(*J*, *WaveletCoef*, *phi*)

This function findes the orthogonality matrix A. First uses the functions *M_AlphaBeta()* and *Integral()* to make the matrix M. Then computes a Cholesky decomposition, which is then inverted.

INPUT:

J [int] The scale.

WaveletCoef [numpy.float64] The wavelet coefficients, must sum to $\sqrt{2}$. For Daubechies 2 they can be found using *np.flipud(pywt.Wavelet('db2').dec_lo)*.

phi [numpy.float64] The phi function, can be made with *pywt.Wavelet(wavelet).wavefun(level=15)*.

OUTPUT:

AL [numpy.float64] Left orthonormalisation matrix; to be used in *boundwave.BoundaryWavelets.BoundaryWavelets()* or *boundwave.FourierBoundaryWavelets.FourierBoundaryWavelets()*.

AR [numpy.float64] Right orthonormalisation matrix; to be used in *boundwave.BoundaryWavelets.BoundaryWavelets()* or *boundwave.FourierBoundaryWavelets.FourierBoundaryWavelets()*.

CHAPTER 2

Test and example scripts

2.1 DataTest module

This module is used for testing the boundary wavelets on ECG data.

DataTest.**Test**(*Name='Data'*, *Row=1*, *J=7*, *N=12*, *Wavelet='db3'*)

This function makes decompositions and reconstructions of several sections of the data, both with boundary wavelets and with mirrored extension. The differences between the orignal signal and the two reconstructions are calculated. The test is run for as many disjoint sections of the signal as possible.

INPUT:

Name [str] The MATLAB data file from whichto load.

Row [int] The row in the dataset to use.

J [int] The scale.

N [int] The number of iterations to use in the cascade algorithm.

Wavelet [str] The name of the wavelet to be used. eg: *'db2'*.

OUTPUT:

Result [float64] 2D array. The first row is the difference between the original signal and the reconstruction using boundary wavelet. The second row is the difference between the original signal and the reconstruction using mirrored extension. The third row is the first row minus the second row. There is one collumn for each section of the signal.

DataTest.**TestPlot**(*Name='Data'*, *Row=1*, *Section=214*, *J=7*, *N=12*, *Wavelet='db3'*)

This function makes decompositions and reconstructions of a chosen section of the data, both with boundary wavelets and with mirrored extension. The difference between the orignal signal and the two reconstructions are calculated and printed and all three signals are plotted in the same figure.

INPUT:

Name [str] The MATLAB data file from which to load.

Row [int] The row in the dataset to use.

Section [int] Which section of the data to use. The samples that will be used are: *[Section*2**N:Section*2**N+2**N]*.

J [int] The scale.

N [int] The number of iterations to use in the cascade algorithm.

Wavelet [str] The name of the wavelet to be used. eg: *'db2'*.

2.2 ReconFunctions module

This is a module which contains reconstruction algorithms for the Daubechies wavelets.

`ReconFunctions.`**`DecomBoundary`**(*Signal*, *J*, *Wavelet*, *phi*)

This function makes a wavelet decomposition of a 1D signal in time, using boundary wavelets at the edge.

INPUT:

Signal [numpy.float64] The signal to be decomposed.

J [int] The scale of the wavelet.

Wavelet [str] The name of the wavelet to use. For instance *'db2'*.

phi [numpy.float64] The scaling function at scale 0. (1d array)

OUTPUT:

x [numpy.float64] The decomposition.

`ReconFunctions.`**`DecomMirror`**(*Signal*, *J*, *Wavelet*, *phi*)

This function makes a wavelet decomposition of a 1D signal in time, using mirroring of the signal at the edge.

INPUT:

Signal [numpy.float64] The signal to be decomposed.

J [int] The scale of the wavelet.

Wavelet [str] The name of the wavelet to use. For instance *'db2'*.

phi [numpy.float64] The scaling function at scale 0. (1d array)

OUTPUT:

x [numpy.float64] The decomposition.

`ReconFunctions.`**`ReconBoundary`**(*WaveletCoef*, *J*, *Wavelet*, *phi*)

This function reconstructs a 1D signal in time from its wavelet coefficients, using boundary wavelets at the edge.

INPUT:

WaveletCoef [numpy.float64] The wavelet decomposition. Can be made using DecomBoundary().

J [int] The scale of the wavelet.

Wavelet [str] The name of the wavelet to use. For instance *'db2'*.

phi [numpy.float64] The scaling function at scale 0. (1d array)

OUTPUT:

x [numpy.float64] The reconstructed signal, the length of the signal is *2**(N-J)*len(WaveletCoef)*.

ReconFunctions.**ReconMirror**(*WaveletCoef*, *J*, *Wavelet*, *phi*)

This function reconstructs a 1D signal in time from its wavelet coefficients, using mirroring of the signal at the edge.

INPUT:

WaveletCoef [numpy.float64] The wavelet decomposition. Can be made using DecomMirror().

J [int] The scale of the wavelet.

Wavelet [str] The name of the wavelet to use. For instance *'db2'*.

phi [numpy.float64] The scaling function at scale 0. (1d array)

OUTPUT:

x [numpy.float64] The reconstructed signal, the length of the signal is *2**(N-J)*len(WaveletCoef)*.

2.3 ReconTest module

This module is used for showing that the Daubechies 2 boundary wavelets can reconstruct constant and linear functions.

ReconTest.**TestOfConFunc**(*J=3*)

Test and plot of constant and linear function.

2.4 WaveletTest module

This module is used for testing and showing the boundary wavelets in the time and frequency domain.

WaveletTest.**GeneralTest**(*Wavelet='db2'*, *J=2*, *epsilon=0.14285714285714285*, *Length=1792*, *TimeOnly=False*)

A test function used in the paper. Plots the boundary wavelets in time and compare them with the boundary wavelets created in frequency by using an inverse Fourier transform.

INPUT:

Wavelet='db2' [str] The wavelet to use in the test.

J=2 [int] The scale. If *J* is too small for the scaling function to be supported within the interval [0,1] it is changed to the smallest possible *J*.

epsilon=1/7 [float] The sampling dencity in the frequency domain, i.e. the distance between samples. In Generalized sampling this must be at least 1/7 for db2.

Length=1792 [int] The number of samples in the frequency domain. The standard is chosen because $1792/7 = 256$.

TimeOnly=False [bool] If *TimeOnly=True* the boundary functions are only constructed in time, not in frequency.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

boundwave, 4
boundwave.BoundaryWavelets, 1
boundwave.FourierBoundaryWavelets, 2
boundwave.Orthonormal, 3

d

DataTest, 5

r

ReconFunctions, 6
ReconTest, 7

w

WaveletTest, 7

Index

B

BoundaryWavelets() (in module boundwave.BoundaryWavelets), 1
boundwave (module), 4
boundwave.BoundaryWavelets (module), 1
boundwave.FourierBoundaryWavelets (module), 2
boundwave.Orthonormal (module), 3

D

DataTest (module), 5
DecomBoundary() (in module ReconFunctions), 6
DecomMirror() (in module ReconFunctions), 6
DownSample() (in module boundwave.BoundaryWavelets), 1

F

FourierBoundaryWavelets() (in module boundwave.FourierBoundaryWavelets), 2

G

GeneralTest() (in module WaveletTest), 7

I

InnerProductPhiX() (in module boundwave.BoundaryWavelets), 2
Integral() (in module boundwave.Orthonormal), 3

M

M_AlphaBeta() (in module boundwave.Orthonormal), 4
Moments() (in module boundwave.BoundaryWavelets), 2

O

OrthoMatrix() (in module boundwave.Orthonormal), 4

R

ReconBoundary() (in module ReconFunctions), 6
ReconFunctions (module), 6
ReconMirror() (in module ReconFunctions), 7
ReconTest (module), 7
Rectangle() (in module boundwave.FourierBoundaryWavelets), 3

S

ScalingFunctionFourier() (in module boundwave.FourierBoundaryWavelets), 3

T

Test() (in module DataTest), 5
TestOfConFunc() (in module ReconTest), 7
TestPlot() (in module DataTest), 5

W

WaveletTest (module), 7