README

This is the readme-file for "a wavelet add-on code for new-generation N-body simulations and data de-noising (JOFILUREN)", written and distributed by Alessandro B. Romeo. The method is introduced in Paper I, and the code is described in Paper II.

Paper I: Romeo A. B., Horellou C. and Bergh J. (2003), "N-Body Simulations with Two-Orders-of-Magnitude Higher Performance Using Wavelets", *Monthly Notices of the Royal Astronomical Society* **342**, 337–344.

Paper II: Romeo A. B., Horellou C. and Bergh J. (2004), "A Wavelet Add-On Code for New-Generation N-Body Simulations and Data De-Noising (JOFILUREN)", *Monthly Notices of the Royal Astronomical Society* **354**, 1208–1222.

Here I give supplementary information about the include-files and the subroutines of the code.

Enjoy JOFILUREN :–)

Alessandro

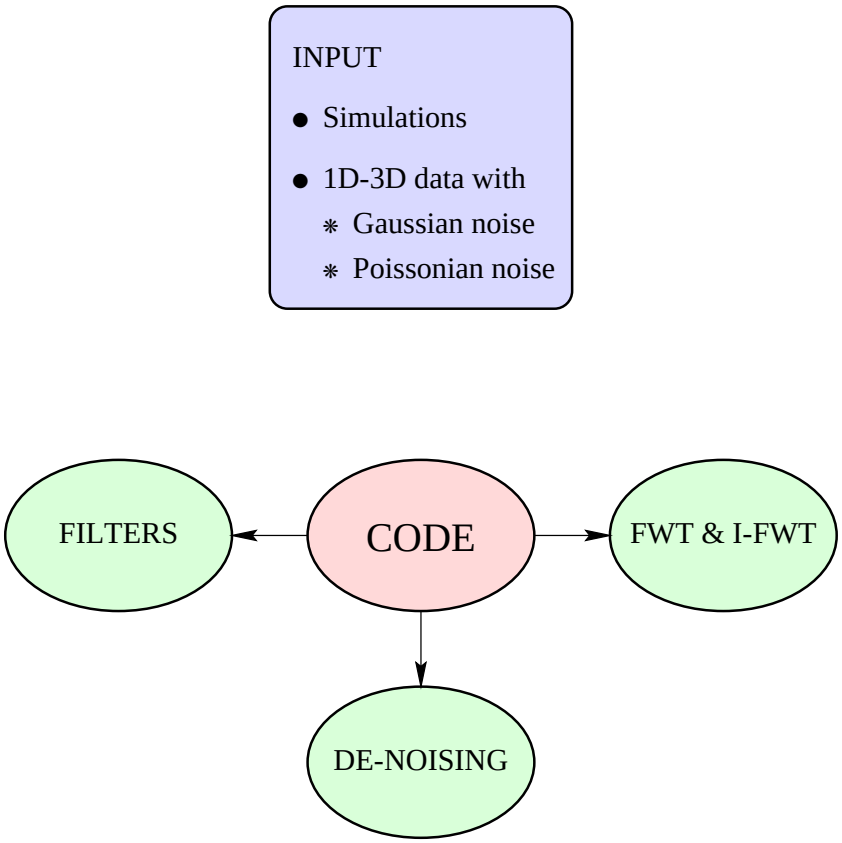

Figure 1: Schematic view of the code.

LIST OF THE MAIN INCLUDE-FILES AND SUBROUTINES

The list is self-explanatory, except for the following: the 3Ds subroutines are generalizations of the 3D subroutines to data with unequal number of points along the three directions, and the subroutines in parenthesis implement a less advanced de-noising.

Include-Files for Wavelet Filters

Main

- `filters.inc`

NOTE: Edit this file for choosing the wavelet (see p. 3)!

Subroutines for De-Noising

N-Body Simulations

- 3Ds: `denoise_sp3ds`
- 3D: `denoise_sp3d`
- 2D: `denoise_sp2d`
- 1D: `denoise_sp1d`

Standard Data with Additive White Gaussian Noise

- 3Ds: `denoise_sg3ds` (`denoise_g3ds`)
- 3D: `denoise_sg3d` (`denoise_g3d`)
- 2D: `denoise_sg2d` (`denoise_g2d`)
- 1D: `denoise_sg1d` (`denoise_g1d`)

Standard Data with Poissonian Noise

- 3Ds: `denoise_sp3ds` (`denoise_p3ds`)
- 3D: `denoise_sp3d` (`denoise_p3d`)
- 2D: `denoise_sp2d` (`denoise_p2d`)
- 1D: `denoise_sp1d` (`denoise_p1d`)

Subroutines for the Fast Wavelet Transform

Direct

- 3Ds: `fwt_3dsf`
- 3D: `fwt_3df`
- 2D: `fwt_2df`
- 1D: `fwt_1df`

Inverse

- 3Ds: `fwt_3dsb`
- 3D: `fwt_3db`
- 2D: `fwt_2db`
- 1D: `fwt_1db`

INCLUDE-FILE AND SUBROUTINE SPECIFICATIONS

The specifications begin with the main include-file, and go on with the 3D subroutines in alphabetical order. The specifications of the 2D and 1D subroutines are the same as in the 3D case, apart from obvious differences. The specifications of the 3Ds subroutines are similar to the 3D case. The non-obvious differences are the following: in the argument list, `ND` is replaced by `ND1,ND2,ND3` (`ND1*ND2*ND3` is the size of the 3-D data), and `NT_MIN` is replaced by `NT_MIN1,NT_MIN2,NT_MIN3` (scale parameters along each direction).

```
C-----------------------------------------------------------------------
C      filters.inc
C-----------------------------------------------------------------------

c      include 'bior2.8.inc'
c      include 'bior4.4.inc'
c      include 'bior5.5.inc'
c      include 'bior6.8.inc'
c      include 'daub4.inc'
c      include 'daub6.inc'
c      include 'daub8_e.inc'
c      include 'daub8_s.inc'
c      include 'daub10_e.inc'
c      include 'daub10_s.inc'
c      include 'haar.inc'
c      include 'rbio2.8.inc'
c      include 'rbio4.4.inc'
c      include 'rbio5.5.inc'
c      include 'rbio6.8.inc'
```

• • • Choose the wavelet by commenting out the line with the proper include-file • • •

Include-Files:

`bior2.8.inc`	: bi-orthogonal spline wavelet
`bior4.4.inc`	: bi-orthogonal spline wavelet, also quasi-orthogonal
`bior5.5.inc`	: bi-orthogonal spline wavelet;
	: the original name 'bior 5.5' is probably improper,
	: in which case the proper name should be 'bior 6.4'
`bior6.8.inc`	: bi-orthogonal spline wavelet, also quasi-orthogonal
`daub4.inc`	: Daubechies wavelet, orthogonal
`daub6.inc`	: Daubechies wavelet, orthogonal
`daub8_e.inc`	: Daubechies wavelet, orthogonal
`daub8_s.inc`	: Daubechies wavelet (symlet), orthogonal
`daub10_e.inc`	: Daubechies wavelet, orthogonal
`daub10_s.inc`	: Daubechies wavelet (symlet), orthogonal
`haar.inc`	: Daubechies (Haar) wavelet, orthogonal
`rbio2.8.inc`	: reverse bi-orthogonal spline wavelet
`rbio4.4.inc`	: reverse bi-orthogonal spline wavelet, also quasi-orthogonal
`rbio5.5.inc`	: reverse bi-orthogonal spline wavelet;
	: the original name 'rbio 5.5' is probably improper,
	: in which case the proper name should be 'rbio 6.4'
`rbio6.8.inc`	: reverse bi-orthogonal spline wavelet, also quasi-orthogonal

```
C-----------------------------------------------------------------------
      subroutine DENOISE_G3D(CHA_CT,CHA_SD,CHA_T,CHA_TN,ND,NT_MIN,SIGMA,
     +                       T,X,Y)

      character*3 CHA_CT, CHA_SD, CHA_T, CHA_TN
      integer     ND, NT_MIN
      real*8      SIGMA, T,
     +            X(0:ND-1,0:ND-1,0:ND-1), Y(0:ND-1,0:ND-1,0:ND-1)
```

Arguments:

CHA_CT	(input)	: thresholding option, 'd' or 'a&d'
CHA_SD	(input)	: thresholding option, 'ng' or 'g'
CHA_T	(input)	: thresholding option, 'h' or 'l'
CHA_TN	(input)	: thresholding option, 'h' or 's'
ND	(input)	: ND**3 is the size of the 3-D data
NT_MIN	(input)	: scale parameter
SIGMA	(output)	: standard deviation of noise
T	(output)	: threshold
X	(input)	: noisy data
Y	(output)	: de-noised data

Calls: FWT_3DF, MEDIAN, FWT_3DB

```
C-----------------------------------------------------------------------
      subroutine DENOISE_P3D(CHA_CT,CHA_SD,CHA_T,CHA_TN,ND,NT_MIN,BIAS,
     +                       SIGMA,T,X,Y)

      character*3 CHA_CT, CHA_SD, CHA_T, CHA_TN
      integer     ND, NT_MIN
      real*8      BIAS, SIGMA, T,
     +            X(0:ND-1,0:ND-1,0:ND-1), Y(0:ND-1,0:ND-1,0:ND-1)
```

Arguments:

CHA_CT	(input)	: thresholding option, 'd' or 'a&d'
CHA_SD	(input)	: thresholding option, 'ng' or 'g'
CHA_T	(input)	: thresholding option, 'h' or 'l'
CHA_TN	(input)	: thresholding option, 'h' or 's'
ND	(input)	: ND**3 is the size of the 3-D data
NT_MIN	(input)	: scale parameter
BIAS	(output)	: local bias of the Anscombe transformation,
		: estimated analytically and subtracted
SIGMA	(output)	: standard deviation of noise
T	(output)	: threshold
X	(input)	: noisy data
Y	(output)	: de-noised data

Calls: DENOISE_G3D

```
C-----------------------------------------------------------------------
      subroutine DENOISE_SG3D(CHA_SD,CHA_T,ND,NT_MIN,C,SIGMA,T,X,Y)

      character*3 CHA_SD, CHA_T
      integer     ND, NT_MIN
      real*8      C, SIGMA, T,
     +            X(0:ND-1,0:ND-1,0:ND-1), Y(0:ND-1,0:ND-1,0:ND-1)
```

NOTE:

• Partial De-Noising at a Pre-Assigned Level •

Arguments:

`CHA_SD`	(input)	: thresholding option, `'ng'` or `'g'`
`CHA_T`	(input)	: thresholding option, `'h'` or `'l'`
`ND`	(input)	: `ND**3` is the size of the 3-D data
`NT_MIN`	(input)	: scale parameter
`C`	(input)	: contraction parameter
`SIGMA`	(output)	: standard deviation of noise
`T`	(output)	: threshold
`X`	(input)	: noisy data
`Y`	(output)	: de-noised data

Calls: `FWT_3DF`, `MEDIAN`, `FWT_3DB`

```
C-----------------------------------------------------------------------
      subroutine DENOISE_SP3D(CHA_SD,CHA_T,ND,NT_MIN,BIAS,C,SIGMA,T,X,Y)

      character*3 CHA_SD, CHA_T
      integer     ND, NT_MIN
      real*8      BIAS, C, SIGMA, T,
     +            X(0:ND-1,0:ND-1,0:ND-1), Y(0:ND-1,0:ND-1,0:ND-1)
```

NOTE:

• Partial De-Noising at a Pre-Assigned Level •
• Numerical Computation of the Global Bias •

Arguments:

`CHA_SD`	(input)	: thresholding option, `'ng'` or `'g'`
`CHA_T`	(input)	: thresholding option, `'h'` or `'l'`
`ND`	(input)	: `ND**3` is the size of the 3-D data
`NT_MIN`	(input)	: scale parameter
`BIAS`	(output)	: global bias of the Anscombe transformation,
		: computed numerically and subtracted
`C`	(input)	: contraction parameter
`SIGMA`	(output)	: standard deviation of noise
`T`	(output)	: threshold
`X`	(input)	: noisy data
`Y`	(output)	: de-noised data

Calls: `DENOISE_SG3D`

```
C-----------------------------------------------------------------------
      subroutine FWT_3DB(ND,NT_MIN,X,Y)

      integer ND, NT_MIN
      real*8  X(0:ND-1,0:ND-1,0:ND-1), Y(0:ND-1,0:ND-1,0:ND-1)
```

Arguments:

ND	(input)	: ND**3 is the size of the 3-D data
NT_MIN	(input)	: scale parameter
X	(input)	: wavelet coefficients
Y	(output)	: inverse transformed data

Calls: BACKTRANS

```
C-----------------------------------------------------------------------
      subroutine FWT_3DF(ND,NT_MIN,X,Y)

      integer ND, NT_MIN
      real*8  X(0:ND-1,0:ND-1,0:ND-1), Y(0:ND-1,0:ND-1,0:ND-1)
```

Arguments:

ND	(input)	: ND**3 is the size of the 3-D data
NT_MIN	(input)	: scale parameter
X	(input)	: original data
Y	(output)	: transformed data

Calls: FORTRANS